Distributed database management system

A distributed Database Management System (DB'S) is a database in which storage devices are not all attached to a common processing unit such as the CUP]. 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. Collections of data (e. G. In a database) can be distributed across ultimate physical locations.

A distributed database can reside on network servers on the Internet, on corporate intranets or extranets, or on other company networks. The replication and distribution of databases improves database performance at end- user workmates. It allows a user to access and manipulate data from several databases that are physically distributed to several sites. A DB'S defers from a Database Management System (DB'S) because It Is responsible for providing transparent and simultaneous access to several databases that are located on dissimilar computer systems whereas a DB'S manages a single site database. This

DB'S would allow an end-user too: 1. Create and store a new part anywhere in the network. 2. Access a part without knowing where the part is physically located. 3. Delete a part from one of the databases without having to worry about the duplicated parts In other databases. 4. Update a part description In one database without having to worry about how and when all the duplicated parts will be updated In other databases. 5. Access a part from an alternative computer when, say, the nearest computer is not available. To ensure that

the distributive databases are up to date and current, there are two processes: Replication and Duplication. Replication involves using specialized software that looks for changes in the distributive database. Once the changes have been identified, the replication process makes all the databases look the same. The replication process can be complex and time consuming depending on the size and number of the distributive databases. This process can also require a lot of time and computer resources. ; Duplication on the other hand is not as complicated. It basically identifies one database as a master and then duplicates that database. The duplication process is normally done at a set time after hours.

This is to ensure that each distributed location has the same data. In the duplication process, changes to the master database only are allowed. This is to ensure that local data will not be overwritten. Both of the processes can keep the data current in all distributive locations. A DB'S can be classified Into two types Heterogeneous Distributed database management systems Homogeneous Distributed database management systems In a homogeneous distributed database all sites have identical software and are aware of each other and agree to cooperate in processing user requests.

Each site renders part of its autonomy in terms of right to change schema or software. Homogeneous DB'S appears to user as a single system. The homogeneous system is much easier to design and manage. The following conditions must be satisfied for homogeneous database. The operating system used, at each location must be same or compatible. The data structures used at each location must be same or compatible. The database application (or DB'S) used at each location must be same or compatible.

Heterogeneous Distributed database management systems In a heterogeneous distributed database different sites may use different schema ND software. Difference in schema is a major problem for query processing and transaction processing. Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing. In heterogeneous system, different nodes may have different hardware & software and data structures at various nodes or locations are also incompatible. Different computers and operating systems, database applications or data models may be used at each of the locations.

For example, one location may have the latest relational database management genealogy, while another location may store data using conventional files or old version of database management system. Similarly, one location may have the Windows NT operating system, while another may have UNIX. Heterogeneous systems are usually used when individual sites use their own hardware and software. On heterogeneous system, translations are required to allow communication between different sites (or DB'S). In this system, the users must be able to make requests in a database language at their local sites.

Usually the SQL database language is used for this purpose. If the hardware is different, then the orientation is straightforward, in which computer codes and word-length is changed. The heterogeneous system is often not technically or economically feasible. In this system, a user at one location may be able to read but not update the data at another location. Characteristics of a Distributed database management systems (DB'S) ; All sites are interconnected. ; Fragments can be replicated. Logically related

shared data can be collected. Data at each and every site is controlled by the DB'S. Each Distributed Database Management System takes part in at least one global application. REQUIREMENTS OF A GENERALIZED Distributed database management systems (DB'S) A DB'S has both Functional and Operational requirements: Functional Requirements The DB'S should be able to support heterogeneous underlying DB'S. ; The user should be able to access data transparently without knowing the actual location and internet format. ; The user should be able to allocate single copy, two copies The user should be able to partition and fragment takes in the network. A common global user view of the data should be supported. ; A common global data manipulation language should be supported. The user should be able to issue distributed transactions which require information from several nodes. Operational Requirements ; Many users should be able to simultaneously update the duplicated database. ; The data integrity should be supported at several levels: transaction level (all duplicate updates must be applied before this transaction ends) and hourly/daily. The distributed transaction processing should be optimized to minimize query processing costs (time, communication, 1/0 etc) ; The communication and node failures should not drastically impact the operation of the DB'S. ; The DB'S would use standard communication protocols and should be transportable among different systems. ; The software design of DB'S should maximize use of asynchronous and concurrent processing. Architecture, Design and Technology Problems of a Distributed Database Management System Architecture of DB'S can be developed to satisfy the requirements stated above.

A database user accesses the distributed database through Local applications: applications which do not require data from other sites; and Global applications: applications which do require data from other sites. This reference architecture is based on the work presented by Gigolo and Fond. This architecture identifies the main components of a DB'S and shows how these components interact with the Transaction Management System (TM'S) and the Database Management System (DB'S). This architecture is presented as sub layers of the application layer of the ISO/OSI reference model.

A brief overview is given as follows: User interface Manager (I-JIM): This module translates the queries into a global form if necessary, determines the location of the data referenced in the queries, and passes control to TM'S if the transaction is local only or to the Global Transaction Analyzer (GTAG) if the transactions need access to remotely located data. This module is also responsible for gathering all user results generated during transaction execution and presenting the results to the user. The two major issues of User Interface Manager are: 1.

Global Schema definition: A common global schema is needed to parse the query in global query format. The Global schema shows all the data in the network and shows where the data is located. The global schema problem is straightforward if the DB'S is homogeneous, but it is nontrivial if it's heterogeneous. 2. Global Directory location: the global directory problem is a file allocation problem. It is customary to show the (data, node assigned) pair in the global schema in a global directory. It is crucial to allocate the global directory carefully due to the number of global directory accesses.

Transaction Plan Generator (TIP): This module is responsible for generating an execution graph or plan to optimize the performance of the arriving transaction. This may involve decomposing the transaction into sub transactions which can run as local transactions on various nodes and translation of the global transaction into local transactions. To illustrate the processing of this sub layer, assume that a transaction T arrives at node no and accesses logical data items del and do where del is located on node ml and do is located on node no.

T can be processed by using one of the following plans: ; T is accesses del and issues remote calls to node no. ; T is routed to no where it accesses do and issues remote calls to node ml. ; T is decomposed into TTL and to where TTL accesses del and do accesses to; TTL is sent to ml and to is sent to no. The TIP will determine and generate one of these transaction processing plans based on an optimality criterion. The problem of determining optimal transaction processing plans is called 'Distributed Query Processing. The TIP is dependent on the following: 1.

Methods used to access remote data: The three basic methods for accessing remote data are transparent access, remote procedure call and transaction routing. ; Transaction access: This method provides a location independent transparent access to data located anywhere in the network. Every data access request is sent to the remote node through the remote access interface of the reference architecture. This method is convenient but may cause excessive communication traffic. Remote Procedure Call: This method allows the data at a remote site to be accessed locally by a remote server.

The requester of the remote procedure call called the 'client' is responsible for preparing and sending the remote procedure call. A remote procedure call can minimize the communication traffic ; Transaction Routing: In this method, a transaction is decomposed into sub transactions where each sub transaction accesses data at only one node. Then each sub transaction is routed to the remote site and the results are sent back to the destination node. This approach examines the concurrent operations of transactions. 2.

Methods for performing remote Joins: Since Joins between remote sites can cause a considerable amount of communication traffic, a semi-Join operation has been introduced to send only the necessary data. Simply stated, it consists of the following steps for Joining RI and RE on attribute A: ; Relation RI is projected on attribute A, giving RI' RI' is transmitted to RE and Joined with RE, giving'; Iris's transmitted to RI and Joined with RI, giving the final Join Translation of a Join into semi-Join is an example of query decomposition. . Optimization strategy: Determination of an optimal transaction processing plan is a difficult problem. Examples of existing methods are: ; The " hill-climbing" algorithm A family of algorithm to minimize either response time or total time for simple queries ; The evaluation algorithm Global Transaction Execution Monitor (STEM): This module is receives the plan generated by Transaction Plan Generator and is responsible for the initiation, execution and integrity control (synchronization, reliability) of the transaction plan.

This module closely interacts with remote access interface and transaction management system. The two main issues addressed here are the database concurrency control and reliability and error recovery. 1 . Database

Concurrency Control: Concurrency control coordinates simultaneous access to shared data. The problem of concurrency control in centralized Dobbs is well understood and the Two-phase locking has been accepted as a standard solution. However, concurrency control in distributed systems is an area of considerable activity with no accepted solutions.

This is due to three main factors; first, data may be duplicated in DB'S, Secondly, if some sites fail or if some communication links fail while an update is Ewing executed, the DB'S must make sure that the effects will be reflected on the failing node after recovery. Thirdly, synchronization of transactions on multiple sites is very difficult because each site cannot obtain immediate information on the actions currently being carried out on other sites.

Different algorithms have being proposed to solve these problems but none of them have successfully solved it. The most common algorithms are variants of two-phase locking and time stamped algorithm. 2. Reliability and Error Recovery: In a DB'S, updates are made armament when a transaction commits and updates are rolled out if a transaction aborts. In a DB'S, a transaction may commit at one node and abort at another. For example, an update completes at node ml and fails at no.

A transaction may terminate abnormally due to two reasons; " Suicide" which indicates that a transaction terminates due to an external error like a program check; or " murder" which indicates an external error like system crash. It is the responsibility of the DB'S to remove all changes made by a failing transaction from all nodes so that the transaction can be reinstated. In

order for atomic actions to be recoverable, the following conditions must be met. ; Updated objects are not released until the action is completed. The initial states of all objects modified by the action can be constructed through the use of a log. Remote Access Interface: This module prepares all of the messages that need to be sent to remote nodes in a particular format and also receives the messages from the presentation layer and passes them to the TM'S or STEM sub layers. The protocols used in this sub layer may be the File Transfer Access Method of the 'SO, the Manufacturing Message System of Manufacturing Automation protocol (MAP) or the X. 400 protocol proposed for electronic mail.

Design Once a database has been developed, the users are faced with three major decisions: where to allocate the database, how to define the different views of the database, and how to access and manipulate the database. 1 . Data allocation strategies It was stated above that the data allocation decision significantly impacts the query processing, concurrency control and reliability of a DB'S. A given database D can be allocated too network by using one of the following strategies: ; D is allocated to a central node NC

D is uniquely allocated too node N where it is most frequently accessed ; D is allocated to N and of a duplicate copy of D is kept at the central node NC ; D is allocated to N and duplicate copies of D are kept at an arbitrary set of nodes IN, NO, NO, etc. ; D is allocated to every node (replicated) D is partitioned into Del, DO, DO which are allocated using the first five strategies The benefit of this decision can be estimated in terms of storage cost, communication costs (cost to read, cost to update), and response time and data availability. A tradeoff between data duplication and cost is made.

This shows that: ; The storage cost increases as the number of copies increases ; The read communication cost decreases as the number of copies increases because most data can be found at local nodes thus eliminating need for communication calls. ; The update data will need to be updated ; The availability of data increases with the number of copies in the system. An optimal data allocation can be theoretically determined which minimizes the total cost (storage + communication + local 1/0) subject to some response time and availability constraints. This problem is referred to as the File

Allocation Problem (FAA). 2. Database Definition After a database has been allocated, the views (schema) of the database have to be defined. The following three levels of schema definitions are based on ANSI/XX/SPARS recommendations for data description. ; Local (internal) schema which show how the data is stored on each node. The format of the internal schema is dependent on the DB'S of each node ; A global (conceptual) schema which describes the data throughout a network and shows what data is at what node. In case of heterogeneous DB'S, the global format is different from the internal schema format.

The global schema is usually stored in a global directory. ; A user (external) schema which shows how a user will view and manipulate the data. The problem of schema design is straightforward if all Dobbs are homogeneous but if is of critical importance in a network with heterogeneous databases. 3. Database Access and Manipulation The user queries (DAML) are written by using the external schema. If one homogeneous query language is used

throughout the network, the user of a DB'S should not see any difference between a DB'S and DB'S. However if different

Dams are supported by the DB'S, then the main user-oriented issue is: what DAML should be used and from what node. A simple database design procedure is described below. A) Determine the database requirements by interviewing end- users and normalize them b) Partition relations into: ; Location specific data which is primarily used at one site ; Location independent data which may be accessed from many sites c)Allocate location specific data to nodes where it is needed most frequently and eliminate it from further considerations d) Reduce the number of allocation independent files due to: ;

Security Management e) Allocate the rest of the files using the following tradeoffs: ; Amount of storage ; Read communications (cost, time) Update communication (cost, time) ; Local 1/0 at each node f) Modify the global schema to show data allocations g) Conduct a local database design for each of the databases and define the external user schema. There are mainly two approaches to store a relation R in a Distributed Database system: Replication and Fragmentation/Partitioning Replication: In replication, the system maintains several identical replicas of the same relation r in different sites.