MTD3033:
(Database Management System)

CHAPTER 23: DISTRIBUTED DATBASE SYSTEM

50%
Distributed Database System

80%
Data Fragmentation, Replication, and Allocation Techniques for Distributed Database Design

5%
Overview of Concurrency Control and Recovery in Distributed Databases

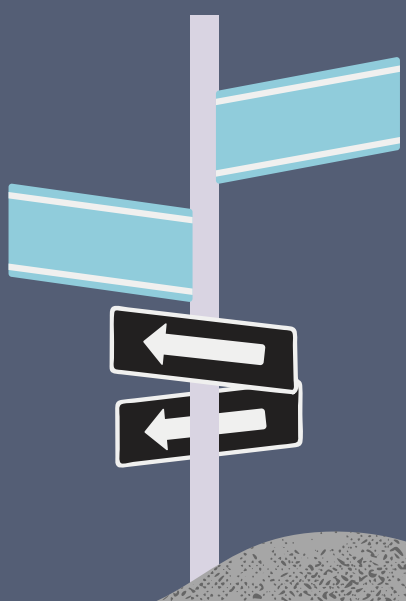
50%
Overview of Transaction Management in Distributed Databases

5%
Query Processing and Optimization in Distributed Databases

50%
Types of Distributed Database Systems

80%
Distributed Database Architectures

80%
Distributed Catalog Management



MOHD IZZUL IKHWAN BIN MOHD YUSOF

D20201095609

DISTRIBUTED DATABASE SYSTEM

Distributed database (DDB) is a collection of multiple logically interrelated database. Distributed database management system (DDBMS) as a software system that manages the DDB and making the distributed transparent to the users.

Database that called distributed must have :

Connection of database nodes over a computer network.

There are multiple computers, called sites or nodes. These sites must be connected by an underlying network to transmit data and commands among sites.

Logical interrelation of the connected database.

It is essential that the information in the various database nodes be logically related.

Possible absence of homogeneity among connected nodes.

It is not necessary that all nodes be identical in terms of data, hardware, and software.

TRANSPARENCY

THE CONCEPT OF TRANSPARENCY EXTENDS THE GENERAL IDEA OF HIDING IMPLEMENTATION DETAILS FROM END USERS. IN THE CASE OF A TRADITIONAL CENTRALIZED DATABASE, TRANSPARENCY SIMPLY PERTAINS TO LOGICAL AND PHYSICAL DATA INDEPENDENCE FOR APPLICATION DEVELOPERS.

This is some type of transparency that possible for DDB is;

DATA ORGANIZATION TRANSPARENCY

Location transparency refers to the fact that the command used to perform a task is independent of the location of the data. **Naming transparency** implies that once a name is associated with an object, the named objects can be accessed unambiguously without additional specification as to where the data is located.

REPLICATION TRANSPARENCY.

Copies of the same data objects may be stored at multiple sites for better availability, performance, and reliability. Replication transparency makes the user unaware of the existence of these copies

FRAGMENTATION TRANSPARENCY.

Horizontal fragmentation distributes a relation (table) into subrelations that are subsets of the tuples (rows) in the original relation; this is also known as sharding in the newer big data and cloud computing systems.

Vertical fragmentation distributes a relation into subrelations where each subrelation is defined by a subset of the columns of the original relation.

AVAILABILITY AND RELIABILITY

Reliability is broadly defined as the probability that a system is running at a certain time point, whereas availability is the probability that the system is continuously available during a time interval. We can directly relate reliability and availability of the database to the faults, errors, and failures associated with it. A failure can be described as a deviation of a system's behavior from that which is specified in order to ensure correct execution of operations. Errors constitute that subset of system states that causes the failure.

SCALABILITY AND PARTITION TOLERANCE

Scalability is the system can expand its capacity while continuing to operate without interruption.

> **Horizontal Scalability.** This refers to expanding the number of nodes in the distributed system. As nodes are added to the system, it should be possible to distribute some of the data and processing loads from existing nodes to the new nodes.

> **Vertical scalability.** This refers to expanding the capacity of the individual nodes in the system, such as expanding the storage capacity or the processing power of a node.

Partition tolerance is the system should have a capacity to continued operating while the network was partitioned.

AUTONOMY

Autonomy can increased flexibility and customized maintenance of an individual nodes. This autonomy can be applied to the design, communication, and execution autonomy.

ADVANTAGES OF THIS DISTRIBUTED DATABASE (DDB)

1. Improved ease and flexibility of application development.
2. Improved availability.
3. Improved performance.
4. Easier expansion via scability.

23.2

DATA FRAGMENTATION, REPLICATION, AND ALLOCATION TECHNIQUES FOR DISTRIBUTED DATABASE DESIGN

DATA FRAGMENTATION

we must decide on a site to store each of the relations EMPLOYEE, DEPARTMENT, PROJECT, WORKS_ON, and DEPENDENT. In many cases, however, a relation can be divided into smaller logical units for distribution. The tuples that belong to the horizontal fragment can be specified by a condition on one or more attributes of the relation, or by some other mechanism.

HORIZONTAL, VERTICLE AND MIXED FRAGMENTATION

HORIZONTAL FRAGMENTATION

Horizontal fragmentation divides a relation horizontally by grouping rows to create subsets of tuples, where each subset has a certain logical meaning. These fragments can be assigned to different sites in the distributed system.

VERTICLE FRAGMENTATIOON

Vertical Fragmentation. Each site may not need all the attributes of a relation, which would indicate the need for a different type of fragmentation. Vertical fragmentation divides a relation «vertically» by columns. A vertical fragment of a relation keeps only certain attributes of the relation.

MIXED FRAGMENTATION

We can intermix the two types of fragmentation, yielding a mixed fragmentation. For example, we may combine the horizontal and vertical fragmentations of the EMPLOYEE relation given earlier into a mixed fragmentation that includes six fragments. In this case, the original relation can be reconstructed by applying UNION and OUTER UNION operations in the appropriate order.

DATA REPLICATION AND ALLOCATION

DATA REPLICATION

Replication is useful in improving the availability of data. The most extreme case is replication of the whole database at every site in the distributed system, thus creating a fully replicated distributed database. This can improve availability remarkably because the system can continue to operate as long as at least one site is up. The disadvantage of full replication is that it can slow down update operations drastically, since a single logical update must be performed on every copy of the database to keep the copies consistent.

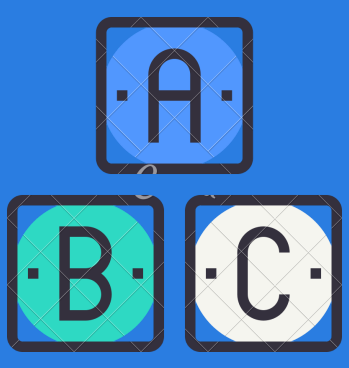
DATA ALLOCATION

This is especially true if many copies of the database exist. Full replication makes the concurrency control and recovery techniques more expensive than they would be if there was no replication. A description of the replication of fragments is sometimes called a replication schema. This process is called data distribution or data allocation .

The choice of sites and the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site. For example, if high availability is required, transactions can be submitted at any site, and most transactions are retrieval only, a fully replicated database is a good choice. However, if certain transactions that access particular parts of the database are mostly submitted at a particular site, the corresponding set of fragments can be allocated at that site only. Data that is accessed at multiple sites can be replicated at those sites.

23.3

Overview of Concurrency Control and Recovery in Distributed Databases

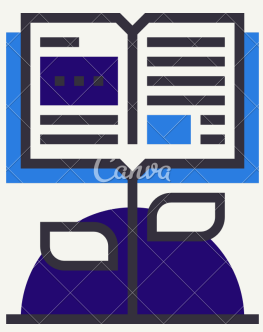


DEALING WITH MULTIPLE COPIES OF THE DATA ITEMS

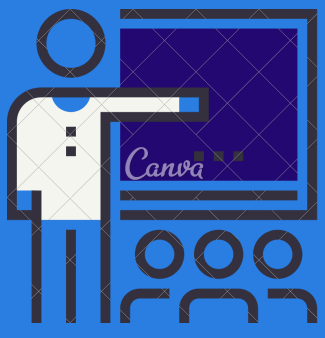
The concurrency control method is responsible for maintaining consistency among these copies. The recovery method is responsible for making a copy consistent with other copies if the site on which the copy is stored fails and recovers later.

FAILURE OF INDIVIDUAL SITES

The DDBMS should continue to operate with its running sites, if possible, when one or more individual sites fail. When a site recovers, its local database must be brought up-to-date with the rest of the sites before it rejoins the system.



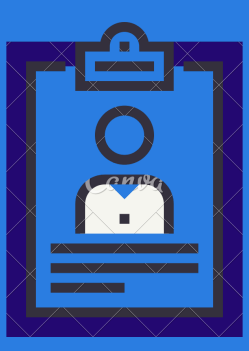
FAILURE OF COMMUNICATION LINKS



The system must be able to deal with the failure of one or more of the communication links that connect the sites. An extreme case of this problem is that network partitioning may occur. This breaks up the sites into two or more partitions, where the sites within each partition can communicate only with one another and not with sites in other partitions.

DISTRIBUTED COMMIT

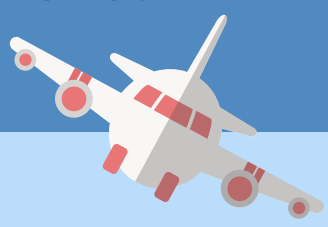
Problems can arise with committing a transaction that is accessing databases stored on multiple sites if some sites fail during the commit process.



DISTRIBUTED DEADLOCK

Deadlock may occur among several sites, so techniques for dealing with deadlocks must be extended to take this into account.

We can distributed concurrency control based primary site technique, primary site with backup, primary copy technique, and choosing a new coordinator site in case of failure.



Primary Site Technique

In this method, a single primary site is designated to be the coordinator site for all database items. Although all locks are accessed at the primary site, the items themselves can be accessed at any site at which they reside. For example, once a transaction obtains a Read_lock on a data item from the primary site, it can access any copy of that data item. However, once a transaction obtains a Write_lock and updates a data item, the DDBMS is responsible for updating all copies of the data item before releasing the lock.

Primary Site with Backup Site

All locking information is maintained at both the primary and the backup sites. This simplifies the process of recovery from failure of the primary site, since the backup site takes over and processing can resume after a new backup site is chosen and the lock status information is copied to that site.

Primary copy technique

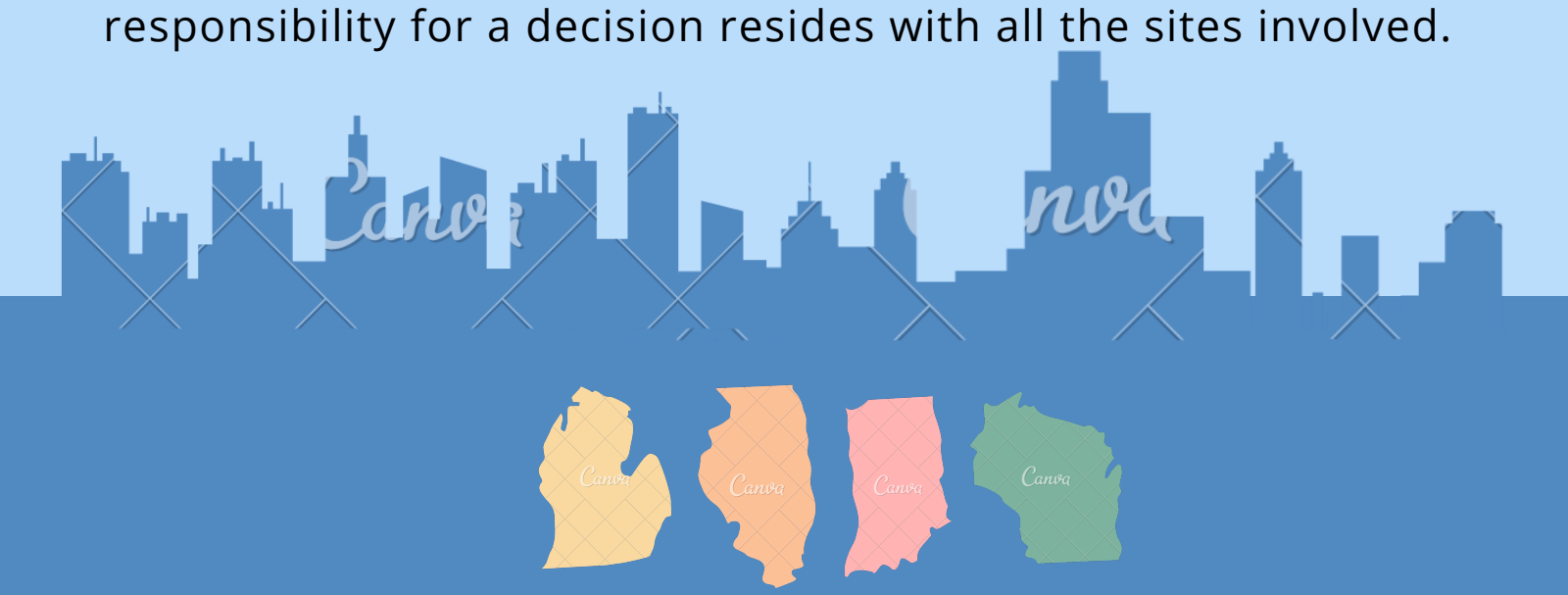
This method attempts to distribute the load of lock coordination among various sites by having the distinguished copies of different data items stored at different sites.

Choosing a new coordinator

In the case of the primary site approach with no backup site, all executing transactions must be aborted and restarted in a tedious recovery process. If a backup site X is about to become the new primary site, X can choose the new backup site from among the system's running sites. However, if no backup site existed, or if both the primary and the backup sites are down, a process called election can be used to choose the new coordinator site. In this process, any site Y that attempts to communicate with the coordinator site repeatedly and fails to do so can assume that the coordinator is down and can start the election process by sending a message to all running sites proposing that Y become the new coordinator. As soon as Y receives a majority of yes votes, Y can declare that it is the new coordinator.

Distributed concurrency control based on voting

If a transaction does not receive a majority of votes granting it will lock within a certain time-out period, it cancels its request and informs all sites of the cancellation. The voting method is considered a truly distributed concurrency control method, since the responsibility for a decision resides with all the sites involved.



23.4

OVERVIEW OF TRANSACTION MANAGEMENT IN DISTRIBUTED DATABASES

An additional component called the global transaction manager is introduced for supporting distributed transactions. The site where the transaction originated can temporarily assume the role of global transaction manager and coordinate the execution of database operations with transaction managers across multiple sites. Transaction managers export their functionality as an interface to the application programs. The manager stores bookkeeping information related to each transaction, such as a unique identifier, originating site, name, and so on. The transaction manager passes to the concurrency controller module the database operations and associated information. If the transaction requires access to a locked resource, it is blocked until the lock is acquired. Once the operation is completed, locks are released and the transaction manager is updated with the result of the operation.

TWO-PHASE COMMIT PROTOCOL

we described the two-phase commit protocol (2PC), which requires a global recovery manager, or coordinator, to maintain information needed for recovery, in addition to the local recovery managers and the information they maintain .

THREE-PHASE COMMIT PROTOCOL

The biggest drawback of 2PC is that it is a blocking protocol. These problems are solved by the three-phase commit protocol, which essentially divides the second commit phase into two subphases called prepare-to-commit and commit. The prepare-to-commit phase is used to communicate the result of the vote phase to all participants. If all participants vote yes, then the coordinator instructs them to move into the prepare-to-commit state.

It can simply ask a crashed participant if it received a prepare-to-commit message. The main idea is to limit the wait time for participants who have prepared to commit and are waiting for a global commit or abort from the coordinator. When a participant receives a precommit message, it knows that the rest of the participants have voted to commit. If a precommit message has not been received, then the participant will abort and release all locks.

23.5

Query Processing and Optimization in Distributed Databases

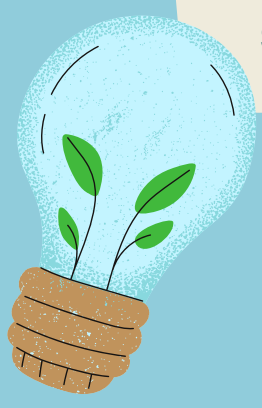
Now i'll give an overview of how a DDBMS processes and optimizes a query. Distributed query processed in this 3 stage;

Query Mapping

It is then translated into an algebraic query on global relations. Hence, this translation is largely identical to the one performed in a centralized DBMS.

Localization

In a distributed database, fragmentation results in relations being stored in separate sites, with some fragments possibly being replicated.



Global query optimization

Since DDBs are connected by a network, often the communication costs over the network are the most significant. This is especially true when the sites are connected through a wide area network .

DATA TRANSFER COSTS OF DISTRIBUTED QUERY PROCESSING

The first is the cost of transferring data over the network. This data includes intermediate files that are transferred to other sites for further processing, as well as the final result files that may have to be transferred to the site where the query result is needed. Although these costs may not be very high if the sites are connected via a high-performance local area network, they become significant in other types of networks. Hence, DDBMS query optimization algorithms consider the goal of reducing the amount of data transfer as an optimization criterion in choosing a distributed query execution strategy.

DISTRIBUTED QUERY PROCESSING USING SEMIJOIN

The idea behind distributed query processing using the semijoin operation is to reduce the number of tuples in a relation before transferring it to another site. Intuitively, the idea is to send the joining column of one relation R to the site where the other relation S is located; this column is then joined with S. Following that, the join attributes, along with the attributes required in the result, are projected out and shipped back to the original site and joined with R. Hence, only the joining column of R is transferred in one direction, and a subset of S with no extraneous tuples or attributes is transferred in the other direction.

23.6

TYPES OF DISTRIBUTED DATABASE SYSTEMS

The main thing that all such systems have in common is the fact that data and software are distributed over multiple sites connected by some form of communication network. If there is no provision for the local site to function as a standalone DBMS, then the system has no local autonomy. On the other hand, if direct access by local transactions to a server is permitted, the system has some degree of local autonomy.

FEDERATED DATABASE MANAGEMENT SYSTEMS ISSUES

The type of heterogeneity present in FDBMS may arise from several sources. We discuss these sources first and then point out how the different types of autonomies contribute to a semantic heterogeneity that must be resolved in a heterogeneous FDBMS.

DIFFERENCES IN DATA MODELS

Even if two databases are both from the Relational Database Management System (RDBMS) environment, the same information may be represented as an attribute name, as a relation name, or as a value in different databases.

DIFFERENCES IN CONSTRAINTS

Triggers may have to be used to implement certain constraints in the relational model. The global schema must also deal with potential conflicts among constraints.

DIFFERENCES IN QUERY LANGUAGES

Even with the same data model, the languages and their versions vary. each system has its own set of data types, comparison operators, string manipulation features, and so on.

SEMANTIC HETEROGENEITY

SEMANTIC HETEROGENEITY OCCURS WHEN THERE ARE DIFFERENCES IN THE MEANING, INTERPRETATION, AND INTENDED USE OF THE SAME OR RELATED DATA. SEMANTIC HETEROGENEITY AMONG COMPONENT DATABASE SYSTEMS (DBS) CREATES THE BIGGEST HURDLE IN DESIGNING GLOBAL SCHEMAS OF HETEROGENEOUS DATABASES. THE DESIGN AUTONOMY OF COMPONENT DBMS REFERS TO THEIR FREEDOM OF CHOOSING THE FOLLOWING DESIGN PARAMETERS, THE DESIGN PARAMETERS IN TURN AFFECT THE EVENTUAL COMPLEXITY OF THE FDBS IS ;

THE UNIVERSE OF DISCOURSE FROM WHICH THE DATA IS DRAWN

For example, for two customer accounts, databases in the federation may be from the United States and Japan and have entirely different sets of attributes about customer accounts required by the accounting practices. Currency rate fluctuations would also present a problem. Hence, relations in these two databases that have identical names—CUSTOMER or ACCOUNT—may have some common and some entirely distinct information.

TRANSACTION AND POLICY CONSTRAINTS

These deal with serializability criteria, compensating transactions, and other transaction policies.

REPRESENTATION AND NAMING

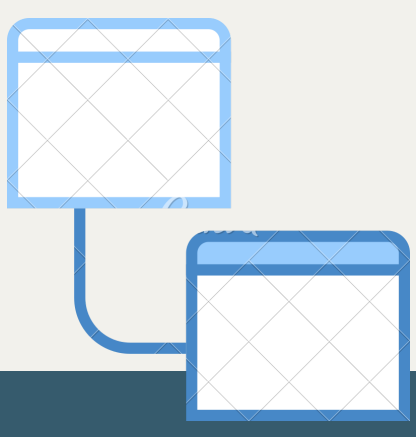
The representation and naming of data elements and the structure of the data model may be prespecified for each local database.

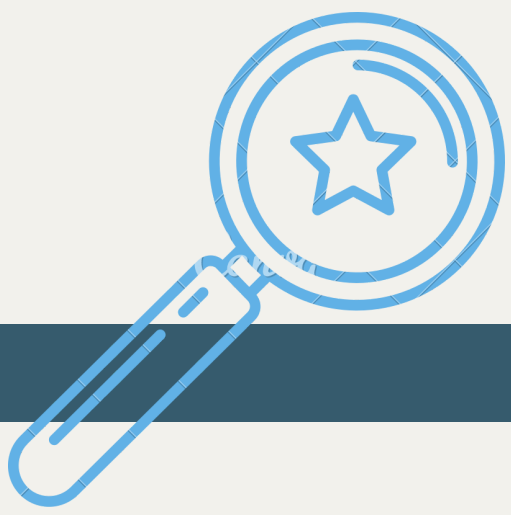
THE UNDERSTANDING, MEANING, AND SUBJECTIVE INTERPRETATION OF DATA

This is a chief contributor to semantic heterogeneity.

DERIVATION OF SUMMARIES

Aggregation, summarization, and other dataprocessing features and operations supported by the system.





23.7

DISTRIBUTED DATABASE ARCHITECTURES

In this section, we first briefly point out the distinction between parallel and distributed database architectures. This is followed by discussions on the architecture of three-tier client/server and federated database systems.

PARALLEL VERSUS DISTRIBUTED DATABASE ARCHITECTURES

There are two main types of multiprocessor system architectures that are commonplace ;

SHARED MEMORY (TIGHTLY COUPLED) ARCHITECTURE

Multiple processors share secondary (disk) storage and also share primary memory.

SHARED DISK (LOOSELY COUPLED) ARCHITECTURE

Multiple processors share secondary (disk) storage but each has their own primary memoy.

These architectures enable processors to communicate without the overhead of exchanging messages over a network. Database management systems developed using the above types of architectures are termed parallel database management systems rather than DDBMS, since they utilize parallel processor technology. Another type of multiprocessor architecture is called shared-nothing architecture In this architecture, every processor has its own primary and secondary (disk) memory, no common memory exists, and the processors communicate over a highspeed interconnection network.

AN OVERVIEW OF THREE-TIER CLIENT/SERVER ARCHITECTURE

10%

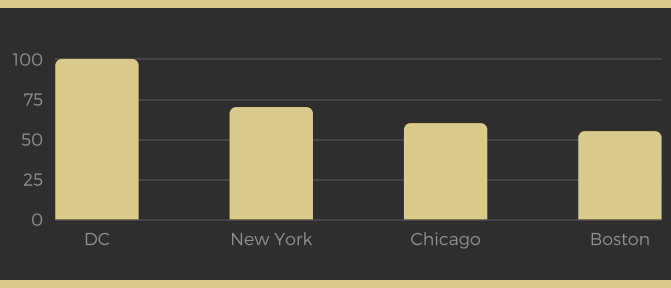
AS WE POINTED OUT IN THE CHAPTER INTRODUCTION, FULL-SCALE DDBMS HAVE NOT BEEN DEVELOPED TO SUPPORT ALL THE TYPES OF FUNCTIONALITIES. INSTEAD, DISTRIBUTED DATABASE APPLICATIONS ARE BEING DEVELOPED IN THE CONTEXT OF THE CLIENT/SERVER ARCHITECTURES. THIS IS THE THREE-TIER CLIENT/SERVER ARCHITECTURE ;

PRESENTATION LAYER (CLIENT)

THIS PROVIDES THE USER INTERFACE AND INTERACTS WITH THE USER. WEB BROWSERS ARE OFTEN UTILIZED, AND THE LANGUAGES AND SPECIFICATIONS USED INCLUDE HTML, XHTML, CSS, FLASH, MATHML, SCALABLE VECTOR GRAPHICS , JAVA, JAVASCRIPT, ADOBE FLEX, AND OTHERS. THE LATTER ARE EMPLOYED WHEN THE INTERACTION INVOLVES DATABASE ACCESS.

APPLICATION LAYER (BUSINESS LOGIC)

THIS LAYER PROGRAMS THE APPLICATION LOGIC. ADDITIONAL APPLICATION FUNCTIONALITY CAN BE HANDLED AT THIS LAYER, SUCH AS SECURITY CHECKS, IDENTITY VERIFICATION, AND OTHER FUNCTIONS. THE APPLICATION LAYER CAN INTERACT WITH ONE OR MORE DATABASES OR DATA SOURCES USING ODBC, JDBC, SQL/CLI, OR OTHER DATABASE ACCESS TECHNIQUES.



DATABASE SERVER

THIS LAYER HANDLES QUERY AND UPDATE REQUESTS FROM THE APPLICATION LAYER, PROCESSES THE REQUESTS, AND SENDS THE RESULTS. USUALLY SQL IS USED TO ACCESS THE DATABASE IF IT IS RELATIONAL OR OBJECT-RELATIONAL, AND STORED DATABASE PROCEDURES MAY ALSO BE INVOKED.

23.8

DISTRIBUTED CATALOG MANAGEMENT

Catalogs are databases themselves containing metadata about the distributed database system. Three popular management schemes for distributed catalogs are centralized catalogs, fully replicated catalogs, and partitioned catalogs. The choice of the scheme depends on the database itself as well as the access patterns of the applications to the underlying data.

CENTRALIZED CATALOGS

Due to its central nature, it is easy to implement. On completion of the read operation, an acknowledgment is sent to the central site, which in turn unlocks this data. All update operations must be processed through the central site.

FULLY REPLICATED CATALOGS

In this scheme, identical copies of the complete catalog are present at each site. However, all updates must be broadcast to all sites. As with the centralized scheme, write-intensive applications may cause increased network traffic due to the broadcast associated with the writes.

PARTIALLY REPLICATED CATALOGS

The system tracks catalog entries for sites where the object was created and for sites that contain copies of this object. In general, fragments of relations across sites should be uniquely accessible.

THANK YOU!

SPECIAL THANKS TO
TS. DR. CHEE KEN NEE

MOHD IZZUL IKHWAN