Security Aspects of Distributed Database

Dr.Mohmad Kashif Qureshi

Assistant Professor, Jazan University, Jazan

Saudi Arabia

Srk1521@gmail.com

ABSTRACT

Security concerns *must be addressed when developing a distributed database. When choosing between the object oriented model and the relational model, many factors should be considered.*
The most important of these factors are, Single level and multilevel access controls, Protection against inference, Maintenance of integrity.
When determining which distributed database model will be more secure for a particular application, the decision should not be made purely on the basis of available security features.
One should also question the efficacy and efficiency of the delivery of these features.
In this paper I tried my level best to enlighten the design issues and security issues of ***distributed database*** *and gave the suggestion to improve design and security issues of* ***distributed database.***
Distributed database *has* ***3 key design issues*** *are* ***Fragmentation****,* ***Allocation, and Replication.***
Because Distributed database has ***disadvantages*** *also like* ***Complexity, Cost, Security, Integrity control is more difficult, Database design is more complex***

This paper will observe the ***fundamental features of the distributed database architecture****. Learning the task of distributed database management system will guide us to a* ***successful design****. The design will improve* ***scalability, accessibility and flexibility*** *while accessing various types of data. Building a triumphant distributed database system requires to acknowledge the* ***importance of security issues*** *that may arise and possibly settlement of the access control and the integrity of the system. I suggested some solutions for some security aspects such as* ***multilevel access control, confidentiality, reliability, integrity and recovery*** *that affect to a distributed database system.*

Keywords—

Distributed database security, Distributed database Design, Fragmentation, Allocation, and Replication distributed database, scalability, accessibility and flexibility ,distributed database management system, distributed database retrieval problems, discretionary security distributed database, query processing.

1. INTRODUCTION

A database implemented on a network. The component partitions are distributed over various nodes (stations) of the network. Depending on the specific update and retrieval traffic, distributing the database can significantly enhance overall performance.

A distributed database is a database in which storage devices are not all attached to a common processing unit such as the CPU. It may be stored in multiple computers located in the same physical location, or may be dispersed over a network of interconnected computers. Unlike parallel systems, in which the processors are tightly coupled and constitute a single database system, a distributed database system consists of loosely coupled sites that share no physical components.

"A distributed database is a collection of databases which are distributed and then stored on multiple computers within a network". A distributed database is also a set of databases stored on multiple computers that typically appears to applications as a single database. "Consequently an application can simultaneously access and modify the data in several databases in a network ". A database link connection allows local users to access data on a remote database ". For this connection to occur, each database in the distributed system must have a unique global database name in the network domain. The global database name uniquely identifies a database server in a distributed system.

Distributed database is a database implemented on a network. The component partitions are distributed over various nodes (stations) of the network. Depending on the specific update and retrieval traffic, distributing the database can significantly enhance overall performance. See also partition.

Distributed database management system is a database management system capable of managing a distributed database

A distributed database management system ('DDBMS') is a software system that permits the management of a distributed database and makes the distribution transparent to the users. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network. Sometimes "distributed database system" is used to refer jointly to the distributed database and the distributed DBMS. A distributed database system consists of loosely coupled sites that share no physical component. Database systems that run on each site are independent of each other.

The main difference between centralized & distributed databases is that the distributed databases are typically geographically separated, are separately administered, have slower interconnection. Also in distributed databases I differentiate between local & global transactions. A local transaction is one that accesses data only from sites where the transaction originated. A global transaction, on the other hand, is one that either accesses data in a site different from the one at which the transaction was initiated, or accessed data in several different sites.

In this paper, I will **review** all the **security and design features & issues of databases in a general form and distributed databases in particular**. I will also investigate the **security problems** found in both models. Moreover, I will evaluate the security problems unique to each

system. Finally, comparing the relative merits of each model with respect to security will be applied as well.

2. DATABASE SYSTEM CONCEPTS

Data management is the control of data from acquisition and input through processing, output, and storage. In microcomputers, hardware manages data by gathering it, moving it, and following instructions to process it. The operating system manages the hardware and ensures that the parts of the system work in harmony so that data is stored safely and accurately. Application programs manage data by receiving and processing input according to the user's commands, and sending results to an output device or to disk storage. The user also is responsible for data management by acquiring data, labeling and organizing disks, backing up data, archiving files, and removing unneeded material from the hard disk.

Database management system is a software interface between the database and the user. A database management system handles user requests for database actions and allows for control of security and data integrity requirements.

A Database Management System **(DBMS)** is a set of programs that enables you to store, modify, and extract information from a database, it also provides users with tools to add, delete, access, modify, and analyze data stored in one location. A group can access the data by using query and reporting tools that are part of the DBMS or by using application programs specifically written to access the data. DBMS's also provide the method for maintaining the integrity of stored data, running security and user's access, and recovering information if the system fails. The information from a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report. Many DBMSs also include a graphics component that enables you to output information in the form of graphs and charts. Database and database management system are essential to all areas of business, they must be carefully managed. There are many different types of DBMSs, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications: computerized library systems, flight reservation systems, and computerized parts inventory systems. It typically supports query languages, which are in fact high-level programming languages, dedicated database languages that considerably simplify writing database application programs. Database languages also simplify the database organization as Ill as retrieving and presenting information from it. A DBMS provides facilities for controlling data access, enforcing data integrity, managing concurrency control, and recovering the database after failures and restoring it from backup files, as Ill as maintaining database security.

3. MAIN FUNCTIONS OF THE DATABASE MANAGEMENT SYSTEM

1. Indexing
2. Views
3. Security
4. Integrity
5. Concurrency
6. Backup/Recovery
7. Design
8. Documentation
9. Update/Query

One of the **main functions** of the database management system is doing the heavy lifting for you. You don't necessarily have to know exactly where all that data is in the system, as long as the database management system knows where it all is, it can deliver a report for you to peruse. This might not seem to matter if you're thinking of just your computer; but throw in a mainframe that contains reams and reams of data, and I am talking about a huge amount of information that can be stored any number of places within the mainframe system. The result is the same, though: a report that you can read, analyze, and act on.

This **functionality** also extends to a multi-user database. Such a database management system under this scenario would allow you as one user to operate all functions within the database without having to know what other users are accessing the same database? The user interacts with the database management system in order to utilize the database and transform data into information. Furthermore, a" database offers many advantages compared to a simple file system with regard to **speed, accuracy, and accessibility** such as: **shared access, minimal redundancy, data consistency, data integrity, and controlled access** ". All of these aspects are enforced by a database management system. Among these things let's review some of the many different types of databases.

The vertical columns are known as the attributes. "Data that is stored on two or more tables establishes a "link" between the tables based on one or more field values common in both tables." A relational database uses a standard user and application program interface called Structure Query Language (SQL). This program language uses statements to access and retrieve queries from the database. Relational databases are the most commonly used due to the reasonable ease of creating and accessing information as well as extending new data categories.

When dealing with complicated data or complex relationships, "**object databases** are more commonly used "."**Object databases in contrast to relational databases** store objects rather than data such as integers, strings, or real numbers". Each object consists of attributes, which define the characteristics of an object. Objects also contain methods that define the behavior of an object (also known as procedures and functions). When storing data in an object database there are **two main types of methods**, **one technique** labels each object with a unique ID. Every unique ID is defined in a subclass of its own base class, where inheritance is used to determine attributes. A **second method** is utilizing virtual memory mapping for object storage and management. Advantages of object databases with regard to relational databases allow more concurrency control, a decrease in paging, and easy navigation. However, there are "some **disadvantages of object databases** compared to relational databases such as: less effective with simple data and relationships, slow access speed, and the fact that relational databases provide suitable standards oppose to those for object database systems".

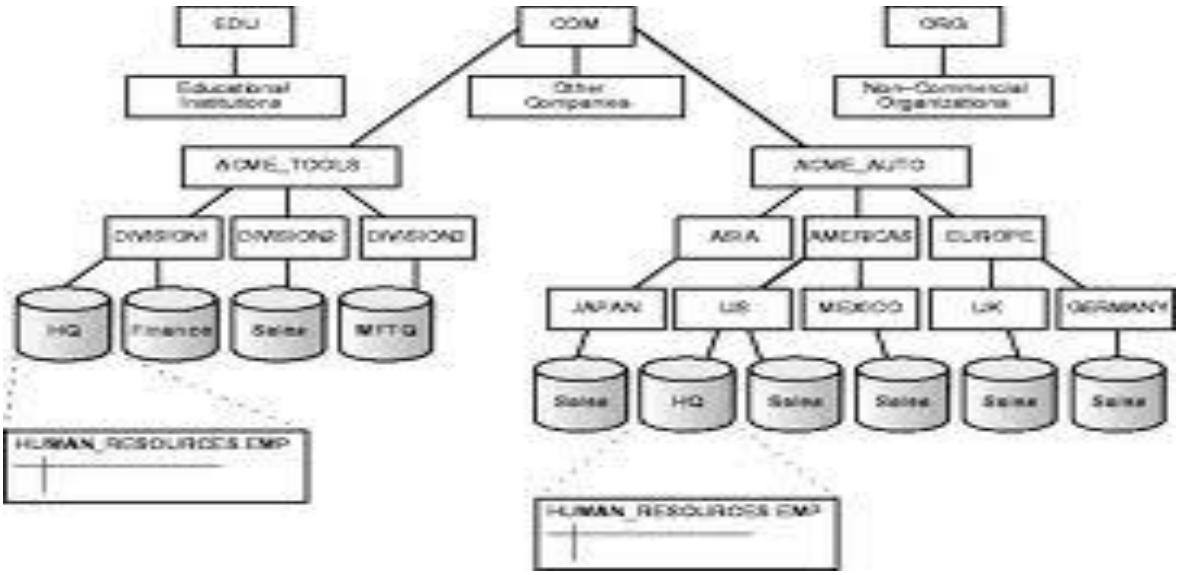
On the other hand, **network databases** ease some of the problem incorporated with hierarchical databases such as data redundancy. "The network model represents the data in the form of a network of records and sets which are related to each other, forming a network of links."

In many ways, the **Network Database model** was designed to solve some of the more serious problems with the Hierarchical Database Model. Specifically, the Network model solves the problem of data redundancy by representing relationships in terms of sets rather than hierarchy.

The model had its origins in the Conference on Data Systems Languages **(CODASYL)** which had created the Data Base Task Group to explore and design a method to replace the hierarchical model.

The **network model is very similar to the hierarchical model** actually. In fact, the hierarchical model is a subset of the network model. HoIver, instead of using a single-parent tree hierarchy, the network model uses set theory to provide a tree-like hierarchy with the exception that child tables Ire allowed to have more than one parent. This allowed the network model to support many-to-many relationships

Visually, a Network Database looks like a hierarchical Database in that you can see it as a type of tree. HoIver, in the case of a Network Database, the look is more like several trees which share branches. Thus, children can have multiple parents and parents can have multiple children.



4. DISTRIBUTED DATABASES DESIGN

Security Problems Unique to Distributed Database Management Systems are:

- Centralized or Decentralized Authorization
- Integrity
- In developing a distributed database, one of the first questions to answer is where to grant system access. Bell and Grisom outline two strategies:
- Users are granted system access at their home site/ remote site/.
- Users are granted system access at their home site
- success of this strategy depends on reliable communication between the different sites

- Since many different sites can grant access, the probability of unauthorized access increases
- Users are granted system access at the remote site.
- This strategy, while perhaps more secure, has several disadvantages.
- Probably the most glaring is the additional processing overhead required, particularly if the given operation requires the participation of several sites.
- A third possibility offered by Woo and Lam centralizes the granting of access privileges at nodes called policy servers.
- These servers are arranged in a network. When a policy server receives a request for access, all members of the network determine whether to authorize the access of the user
- Preservation of integrity is much more difficult in a heterogeneous distributed database than in a homogeneous one.
- The degree of central control dictates the level of difficulty with integrity constraints
- The homogeneous distributed database has strong central control and has identical DBMS schema
- Integrity
- Preservation of integrity is much more difficult in a heterogeneous distributed database than in a homogeneous one.
- The degree of central control dictates the level of difficulty with integrity constraints
- The homogeneous distributed database has strong central control and has identical DBMS schema
- If the nodes in the distributed network are heterogeneous several problems can arise that will threaten the integrity of the distributed data. They list three problem areas:
- Inconsistencies between local integrity constraints
- Difficulties in specifying global integrity constraints
- Inconsistencies between local and global constraints
- Local integrity constraints are bound to differ in a heterogeneous distributed database.
- The differences stem from differences in the individual organizations.
- These inconsistencies can cause problems, particularly with complex queries that rely on more than one database
- Development of global integrity constraints can eliminate conflicts between individual databases.
- Yet these are not always easy to implement.
- Global integrity constraints on the other hand are separated from the individual organizations. It may not always be practical to change the organizational structure in order to make the distributed database consistent.
- Ultimately, this will lead to inconsistencies between local and global constraints.
- Conflict resolution depends on the level of central control.

- If there is strong global control, the global integrity constraints will take precedence.

Distributed Databases Design have some issue regarding design, are as follows

- Design problem
- Design strategies(top-down, bottom-up)
- Fragmentation
- Allocation and replication of fragments, optimality, heuristics

Design problem of distributed systems making decisions about the placement of data and programs across the sites of a computer network as well as possibly
designing the network itself.

In DDBMS, the distribution of applications involves

– Distribution of the DDBMS software

– Distribution of applications that run on the database

- Distribution of applications will not be considered in the following; instead the distribution of data is studied

- Dimension for the analysis of distributed systems

– Level of sharing: no sharing, data sharing, data + program sharing

– Behavior of access patterns: static, dynamic

– Level of knowledge on access pattern behavior: no information, partial information, Complete information

- Distributed database design should be considered within this general framework

Design Strategies

Top-down approach

– Designing systems from scratch

– Homogeneous systems

Bottom-up approach

– The databases already exist at a number of sites

– The databases should be connected to solve common tasks

Distribution design is the central part of the design in DDBMSs (the other tasks are Similar to traditional databases)

– **Objective**: Design the LCSs by distributing the entities (relations) over the sites

Two main aspects have to be designed carefully

Fragmentation

- Relation may be divided into a number of sub-relations, which are distributed

Allocation and replication

- Each fragment is stored at site with ”optimal” distribution
- Copy of fragment may be maintained at several sites

In this chapter I mainly concentrate on these two aspects

• Distribution design issues

– Why fragment at all?

– How to fragment?

– How much to fragment?

– How to test correctness?

– How to allocate?

Fragmentation

What is a reasonable unit of distribution? Relation or fragment of relation?

• Relations as unit of distribution:

– If the relation is not replicated, I get a high volume of remote data accesses.

– If the relation is replicated, I get unnecessary replications, which cause problems in executing updates and waste disk space

– Might be an Ok solution, if queries need all the data in the relation and data stays at the only sites that uses the data

Fragments of relations as unit of distribution:

– Application views are usually subsets of relations

– Thus, locality of accesses of applications is defined on subsets of relations

– Permits a number of transactions to execute concurrently, since they will access different portions of a relation

– Parallel execution of a single query (intra-query concurrency)

– However, semantic data control (especially integrity enforcement) is more difficult

- Fragments of relations are (usually) the appropriate unit of distribution.

Conclusion

• Distributed design decides on the placement of (parts of the) data and programs across the sites of a computer network

• On the abstract level there are two patterns: Top-down and Bottom-up

• On the detail level design answers two key questions: fragmentation and Allocation/replication of data

- Horizontal fragmentation is defined via the selection operation σp(R)

-Rewrites the queries of each site in the conjunctive normal form and finds a Minimal and complete set of conjunctions to determine fragmentation

– Vertical fragmentation via the projection operation πA(R) computes the attribute affinity Matrix and groups "similar" attributes together

– Mixed fragmentation is a combination of both approaches

- Allocation/Replication of data

– Type of replication: no replication, partial replication, full replication

– Optimal allocation/replication modeled as a cost function under a set of constraints

– The complexity of the problem is NP-complete

– Use of different heuristics to reduce the complexity

A development of a distributed database structure for centered databases offers scalability and flexibility, allowing participating centers to maintain ownership of their own data, without introducing duplication and data integrity issues.

Moreover, Distributed database system functions include distributed query management, distributed transaction processing, distributed metadata management and enforcing security and integrity across the multiple nodes.

"The centralized database system is one of the many objectives of a distributed database system". This system will be accomplished by using the following transparencies: Location Transparency, Performance Transparency, Copy Transparency, Transaction Transparency, Transaction Transparency, Fragment Transparency, Schema Change Transparency, and Local DBMS Transparency. These eight transparencies are believed to incorporate the desired functions of a distributed database system.

The design of responsive distributed database systems is a key concern for information systems. In high bandwidth networks, latency and local processing are the most significant factors in query and update response time. Parallel processing can be used to minimize their effects, particularly if it is considered at design time. It is the judicious replication and placement of data within a network that enable parallelism to be effectively used. Distributed database design can thus be seen as an optimization problem requiring solutions to various interrelated problems: data fragmentation, data allocation, and local optimization.

A successful distributed database could include free object naming. "Free object naming means that it allows different users the ability to access the same object with different names, or different objects with the same internal name." Thus, giving the user complete freedom in naming the objects while sharing data without naming conflicts.

Concurrency control (CC) is another issue among database systems. Process of managing simultaneous execution of transactions in a shared database, to ensure the serializability of transactions, is known as concurrency control.

Simultaneous execution of transactions over a shared database can create several data integrity and consistency problems:

- **Lost Updates.**
- **Uncommitted Data.**
- **Inconsistent retrievals**.

Concurrency Control Techniques

- Pessimistic concurrency control

– Locking

- Optimistic concurrency control

It permits users to access a distributed database in a multi-programmed fashion while preserving the illusion that each user is executing alone on a dedicated system. For this, CC mechanisms are required that interleave the execution of a set of transactions under certain consistency constrains while maximizing concurrency. Two main categories of CC mechanisms are as mentioned above:

Optimistic concurrency

Delay the synchronization for transactions until the operations are actually performed. Conflicts are less likely but won't be known until they happen, making rollback operations more expensive.

Pessimistic - The potentially concurrent executions of transactions are synchronized early in their execution life cycle. Blocking is thus more likely but will be known earlier avoiding costly rollbacks.

Another activity of CC is to "coordinating, concurrent accesses to a database in a multi-user database management system (DBMS)." There are a number of methods that provide "concurrency control, such as: Two phase locking, Time stamping, Multi-version timestamp, and optimistic non-locking mechanisms. Some methods provide better concurrency control than others depending on the system"

5. THE FEATURES OF DISTRIBUTED DATABASE SYSTEM

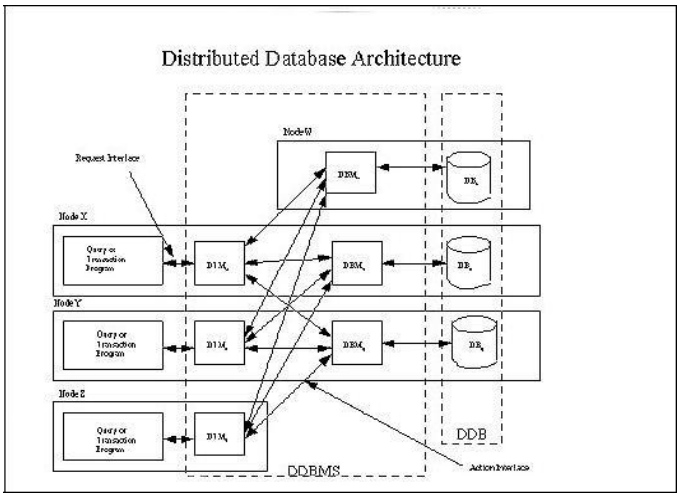
The most common features of the distributed database system. One of the **main features** in a DDBMS is the Database Manager. "**A Database Manager**

A Database Management System (DBMS) is a set of programs that enables you to store, modify, and extract information from a database, it also provides users with tools to add, delete, access, modify, and analyze data stored in one location. A group can access the data by using query and reporting tools that are part of the DBMS or by using application programs specifically written to access the data. DBMS's also provide the method for maintaining the integrity of stored data, running security and users access, and recovering information if the system fails. The information from a database can be presented in a variety of formats. Most DBMSs include a report writer program that enables you to output data in the form of a report. Many DBMSs also include a graphics component that enables you to output information in the form of graphs and charts. Database and database management system are essential to all areas of business, they must be carefully managed. There are many different types of DBMSs, ranging from small systems that run on personal computers to huge systems that run on mainframes. The following are examples of database applications: computerized library systems, flight reservation systems, and computerized parts inventory systems. It typically supports query languages, which are in fact high-level programming languages, dedicated database languages that considerably simplify writing database application programs. Database languages also simplify the database organization as well as retrieving and presenting information from it. A DBMS provides facilities for controlling data access, enforcing data integrity, managing concurrency control, and recovering the database after failures and restoring it from backup files, as well as maintaining database security.

Another main component is the User Request Interface, known some times as a customer user interface, which is usually a client program that acts as an interface to the distributed Transaction Manager. A customizable user interface is provided for entering requested parameters related to a database query. The customized parameter user interface provides parameter entry dialogs/windows in correlation to a data view (e.g., form or report) that is produced according to a database query. The parameters entered may provide for modification of the data view. Also, the manager of the database may structure data views of a database to automatically include prompts

for parameters before results are returned by the database. "These prompts may be customized by the manager and may be provided according to dialogs such as pop-ups, pull-down menus, fly-outs, or a variety of other user interface components".

"A **Distributed Transaction, Manager** is a program that translates requests from the user and converts them into actionable requests for the database managers, which are typically distributed. A distributed database system is made of both the distributed transaction manager and the database manager". The components of a DDBMS are shown in the "fig. 1" below:



6. DISTRIBUTED QUERY PROCESSING

For **centralized systems**, the primary criterion for measuring the **cost** of a particular strategy is the **number of disk accesses**.

In a **distributed system**, other issues must be taken into account:

The **cost** of a **data transmission** over the **network**.

The **potential gain** in performance from having **several sites** process **parts of the query** in **parallel**.

Several issues in **query** processing in a **heterogeneous** database

Schema translation

Write a **wrapper** for each **data source** to translate data to a global schema

Wrappers must also **translate updates** on global schema to updates on local schema

Limited query **capabilities**:

Some data sources allow only **restricted** forms of **selections**

E.g. web forms, **flat file** data sources

Queries have to be **broken** up and processed partly at the source and partly at a different site

Removal of **duplicate information** when sites have overlapping information. **Decide which** sites

to execute query **Global** query **optimization**

Distributed query processing in a large scale distributed system; it is often difficult to find an optimal plan for a distributed query: distributed systems can become very large, involving thousands of heterogeneous sites. As new databases are added/ removed to/from the system, it becomes harder for a query processor to maintain accurate statistics of the participating relations stored at the different sites and of the selectivity's of the related query operations. Also, as the workload at the various interacting processing servers and the transmission speeds of the links between them fluctuate at runtime, there is the need of distributed query engines that dynamically adapt to large- scale distributed environments. The query will be request usually from the user or the client host from a proper user interface.

By definition the client/server programs are considered distributed applications. There are two types of client/server applications. One particular client/server application is an implementation of a protocol standard defined in an RFC (request for comments).

This type of application forces the client and server programs to abide by certain rules dictated by the RFC. Another sort of client/server application is a proprietary client/server application. “In this case, the client and server programs do not necessarily conform to any existing RFC”.

In developing a proprietary client/server application the developer must decide whether to run the application over TCP or UDP. TCP connection oriented and ensures reliable byte-stream channel. However "UDP is, connectionless and forwards independent packets of data and does not guarantee delivery".

7. SECURITY ASPECTS IN DISTRIBUTED DATABASE

Approximately all of the early work on secure databases was on discretionary security. But the most **important issues in security** are **authentication, identification and enforcing appropriate access controls**. For example, the mechanisms for identifying and authenticating the user, or if a simple password mechanisms suffice? With respect to access control rules, "languages, such as SQL have incorporated GRANT and REVOKE statements to grant and revoke access to users". For many applications, simple GRANT and REVOKE statements are not sufficient. There may be more complex authorizations based on database content. Negative authorizations may also be needed. Access to data based on the roles of the user is also being investigated. "Numerous papers have been published on discretionary security in databases ".”Fig. 2", shows a simple security model.

Important security requirements for database management systems are: Multi-Level Access Control: Confidentiality. Reliability, Integrity, Recovery"."Security in distributed, database systems have focused on multilevel security".

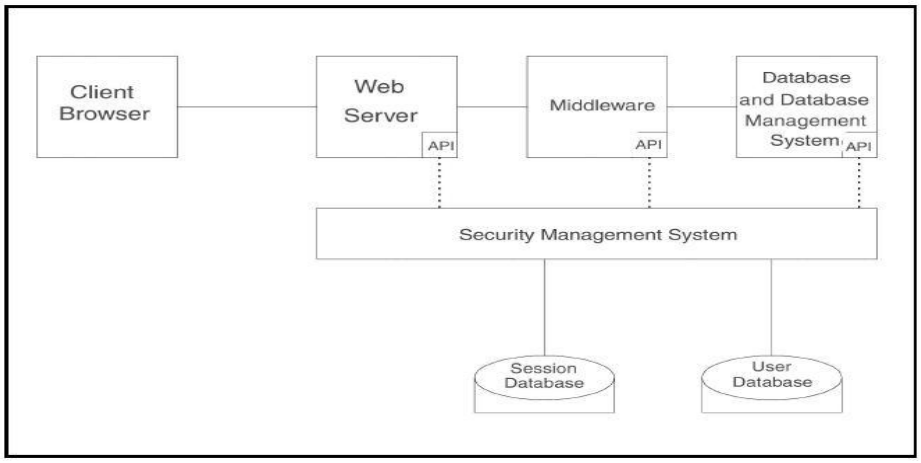


Figure 2: A Simple Security Mode

8. EMERGING SECURITY USED IN DISTRIBUTED SYSTEM TOOLS

Security impact in most of the distributed database tools became an emerging technology that has evolved in some way from distributed databases and discussions. These **include data warehouses and data mining systems, collaborative computing systems, distributed object systems and the web**. First, let us consider data warehousing systems. The major issues here are ensuring that security is maintained in building a data warehouse from the backend database systems and also enforcing appropriate access control techniques when retrieving the data from the warehouse. For example, security policies of the different data sources that form the warehouse have to be integrated to form a policy for the warehouse. This is not a straightforward task, as one has to maintain security rules during the transformations. For example, one cannot give access to an entity in the warehouse, while the same person cannot have access to that entity in the data source. Next, the Warehouse security policy has to be enforced. In addition, the warehouse has to be audited.

The **retrieval problem also becomes an issue** here. For example, the warehouse may store average salaries. A user can access average salaries and then deduce the individual salaries in the data sources, which may be sensitive and therefore, the inference problem could become an issue for the warehouse. To date, little work has been reported on security for data warehouses as Ill as the retrieval problem for the warehouse. This is an area that needs much research intention.

Data mining causes serious security problems. For example, consider a user who has the ability to apply data mining tools. This user can pose various queries and infer a sensitive hypothesis. That is, the retrieval problem occurs via data mining. There are various ways to handle this problem. Given a database and a particular data-mining tool, one can apply the tool to see if sensitive information can be deduced from legitimately obtained unclassified information. If so, then there is a retrieve problem. There are some issues with this approach. One is that I are applying only one tool. In reality, the user may have several tools available to him or to her. Furthermore, it is impossible to cover all of the ways that the retrieval problem could occur. Another solution to the retrieval problem is to build a retrieval controller that can detect the motives of the user and prevent the retrieval problem from occurring. Such a retrieval controller lies between the data-

mining tool and the data source or database, possibly managed by a DBMS.

Data mining systems are being extended to function in a distributed environment. These systems are called distributed data mining systems. Security problems may be exacerbated in distributed data mining systems. This area has received very little attention. Other emerging technologies that have evolved in some way from distributed databases are collaborative computing systems, distributed object management systems and the Ib. Much of the work on securing distributed databases can be applied to securing collaborative computing systems. With respect to distributed object systems security, there is a lot of work by the Object Management Group's Security Special Interest Group.

More recently, there has been much work on securing the web as well. The main issue here is ensuring that the databases, the operating systems, the applications, the Ib servers, the clients and the network are not only secure, but are also securely integrated.

9. CONCLUSION

In this paper I introduced distributed database and all common aspects related to **database system concepts, features and design issues of distributed database and distributed query processing** as well.

I also indicate some related **security and design issues** including **multilevel security** in distributed database systems. The security aspects lead us to investigate the distributed data and centralized control, distributed data and distributed control and some retrieval problems in distributed databases in since of accessing control and integrity. Moreover, I described the most **common mechanisms of discretionary security** and stated the emerging security used in distributed system tools.

Finally I believe that as more and more distributed database tools, the impact of secure distributed database systems on these tools will be a significant requirement.

REFERENCES

[1] Bell, David and Jane Grisom, Distributed Database Systems. Workinham, England: Addison Isley, 1992.

[2] Charles P. Pfleeger and Shari Lawrence Pfleeger, Security in Computing, Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2003.

[3] Haigh, J. T. et al., "The LDV Secure Relational DBMS Model," In Database Security, IV: Status andProspects, S. Jajodia and C.E. LandIhr eds., pp. 265-269, North Holland: Elsevier, 1991.

[4] İlker Köse, GYTE, Veri ve Ağ Güvenliği, Distributed Database Security, Spring 2002.

[5] James F. Kurose and Keith W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, Pearson Education, Inc, New York, 2003.

[6] Paul Lothian and Peter Inham, Database Security in Ib Environment, 2001.

[7] Pfleeger, Charles P., (1989) Security in Computing. New Jersey: Prentice Hall. 1989.

[8] Simon Wiseman, DERA, Database Security: Retrospective and Way Forward, 2001.

[9] Stefano Ceri, Giuseppe Pelagatti: Distributed Databases: Principles and Systems. McGraw-Hill Book Company 1984, ISBN 0-07-010829-3.

[10] Thuraisingham B., Security for Distributed Database Systems, Computers & Security, 2000.

[11] Thuraisingham, Bhavani and William Ford, "Security Constraint Processing In A Multilevel Secure Distributed Database Management System," IEEE Transactions on Knowledge and Data Engineering, v7 n2, pp. 274-293, April 1995.

[12] "Components of a Distributed Database System," http://www.fi/~hhyotyni/latex/Final/node44.html, October 24, 2008.

[13] "Object Oriented Databases," http://www.comptechdoc.org/independent/database/basi cdb/dataobject.html,October 25, 2008. "Network Databases," http://wwwdb.Ib.cern.ch/wwwdb/aboutdbs/classificati on/network.html, October 25, 2008.

[14] www.Wikipedia.com

[15] www.wikianswers.com

