Advanced Management of Data

Concepts of Distributed Databases (3)

Query Processing

Last lecture we discussed processing of a distributed database query:

1. **Query Mapping** The input query on distributed data is specified formally using a query language. It is then translated into an algebraic query on global relations using the global conceptual schema.

2. **Localization** The distributed query is mapped on the global schema to separate queries on individual fragments using data distribution and replication information.

3. **Global Query Optimization** Selecting a strategy from a list of candidates that is closest to optimal. A list of candidate queries can be obtained by permuting the ordering of operations within a fragment query generated by the previous stage. The total cost is a weighted combination of CPU cost, I/O costs, and communication costs.

4. **Local Query Optimization** The techniques are similar to those used in centralized systems.

Query Processing and Optimization

Communication time (CT)

The time taken to send a message depends upon the length of the message and the type of network being used. It can be calculated using the following formula:

CT = AD + (MS / TR)

AD:	access delay = fixed costs of initiating a message
MS:	message size = number of bits in message
TR:	transmission rate

Examples

Using AD = 1 second, TR = 10 000 bits per second, MS = 100 bits, we need to transfer 100 000 records

- as a whole: CT = 1 + (100 000 * 100 / 10 000) = 1001 seconds
- one at a time: CT = 100 000 * (1 + (100 / 10 000)) = 101 000 seconds

Example (Task)

Consider the following query over three relations A, B, and R with tuple size MS=100 bits, Transmission Rate TR=10 000 bits/s, and access delay AD=1s:

Relation	location	tuple number
A(<u>a</u>, v)	site 1	10 000
B(<u>b</u>, v)	site 2	100 000
R(<u>a</u>, <u>b</u>)	site 1	1 000 000

```
select A.a
from   A join (B join R on B.b = R.b) on A.a = R.a
where  A.v = 1 and B.v = 2;
```

100 000 tuples in (A join R) fulfill the condition v = 1, 10 tuples in B fulfill the condition v = 2

Some Strategies

- Strategy 1: Move B to site 1 and process query there
- Strategy 2: Move A and R to site 2 and process query there
- Strategy 3: Select B tuples with v=2 at site 2, move result to site 1 for matching with A.v=1
- Strategy 4: Join A and R at site 1, select tuples for v = 1 and then for each of these tuples in turn check at site 2 to determine whether the associated B.v = 2 holds.

Multidatabase Systems

Multidatabase System (MDBS)

- distributed DBMS in which each site maintains complete autonomy
- distribution is realized by an additional software layer on top of the local systems
- users can access and share data without requiring full database schema integration
- users can manage their own / local databases without centralized control

Export Schema

The administrator of a local DBMS can authorize access to particular parts of a database by specifying a distinct schema.

This „exported“ schema defines the parts of the database that may be accessed by nonlocal users.

Multidatabase Systems

Unfederated MDBS

no local users

Federated MDBS (FDBS)

Applications share a global view (schema) of the federation of databases.

A federated database (FDB) system is a hybrid of a distributed DBMS and a centralized DBMS:

- a distributed system for global users
- a centralized system for local users

Federated DB Systems

Sources of Heterogeneity

• data models	Databases in an organization may come from a variety of data models, e.g. legacy models (hierarchical, network), relational, object data models, and even files ➔ how to process them in a single query language?
• constraints	Constraint facilities may vary from system to system.
• query languages	Even with the same data model, the versions of query languages and their capabilities may vary, which can result in conflicts regarding data, naming, domains, precision, schema, … .
• semantics	Differences in the meaning, interpretation, and intended use of the data, e.g. homonyms and synonyms

Federated DB Systems

The complexity of the FDBS will be directly influenced by the degree of autonomy of component DBSs.

Design autonomy

- universe of discourse from which the data is drawn
- representation and naming
- understanding, meaning, and subjective interpretation of data
- transaction and policy constraints
- derivation of summaries

Federated DB Systems

The following types of autonomy should be provided to component DBS:

Communication autonomy

- ability to decide whether to communicate with another component DBS

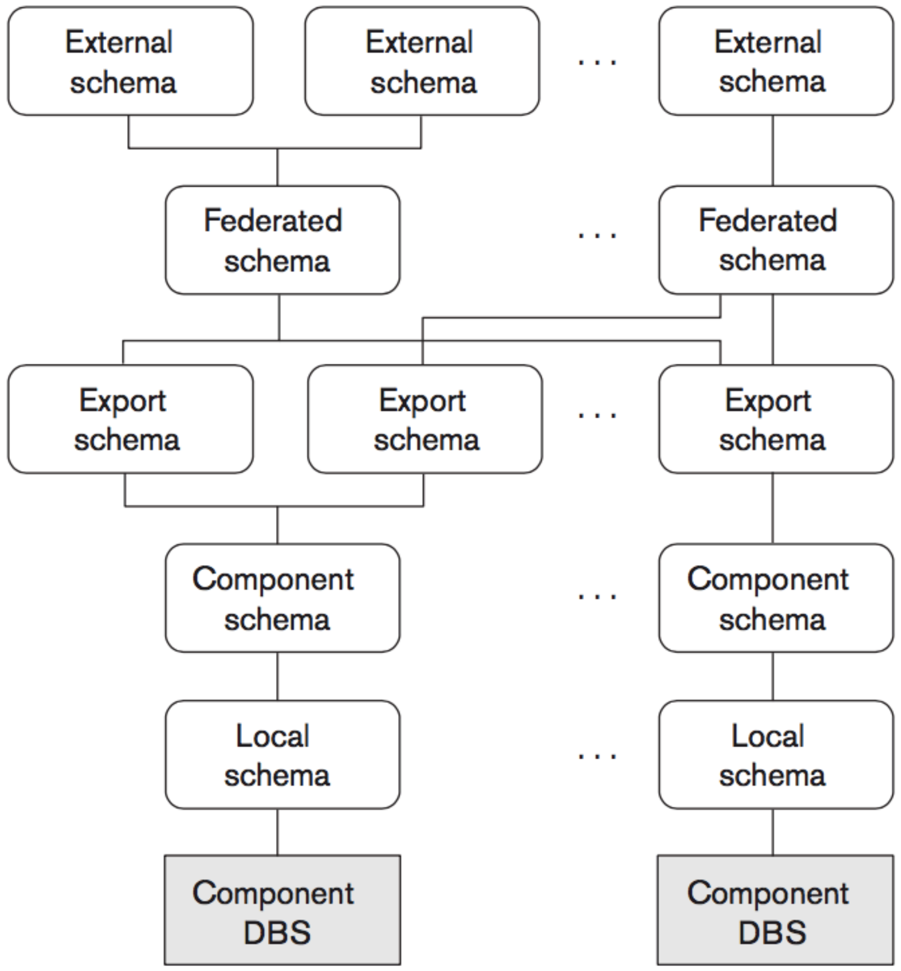
Execution autonomy

- ability to perform local operations without interference from external operations by other component DBSs
- ability to decide the execution order of local operations

Association autonomy

- ability to decide whether and how much to share its functionality and data with other component DBSs

FDB Schema Architecture (five-level schema architecture)



schema for a user group or an application, as in the three-level schema architecture

global schema or view, which is the result of integrating all shareable export schemas

subset of a component schema that is available to the FDBS

result of translating the local schema into a common data model. Mappings transform commands into commands on the corresponding local schema

conceptual schema of a component database

[Elmasri & Navathe]

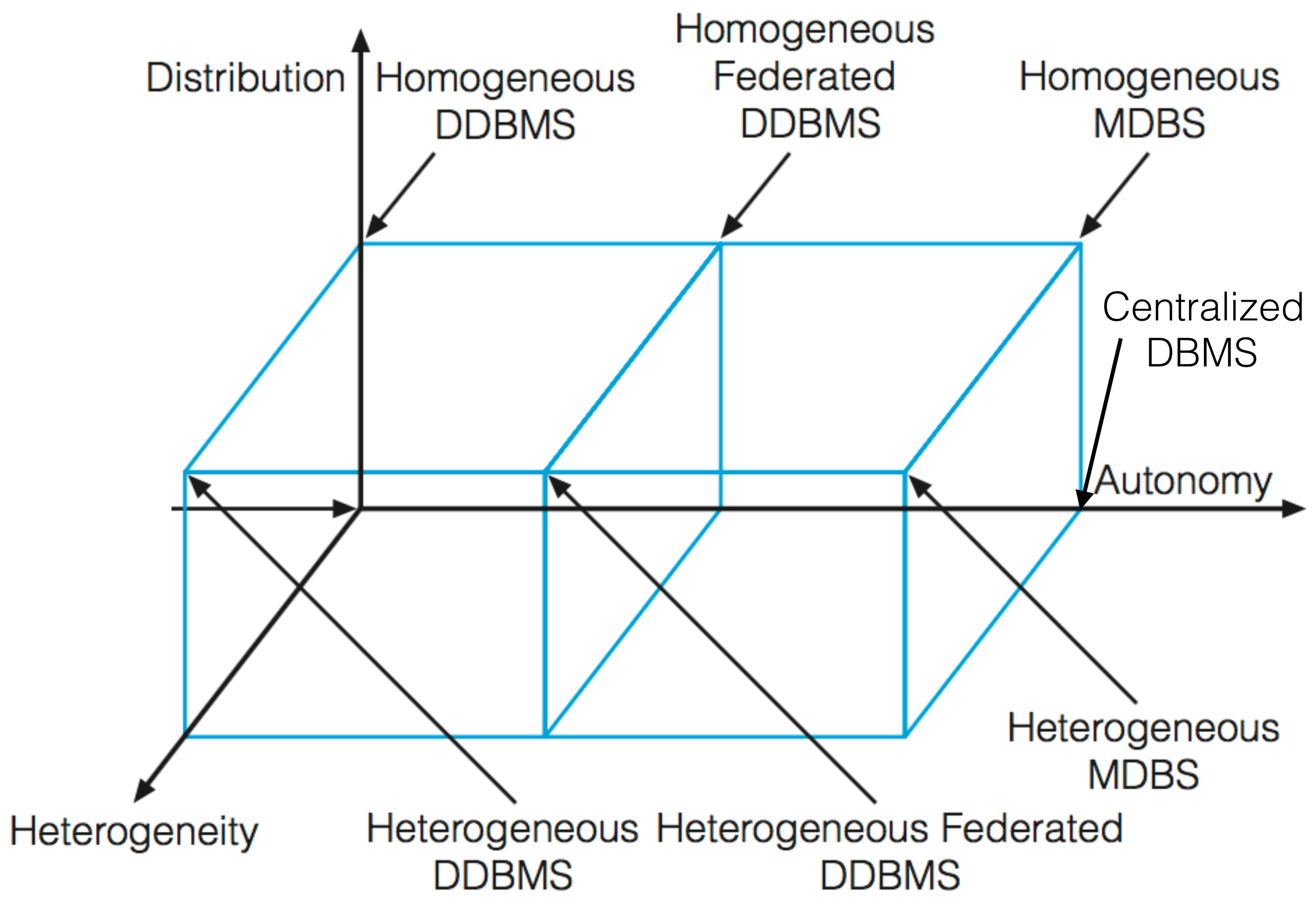
Types of Distributed DB Systems

We already mentioned that the term distributed database management system can describe various systems that differ from one another in many respects.

To classify DDBS we consider

• degree of homogeneity	If all DBMS servers and clients use identical software, the DDBMS is called homogeneous, otherwise, it is called heterogeneous.
• degree of local autonomy	If there is no provision for the local site to function as a standalone DBMS, then the system has no local autonomy. If direct access by local transactions to a server is permitted, the system has some degree of local autonomy.
• degree of distribution	

Types of Distributed DB Systems



[Connolly & Begg]

Distributed Catalog Management

Option 1 - Centralized Catalogs

The entire catalog is stored in one site. For read operations from non-central sites, the requested catalog data is locked at the central site and is then sent to the requesting site. On completion of the read operation, an acknowledgment is sent to the central site, and the locked data is unlocked.

Advantage:	easy to implement
Disadvantages:	Since all update operations must be processed through only one site, performance for write-intensive applications becomes negatively impacted.
	Also, reliability, availability, autonomy, and distribution of processing load will be impacted adversely.

Distributed Catalog Management

Option 2 - Fully Replicated Catalogs

Identical copies of the complete catalog are available at each site.

Advantage:	Read operations are very fast since they can be answered locally.
Disadvantages:	All updates must be broadcast to all sites.
	Catalog consistency must be ensured.
	Write-intensive applications cause increased network traffic due to the broadcast associated with the writes.

Distributed Catalog Management

Option 3 - Partially Replicated Catalogs

Each site maintains complete catalog information on data stored locally at that site.

Each site is also permitted to cache entries retrieved from remote sites. There are no guarantees that these cached copies contain the most recent data.

The system tracks catalog entries for sites where the object was created and for sites that contain copies of this object.

Any changes of copies are propagated immediately to the original site.

Retrieving updated copies to replace data that is not up to date may be delayed until an access to this data occurs.

Concurrency Control

Potential problems

- dealing with multiple copies of data items
- failure of individual sites
- failure of communication links
- distributed commit
- distributed deadlock

Approach

We extend centralized locking mechanisms to deal with distribution.

Concurrency Control

Primary Site Technique

A single primary site is designated to be the coordinator site for all database items.

➔ All requests for locking or unlocking are sent at the primary site.

Advantage:	simple extension of the centralized lock mechanism
Disadvantages:	potential system bottleneck
	system reliability and availability is limited
	a failure of the primary site stops the entire system

Concurrency Control

Primary Site with Backup Site

The primary site technique is extended by designating a second site to be a backup site.

➔ All locking information is maintained at both the primary and the backup site.

Advantage:	The risk of paralyzing the whole system is alleviated, since the backup site takes over in case of a failure of the primary site.
Disadvantages:	The process of acquiring locks is slowed down, because all lock requests and granting of locks must be recorded at both the primary and the backup sites.
	The problem of the primary and backup sites becoming overloaded with requests and slowing down the system remains undiminished.

Concurrency Control

Primary Copy Technique

- a particular copy of each data item is designated as a distinguished copy
- the distinguished copies of different data items are stored at different sites to distribute the load of lock coordination among various sites
- a failure of one site affects any transactions that are accessing locks on items whose primary copies reside at that site, but other transactions are not affected
- reliability and availability can be further enhanced by using backup sites

Further Techniques

There are other approaches available (e.g. election, voting), which may show increased network traffic and can become very complex.

Also, a distributed recovery process is quite involved.

Advantages of DDB

- better representation of organizational structures
- improved shareability and local autonomy
- increased availability and reliability (due to replication)
- improved performance
- economics - it may cost less to create a network of smaller computers with the power of a single large computer
- modular expansion via scalability
- integration (of existing systems)

Disadvantages of DDB

- increased complexity
- increased cost
- security (e.g. access to replicated data and networks)
- more difficult integrity control
- lack of standards
- lack of experience
- more complex database design