Principles of Distributed Database Systems

M. Tamer Özsu
Patrick Valduriez

Outline

- Introduction
- Distributed and Parallel Database Design
- Distributed Data Control
- Distributed Query Processing
- Distributed Transaction Processing
- Data Replication
- Database Integration – Multidatabase Systems
- Parallel Database Systems
- Peer-to-Peer Data Management
- Big Data Processing
- NoSQL, NewSQL and Polystores
- Web Data Management

Outline

- Distributed and Parallel Database Design
  - Fragmentation
  - Data distribution
  - Combined approaches

Distribution Design

The design of a distributed computer system involves making decisions on the placement of data and programs across the sites of a computer network, as well as possibly designing the network itself.

In the case of distributed DBMSs, the distribution of applications involves two things:

1. The distribution of the distributed DBMS software
2. The distribution of the application programs that run on it.

Distribution Design

The organization of distributed systems can be investigated along three orthogonal dimensions

1. Level of sharing
2. Behavior of access patterns
3. Level of knowledge on access pattern behavior

Distribution Design

In terms of the level of sharing, there are three possibilities:

1. No sharing: each application and its data execute at one site, and there is no communication with any other program or access to any data file at other sites.

This characterizes the very early days of networking and is probably not very common today.

2. Level of data sharing; all the programs are replicated at all the sites, but data files are not.

Accordingly, user requests are handled at the site where they originate, and the necessary data files are moved around the network.

Finally, in data-plus-program sharing, both data and programs may be shared, meaning that a program at a given site can request a service from another program at a second site, which, in turn, may have to access a data file located at a third site.

Distribution Design

Along the second dimension of access pattern behavior, it is possible to identify two alternatives.

1. The access patterns of user requests may be static, so that they do not change over time, or dynamic. It is obviously considerably easier to plan for and manage the static environments than would be the case for dynamic distributed systems.

Unfortunately, it is difficult to find many real-life distributed applications that would be classified as static.

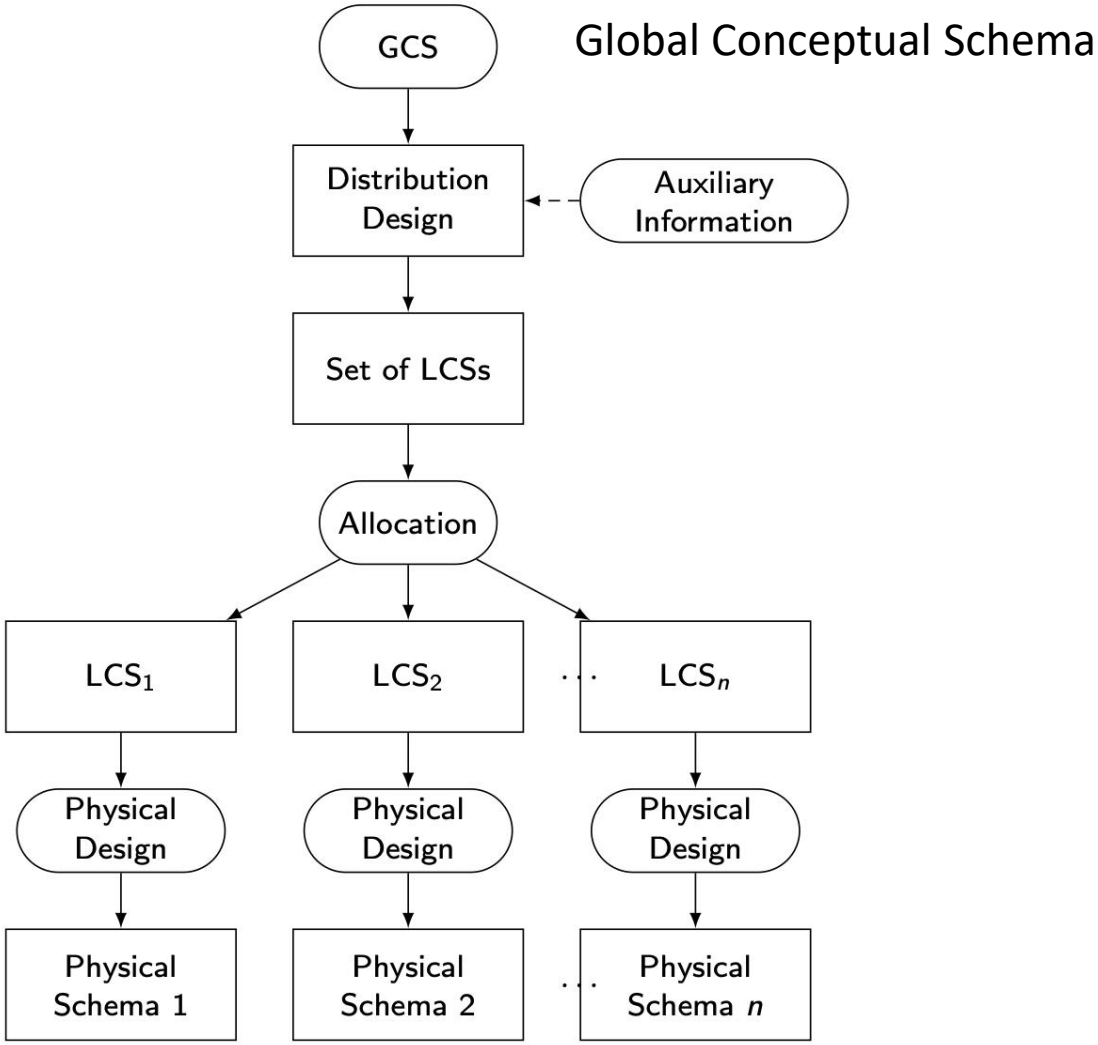
1. The third dimension of classification is the level of knowledge about the access pattern behavior. One possibility, is that the designers do not have any information about how users will access the database.

Distribution Design

Two major strategies that have been identified for **designing distributed databases** are the top-down approach and the bottom-up approach .

As the names indicate, they constitute very different approaches to the design process.
Top-down approach is more suitable for tightly integrated, homogeneous distributed DBMSs, while bottom-up design is more suited to multidatabases

Distribution Design



Distribution Design

The global conceptual schema (GCS) and access pattern information collected as a result of view design are inputs to the *distribution design* step.

The objective at this stage, is to design the local conceptual schemas (LCSs) by distributing the entities over the sites of the distributed system.
It is possible, to treat each entity as a unit of distribution. Given that we use the relational model as the basis of discussion in this book, the entities correspond to relations.

Rather than distributing relations, it is quite common to divide them into subrelations, called *fragments*, which are then distributed. Thus, the distribution design activity consists of two steps: *fragmentation* and *allocation*. The reason for separating the distribution design into two steps is to better deal with the complexity of the problem. .

Distribution Design

The last step in the design process is the physical design, which maps the local conceptual schemas to the physical storage devices available at the corresponding sites. The inputs to this process are the local conceptual schema and the access pattern information about the fragments in them.
It is well known that design and development activity of any kind is an ongoing process requiring constant monitoring and periodic adjustment and tuning. We have therefore included observation and monitoring as a major activity in this process. Note that one does not monitor only the behavior of the database implementation but also the suitability of user views. The result is some form of feedback, which may result in backing up to one of the earlier steps in the design.

Outline

- Distributed and Parallel Database Design
- **Distribution Design Issues**
  - Fragmentation
    - Data distribution
    - Combined approaches

Fragmentation

- Can't we just distribute relations?
- What is a reasonable unit of distribution?
  - relation
    - views are subsets of relations ➔ locality
    - extra communication
  - fragments of relations (sub-relations)
    - concurrent execution of a number of transactions that access different portions of a relation
    - views that cannot be defined on a single fragment will require extra processing
    - semantic data control (especially integrity enforcement) more difficult

Example Database

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal.
E3	A. Lee	Mech. Eng.
E4	J. Miller	Programmer
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E7	R. Davis	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	SAL
Elect. Eng.	40000
Syst. Anal.	34000
Mech. Eng.	27000
Programmer	24000

Fragmentation Alternatives – Horizontal

$PROJ_1$: projects with budgets less than $200,000

$PROJ_2$: projects with budgets greater than or equal to $200,000

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	255000	New York
P4	Maintenance	310000	Paris

Fragmentation Alternatives – Vertical

$PROJ_1$: information about project budgets

$PROJ_2$: information about project names and locations

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAM	New York
P4	Maintenance	Paris

Correctness of Fragmentation

- Completeness
  - Decomposition of relation R into fragments R_1, R_2, ..., R_n is complete if and only if each data item in R can also be found in some R_i
- Reconstruction
  - If relation R is decomposed into fragments R_1, R_2, ..., R_n, then there should exist some relational operator ∇ such that

$$R = \nabla_{1 \le i \le n} R_i$$

- Disjointness
  - If relation R is decomposed into fragments R_1, R_2, ..., R_n, and data item d_i is in R_j, then d_i should not be in any other fragment R_k ($k \neq j$).

Allocation Alternatives

- Non-replicated
  - partitioned : each fragment resides at only one site
- Replicated
  - fully replicated : each fragment at each site
  - partially replicated : each fragment at some of the sites
- Rule of thumb:

$$\text{If } \frac{\text{read-only queries}}{\text{update queries}} << 1, \text{ replication is advantageous,}$$
$$\text{otherwise replication may cause problems}$$

Comparison of Replication Alternatives

	Full replication	Partial replication	Partitioning
QUERY PROCESSING	Easy	Same difficulty	
DIRECTORY MANAGEMENT	Easy or nonexistent	Same difficulty	
CONCURRENCY CONTROL	Moderate	Difficult	Easy
RELIABILITY	Very high	High	Low
REALITY	Possible application	Realistic	Possible application