Distributed Database Systems and Data Warehouses

Dr. Volodymyr Sokol
(vlad.sokol@gmail.com)

© Volodymyr Sokol

Further Reading

- Connolly, Begg - Database Systems: A Practical Approach to Design, Implementation, and Management (Базы данных. Проектирование, реализация и сопровождение. Теория и практика)
- Elmasri, Navathe - Fundamentals of Database Systems
- Inmon W. H.: Building the Data Warehouse, Wiley &Sons, 2002.
- Kimball R., Ross M.: The Data Warehouse Toolkit. The Complete Guide to Dimensional Modeling, Wiley &Sons, 2002.
- Ponniah P.: Data Warehousing Fundamental, Wiley &Sons, 2001.

© Volodymyr Sokol

LECTION 1

© Volodymyr Sokol

Definitions

- **Distributed database** is a logically interrelated collection of shared data (and a description of this data) physically distributed over a computer network
- **Distributed DBMS** is the software system that permits the management of the distributed database and makes the distribution transparent to users

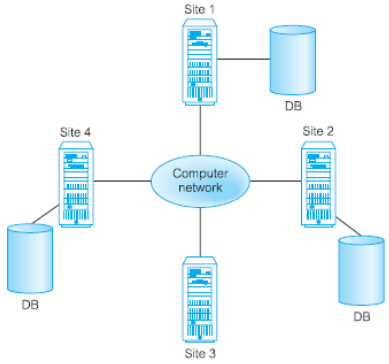
© Volodymyr Sokol

Definitions

- DDBMS consists of a single logical database that is split into a number of **fragments**
- Users access the distributed database via applications, which are classified as those that do not require data from other sites (**local applications**) and those that do require data from other sites (**global applications**). We require a DDBMS to have at least one global application

© Volodymyr Sokol

Characteristics of DDBMS



- collection of logically related shared data
- data is split into a number of fragments
- fragments may be replicated
- fragments/replicas are allocated to sites
- sites are linked by a communications network
- data at each site is under the control of a DBMS
- DBMS at each site can handle local applications, autonomously
- each DBMS participates in at least one global application

© Volodymyr Sokol

Transparency

- From the definition of the DDBMS, the system is expected to make the distribution **transparent** (invisible) to the user. Thus, the fact that a distributed database is split into fragments that can be stored on different computers and perhaps replicated, should be hidden from the user
- The objective of transparency is to make the distributed system appear like a centralized system. This is sometimes referred to as the **fundamental principle** of DDBMSs

© Volodymyr Sokol

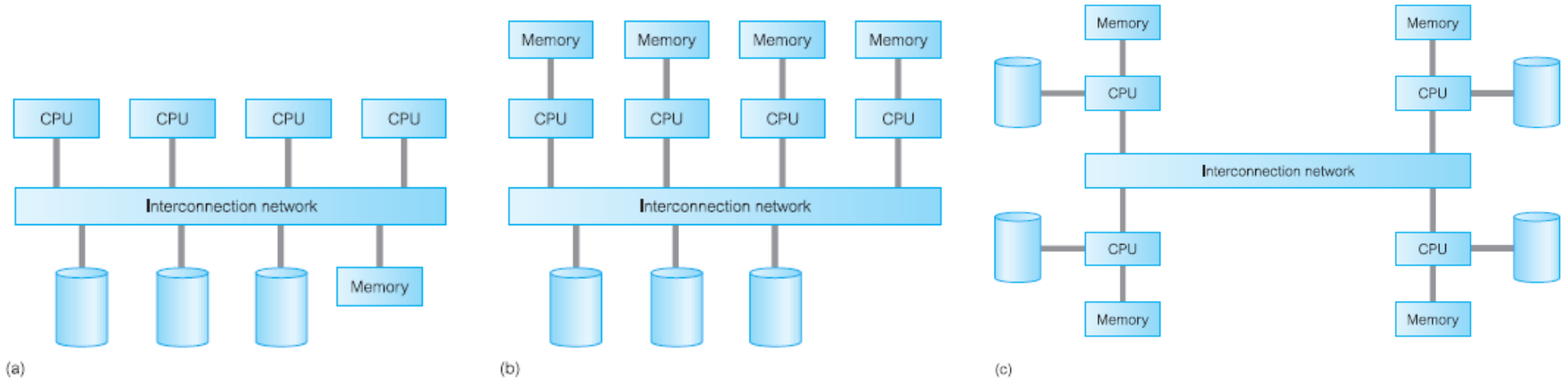
Distributed processing

- **Distributed processing** is a centralized database that can be accessed over a computer network
- The key point with the definition of a distributed DBMS is that the system consists of data that is physically distributed across a number of sites in the network. If the data is centralized, even though other users may be accessing the data over the network, we do not consider this to be a distributed DBMS, simply distributed processing

© Volodymyr Sokol

Parallel DBMSs

- **Parallel DBMS** is a DBMS running across multiple processors and disks that is designed to execute operations in parallel, whenever possible, in order to improve performance



(a) shared memory;
(b) shared disk;
(c) shared nothing.

© Volodymyr Sokol

Advantages of DDBMSs

- Reflects organizational structure
- Improved shareability and local autonomy
- Improved availability
- Improved reliability
- Improved performance
- Economics
- Modular growth
- Integration
- Remaining competitive

© Volodymyr Sokol

Disadvantages of DDBMSs

- Complexity
- Cost
- Security
- Integrity control more difficult
- Lack of standards
- Lack of experience
- Database design more complex

© Volodymyr Sokol

Homogeneous and Heterogeneous DDBMSs

- In a **homogeneous** system, all sites use the same DBMS product
- In a **heterogeneous** system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical, and object-oriented DBMSs

© Volodymyr Sokol

Multidatabase system (MDBS)

- **Multidatabase system (MDBS)** is a distributed DBMS in which each site maintains complete autonomy

© Volodymyr Sokol

Functions of a DDBMS

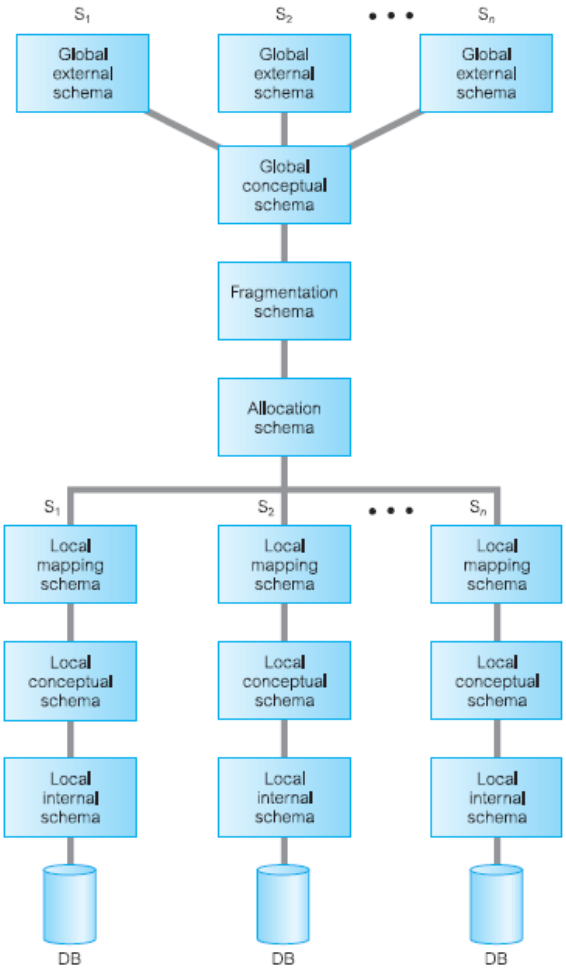
- extended communication services to provide access to remote sites and allow the transfer of queries and data among the sites using a network
- extended system catalog to store data distribution details
- distributed query processing, including query optimization and remote data access
- extended security control to maintain appropriate authorization/access privileges to the distributed data
- extended concurrency control to maintain consistency of distributed and possibly replicated data
- extended recovery services to take account of failures of individual sites and the failures of communication links

© Volodymyr Sokol

Reference Architecture for a DDBMS

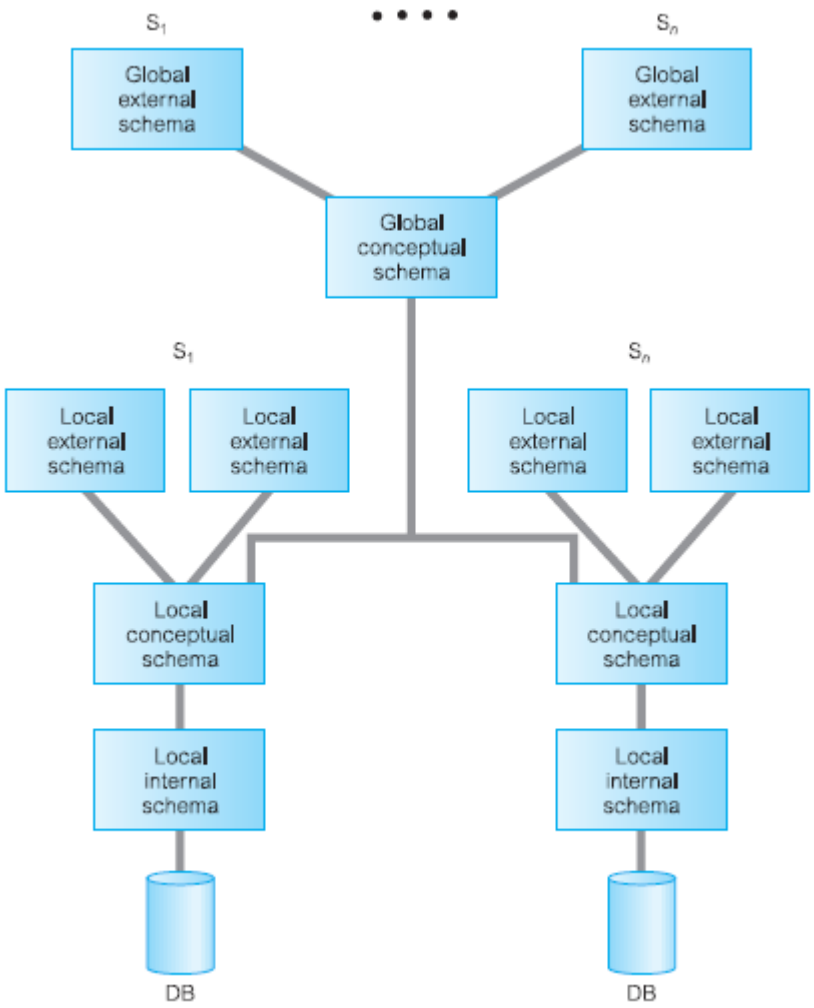
The reference architecture consists of the following schemas:

- set of global external schemas
- global conceptual schema;
- fragmentation schema and allocation schema
- set of schemas for each local DBMS conforming to the ANSI-SPARC three-level architecture



© Volodymyr Sokol

Reference Architecture for a Federated MDBS

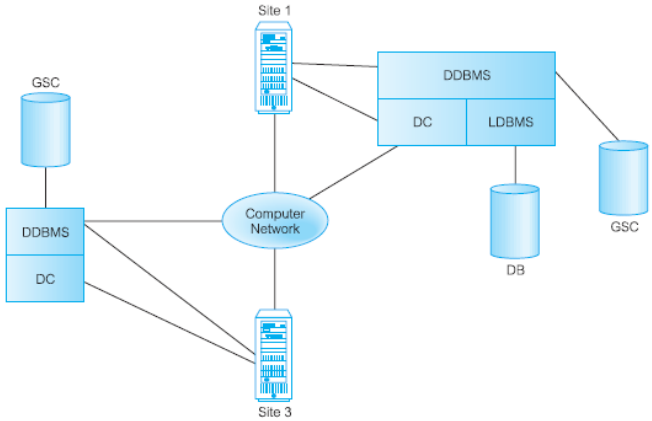


© Volodymyr Sokol

Component Architecture for a DDBMS

We can identify a component architecture for a DDBMS consisting of four major components:

- local DBMS (LDBMS) component
- data communications (DC) component
- global system catalog (GSC)
- distributed DBMS (DDBMS) component



© Volodymyr Sokol

Distributed Relational Database Design

Factors that have to be considered for the design of a distributed relational database:

- **Fragmentation.** A relation may be divided into a number of subrelations, called fragments, which are then distributed. There are two main types of fragmentation: horizontal and vertical. Horizontal fragments are subsets of tuples and vertical fragments are subsets of attributes.
- **Allocation.** Each fragment is stored at the site with 'optimal' distribution.
- **Replication.** The DDBMS may maintain a copy of a fragment at several different sites.

© Volodymyr Sokol

Distributed Relational Database Design

The design should be based on both **quantitative** and **qualitative** information. Quantitative information is used in allocation; qualitative information is used in fragmentation.
The quantitative information may include:

- frequency with which a transaction is run
- site from which a transaction is run
- performance criteria for transactions

The qualitative information may include information about the transactions that are executed, such as:

- relations, attributes, and tuples accessed
- type of access (read or write)
- predicates of read operations

© Volodymyr Sokol

Distributed Relational Database Design

The definition and allocation of fragments are carried out strategically to achieve the following objectives:

- **Locality of reference.** Where possible, data should be stored close to where it is used. If a fragment is used at several sites, it may be advantageous to store copies of the fragment at these sites
- **Improved reliability and availability.** Reliability and availability are improved by replication: there is another copy of the fragment available at another site in the event of one site failing
- **Acceptable performance.** Bad allocation may result in bottlenecks occurring, that is a site may become inundated with requests from other sites, perhaps causing a significant degradation in performance. Alternatively, bad allocation may result in underutilization of resources
- **Balanced storage capacities and costs.** Consideration should be given to the availability and cost of storage at each site so that cheap mass storage can be used, where possible. This must be balanced against locality of reference
- **Minimal communication costs.** Consideration should be given to the cost of remote requests. Retrieval costs are minimized when *locality of reference* is maximized or when each site has its own copy of the data. However, when replicated data is updated, the update has to be performed at all sites holding a duplicate copy, thereby increasing communication costs

© Volodymyr Sokol