CHAPTER 1

INTRODUCTION

1.1 TEMPORAL DATABASES

Relational database systems were designed to maintain accurate, complete and consistent data pertaining to an organization. In such databases, the most recent data i.e. current data is only stored. When new data values are available through updates, the existing data values are removed from these databases permanently and hence captures a snapshot of reality. Although conventional databases serve some applications well, they are insufficient for those applications in which past, present and future data are required to be stored and retrieved. Hence, a database is needed that fully supports the storage and querying of information that varies over time (Abdullah uz Tansel et al 1993). A database management system that maintains past, present and future data is called a Temporal Data Base Management System (TDBMS) and it supports both valid time and transaction time (Snodgrass and Illsoo Ahn 1986). Valid time denotes the time at which an event took place in the real world. Transaction time, on the other hand, refers to the time at which a transaction has been processed to update the database. These two times are orthogonal and can be supported separately, or both can be supported in concert. If a user wishes to have a user defined time in a conventional database, such user defined time is an uninterpreted time domain, managed by the user for which the data model supports only the operations of input, output and comparison.
There are some applications such as weather monitoring and forecasting in which the database systems cannot operate with either real world time or system time alone. As they need to perform analysis based on the past and the current data in emerging situations, they need to have the system time for continuous monitoring of events. The support of transaction time and valid time in the same relation allows tuples to be viewed as valid at the same moment, seen as of other moment, thereby allowing complete capture of the history of retroactive or proactive changes (Snodgrass 1997). These kinds of time induce different types of databases. A traditional database supporting neither valid time nor transaction time as a special dimension is termed as a Snapshot database. It contains only a snapshot of the real world. On the other hand, a Static Rollback database supports only transaction time and hence allows rolling back the database to a previous state. This database records all states of a database and provides a complete audit trail (Christian et al 1991). A Historical database contains the entire history of an enterprise and it supports only valid time (Sarda 1990). According to Snodgrass and Illsoo Ahn (1986), there are four approaches to design temporal databases. They are

(i) Instant Stamping of Tuples - Each tuple includes the time value at which the data in the tuple becomes current.

(ii) Interval Stamping of Tuples - Each tuple includes two time values that define an interval during which the data are current.

(iii) Instant Stamping of Attributes - A time value is associated with each attribute value.

(iv) Interval Stamping of Attributes - Two time values defining an interval are associated with each attribute value in a tuple.
Any database system that needs to store and retrieve temporal data requires to select one or more of these approaches. In this work, Interval Stamping of Tuples is taken for valid time and Instant Stamping of Tuples is taken for transaction time.

Time is continuous in nature. There are two common views of time, Continuous and Discrete time. Continuous time is considered to be isomorphic to real numbers, whereas discrete time is considered to be isomorphic to natural numbers or a subset of real numbers. Both views assume that time is linearly ordered. Discrete interpretation of time has been adopted in temporal databases because of its simplicity and relative ease of implementation. Hence, time can be interpreted as a set of equally spaced and ordered time points which is denoted by T, where T = { 0,1,2,... NOW }. The symbol 0 is the relative beginning, and NOW is a special constant to represent current time. An intelligent temporal database management system is a system that can be developed by the integration of Knowledge Based Systems and Database Technology with Temporal reasoning operators. In the past, many works have been carried out in the development of intelligent databases and query languages. The intelligent query language called Extended Structured Query Language (ETSQL) has been designed and implemented for querying the intelligent TDBMS called TEMIBASE (Kannan et al 1994, 1998b). This system provides intelligence through four groups of production rules (Kannan et al 1995) which has two user interfaces namely a natural language interface (Pandiaraj et al 1996) and a prolog interface (Swamynathan et al 1996). Moreover, it provides a null handling facility (Bhuvaneswari et al 1998) and the query language has been developed using a temporal relational algebra for effective forecasting (Kannan et al 1998a). In the commercial database management systems, there are no separate features for forecasting, explanation regarding the inferences made during a prediction or planning operation, knowledge based planning and learning new rules.
In this work, an intelligent temporal database management system that provides temporal mining and reasoning facilities through the query language Extended Temporal Mining Structured Query Language (ETMSQL), which performs non-monotonic temporal reasoning during query processing by firing rules. For this purpose, this query language provides additional constructs in addition to the Structured Query Language (SQL) features such as automatic insertion of start_time, end_time (both for valid time) and system_time (for transaction time) attributes in the CREATE TABLE statement, an additional CREATE RULE command, history maintenance options in UPDATE and DELETE statements, WHEN clause for interval operations and a special MWHEN clause for mining options. Moreover, this system supports temporal reasoning in the form of prediction, explanation, planning and learning using the temporal reasoning option. Another feature of this system is the automatic provision of security checking mechanisms using access control and intrusion detection.

1.2 TEMPORAL REASONING

Temporal reasoning involves a number of capabilities such as the provision of finding the dependencies between intervals and instants and prediction of the future using the past and present data. There are a number of applications such as weather forecasting, market trend analysis, health care management and banking systems which require the use of temporal reasoning in decision making, planning and to learn new rules. Temporal mining helps to perform temporal reasoning in an effective way, since it finds temporal patterns that can be used for analysis and prediction. Moreover, temporal reasoning on temporal data requires the use of temporal mining in order to provide the ability to handle dependencies among different temporal data using rules. In order to perform temporal reasoning, Allen (1983) has
introduced a most influential theory called Allen’s interval algebra. His algebra is efficient in the areas of natural language processing and problem solving with temporal intervals.

Temporal reasoning with database systems should provide the possibility of managing temporal information not only to retrieve the available information from the database but also the ability to derive new data through predictions. Temporal query languages are normally developed based on temporal relational algebra. The model used to design Temporal QUEry Language (TQUEL) (Richard Snodgrass 1987) uses interval stamping of tuples for both valid time and transaction time. A temporal query language that includes additional temporal reasoning constructs can perform temporal reasoning in addition to database management functionalities. Reasoning on temporal data requires a number of different capabilities (Kannan et al 1996). These include the ability to handle dependencies among different temporal data, to handle incomplete temporal data, to determine the period of validity for data values to handle real and apparent contradictions, and to recognize the incorrect data.

In this research work, rules are generated using data mining algorithms and additionally a knowledge base consisting of general rules and domain rules are used to perform temporal reasoning effectively.

1.2.1 Rules for Temporal Reasoning

Works on rule based systems have originated from logic programming communities. The integration of rule systems and database systems can provide a reliable environment for temporal reasoning. This integration can be carried out either by adding rules to the database systems or by adding database functionalities to a rule system. Abiteboul et al (1991)
focus on using database technology to efficiently support production rules. Rules supporting temporal database systems are of two types namely passive rules and active rules. Passive rules respond only to queries from users and application programs, while active rules take action in response to updates and events.

There are many database systems that support rules (Bogdan et al 1993, Kannan et al 1995, 1996). The POSTGRES database system provides a rule manager that allows four types of cascading rules namely retrieve..retrieve, retrieve..update, update..update and update..retrieve, to provide forward chaining and backward chaining control flows and also to provide an alerter and audit trail (Stonebraker et al 1988, 1990). Jennifer et al (1990) discussed the use of set-oriented production rules in relational database management systems. The Ariel DBMS provides a rule system that uses a set of active Event Condition Action (ECA) rules, which provides a time specifier clause to specify instants or intervals during which rules can be fired (Hanson 1989).

The High Performance ACtive (HiPAC) system (Sharma Chakravarthy 1989) has a rule management and evaluation subsystem which recognizes ECA rules explicitly and encapsulates them into a rule base. It supports temporal events as absolute time instants as well as relative time instants. These rules are used to serve various object manipulation and constraint management tasks. Generally rules can be active or passive. Active ECA rules can be defined by users, applications or database administrators. The purpose of defining active ECA rules is to monitor the occurrence of events actively. The events specify what causes the rule to be triggered. Moreover, Data Base Management Systems (DBMSs) have been passive traditionally since queries or transactions are executed only when explicitly requested. If the action of one rule triggers another rule, such rules are called
as cascading rules. A diagnostic decision support system proposed by Khanna Nehemiah et al (2006) provides explanation facility which uses two types of rules namely IF..THEN rules and ECA rules. In their work, the rules are applied using a backward chaining flow control.

In this work, a rule manager which supports four types of rules namely active IF..THEN rules, passive IF..THEN rules, active cascading IF..THEN rules and passive cascading IF..THEN rules has been proposed and implemented. The combination of rule manager and the knowledge base provides an effective subsystem called temporal reasoning subsystem which supports temporal reasoning for application as well as temporal database security. The rule system proposed in this work enables the implementation of both access control and intrusion detection to provide effective security.

1.2.2 Temporal Mining Tasks in Temporal Reasoning

Temporal data mining techniques allow for the possibility of computer driven, automatic exploration of the data and intelligent data analysis. Intelligent data analysis refers to all methods that are committed to automatically transform data into information exploiting the background knowledge on the domain (Hand 1997). The primary goal of intelligent data analysis is to provide methods that support data understanding. In most application areas that have been studied for data mining, the time at which an event has happened is recorded effectively. However, most data mining techniques treat data in temporal databases at best as data series in chronological order and ignore the time stamps. Moreover, data mining is application-dependent and user-dependent. Therefore, Knowledge Discovery in Databases (KDD) process provides users with flexible and powerful descriptive languages to express data mining tasks.
Xiaodong et al (1998) have proposed a prototype system and its architecture for mining temporal patterns in which the generic temporal and periodic patterns are formally defined and the mining problem for temporal patterns is described. According to them, the user of a KDD system has to select the relevant data subset, identify suitable classes of patterns and define good criteria for interestingness of the patterns. For this purpose, an SQL-based mining language called Temporal Query and Mining Language (TQML) was presented by them and it has been demonstrated that this language can be used to discover various types of temporal patterns or rules. However, their system provides the facilities to find only a few temporal patterns.

In this work, the query language ETMSQL provides additional constructs namely MWHERE and MWITH clauses which describes the periodicity that mining users are interested in with or without a specific time framework for effective mining process. The data relevant to the data mining task can be stated in the SELECT-FROM-WHERE-GROUP clause. Moreover, it can be used as a temporal query language as well as to call the data mining option. The main advantage of this work over TQML and ETSQL is that it integrates temporal mining with temporal reasoning.

1.3 SECURITY IN TEMPORAL DATABASES

In this work, security is provided to the proposed TDBMS using two approaches namely access control and intrusion detection. Both these techniques use intelligent agents and production rules for providing effective security.
1.3.1 Access Control

According to Ravi et al (1999), organizations today have assumed a global presence and are no longer restricted within their narrow confines. Organizations are increasingly under pressure to accommodate their ever-expanding user list. Typically, each application system has its specific user administration control and is managed by the respective system administrator. This is because owing to the change in need of business, more new applications are added. As each application system can behave differently than others, there is a lack of common user privilege policy. Different administrators maintain the users lists, giving rise to duplicate, and at times, inconsistent user entries. If user records in one system are updated, there is no way to communicate this to the other systems. This has led to controlling user access to information and other resources becoming even more important and complex. Hence, it is necessary to develop systems for role based access control. Access control is a mechanism for limiting access to resources based on user’s identities and their membership in various predefined groups. Access control techniques are sometimes categorized as either discretionary or mandatory. The authors define Discretionary Access Control (DAC) as an access policy which is determined by the owner of an object. The owner decides who is allowed access to the object and what privileges they have. On the other hand, Mandatory access control system is an access policy determined by the system, not the owner.

Currently, Role-Based Access Control (RBAC) models are receiving increased attention as a generalized approach to access control. Roles may be available to users at certain time periods, and unavailable at others. It differs from traditional identity based access control in that it takes advantage of the concept of role relations. In such models, user's rights to access computer resources (objects) are determined by the user's assignment
to a role and by the roles' permissions to perform operations on objects. Thus, a role is a collection of permissions or operations on a set of objects assigned by the system. Based on the role's intended function as well as policies within an organization, it is possible to control the operations by a role on database objects (Ravi et al 1999).

In order to control the time-sensitive activities present in various applications like work flow management systems and real time databases, access control systems are required to be augmented with temporal constraints. The ability of the underlying access control mechanisms to accurately enforce these constraints depends on the implementation, conformance with the specifications and absence of any violations in the implementation. Therefore, it becomes essential to assure that the underlying implementation realizes the given access control policy completely and has no additional unspecified functionality.

There are many works that are found in the literature that deal with access control (Elisa Bertino et al 2001, James et al 2001a). One of the important aspects of access control is that it provides a means for time constraining accesses to limit the use of resource. Such constraints are essential for controlling time-sensitive activities that may be present in various applications such as Work Flow Management Systems (WFMSs), where various workflow tasks, each having some timing constraints, need to be executed in some order. To address general time-based access control needs, Elisa Bertino et al (2001) proposed a Temporal Role Based Access Control model (TRBAC), which has been generalized recently by James et al (2001a) called as the Generalized-Temporal Role Based Access Control (GTRBAC) model. This GTRBAC incorporates a set of language constructs for the specification of various temporal constraints on roles, including constraints on their activations as well as on their enabling times, user-role
assignment and role-permission assignments. Among them, James et al (2005) proposed a generalized temporal model for role-based access control. The main advantage of their work was the provision of temporal support for effective access control. They provide various roles that were assigned to particular service, to access the resources at particular time. The model proposed by Eliza et al (2001) provides temporal constructs for effective implementation of the TRBAC. TRBAC is an extension of RBAC models that supports temporal constraints on the enabling or disabling of roles. TRBAC supports periodic role enabling or disabling, and temporal dependencies among such actions. Such dependencies expressed by means of role triggers can also be used to constrain the set of roles that a particular user can activate at a given time instant. The firing of a trigger may cause a role to be enabled or disabled either immediately, or after an explicitly specified amount of time. Enabling or disabling actions may be given a priority that may help in solving conflicts, such as the simultaneous enabling and disabling of a role.

In this work, a Temporal Role Based Access Control (TRBAC) model with intelligent agents which provides seamless access to information present in the database has been proposed. Moreover, this model supports temporal constraints on the enabling or disabling of roles. We use various agents such as Access Checking Agent, Decision Making Agent, Rule Priority Checking Agent, Application Agent and Communication Agent for providing effective access control.

1.3.2 Intrusion Detection Systems

Intrusion Detection is necessary for intelligently monitoring the events occurring in computer systems, networks and databases in order to detect the signs of violations of the security policy. The primary aim of an Intrusion Detection Systems (IDSs) is to protect the availability,
confidentiality and integrity of critical networked information systems such as distributed databases. The effectiveness of IDSs are characterized by both the method used to detect attacks and the placement of the Intrusion Detection System (IDS) on the network. IDSs are monitoring programs, which aim at detecting intruders who are trying to access the resources illegally in a computer network (Dorothy 1987, Rebecca 2000, McHugh 2001, Fernando et al 2004 and Nong et al 2004). Intrusion detection system can be classified into the two types namely misuse detection and anomaly detection system based on how data analysis is carried out (Ingo et al 2005). In misuse detection, patterns are learned from already known attacks. These learned patterns are searched through the unseen data to find intrusions of the already known types. On the other hand, in anomaly intrusion detection, patterns are learned from normal data.

An IDS may perform either misuse detection or anomaly detection and may be deployed as either a network-based system or a host-based system. This results in four general groups for intrusion detection namely misuse-host, misuse-network, anomaly-host and anomaly-network. Misuse detection relies on matching known patterns of hostile activity against databases of past attacks. They are highly effective at identifying known attack and vulnerabilities, but rather poor in identifying new security threats. Anomaly detection searches for something rare or unusual by applying statistical measures or Artificial Intelligence (AI) methods to compare current activity against historical knowledge. Common problems with anomaly based systems are that, they often require extensive training data for artificial learning algorithms, and they tend to be computationally expensive, because several metrics are often maintained, and need to be updated against every systems activity.
Some IDSs combine qualities from all these categories (usually implementing both misuse and anomaly detection) and are known as hybrid systems. In the past, AI techniques have been applied both to misuse detection and also for anomaly detection. Standford Research Institute’s (SRI’s) Intrusion Detection Expert System (IDES) (Lunt 1993) encodes an expert’s knowledge of known patterns of attack and system vulnerabilities as IF-THEN rules. Time-based Inductive Machine (TIM) (Teng 1990) for intrusion detection learns sequential patterns.

The existing commercial intrusion detection systems are based on misuse detection approach, which means these systems will only be able to detect known attack types. In most cases, they tend to be ineffective due to various reasons like non-availability of attack patterns, time consumption for developing new attack patterns, insufficient attack data etc., These existing techniques do not consider temporal aspects. On the other hand, with the explosive growth of data in the World Wide Web (WWW), either data warehouses or advanced data models such as temporal databases are necessary to store such a large amount of information for fast retrieval and decision making. Recently, techniques from data mining area have been used to mine normal patterns from audit data (Lee et al 1998, Mukkamala et al 2000, Stolfo et al 2001) which are based on trend analysis, fuzzy logic and neural networks to minimize and control intrusions. Gomez et al (2003) proposed a technique to generate fuzzy classifiers using genetic algorithms that can detect anomalies and some specific intrusions. Their main idea is to evolve two rules, one for the normal class and the other one for the abnormal class using a profile data set with information related to the computer network during the normal behavior and during intrusive (abnormal) behavior. Mrutyunjaya et al (2007) developed an efficient data mining algorithm called Naïve Bayes Classification Algorithm for anomaly based intrusion detection.
In this thesis, a new Agent Based Intelligent Intrusion Detection System (ABIIDS) has been proposed for effective intrusion detection that provides highly accurate results, low in false alarms, not possible to cheat by small variations in patterns, and is of real time along with high performance computing. This intrusion detection subsystem uses a Modified Bayesian Classification Algorithm for classifying the intruders based on their access patterns. This proposed Modified Bayesian Classification Algorithm operates on a strong independence assumption. This means that the probability of one attribute does not affect the probability of other. A temporal data set has been used for experimental results in this algorithm. It obtains the dataset for the particular interval time. The classifier identifies the type of intruder and takes the necessary action with the help of Action Agent. In case of time instants, the start time and end time of an interval are equal and hence, the interval based approach satisfies both instants and intervals. Moreover, we use a semantic relation extraction method to form a data set for intrusion detection when it is applied for text databases.

1.4 THESIS CONTRIBUTIONS

The major contributions of this work are the proposal of a new architecture, a special Query language called Extended Temporal Mining Structured Query Language (ETMSQL), temporal role based access control methods and intrusion detection method for identifying the intrusions effectively. The proposed query language ETMSQL provides constructs for querying the intelligent temporal databases for reasoning and mining with a focus on security. It has been designed by integrating and extending the query languages ETSQL and TQML. ETMSQL provides the following additional constructs.
• Automatic insertion of start_time, end_time (both for valid time) and stime (for transaction time) attributes in the CREATE TABLE statement.

• Automatic insertion of system_time for system time (stime) attribute.

• History maintenance options in UPDATE and DELETE statements.

• WHEN clause for interval operations.

• Constructs for specification of time granularity.

• Extensions of group functions to include interval value functions instant comparison operators and TIME SLICE clause in SELECT statement.

• MINE options for security in which security is provided with both temporal role based access control and intrusion detection.

• Options for Forecasting, Analysis of past, Planning and Learning.

• Special constructs for rules.

This Temporal Role Based Access Control has been provided effectively using multi agents that support temporal constraints on the enabling or disabling of roles. Moreover, this system provides a number of intelligent agents that are assigned a specific type of security handling job and has been developed by extending TRBAC. In order to achieve the intrusion detection effectively, an extended algorithm called Modified Bayesian Classification Algorithm has been proposed and implemented by extending
the Naïve Bayesian Classification algorithm with temporal constraints to
distinguish the intruders from normal users. In this system, five types of
intelligent agents have been proposed and implemented for providing an
effective temporal security subsystem. Finally, a rule manager has been
designed and implemented in which rules have been generated, validated and
stored in knowledge base. This rule manager fires rules using forward
chaining and backward chaining flows for performing temporal reasoning.

1.5 THESIS ORGANIZATION

The remainder of this thesis is organized as follows:

Chapter 2 discusses the related works in this area and compares
them with this work. Chapter 3 describes the overall architecture of the
system proposed in this thesis. Chapter 4 explains the temporal reasoning
subsystem along with algorithms proposed and provides the results obtained
in this research work. Chapter 5 details the temporal security subsystem with
algorithms used for access control and intrusion detection and produces the
results obtained in this work. Chapter 6 gives the conclusions on this work
and presents some possible future enhancements.