**Data Parallelism:**

Database parallelism in a data warehouse means splitting data processing tasks across multiple processors or machines to handle large datasets and complex queries faster and more efficiently.

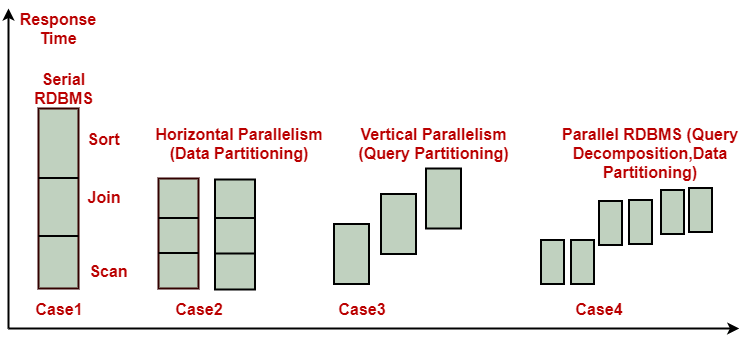
**Types of Database Parallelism:**

* Parallelism in databases speeds up query execution by using more resources and manages larger workloads without delays by increasing parallel processing.
* It is implemented using architectures like shared-memory, shared-disk, shared-nothing, and hierarchical structures.

**(a)Horizontal Parallelism:**

Horizontal parallelism in a data warehouse splits data rows across nodes to process the same task simultaneously, boosting performance.

**(b)Vertical Parallelism:**

Vertical parallelism in a data warehouse runs different tasks, like scanning or sorting, simultaneously to improve efficiency.

## **Intraquery Parallelism:**

• Defines execution of a single query in parallel on multiple processors and disks.  
• Essential for speeding up long-running queries.  
• DBMS vendors use intraquery parallelism to improve performance.  
• Decomposes serial SQL query into lower-level operations like scan, join, sort, and aggregation.  
• Lower-level operations are executed concurrently in parallel.

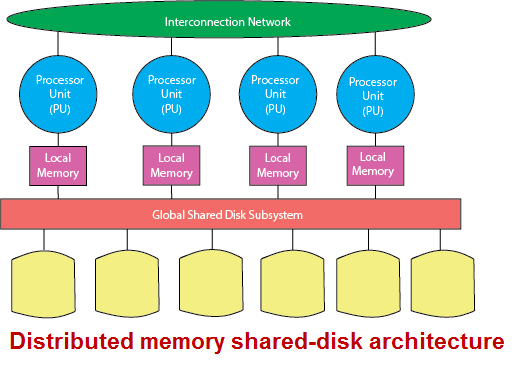
## **Interquery Parallelism:**

• Interquery parallelism allows multiple queries or transactions to execute in parallel.

• Database vendors use parallel hardware architectures to handle large client requests efficiently.  
• Successful implementation on SMP systems increases throughput and supports more concurrent users.

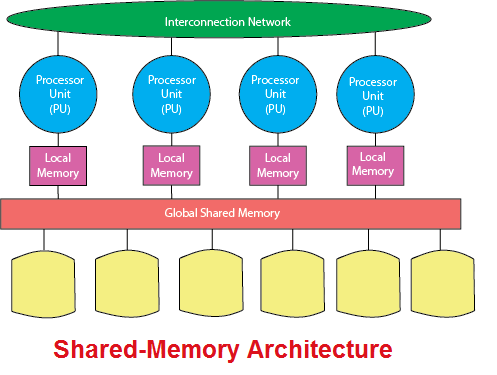
## **Shared Disk Architecture:**

• Implements shared ownership of the entire database between RDBMS servers.  
• Each server can read, write, update, and delete information from the same shared database.  
• DLM components can be found in hardware, operating system, and separate software layer.  
• Reduces performance bottlenecks from data skew and increases system availability.  
• Eliminates memory access bottleneck of large SMP systems and reduces DBMS dependency on data partitioning.



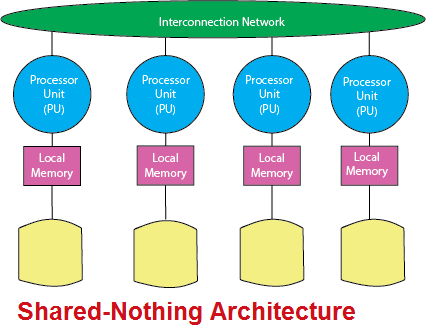
## **Shared-Memory Architecture:**

Shared-Memory RDBMS Implementation  
• Traditional RDBMS implementation on SMP hardware.  
• Simple to implement, but faces scalability limitations.  
• Single RDBMS server can apply all processors, access all memory, and the entire database.  
• Multiple database components communicate via shared memory.  
• All processors have access to all data partitioned across local disks.



## **Shared-Nothing Architecture:**

• Data partitioned across all disks.  
• DBMS partitioned across multiple co-servers.  
• Each node owns its disk and database partition.  
• Parallelizes SQL query execution across multiple processing nodes.  
• Each processor communicates with other processors via interconnection network.  
• Optimized for Multi-Process-Performer-Node (MPP) and cluster systems.  
• Offers near-linear scalability, with each node capable of being a powerful SMP system.



### **Application of Data Parallelism:**

* **Query Processing:** Parallel execution of queries on large datasets to improve performance.
* **Data Aggregation:** Distributing data across nodes to perform aggregations simultaneously.
* **ETL Processes:** Dividing ETL tasks (Extract, Transform, Load) into smaller, parallelizable units.
* **Indexing and Searching:** Splitting indexing tasks to quickly process large volumes of data.

#### **Advantages:**

1. **Improved Performance:** Faster query execution by processing data in parallel.
2. **Scalability:** Efficiently handles large volumes of data as workloads can be distributed.
3. **Better Resource Utilization:** Makes full use of available CPU, memory, and disk resources.
4. **Reduced Processing Time:** Divides tasks into smaller units, significantly reducing overall processing time.

#### **Disadvantages:**

1. **Complexity in Data Distribution:** Proper partitioning and managing data across nodes can be complex.
2. **Overhead for Small Tasks:** For small datasets, the overhead of managing parallelism may outweigh the benefits.
3. **Data Skew Issues:** Uneven data distribution can lead to performance bottlenecks.
4. **Resource Contention:** Multiple processes may compete for limited resources, potentially causing delays.