Distributed Data Management

Big Data Management

Knowledge objectives

1. Give a definition of Distributed System
2. Enumerate the 6 challenges of a Distributed System
3. Give a definition of Distributed Database
4. Explain the different transparency layers in DDBMS
5. Identify the requirements that distribution imposes on the ANSI/SPARC architecture
6. Draw a classical reference functional architecture for DDBMS
7. Enumerate the 8 main features of Cloud Databases
8. Explain the difficulties of Cloud Database providers to have multiple tenants
9. Enumerate the 4 main problems tenants/users need to tackle in Cloud Databases
10. Distinguish the cost of sequential and random access
11. Explain the difference between the cost of sequential and random access
12. Distinguish vertical and horizontal fragmentation
13. Recognize the complexity and benefits of data allocation
14. Explain the benefits of replication
15. Discuss the alternatives of a distributed catalog

Understanding Objectives

- Decide when a fragmentation strategy is correct

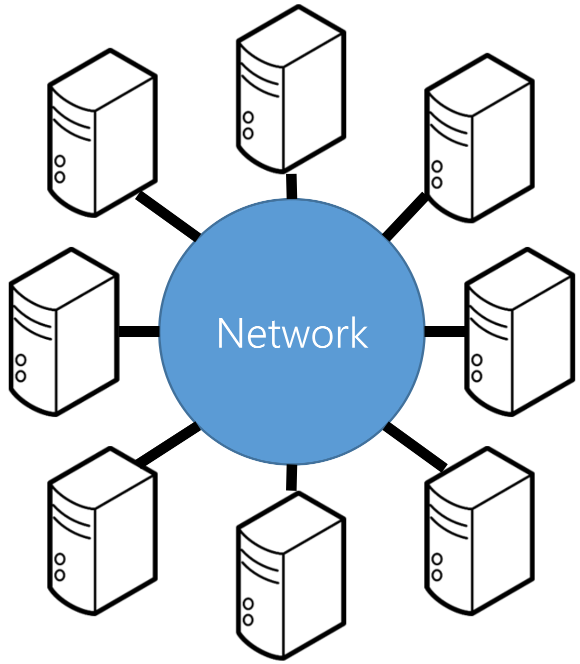
Distributed Systems

Distributed system

"One in which components located at networked computers communicate and coordinate their actions only by passing messages."

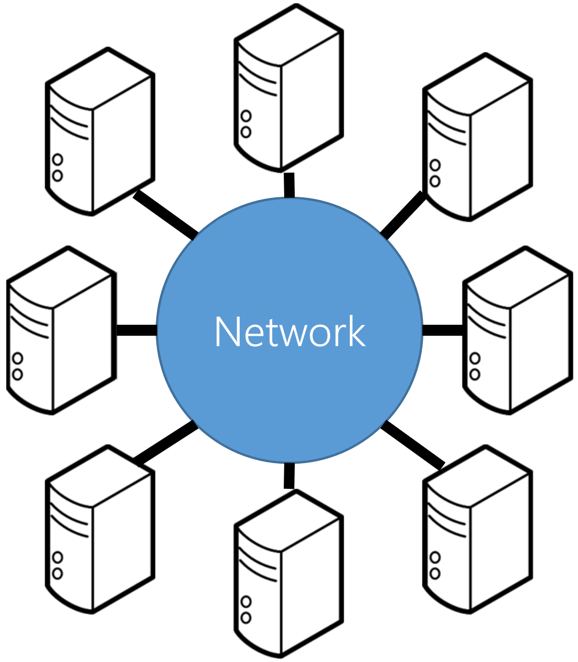
G. Coulouris et al.

- Characteristics:
  - Concurrency of components
  - Independent failures of components
  - Lack of a global clock



Challenges of distributed systems

- ~~Openness~~
- Scalability
- Quality of service
  - Performance/Efficiency
  - Reliability/Availability
  - ~~Confidentiality~~
- Concurrency
- Transparency
- ~~Heterogeneity of components~~



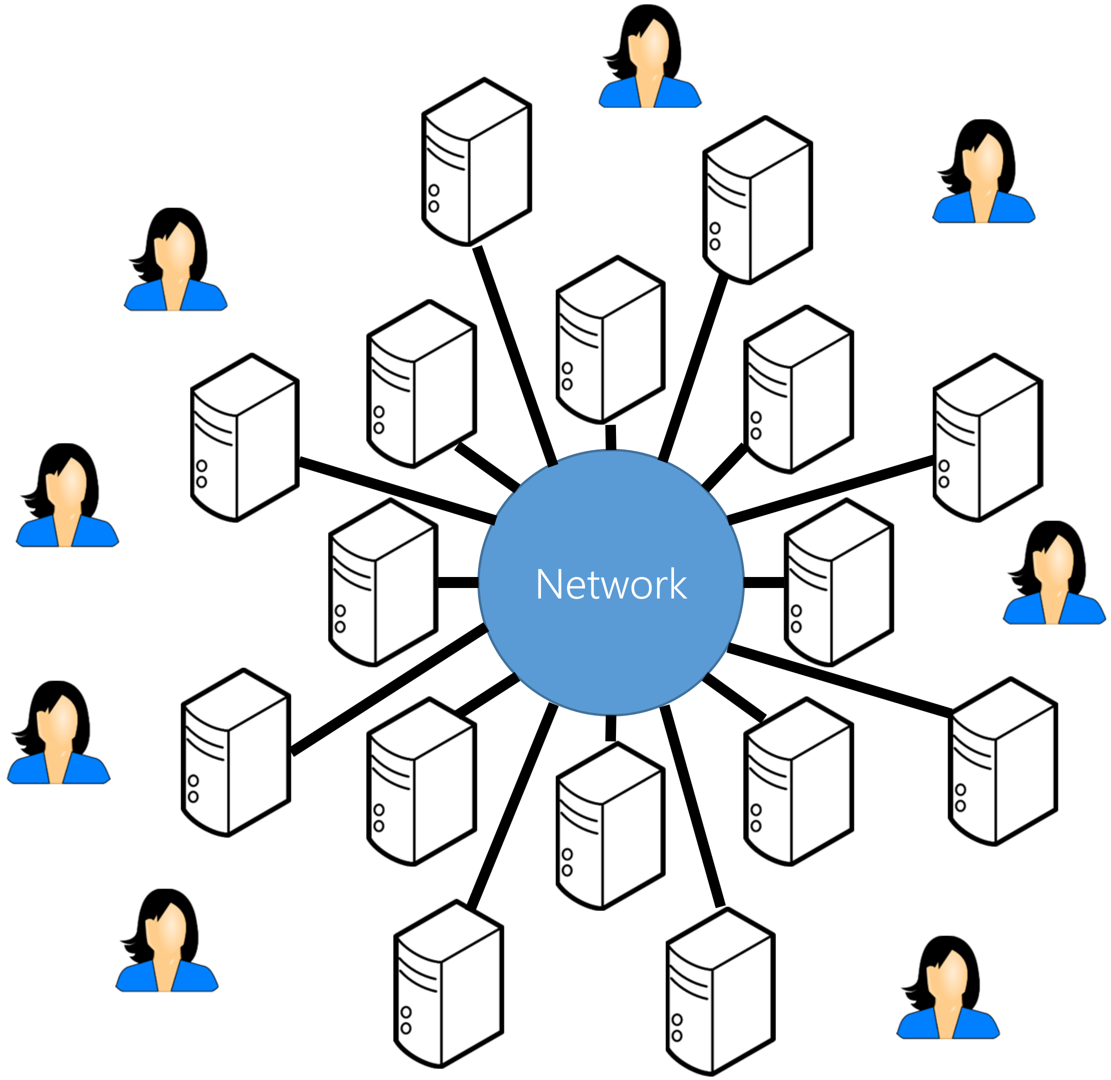
Scalability

Cope with large workloads

- Scale out

- Use:
  - Automatic load-balancing

- Avoid:
  - Bottlenecks
  - Unnecessary communication
    - Peer-to-peer

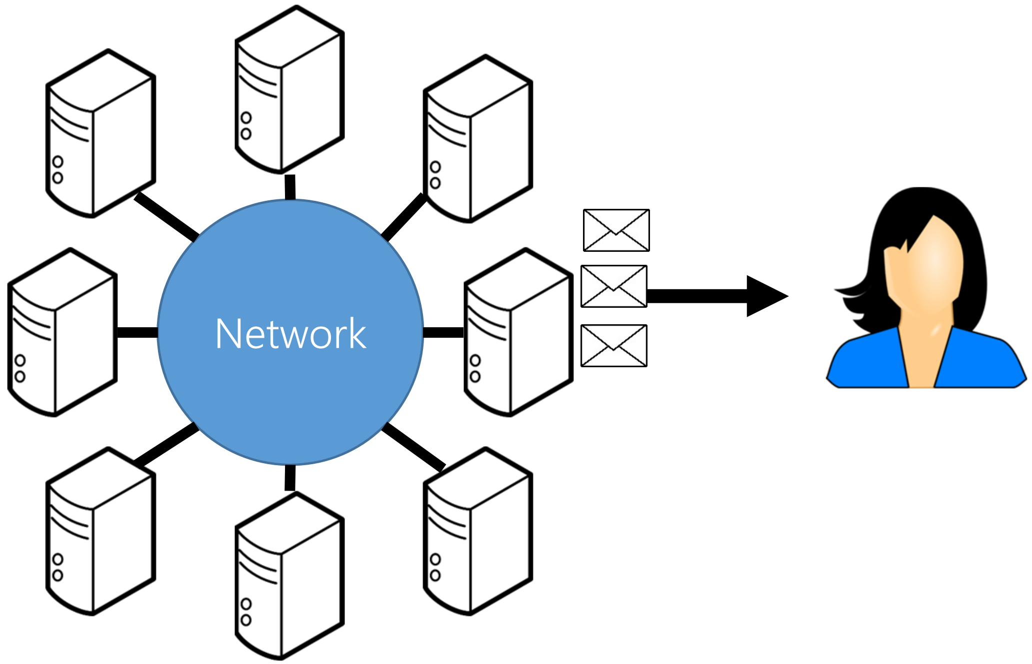


Performance/Efficiency

Efficient processing

- Minimize latencies
- Maximize throughput

- Use
  - Parallelism
  - Network optimization
  - Distributed indexes

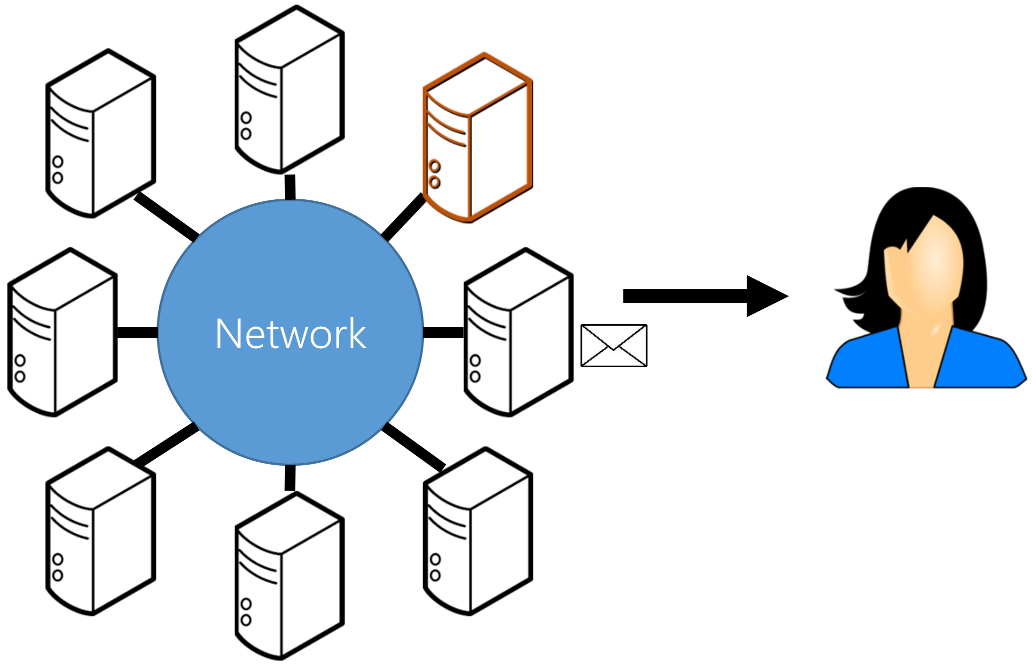


Reliability/Availability

a) Keep data consistency

b) Keep the system running
   - Even in the case of failures

- Use
  - Replication
  - Flexible routing
  - Heartbeats
  - Automatic recovery

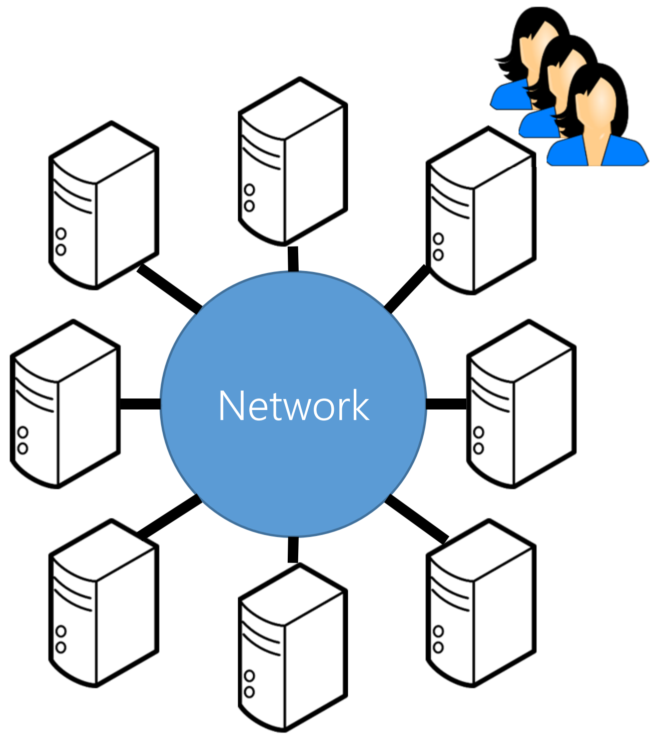


Concurrency

Share resources as much as possible

- Use
  - Consensus Protocols

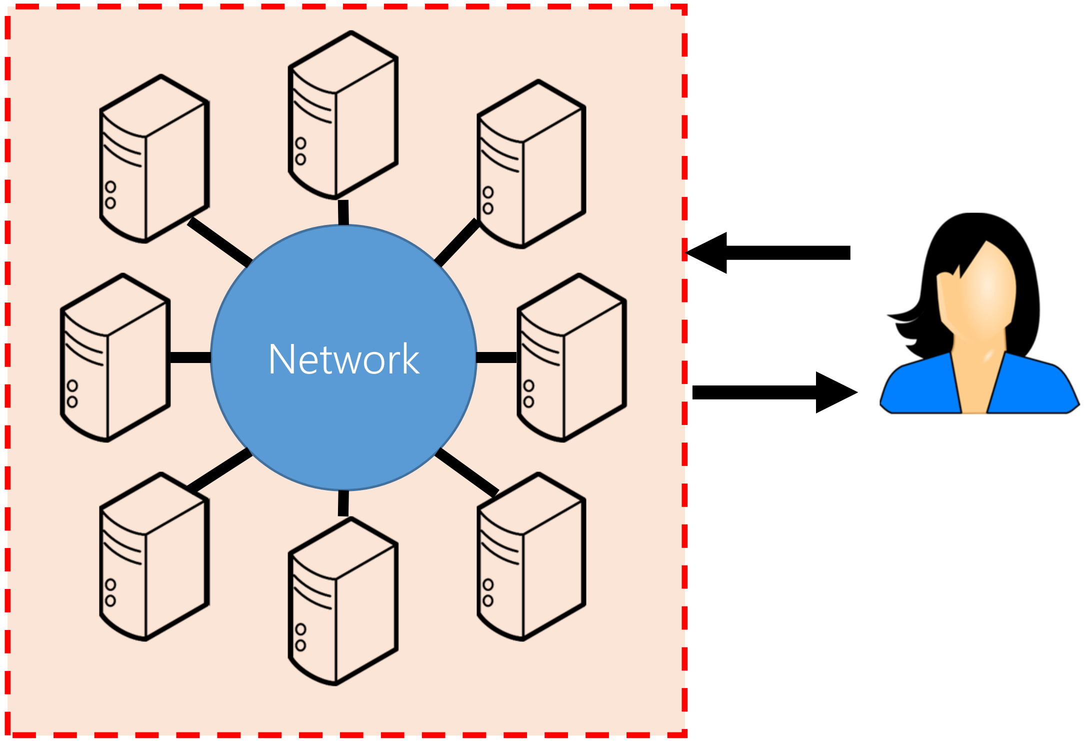
- Avoid
  - Interferences
  - Deadlocks



Transparency

a) Hide implementation (i.e., physical) details to the users

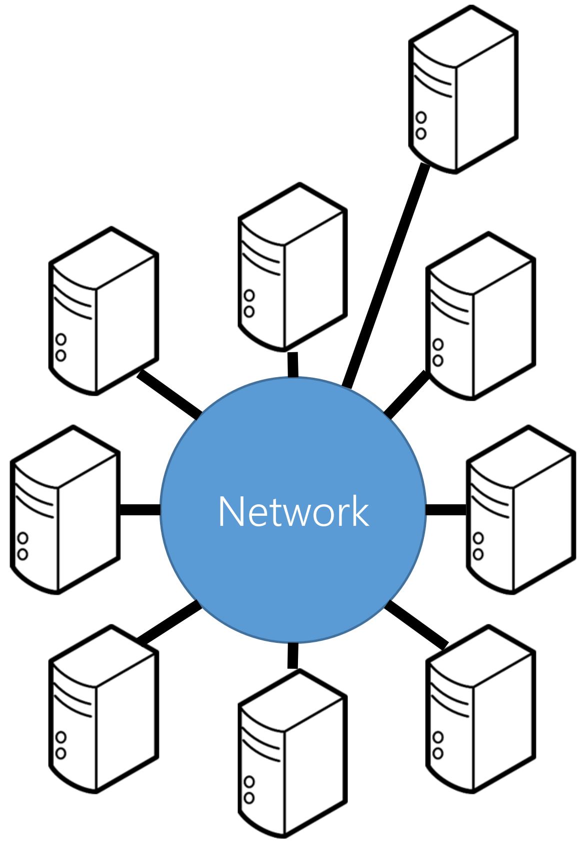
b) Make transparent to the user all the mechanisms to solve the other challenges



Further objectives

- Use
  - Platform-independent software

- Avoid
  - Complex configurations
  - Specific hardware/software

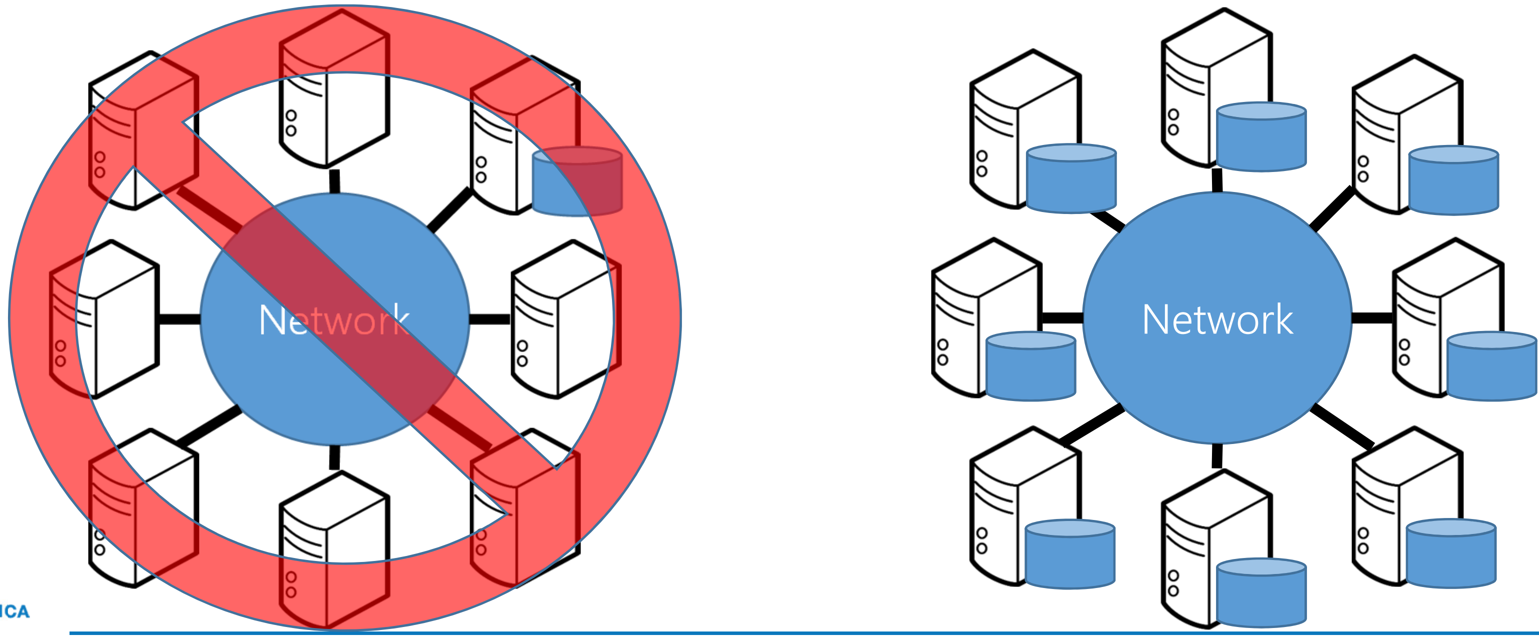


Distributed Database Systems

Distributed database

"A Distributed DataBase (DDB) is an integrated collection of databases that is physically distributed across sites in a computer network. A Distributed DataBase Management System (DDBMS) is the software system that manages a distributed database such that the distribution aspects are transparent to the users."

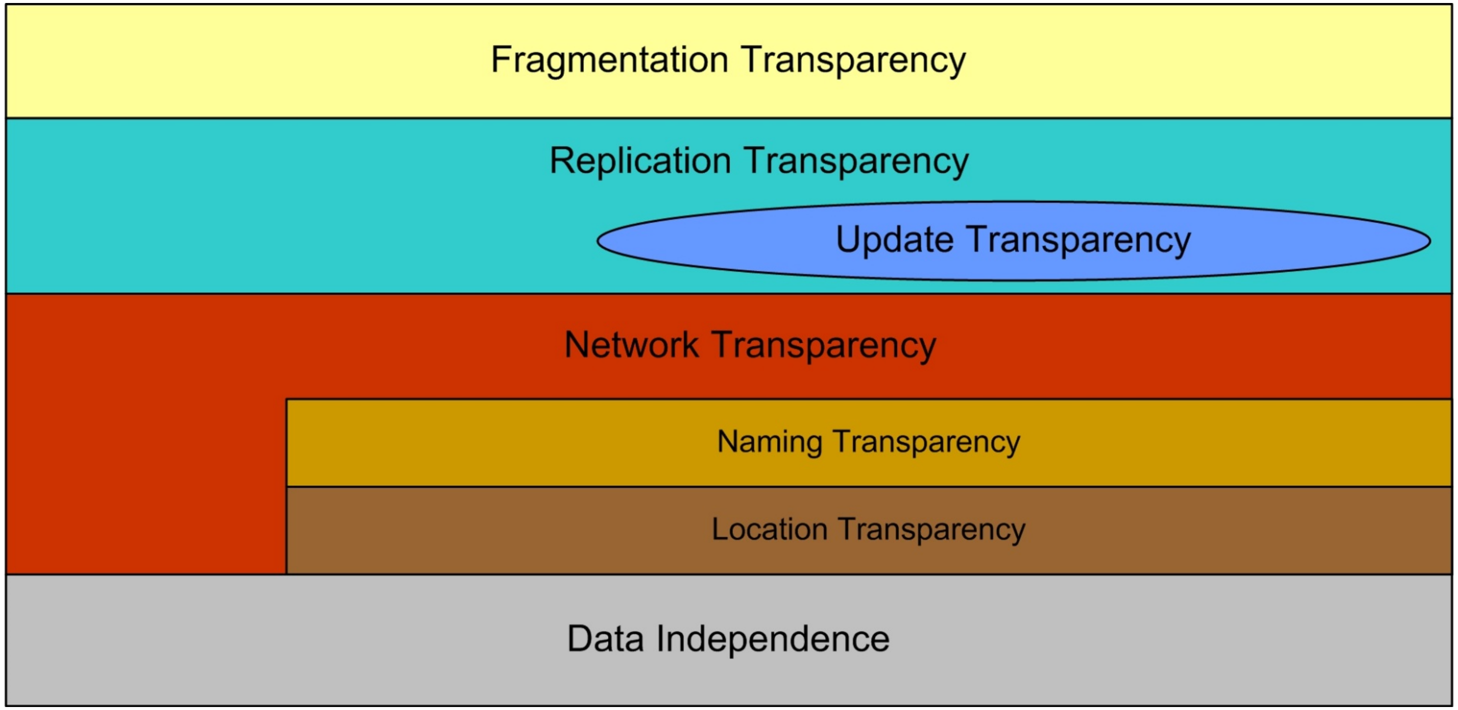
Encyclopedia of Database Systems



Transparency layers (I)

- Data independency at the logical and physical level must be guaranteed
  - Inherited from centralized DBMSs (ANSI SPARC)
- Network transparency
  - Data access must be independent regardless where data is located
  - Each data object must have a unique name
- Replication transparency
  - The user must not be aware of the existing replicas
- Fragmentation transparency
  - The user must not be aware of partitioning

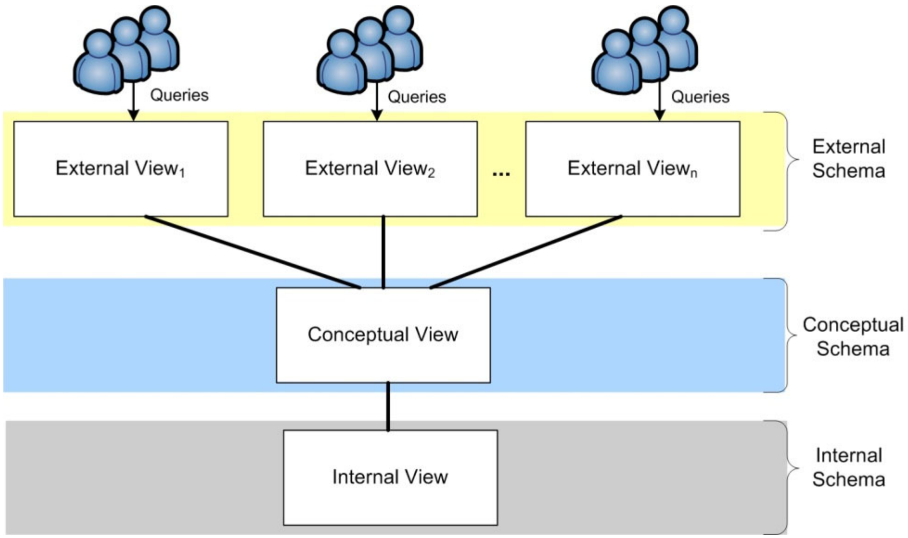
Transparency layers (II)

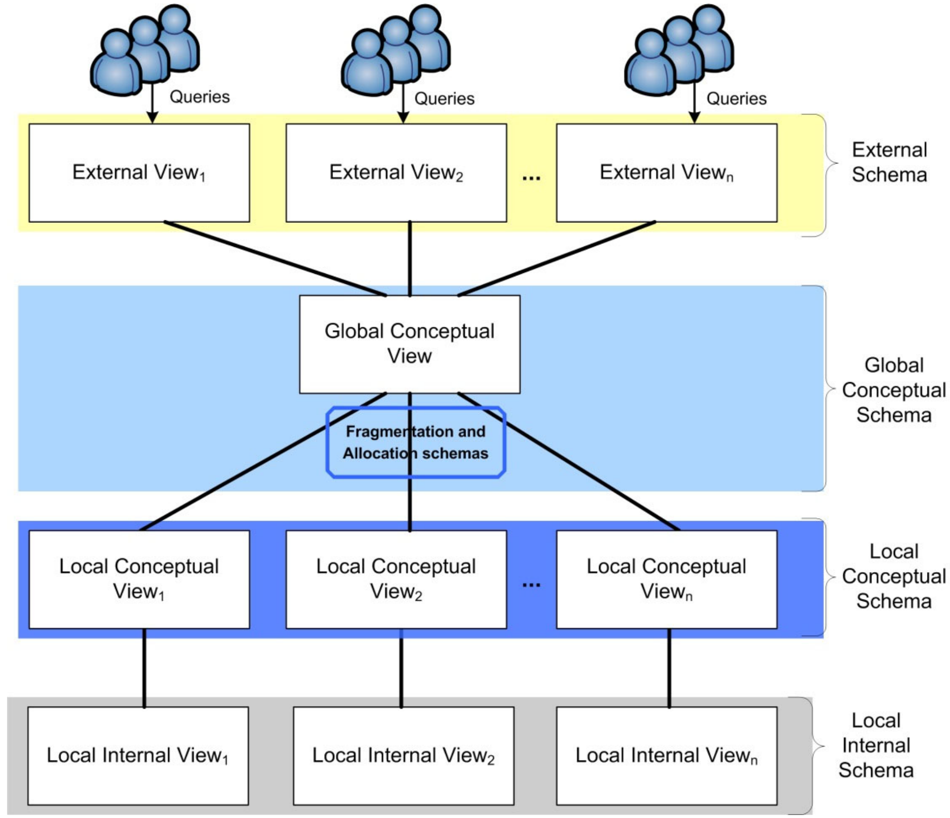


Classification According to Degree of Autonomy

	Autonomy	Central schema	Query transparency	Update transparency
DDBMS	No	Yes	Yes	Yes
T.C. Federated	Yes	Yes	Yes	Limited
L.C. Federated	Yes	No	Yes	Limited
Multi-database	Yes	No	No	No

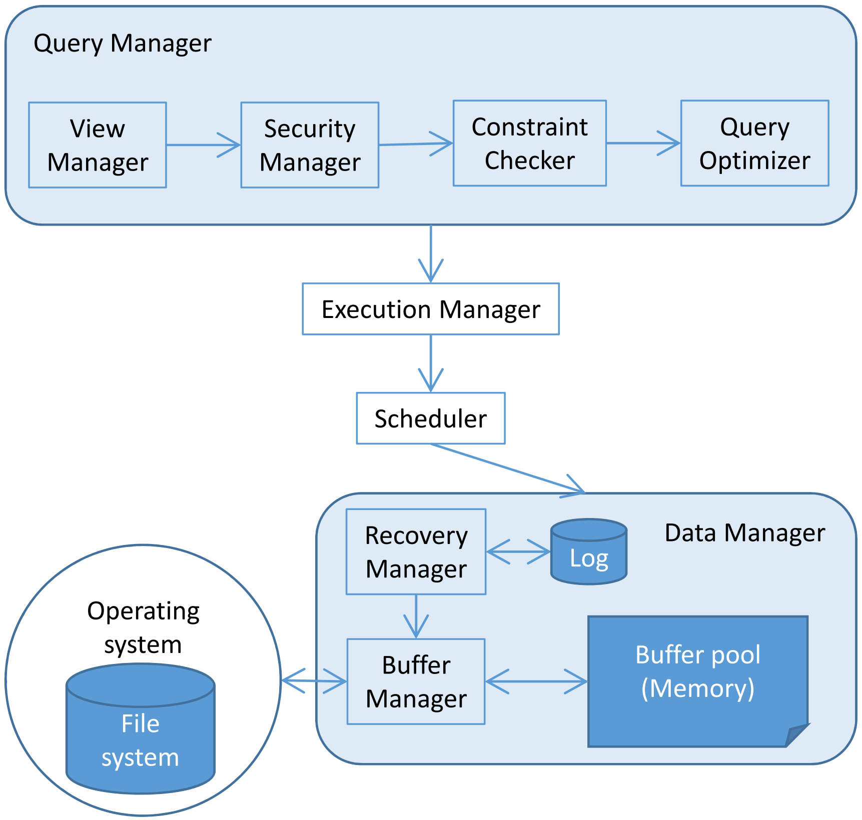
Extended ANSI-SPARC Architecture of Schemas



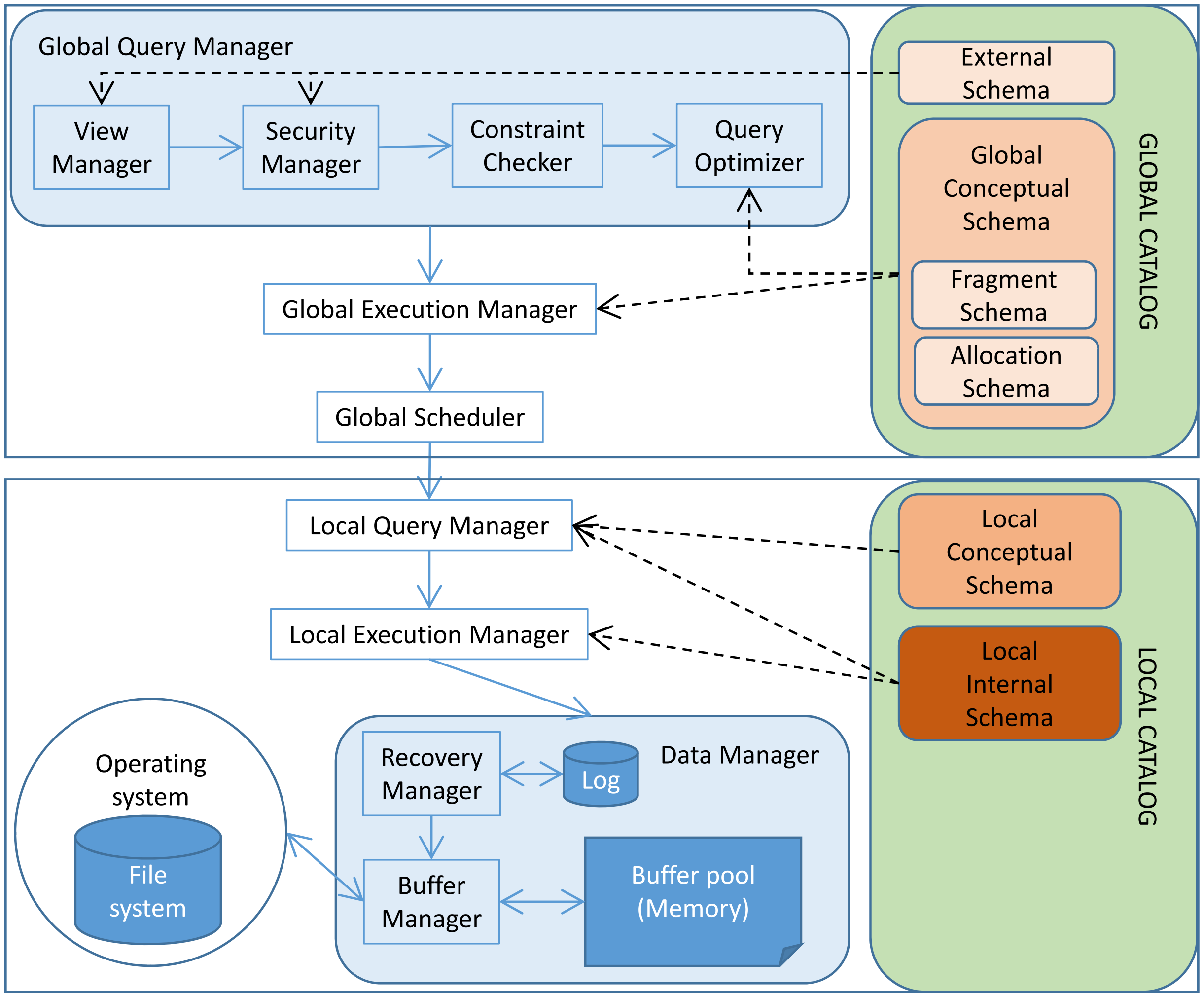


- Global catalog (Mappings between ESs – GCS and GCS – LCSs)
- Each node has a local catalog (Mappings between LCS_i – IS_i)

Centralized DBMS Functional Architecture

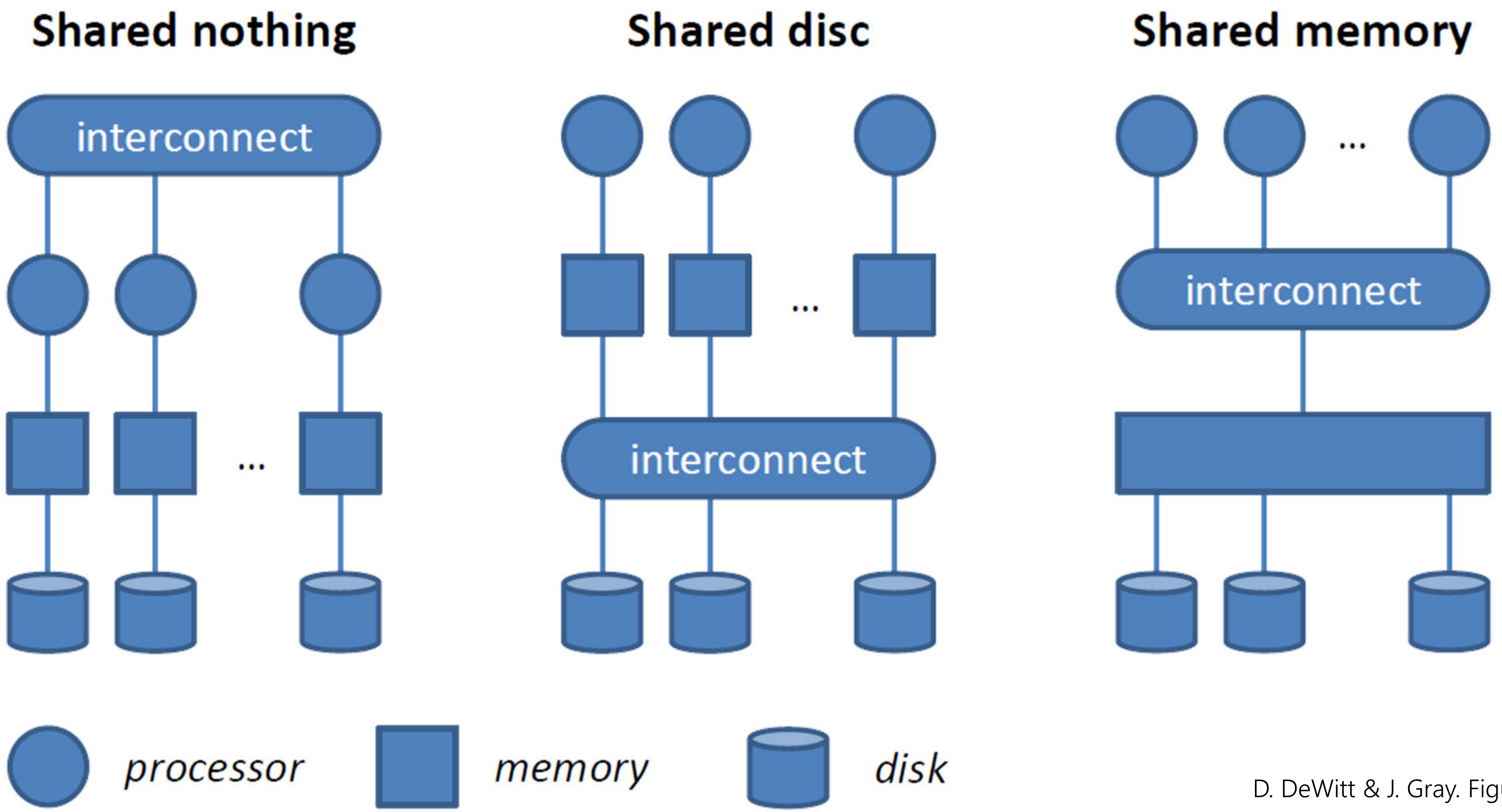


Distributed DBMS Functional Architecture



Cloud Databases

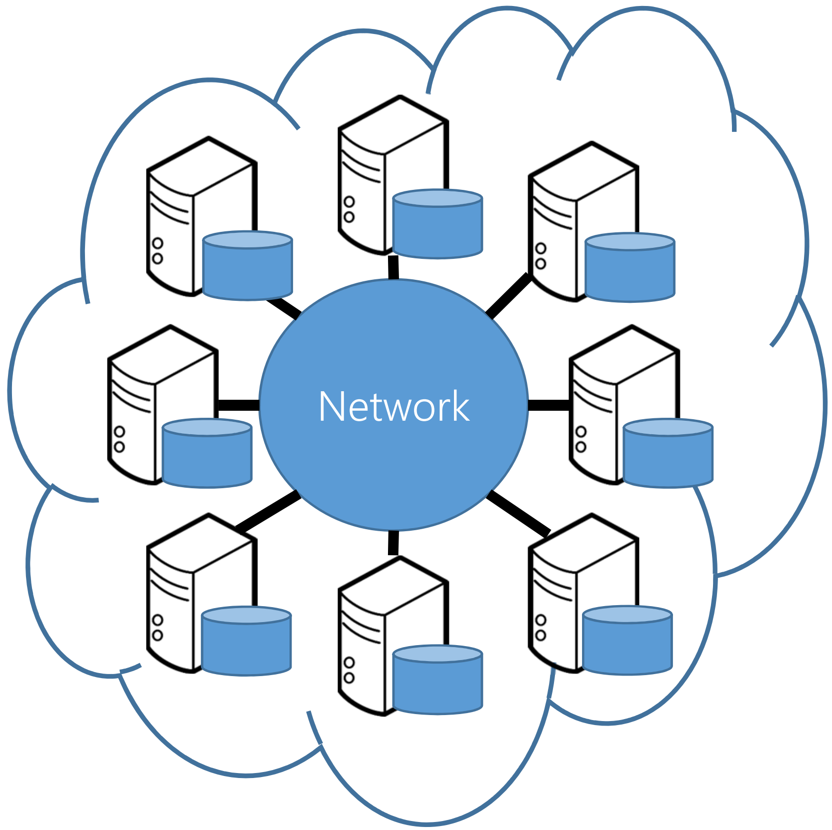
Parallel database architectures



D. DeWitt & J. Gray. Figure by D. Abadi

Key Features of Cloud Databases

- Scalability
  a) Ability to horizontally scale
- Quality of service
  - Performance/Efficiency
    b) Fragmentation: Replication & Distribution
    c) Indexing: Distributed indexes and RAM
  - Reliability/Availability
- Concurrency
  d) Weaker concurrency model than ACID
- Transparency
  e) Simple call level interface or protocol
    - No declarative query language
- Further objectives
  f) Quick/Cheap set up
  g) Multi-tenancy
  h) Flexible schema
    - Ability to dynamically add new attributes



Multi-tenancy platform problems (provider side)

- Difficulty: Unpredictable load characteristics
  - Variable popularity
  - Flash crowds
  - Variable resource requirements
- Requirement: Support thousands of tenants
  a) Maintain metadata about tenants (e.g., activated features)
  b) Self-managing
  c) Tolerating failures
  d) Scale-out is necessary (sooner or later)
     - Rolling upgrades one server at a time
  e) Elastic load balancing
     - Dynamic partitioning of databases

Data management problems (tenant side)

I. Distributed data design
  - Data fragmentation
  - Data allocation
  - Data replication

II. Distributed catalog management
  - Metadata fragmentation
  - Metadata allocation
  - Metadata replication

III. Distributed transaction management
  - Enforcement of ACID properties
    - Distributed recovery system
    - Distributed concurrency control system
    - Replica consistency
      - Latency&Availability vs. Update performance

IV. Distributed query processing
  - Optimization considering
    - Distribution/Parallelism
      - Communication overhead
    - Replication

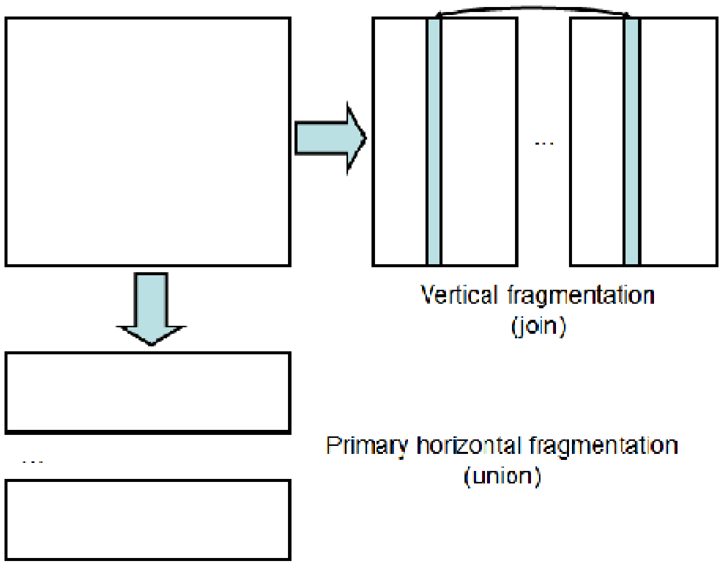
Distributed Data Design

Problem I

DDB Design

- Given a DB and its workload, how should the DB be split and allocated to sites as to optimize certain objective functions
    - Minimize resource consumption for query processing

- Two main issues:
    - Data fragmentation
    - Data allocation
        - Data replication



Data Fragmentation

- Usefulness
  - An application typically accesses only a subset of data
  - Different subsets are (naturally) needed at different sites
  - The degree of concurrency is enhanced
    - Facilitates parallelism
      - Fragments can be even defined dynamicaly (i.e., at query time, not at design time)
- Difficulties
  - Complicates the catalog management
  - May lead to poorer performance when multiple fragments need to be joined
    - Fragments likely to be used jointly can be colocated to minimize communication overhead
  - Costly to enforce the dependency between attributes in different fragments

Fragmentation Correctness

- Completeness
  - Every tuple in the relation must be assigned to a fragment
- Disjointness
  - There is no redundancy and every datum is assigned to only one fragment
    - The decision to replicate data is in the allocation phase
- Reconstruction
  - The original relation can be reconstructed from the fragments
    - Union for horizontal fragmentation
    - Join for vertical fragmentation

Finding the best fragmentation strategy

- Consider it per table
  - Computational cost is NP-hard
- Needed information
  - Workload
    - Frequency of each query
    - Access plan and cost of each query
      - Take intermediate results and repetitive access into account
  - Value distribution and selectivity of predicates
- Work in three phases
  1. Determine primary partitions (i.e., subsets of attributes always accessed together)
  2. Generate a disjoint and covering combination of primary partitions
  3. Evaluate the cost of all combinations generated in the previous phase

Data Allocation

- Given a set of fragments, a set of sites on which a number of applications are running, **allocate** each fragment such that some optimization criterion is met (subject to certain constraints)
- It is known to be an NP-hard problem
  - The optimal solution depends on many factors
    - Location in which the query originates
    - The query processing strategies (e.g., join methods)
  - Furthermore, in a dynamic environment the workload and access patterns may change
- The problem is typically simplified with certain assumptions (e.g., only communication cost considered)
- Typical approaches build *cost models* and any optimization algorithm can be adapted to solve it
  - Heuristics are also available: (e.g., best-fit for non-replicated fragments)
  - Sub-optimal solutions

Data Replication

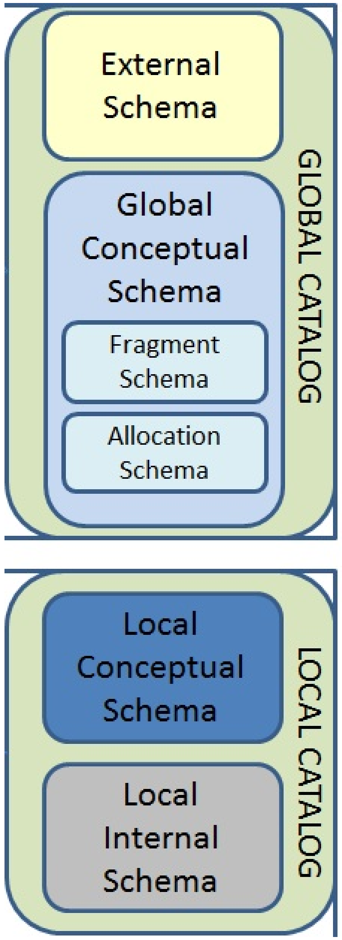
- Generalization of Allocation (for more than one location)
- Provides execution alternatives
- Improves availability
- Generates consistency problems
  - Specially useful for read-only workloads
    - No synchronization required

Distributed Catalog Management

Problem II

DDBMS Catalog Characteristics

- Fragmentation
  - Global metadata
    - External schemas
    - Global conceptual schema
      - Fragment schema
      - Allocation schema
  - Local metadata
    - Local conceptual schema
    - Physical schema
- Allocation
  - Global metadata in the coordinator node
  - Local metadata in the workers
- Replication
  a) Single-copy (Coordinator node)
    - Single point of failure
    - Poor performance (potential bottleneck)
  b) Multi-copy (Mirroring, secondary node)
    - Requires sinchronization



Closing

Summary

- Distributed Systems
- Distributed Database Systems
  - Distributed Database Systems Architectures
- Cloud Databases
- Sequential vs Random Access
- Distributed Database Design
  - Fragmentation
    - Kinds
    - Characteristics
  - Allocation
    - Replication
- Distributed Catalog

References

- D. DeWitt & J. Gray. *Parallel Database Systems: The future of High Performance Database Processing*. Communications of the ACM, June 1992
- N. J. Gunther. *A Simple Capacity Model of Massively Parallel Transaction Systems*. CMG National Conference, 1993
- L. Liu, M.T. Özsu (Eds.). *Encyclopedia of Database Systems*. Springer, 2009
- M. T. Özsu & P. Valduriez. *Principles of Distributed Database Systems*, 3rd Ed. Springer, 2011
- G. Coulouris et al. *Distributed Systems: Concepts and Design*, 5th Ed. Addisson-Wesley, 2012