Types of Architecture Database

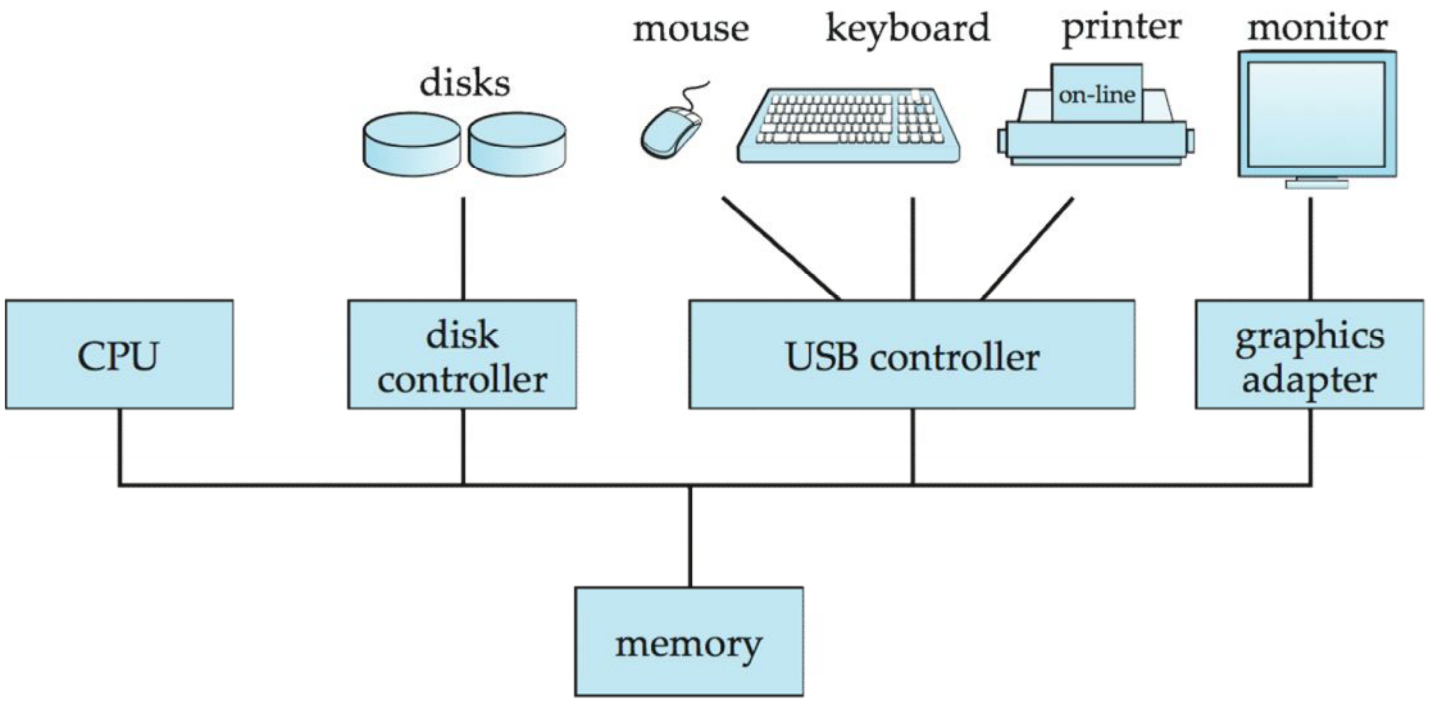
Database System Concepts, 6th Ed.

Chapter 17: Database System Architectures

- 17.1 Centralized and Client-Server Database
- 17.3 Parallel Database
- 17.4 Distributed Database
- 17.5 Cluster Database
- Grid Database
- Cloud Dtabase
- P2P Database
- Memory Database

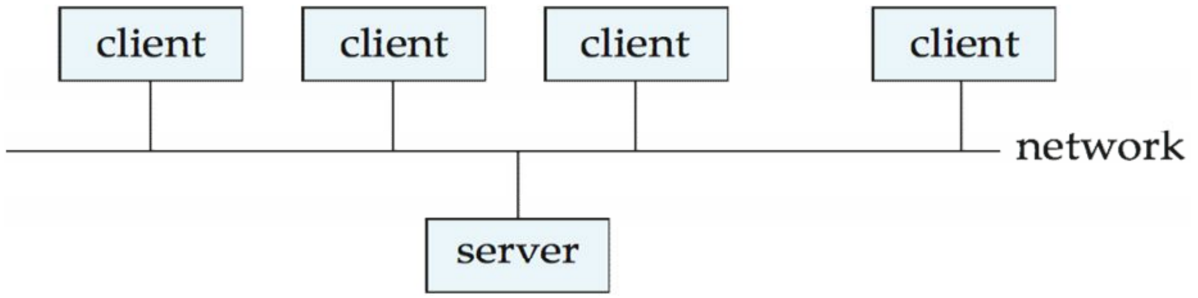
Centralized Systems

- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provides access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system vie terminals. Often called *server* systems.

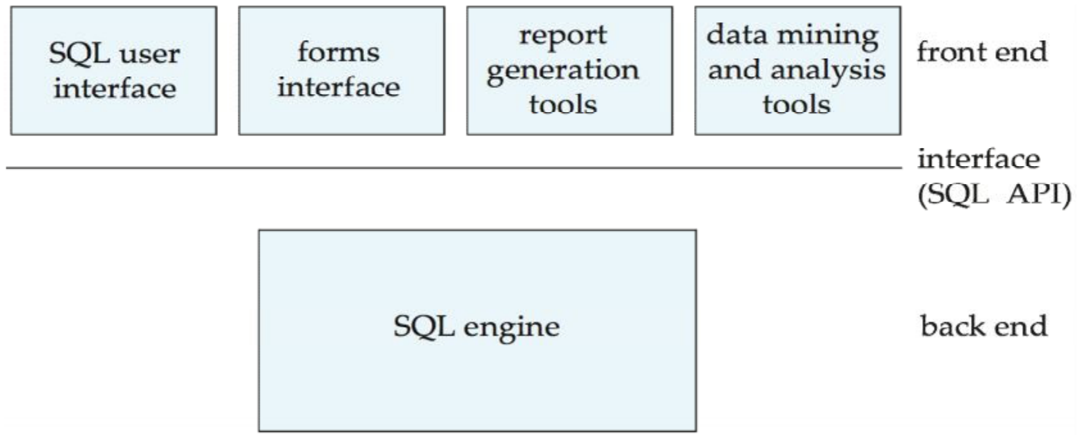


Client-Server Systems

- Server systems satisfy requests generated at *m* client systems, whose general structure is shown below:



- Database functionality can be divided into:
  - **Back-end**: manages access structures, query evaluation & optimization, CC and recovery.
  - **Front-end**: consists of tools such as *forms*, *report-writers*, and graphical UI facilities.
- The interface between the front-end and the back-end is through SQL or through an application program interface

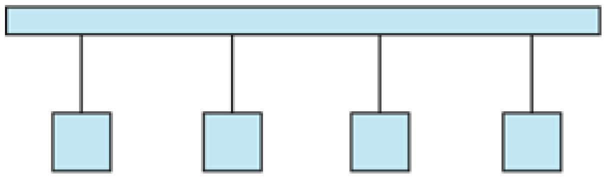


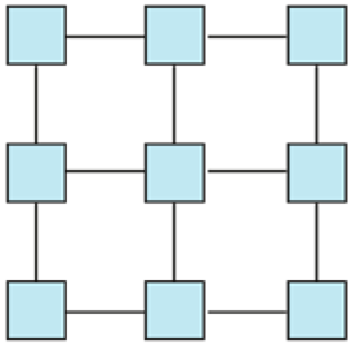
Parallel Systems

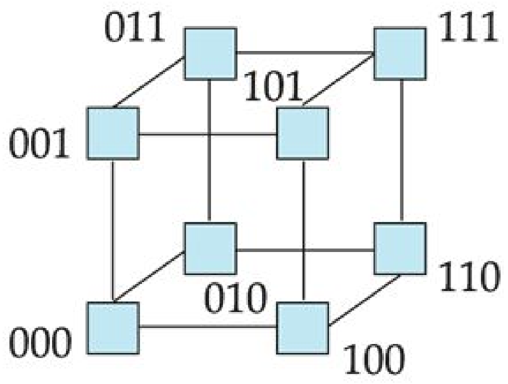
- Parallel database systems consist of multiple processors and multiple disks connected by a fast interconnection network.
  - A **coarse-grain parallel** machine consists of a small number of powerful processors
  - A **fine grain parallel** or **massively parallel** machine utilizes thousands of smaller processors.
- Two main performance measures:
  - **throughput** --- the number of tasks that can be completed in a given time interval
  - **response time** --- the amount of time it takes to complete a single task from the time it is submitted

Interconnection Network Architectures

- Bus
  - System components send data on and receive data from a single bus
  - Does not scale well with increasing parallelism
- Mesh
  - Components are arranged as nodes in a grid, and each component is connected to all adjacent components
  - Communication links grow with growing number of components, and so scales better
  - But may require 2√*n* hops to send message to a node (or √*n* with wraparound connections at edge of grid)
- Hypercube
  - Components are numbered in binary
  - Components are connected to one another if their binary representations differ in exactly one bit
  - *N* components are connected to *log(n)* other components and can reach each other via at most *log(n)* links; reduces communication delays


(a) bus

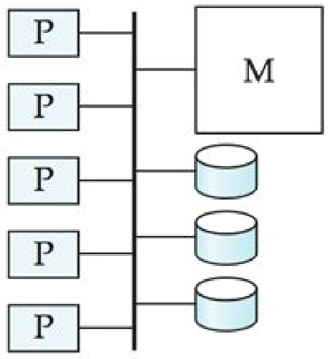

(b) mesh



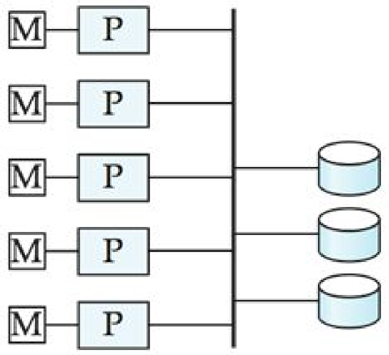
(c) hypercube

Parallel Database Architectures

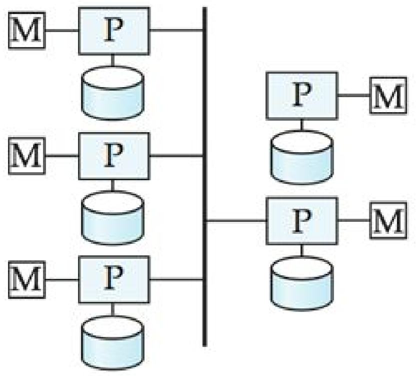
- **Shared memory** -- processors share a common memory
- **Shared disk** -- processors share a common disk, sometimes called clusters
- **Shared nothing** -- processors share neither a common memory nor common disk
- **Hierarchical** -- hybrid of the above architectures



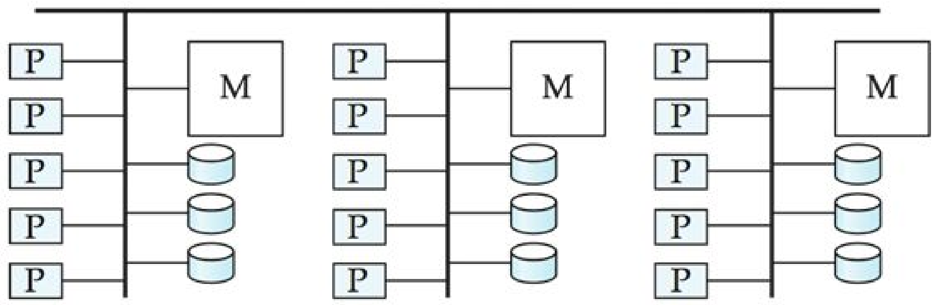
(a) shared memory



(b) shared disk



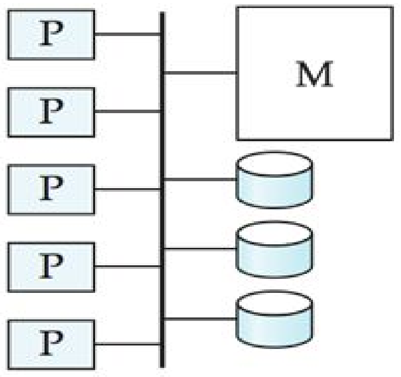
(c) shared nothing



(d) hierarchical

Shared-Memory Parallel DB

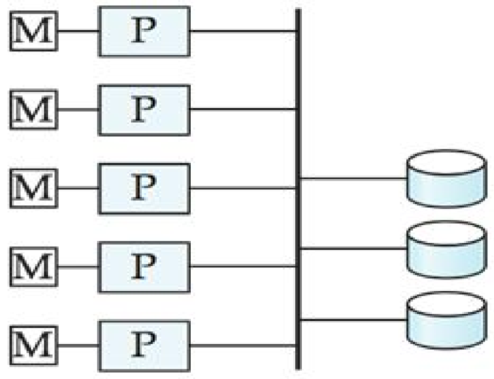
- Processors and disks have access to a common memory, typically via a bus or through an interconnection network.
- Advantage
  - Extremely efficient communication between processors
  - Data in shared memory can be accessed by any processor without having to move it using software.
- Downside
  - Architecture is not scalable beyond 32 or 64 processors since the bus or the interconnection network becomes a bottleneck
- Widely used for lower degrees of parallelism (4 to 8).



(a) shared memory

Shared-Disk Parallel DB

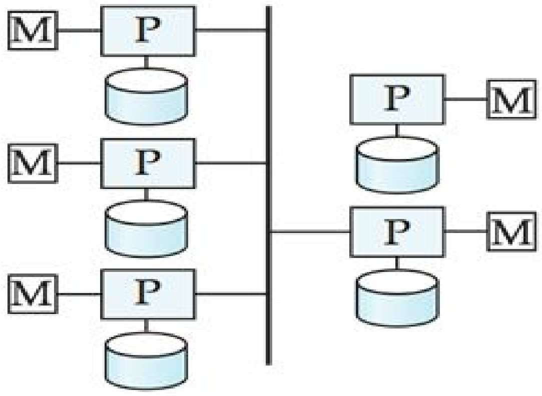
- All processors can directly access all disks via an interconnection network, but the processors have private memories.
  - The memory bus is not a bottleneck
  - Architecture provides a degree of **fault-tolerance** — if a processor fails, the other processors can take over its tasks since the database is resident on disks that are accessible from all processors.
- Examples: IBM Sysplex and DEC clusters (now part of Compaq) running RDBMS (now Oracle RDBMS) were early commercial users of this architecture
- Downside: bottleneck now occurs at interconnection to the disk subsystem.
- Shared-disk systems can scale to a somewhat larger number of processors, but communication between processors is slower.



(b) shared disk

Shared-Nothing Parallel DB

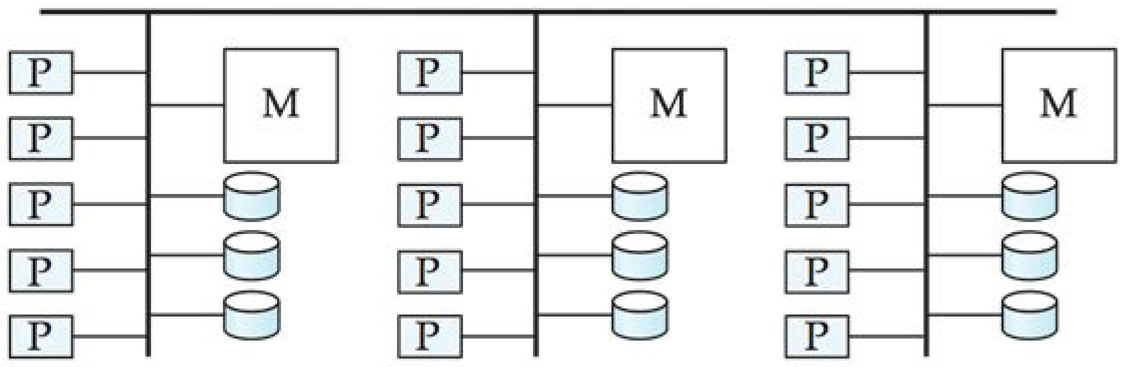
- Node consists of a processor, memory, and one or more disks.
  - Processors at one node communicate with another processor at another node using an interconnection network.
  - A node functions as the server for the data on the disk the node owns.
- Examples: Teradata, Tandem, Oracle n-CUBE
- Data in local disks (and local memory) cannot be accessed through interconnection network, thereby minimizing the interference of resource sharing.
- Shared-nothing multiprocessors can be scaled up to thousands of processors without interference.
- Main drawback: cost of communication and non-local disk access
  - Sending data involves software interaction at both ends.



(c) shared nothing

Hierarchical Parallel DB

- Combines characteristics of shared-memory, shared-disk, and shared-nothing
  - Top level is a shared-nothing architecture – nodes connected by an interconnection network, and do not share disks or memory with each other.
  - Each node could be a shared-memory system with a few processors.
  - Alternatively, each node could be a shared-disk system, and each of the systems sharing a set of disks could be a shared-memory system.
- Reduce the complexity of programming such systems by **distributed virtual-memory** architectures
  - Also called **non-uniform memory architecture (NUMA)**



(d) hierarchical

Parallel Databases

- I/O Parallelism

Reduce the time required to retrieve relations from disk by partitioning the relations on multiple disks.

- Interquery Parallelism

Different queries / transactions execute in parallel with one another
Increases transaction throughput

- Intraquery Parallelism

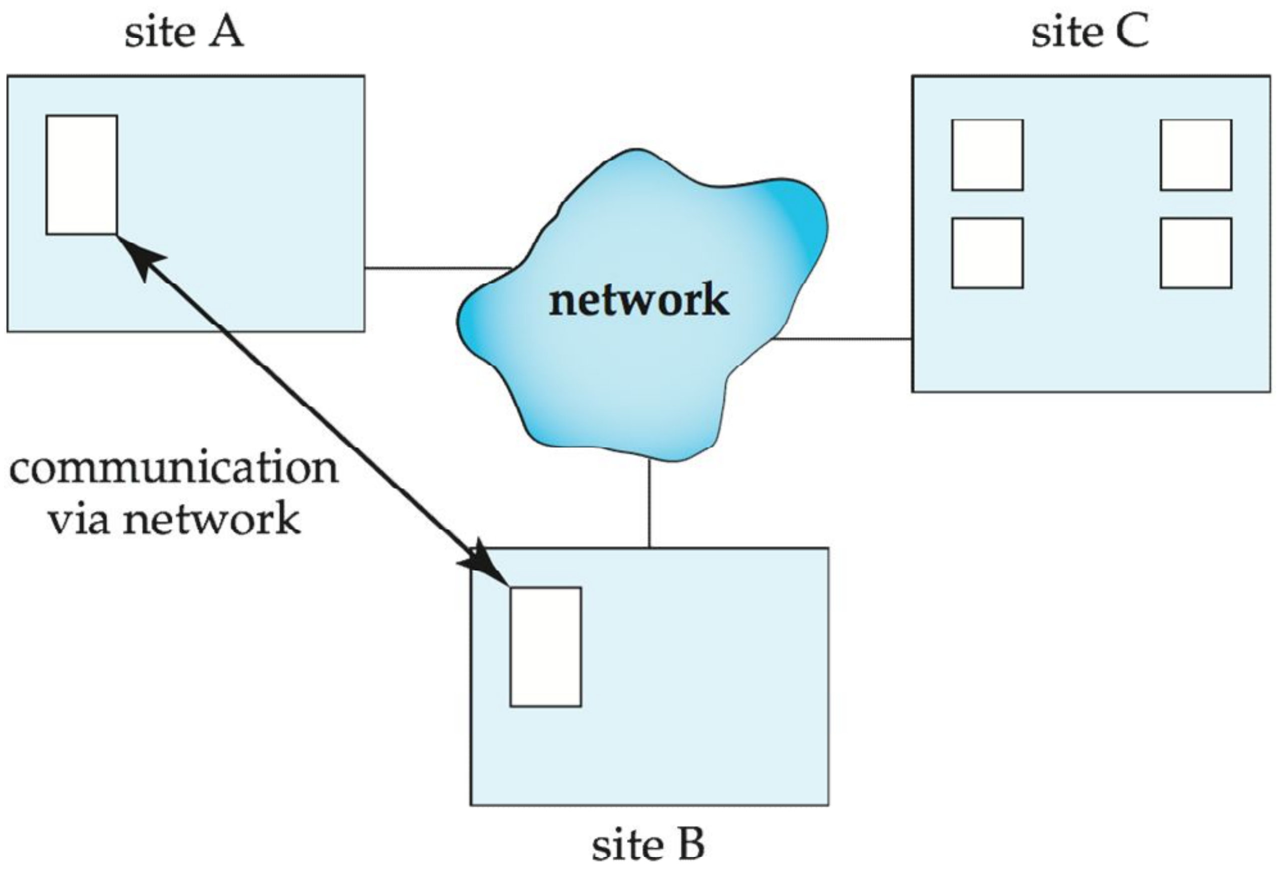
Execution of a single query in parallel on multiple processors / disks

- Intraoperation e Interoperation Parallelism

Sort, Join, Fragment-and-Replicate Join, Selection

Distributed Systems

- Data spread over multiple machines (also referred to as **sites** or **nodes**)
- Network interconnects the machines
- Data shared by users on multiple machines



Trade-offs in Distributed Systems

- Advantages
  - Sharing data
    - Users at one site able to access the data residing at some other sites.
  - Autonomy
    - Each site is able to retain a degree of control over data stored locally.
  - Higher system availability through redundancy
    - Data can be replicated at remote sites, and system can function even if a site fails.
- Disadvantages: added complexity for proper coordination among sites
  - Software development cost
  - Greater potential for bugs
  - Increased processing overhead

Distributed Databases

- Homogeneous distributed databases
  - Same software/schema on all sites, data may be partitioned among sites
  - Goal: provide a view of a single database, hiding details of distribution
- Heterogeneous distributed databases
  - Different software/schema on different sites
  - Goal: integrate existing databases to provide useful functionality
- Differentiate between *local* and *global* transactions
  - A local transaction accesses data in the *single* site at which the transaction was initiated.
  - A global transaction either accesses data in a site different from the one at which the transaction was initiated or accesses data in several different sites.

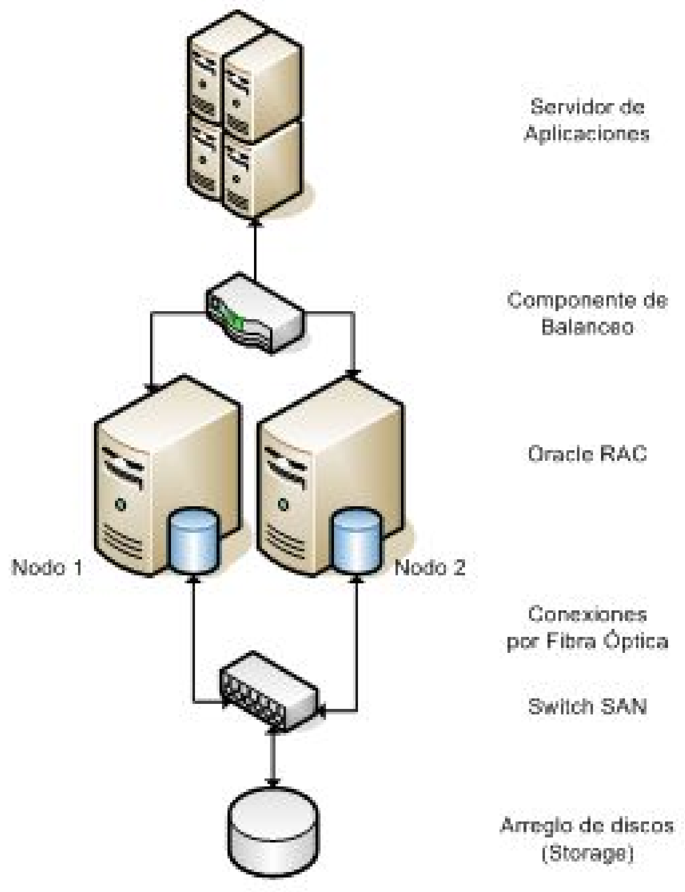
Implementation Issues for Distributed Databases

- Atomicity needed even for transactions that update data at multiple sites
- The two-phase commit protocol (2PC) used to ensure atomicity
  - Basic idea: each site executes transaction till just before commit, and the leaves final decision to a coordinator
  - Each site must follow decision of coordinator: even if there is a failure while waiting for coordinators decision
  - 2PC is not always appropriate
    - Other transaction models (persistent messaging, workflows), may be used
- Distributed concurrency control (and deadlock detection) required
- Distributed query processing: reducing communication overhead
- Replication of data items required for improving data availability

Cluster Database

Cluster

A cluster is a group of independent servers that cooperate behaving as if they were a single system.

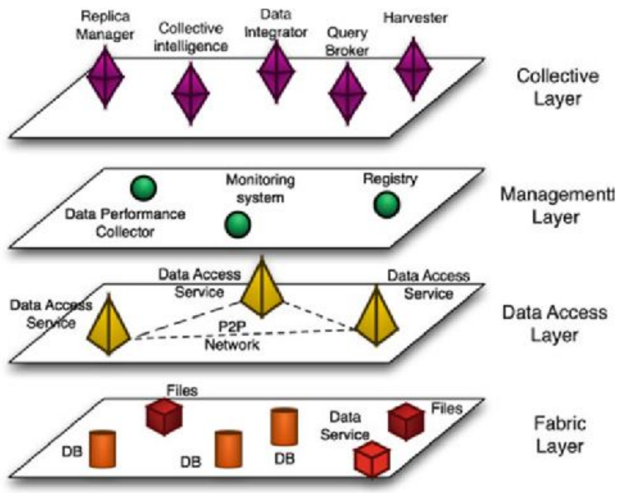


- **An Clusters Database** , it is software that allows you to use a cluster of servers running multiple instances on the same database.
- The database files are stored on physical or logical disks attached to each node, so that all active instances can read or write them.
- RAC software manages data access so that changes in data are coordinated between instances and each instance sees consistent images of the database.
- The cluster interconnect allows instances to pass coordination information and data images to each other.

Grid Database

- Grid vision represent a pool of database servers, storage and networks in an inter-related resource platform and Effective Management of workload
- Clustering technology helped to set up and manage the overall Grid strategy. Thus, RAC is an integral part of Grid computing, but it is not the sum total of it.
- The goal of Grid computing is to supply high availability information sharing supported by an architecture such that users can getting as much information or processing as they need on demand.
- Users no longer have to consider where their data is being stored or where their data request are processed because the grid storage.
- Grids, which can consist of multiple clusters and stand alone servers, are dynamic resource pools and they are shareable among many different applications and users.
- A grid does not assume that all servers in the grid are running the same set of applications.

Grid Database



Source: Grid and Cloud Database Management
Sandro Fiore Giovanni Aloisio
Springer-Verlag Berlin Heidelberg 2011

- The Fabric layer refers to the database resources, and it is characterized by a high level of heterogeneity with regard to the DBMS servers, the data models, the supported data formats, the available APIs, the security frameworks, the supported platforms, etc.
- The Data Access refers to the grid database access interface. According to the hourglass model, such an interface must provide a uniform and grid-enabled entry point to the underlying and heterogeneous database resources. From a security point of view, a data access service must provide support for the Grid Security Infrastructure

Grid Database

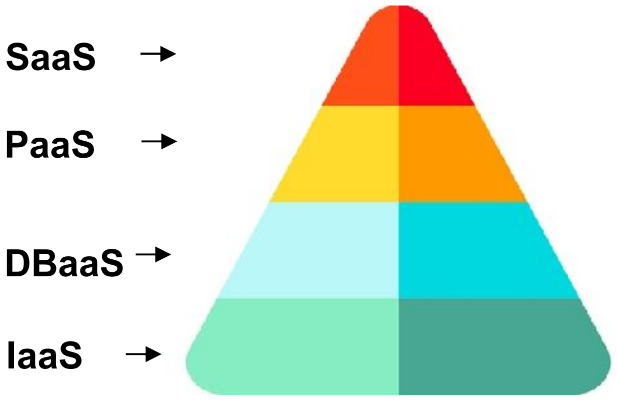
- The Management layer relates to a user-transparent level devoted to monitoring, management, and control functionalities. At this level, several metrics can be collected to check the status of the underlying data access services. ECA rules mechanisms can also be implemented at this layer along with global, advanced, and automatic detection tools or completely/semiautomatic diagnosis tools.

- Grids, which can consist of multiple clusters and stand alone servers, are dynamic resource pools and they are shareable among many different applications and users.
- A grid does not assume that all servers in the grid are running the same set of applications.

Cloud Database

The Cloud stack

- All computing platforms are built on the concept of a technology stack, where one layer of technology is built on another. In the Cloud computing arena, different categories of products bring Cloud computing to different levels of the stack.
- There are four main categories of Cloud computing solutions



- Infrastructure-as-a-Service (IaaS), which gives users access to infrastructure components on a Cloud platform, such as operating systems and other system software
- Database-as-a-Service (DBaaS), which gives users access to databases running on a Cloud computing platform
- Platform-as-a-Service (PaaS), which gives users access to development and deployment environments running on a Cloud platform
- Software-as-a-Service (SaaS), which gives users access to solutions running on a Cloud computing platform

Fuente: https://www.oracle.com/technetwork/database/database-cloud/public/oracle-db-and-db-cloud-service-wp-1844127.pdf

Cloud Database

Differences between Cloud computing categories

- Different categories of Cloud computing offer different sets of value propositions to customers. These value propositions can be differentiated based on some common principals
- A Cloud computing category describes the level at which users interact with the offering.
  - For a Database-as-a-Service offering (DBaaS), you interact with the database
  - For Infrastructure-as-a-Service (IaaS), you interact with the infrastructure software
- All technologies beneath the Cloud computing level are transparent, both in terms of maintenance and operations.
  - For Platform-as-a-Service (PaaS), you do not have to perform any maintenance operations on the database or system software.

Cloud Database

Differences between Cloud computing categories (Cont.)

- You are responsible for all layers of technology above the level of the Cloud computing category.
  - For IaaS, you would be responsible for the database and platform tools, both in terms of license acquisition and maintenance operations.
  - Because of this, Software-as-a-Service (SaaS) offerings have the lowest overall overhead, since the entire technology stack is within the Cloud, reducing overhead across the broadest range of operations.
- All technology beneath the Cloud computing level is inaccessible for configuration changes. For PaaS, you would not be able to configure the underlying database.

Cloud Database

Database Cloud attributes

Based on the distinctions described, there are three basic attributes of the Database Cloud that match those of a PaaS

- The overall operational effort required for the Data ase Cloud Service. The Database Cloud Service is a fully managed service, which does not require any operational effort for the underlying Oracle Database, as with a PaaS offering.
- The configuration options allowed with the Database Cloud Service. As with PaaS, you have very limited control over database instance level configuration parameters, including settings like the overall size of the SGA or whether to pin a particular table into cache
- The Database Cloud manages the settings to provide optimal performance for all tenants

Cloud Database

Database Cloud attributes (Cont)

- Gives you programmatic access to the underlying Database through SQL or PL/SQL, executed from inside the Database Cloud or through RESTful Web Services.
- This form of interaction is associated with PaaS.
- You do not have access to the Database Cloud Service over the Internet with standard SQL*Net connectivity.
- The Database Cloud Service is a public Cloud service – fully manages and available on a monthly subscription basis.
- The Database Cloud Service gives you everything you need to create Software-as-a-Service products, as well as making the data in your Database

Memory Database

- In-memory databases work faster than databases with disk storage. This is because they use “internal” optimization algorithms, which are simpler and faster, and this type of system requires fewer CPU instructions than a disk storage system.
- Additionally, accessing data that has been stored “in-memory” eliminates the need for seek time while searching for data.
- Traditionally, data has been stored on disk drives, with RAM used for short-term memory while the computer is in use. in-memory database architecture uses a database management system that relies primarily on a computer’s main memory (RAM), and is organized by an In-Memory Database Management System (IMDBMS). In-memory database (IMDB) architecture requires a management system designed to use the computer’s main memory as the primary location to store and access data, rather than a disk drive.

Memory Database

- Though in-memory database systems do have broad uses, they are used primarily for real-time applications requiring high performance technology. The use cases for these systems include applications for real-time responses, such as with the finance, defense, telecom, and intelligence industries. Applications requiring real-time data access such as streaming apps, call center apps, reservations apps, and travel apps also work well with IMDBMS.

- The two primary reasons in-memory databases have not historically been popular have to do with costs and a lack of ACID (atomicity, consistency, isolation, and durability

https://www.dataversity.net/in-memory-database-architecture-overview/

Memory Database

Durability

For full durability, they need supplementing with one of the following:

- Transaction logging, which records changes to the database in a journal file and facilitates automatic recovery of an in-memory database.
- Non-Volatile DIMM (NVDIMM), a memory module that has a DRAM interface, often combined with NAND flash for the Non-Volatile data security. The first NVDIMM solutions were designed with supercapacitors instead of batteries for the backup power source. With this storage, IMDB can resume securely from its state upon reboot.
- Non-volatile random access memory (NVRAM), usually in the form of static RAM backed up with battery power (battery RAM), or an electrically erasable programmable ROM (EEPROM). With this storage, the re-booting IMDB system can recover the data store from its last consistent state.

https://en.wikipedia.org/wiki/In-memory_database

Memory Database

Durability (Cont.)

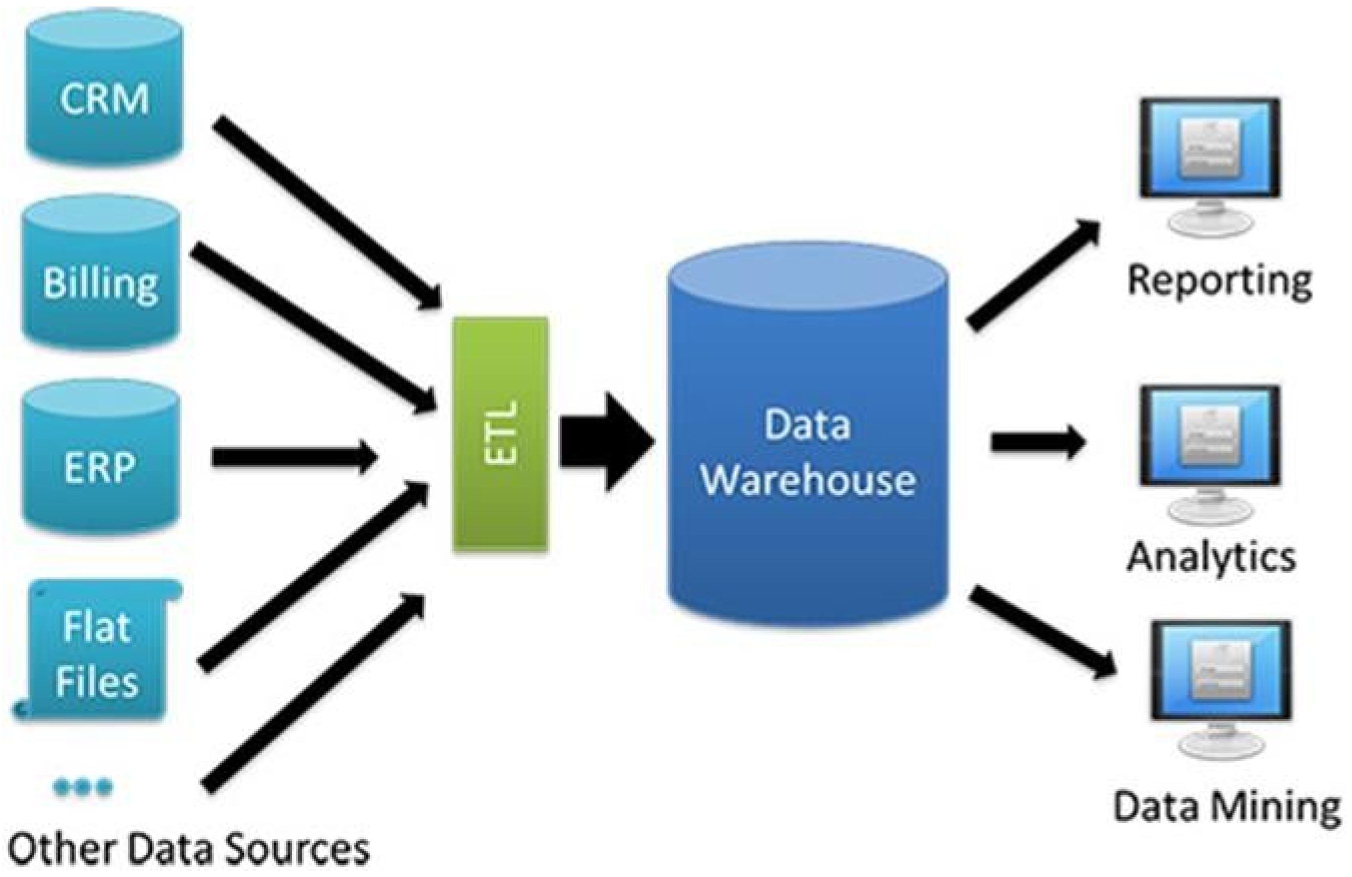
For full durability, they need supplementing with one of the following:

- High availability implementations that rely on database replication, with automatic failover to an identical standby database in the event of primary database failure. To protect against loss of data in the case of a complete system crash, replication of an IMDB is normally used in addition to one or more of the mechanisms listed above.

Databases according to purpose

- DataWarehouse
- Gis

Datawarehouse



Databases according to purpose

Enterprise Resource Planning ERP

Set of information systems that allow the integration of certain operations of a company, especially those that have to do with production, logistics, inventory, shipping and accounting.

Customer Relationship Management (CRM)

- Is an application that allows to centralize in a single Database all the interactions between a company and its clients.
- CRM software, by definition, allows you to share and maximize the knowledge of a given customer and thus understand and anticipate their needs. By definition, the CRM collects all the information of the commercial management keeping a detailed history.
- A CRM solution allows directing and managing customer acquisition and loyalty campaigns.
- Thanks to data mining and CRM, you can control the set of actions carried out on clients or potential clients, and manage commercial actions from a detailed dashboard.

Databases according to purpose

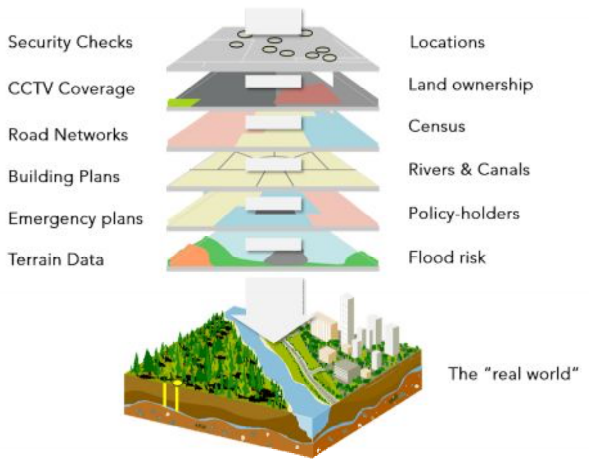
Online Analytical Processing OLAP

- A category of software tools which provide analysis of data for business decisions.
- OLAP systems allow users to analyze database information from multiple database systems at one time.
- The primary objective is data analysis and not data processing.

Online transaction processing OLTP

- supports transaction-oriented applications in a n-tier architecture.
- OLTP administers day to day transaction of an organization.
- The primary objective is data processing and not data analysis

Geographic information System GIS



A set tool for:

- Collecting
- Storing
- Manipulating
- Retrieving
- Transforming and display of spatial data from the real world

- Link databases and maps
- manage information about places
- help answer questions such as:
- Where is it?
- What else is near by?
- Where is the highest concentration of "X"
- Where can I find things with characteristics "Y

Geographic information System GIS

A set tool for:

- Collecting
- Storing
- Manipulating
- Retrieving
- Transforming and display of spatial data from the real world

End of Chapter