Overview of Database Architecture and Security Measures – Attacks and Control Methods

Otusile Oluwabukola [1], S. A. Idowu[2] and Ajayi Adebowale [3]

[1] Computer Science Department, School of Computing and Engineering Sciences
Babcock University, Ilishan Remo, Ogun State, Nigeria.
Email: buhkieotusile {at} yahoo.com

[2] Computer Science Department, School of Computing and Engineering Sciences
Babcock University, Ilishan Remo, Ogun State, Nigeria.
Email: saidowu {at} yahoo.com

[3] Computer Science Department, School of Computing and Engineering Sciences
Babcock University, Ilishan Remo, Ogun State, Nigeria.
Email: deboxyl {at} gmail.com

ABSTRACT— ***In this paper, we describe the different database architecture and security measures. The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, where one server machine executes operations on the database. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines. Ensuring the security of databases is also a complex issue when discussing database architecture. The purpose of this paper is to highlight and identify the architectures of database systems running on server systems, which are used in centralized and client–server architectures and the security measures pertaining to database systems.***

Keywords— Database, Database-Management System (DBMS), Database Architecture, security.

1. INTRODUCTION

A database-management system (DBMS) is a collection of interrelated data and a set of programs, the collection of data, usually referred to as the database contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient. A primary goal of a database system is to retrieve information from and store new information into the database.

Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access [11]. If data are to be shared among several users, the system must avoid possible anomalous results. Because information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data. These concepts and techniques form the focus of this paper. This article briefly introduces the principles of database systems.

The database architecture is the set of specifications, rules, and processes that dictate how data is stored in a database and how data is accessed by components of a system. It includes data types, relationships, and naming conventions. The database architecture describes the organization of all database objects and how they work together. It affects integrity, reliability, scalability, and performance. The database architecture involves anything that defines the nature of the data, the structure of the data, or how the data flows [10].

The architecture of a database system is greatly influenced by the underlying computer system on which the database system runs. Database systems can be centralized, or client-server, where one server machine executes work on behalf of multiple client machines. Database systems can also be designed to exploit parallel computer architectures. Distributed databases span multiple geographically separated machines [1].

Most users of a database system today are not present at the site of the database system, but connect to it through a network. We can therefore differentiate between client machines, on which remote database users' work, and server machines, on which the database system runs and discuss the security means involved in securing the database system.

1.1 Centralized

Centralized database systems run entirely on a single computer. A modern, general-purpose computer system consists of one to a few processors and a number of device controllers that are connected through a common bus that provides access to shared memory [7]. The processors have local cache memories that store local copies of parts of the memory, to speed up access to data. Each processor may have several independent cores, each of which can execute a separate instruction stream. Each device controller is in charge of a specific type of device. The processors and the device controllers can execute concurrently, competing for memory access. Cache memory reduces the contention for memory access, since it reduces the number of times that the processor needs to access the shared memory. Centralized systems can be categorized into two: as single-user systems and as multiuser systems: designed for use by single users usually does not provide many of the facilities that a multiuser database provides. In particular, they may not support concurrency control, which is not required when only a single user can generate updates. Provisions for crash recovery in such systems are either absent or primitive. In contrast, database systems designed for multiuser systems support the full transactional features that we have studied earlier.

1.2 Client–Server Systems

As personal computers became faster, more powerful, and cheaper, there was a shift away from the centralized system architecture. Personal computers supplanted terminals connected to centralized systems [1]. Correspondingly, personal computers assumed the user-interface functionality that used to be handled directly by the centralized systems. As a result, centralized systems today act as server systems that satisfy requests generated by client systems. Client–server interface protocols have helped the growth of client–server database systems.

Database applications are usually partitioned into two or three parts. In two-tier architecture, the application resides at the client machine, where it invokes database system functionality at the server machine through query language statements. Application program interface standards like Open Database Connectivity (ODBC) for C language and java database connectivity (JDBC) for java language are used for interaction between the client and the server, increasingly; programmers use these APIs to access databases.

In contrast, in three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface.

The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web. Functionality provided by database systems can be broadly divided into two parts—the front end and the back end. The back end manages access structures, query evaluation and optimization, concurrency control, and recovery. The front end of a database system consists of tools such as the SQL user interface, forms interfaces, report generation tools, and data mining and analysis tools. The interface between the front end and the back end is through SQL, or through an application program.

1.3 Server System Architectures

Server systems can be broadly categorized as transaction servers and data servers.

• Transaction-server systems also called query-server systems provide an interface to which clients can send requests to perform an action, in response to which they execute the action and send back results to the client. Usually, client machines ship transactions to the server systems, where those transactions are executed, and results are shipped back to clients that are in charge of displaying the data. Requests may be specified by using SQL, or through a specialized application program interface.

• Data-server systems allow clients to interact with the servers by making requests to read or update data, in units such as files or pages. For example, file servers provide a file-system interface where clients can create, update, read, and delete files. Data servers for database systems offer much more functionality; they support units of data—such as pages, tuples, or objects —that are smaller than a file. They provide indexing facilities for data, and provide transaction facilities so that the data are never left in an inconsistent state if a client machine or process fails.

1.4 Parallel Systems

Parallel processing within a computer system allows database-system activities to be speeded up, allowing faster response to transactions, as well as more transactions per second. Queries can be processed in a way that exploits the parallelism offered by the underlying computer system. The need for parallel query processing has led to parallel database systems. In parallel processing, many operations are performed simultaneously, as opposed to serial processing, in which the computational steps are performed sequentially.

Parallel systems improve processing and I/O speeds by using multiple processors and disks in parallel. Parallel machines are becoming increasingly common, making the study of parallel database systems correspondingly more important. The driving force behind parallel database systems is the demands of applications that have to query extremely large databases (of the order of terabytes—that is, 1012 bytes) or that have to process an extremely large number of transactions per second (of the order of thousands of transactions per second). Centralized and client –server database systems are not powerful enough to handle such applications.

Parallelism is emerging as a critical issue in the future designs of database systems. Whereas today those computer systems with multicore processors have only a few cores, future processors will have large numbers of cores; the reasons for this pertain to issues in computer architecture related to heat generation and power consumption.

Rather than make processors significantly faster, computer architects are using advances in chip design to put more cores on a single chip, a trend likely to continue for some time. As a result, parallel database systems, which once were specialized systems running on specially designed hardware will become the norm.

There are two main measures of performance of a database system: (1) throughput, the number of tasks that can be completed in a given time interval, and (2) response time, the amount of time it takes to complete a single task from the time it is submitted. A system that processes a large number of small transactions can improve throughput by processing many transactions in parallel. A system that processes large transactions can improve response time as well as throughput by performing subtasks of each transaction in parallel.

Two important issues in studying parallelism are speedup and scaleup. Running a given task in less time by increasing the degree of parallelism is called speedup. Handling larger tasks by increasing the degree of parallelism is called scaleup.

Parallel database architectures include the shared-memory, shared-disk, shared-nothing, and hierarchical architectures [14]. These architectures have different trade-offs of scalability versus communication speed.

1.5 Distributed Systems

A distributed database system is a collection of partially independent database systems that (ideally) share a common schema, and coordinate processing of transactions that access nonlocal data. In a distributed database system, the database is stored on several computers. The computers in a distributed system communicate with one another through various communication media, such as high-speed private networks or the Internet. They do not share main memory or disks. The computers in a distributed system may vary in size and function, ranging from workstations up to mainframe systems. The computers in a distributed system are referred to by a number of different names, such as sites or nodes, depending on the context in which they are mentioned. There are several reasons for building distributed database systems, including sharing of data, autonomy, and availability.

Although recovery from failure is more complex in distributed systems than in centralized systems, the ability of most of the system to continue to operate despite the failure of one site results in increased availability. Availability is crucial for database systems used for real-time applications. Loss of access to data by, for example, an airline may result in the loss of potential ticket buyers to competitors. However, in reality a distributed database has to be constructed by linking together multiple already-existing database systems, each with its own schema and possibly running different database-management software. Such systems are sometimes called multidatabase systems or heterogeneous distributed database systems.

2. DATABASE SECURITY

Database security concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability [2]. Safety measures must be an integral part of any database. More recently, due to the rapidly changing and increased size as well as complexity and expansion of company information systems, AAA type measures began to be used (Authentication, Authorization, Access). It involves various types or categories of controls, such as technical, procedural/administrative and physical. The focus of attacks on the database is motivated by the following factors:

- Databases are the mass of information which the company works with;
- Databases can reveal private data by processing public data.

Security risks to database systems include [1]:

- Unauthorized or unintended activity or misuse by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers i.e theft and fraud;

- Malware infections causing incidents such as unauthorized access, leakage or disclosure of personal or proprietary data, deletion of or damage to the data or programs, interruption or denial of authorized access to the database, attacks on other systems and the unanticipated failure of database services;
- Overloads, performance constraints and capacity issues resulting in the inability of authorized users to use databases as intended;
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, accidental liquid spills, static discharge, electronic breakdowns/equipment failures and obsolescence;
- Design flaws and programming bugs in databases and the associated programs and systems, creating various security vulnerabilities (e.g. unauthorized privilege escalation), data loss/corruption, performance degradation etc.;
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc.

The hazards which make these things happen are sometimes due to deliberate human action. Natural type hazards only have an impact on data integrity and availability and to ensure a minimum security of the databases the following requirements must be satisfied:

• Physical integrity of databases;
• Logical integrity of databases;
• The integrity of each element which composes the database;
• Access control;
• User identification;
• Availability.

The physical and **logical integrity** of databases will require the focus of efforts for protecting the physical integrity of databases, especially the recordings against destruction. The easiest way to do that is represented by regular backups.
The integrity of each element forming the database will assume that the value of each field may be written or changed only by authorized users and only if there are correct values.
The access control is being done taking into consideration the restrictions of the database administrator. DBMS will apply the security policy of the database administrator (DBA).
This must meet the following requirements:
• **Server security**: Server security involves limiting access to data stored on the server.
It is the most important option that has to be taken in consideration and planned carefully.
• **Connections to the database**: Using the ODBC will have to be followed by checking that each connection corresponds to a single user who has access to data.
• **Access control table**: The access control table is the most common form of securing a database. An appropriate use of the table access control involves a close collaboration between the administrator and the base developer.
• **Restriction tables**: Restriction tables will include lists of unsure subjects who could open set off sessions.
Secure IP addresses: Some servers may be configured to receive only queries from hosts that are in a list. Oracle servers allow blocking queries that are not related to the database.
Cancellation of the Server Account: The ability to suspend an account when guessing the password is tried after a predefined number of attempts (usually 3).
Special tools: Special programs such as Real Secure by ISS which will alert in case of intrusion attempts. Oracle has an additional set of authentication methods: Kerberos security; Virtual private databases; Role-based security; Grant-execute security; Authentication servers; Port access security.
User identification will allow at any time to be known who does anything in the system. All the operations performed by users will be stored and will form a history of access. Checking the history of all hits is sometimes hard and requires a considerable workload.
Availability will allow the required data to be available for an authorized user.

2.1 Control methods

The classical methods of ensuring database security, the partition of database, cards, encryption, etc. are able to accomplish their tasks. These are meant to stop the highly sophisticated attacks from external attackers, and especially, from those who may very well have access to the company's internal network. Traditionally databases have been largely secured against hackers through network security measures such as firewalls, and network-based intrusion detection systems. While network security controls remain valuable in this regard, securing the database systems themselves, and the programs/functions and data within them, has arguably become more critical as networks are increasingly opened to wider access, in particular access from the Internet. Furthermore, various security-related activities (manual controls) are normally incorporated into the procedures, guidelines etc. relating to the design, development, configuration, use, management and maintenance of databases. Different technique for evaluating database

security include vulnerability assessment, abstraction, database activity monitoring. native audit, database security applying statistical method and process and procedures.

2.1.1 Vulnerability Assessments and Compliance

Testers attempt to find security vulnerabilities that could be used to defeat or bypass security controls, break into the database, compromise the system etc. Database administrators or information security administrators use automated vulnerability scan to search out for mis-configuration of controls within the layers along with known vulnerabilities within the database software. The results of such scans are then used to improve the security controls of the database and close off the specific vulnerabilities identified, but unfortunately some other vulnerability may typically remain unrecognized. A program of continual monitoring for compliance with database security standards is another important task for mission critical database environments. Two crucial aspects of database security compliance include patch management and the review and management of permissions granted to objects within the database. The permissions granted for SQL language commands on objects are considered in this process. Essentially, vulnerability assessment is a preliminary procedure to determine risk where a compliance program is the process of on-going risk assessment.

2.1.2 Abstraction

Application level authentication and authorization mechanisms should be considered as an effective means of providing abstraction from the database layer. The primary benefit of abstraction is that of a single sign-on capability across multiple databases and database platforms. A Single sign-on system should store the database user's credentials (login id and password), and authenticate to the database on behalf of the user.

2.1.3 Database activity Monitoring (DAM)

This requires the use of agents or native logging to capture activities executed on the database server, which typically include the activities of the database administrator. Agents allow this information to be captured in a fashion that cannot be disabled by the database administrator, who has the ability to disable or modify native audit logs. Analysis is performed to identify known exploits or policy breaches, or baselines can be captured over time to build a normal pattern used for detection of anomalous activity that could be indicative of intrusion. These systems provides a comprehensive Database audit trail in addition to the intrusion detection mechanisms, and some systems can also provide protection by terminating user sessions and/or quarantining users demonstrating suspicious behavior. Some systems are designed to support separation of duties (SOD), which is a typical requirement of auditors. SOD requires that the database administrators, who are typically monitored as part of the DAM, not be able to disable or alter the DAM functionality. This requires the DAM audit trail to be securely stored in a separate system not administered by the database administration group. It is basically used for real-time database monitoring activities

2.1.4 Native Audit

Native database audit capabilities are available for many database platforms. The native audit trails are extracted on a regular basis and transferred to a designated security system where the database administrators do not have access. This ensures a certain level of segregation of duties that may provide evidence the native audit trails were not modified by authenticated administrators. Turning on native impacts the performance of the server. Generally, the native audit trails of databases do not provide sufficient controls to enforce separation of duties; therefore, the network and/or kernel module level host based monitoring capabilities provides a higher degree of confidence for forsenics and preservation of evidence.

2.1.5 Process and Procedures

A database security program should include the regular review of permissions granted to individually owned accounts and accounts used by automated processes. The accounts used by automated processes should have appropriate controls around password storage such as sufficient encryption and access controls to reduce the risk of compromise. In conjunction with a sound database security program, an appropriate disaster recovery program should exist to ensure that service is not interrupted during a security incident or any other incident that results in an outage of the primary database environment.

2.1.6 Database Security applying Statistical Method

Unauthorized changes introduced by internal as well as external users directly in the database without keeping any track are considered as biggest threat – algorithm based on cryptology and other statistical methods should be deployed

to identify such events and report to owners. Such shield DB approach maps large dataset into its small digital fingerprint which, is continuously updated with every change in main database by registered applications. Desired fingerprints are then matched with actual at preset intervals for identifying the changed location/s (rows and columns) in main database, date and time of unauthorized changes, even made through privileged authority.

3. CONCLUSIONS

As stated in the introduction, this database architecture may not be suitable exactly as-is for every situation. It provides an excellent foundation for a tailored solution that meets specific needs. It also points out several important considerations. Database security presents features that must be seriously taken into account. The first option, for a secure database is represented by its optimal protection. Ensuring database security must be done from outside to inside, this involving ensuring security starting from the physical level and ending with the data level (physical, network, host, applications and data) since databases are a favorite target for attackers because of the data these are containing and also because of their volume. Therefore the research would recommend combination of the security measures to ensure a secure database or the implementation of datawarehouse as the ultimate goal

4. REFERENCES

[1] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database Concept, McGraw-Hill, New York, 2006
[2] Burtescu, E. "Problems of Inference in Databases in Education, Research & Business Technologies", in Proceeding of the 9th International Conference on Economic Informatics, INFOREC Printing House, Bucharest, 2009.
[3] Burtescu, E, "Databse security in Knowledge Management. Projects, Systems and Technologies", International Conference Knowledge Management. Projects, Systems and Technologies, INFOREC Printing House, Bucharest, November 9-10, 2006.
[4] Hsiao, S.B. and Stemp, R, "Computer Security, course, CS 460" Monterey, California, 1995.
[5] McCarthy, L, IT Security: Risking the Corporation, Prentice Hall PTR, 2003
[6] Proctor, P.E. and Byrnes, F.C. (2002). The Secured Enterprise, Prentice Hall PTR, 2002
[7] Security Complete. Second Edition, SYBEX Inc, 2002.
[8] Harder, T., Reuter, A., "Concepts for implementing a centralized database management system", In Proceedings of International Computing Symposium on Application Systems Development, March 1983
[9] Reuter, A., 2005. Databases: the integrative force in cyberspace. In: Data Management in a Connected World, LNCS 3551. Springer Verlag, pp. 3e16, 2005.
[10] Benyon D. *Information and Data-Modelling* (2nd. Edition), Mc Graw Hill, 1997.
[11] C.J. *Introduction to Database Systems* (6th Edition), Addison- Wesley Publ. company, 1995.
[12] Elmaseri, K. and Navathe, D. *Fundamentals of Database Systems* (2nd. Edition), Benjamin / Cumming, 1994.
[13] Ullmann & Widom, S. *A First Course in Database* Systems, Prentice Hall, 1997.
[14] Joseph M. Hellerstein, Michael Stonebraker and James Hamilton, Architecture of a Database System Foundations and Trends_ in Databases Vol. 1, No. 2 pp 141–259, 2007.