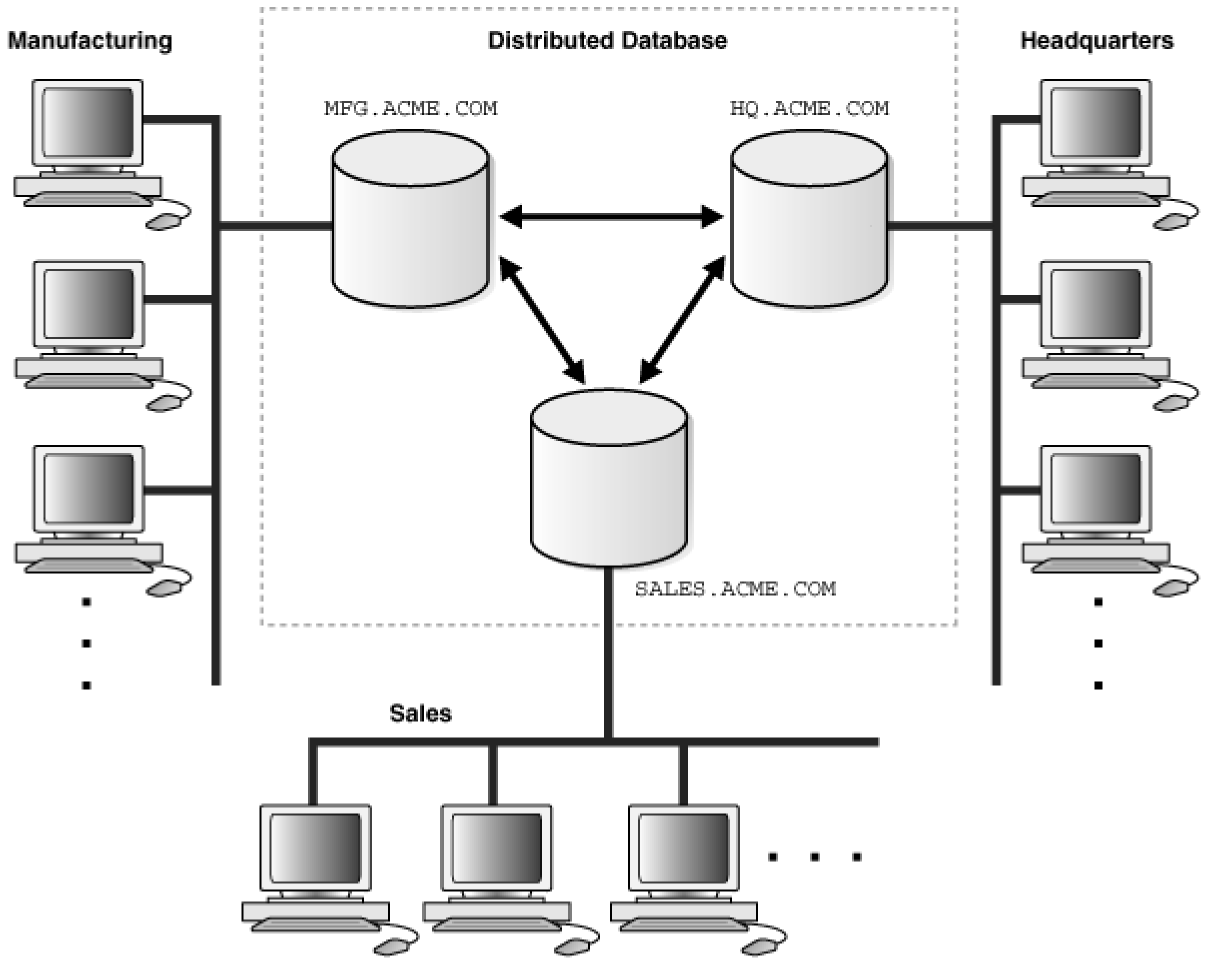
Introduction to distributed Database

Ram Datta Bhatta

Distributed Database

- ✓A distributed database is a collection of interconnected databases that are spread across multiple physical or geographical locations but function as a single, unified database system.
- ✓ Distributed database is designed to store, manage, and process large volumes of data in a distributed and decentralized manner, offering numerous benefits over traditional centralized databases.
- ✓ADVANCED DATABASE CONCEPTS- PART 7 (DISTRIBUTED DATABASES-BASICS) - YouTube

Distributed Database System



Characteristics of DDBS

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

Distributed Database Design

- ✓Data Fragmentation
- ✓Data Replication

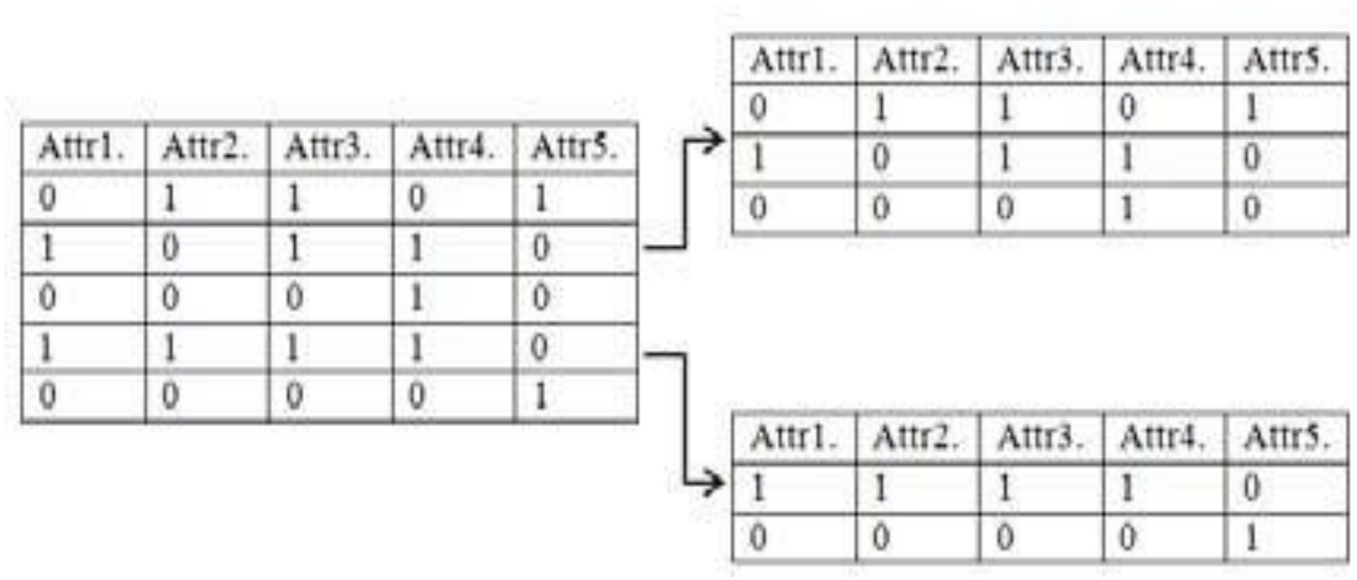
Data Fragmentation

- ✓Fragmentation is a process of dividing the whole or full database into various sub tables or sub relations so that data can be stored in different systems.
- ✓The small pieces of sub relations or sub tables are called *fragments*.
- ✓Division of relation r into fragments r1, r2, …, rn which contain sufficient information to reconstruct relation r.
- ✓There are two types of data fragmentation: Horizontal and Vertical Fragmentation

Horizontal fragmentation

✓It refers to the division of a relation into subsets (fragments) of tuples (rows).

✓Each fragment is stored at a different node, and each fragment has unique rows. However, the unique rows all have the same attributes (columns).



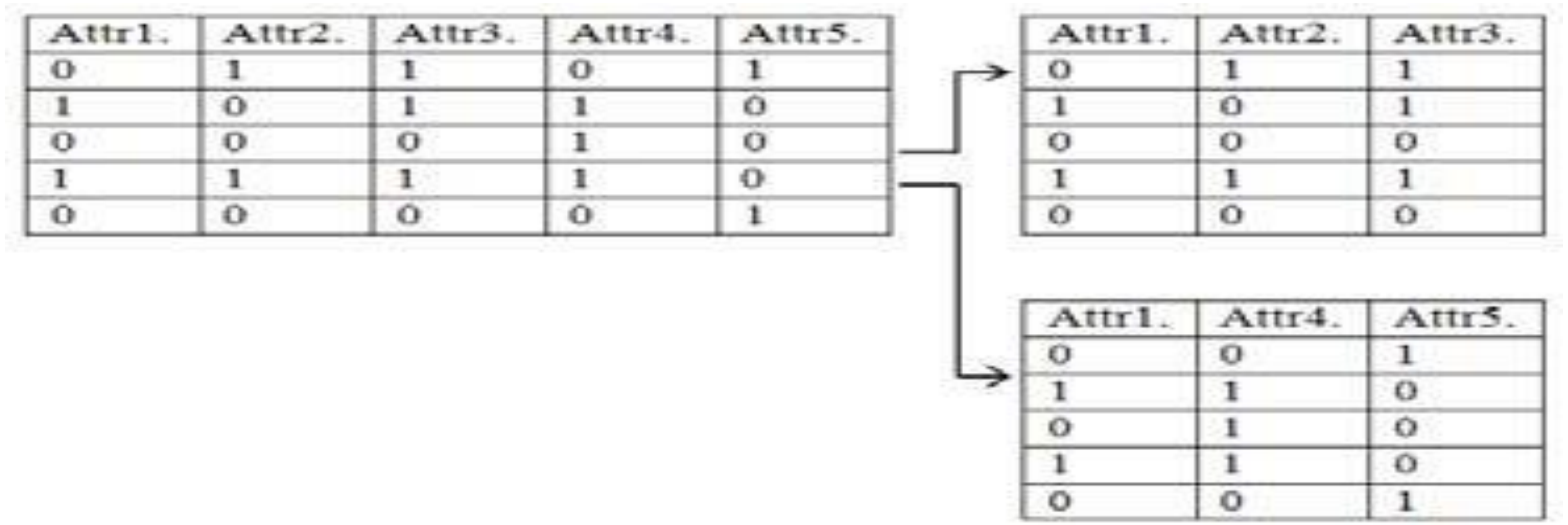
Attr1.	Attr2.	Attr3.	Attr4.	Attr5.
0	1	1	0	1
1	0	1	1	0
0	0	0	1	0
1	1	1	1	0
0	0	0	0	1

Attr1.	Attr2.	Attr3.	Attr4.	Attr5.
0	1	1	0	1
1	0	1	1	0
0	0	0	1	0

Attr1.	Attr2.	Attr3.	Attr4.	Attr5.
1	1	1	1	0
0	0	0	0	1

Vertical Fragmentation

- ✓Vertical fragmentation refers to the process of decomposing a table vertically by attributes are columns.
- ✓ Some of the attributes are stored in one system and the rest are stored in other systems.
- ✓Each site may not need all columns of a table. In order to take care of restoration, each fragment must contain the primary key field(s) in a table.



Attr1.	Attr2.	Attr3.	Attr4.	Attr5.
0	1	1	0	1
1	0	1	1	0
0	0	0	1	0
1	1	1	1	0
0	0	0	0	1

Attr1.	Attr2.	Attr3.
0	1	1
1	0	1
0	0	0
1	1	1
0	0	0

Attr1.	Attr4.	Attr5.
0	0	1
1	1	0
0	1	0
1	1	0
0	0	1

Advantages of Fragmentation

Horizontal:

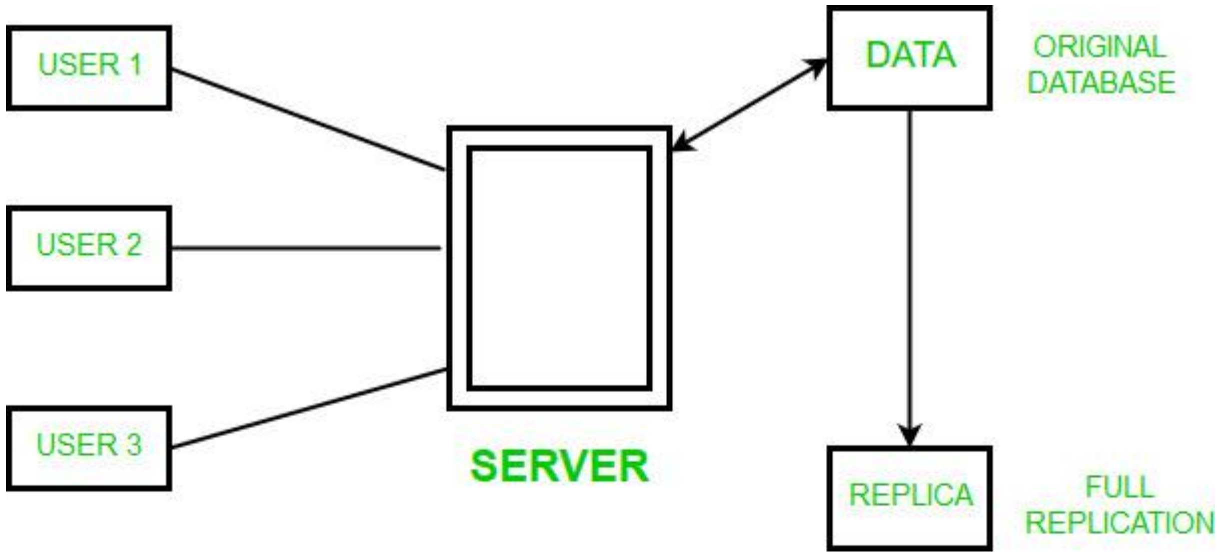
- ✓allows parallel processing on fragments of a relation
- ✓allows a relation to be split so that tuples are located where they are most frequently accessed

Vertical:

- ✓allows tuples to be split so that each part of the tuple is stored where it is most frequently accessed
- ✓tuple-id attribute allows efficient joining of vertical fragments
- ✓allows parallel processing on a relation

Data Replication

- ✓A relation or fragment of a relation is **replicated** if it is stored redundantly in two or more sites.
- ✓**Full replication** of a relation is the case where the relation is stored at all sites.
- ✓Fully redundant databases are those in which every site contains a copy of the entire database.



Data Replication

- ✓ It is simply copying data from a database from one server to another server so that all the users can share the same data without any inconsistency.
- ✓The result is a **distributed database** in which users can access data relevant to their tasks without interfering with the work of others.
- ✓The choice of sites and the degree of replication depend on the performance and availability goals of the system and on the types and frequencies of transactions submitted at each site.

Advantages of Distributed Database

- ✓**Scalability**: As the data is distributed across multiple nodes, a distributed database can easily scale by adding more nodes to handle increased data loads.
- ✓**Fault Tolerance:** Distributed databases can tolerate node failures without losing data or compromising data availability. Replication and data distribution help maintain data integrity in the event of hardware failures.
- ✓**High Availability:** With data replication and redundancy, distributed databases offer high availability, ensuring that data remains accessible even if some nodes are offline or unavailable.
- ✓**Geographical Distribution**: Distributed databases can span multiple geographic locations, allowing for reduced latency and better data access for users located in different regions.
- ✓**Load Balancing:** Distributing data across nodes enables load balancing, ensuring that no single node becomes overwhelmed with excessive read or write requests

Challenges of Distributed Database

- ✓**Data Consistency:** Achieving strong data consistency in a distributed environment is complex and may require trading off consistency for availability (CAP theorem).
- ✓**Network Latency:** Communication between distributed nodes may introduce latency, affecting query performance and response times.
- ✓**Distributed Transactions:** Coordinating transactions that involve multiple nodes can be challenging to maintain ACID properties.
- ✓**Security and Privacy**: Protecting data in a distributed setup requires robust security measures to prevent unauthorized access or data breaches

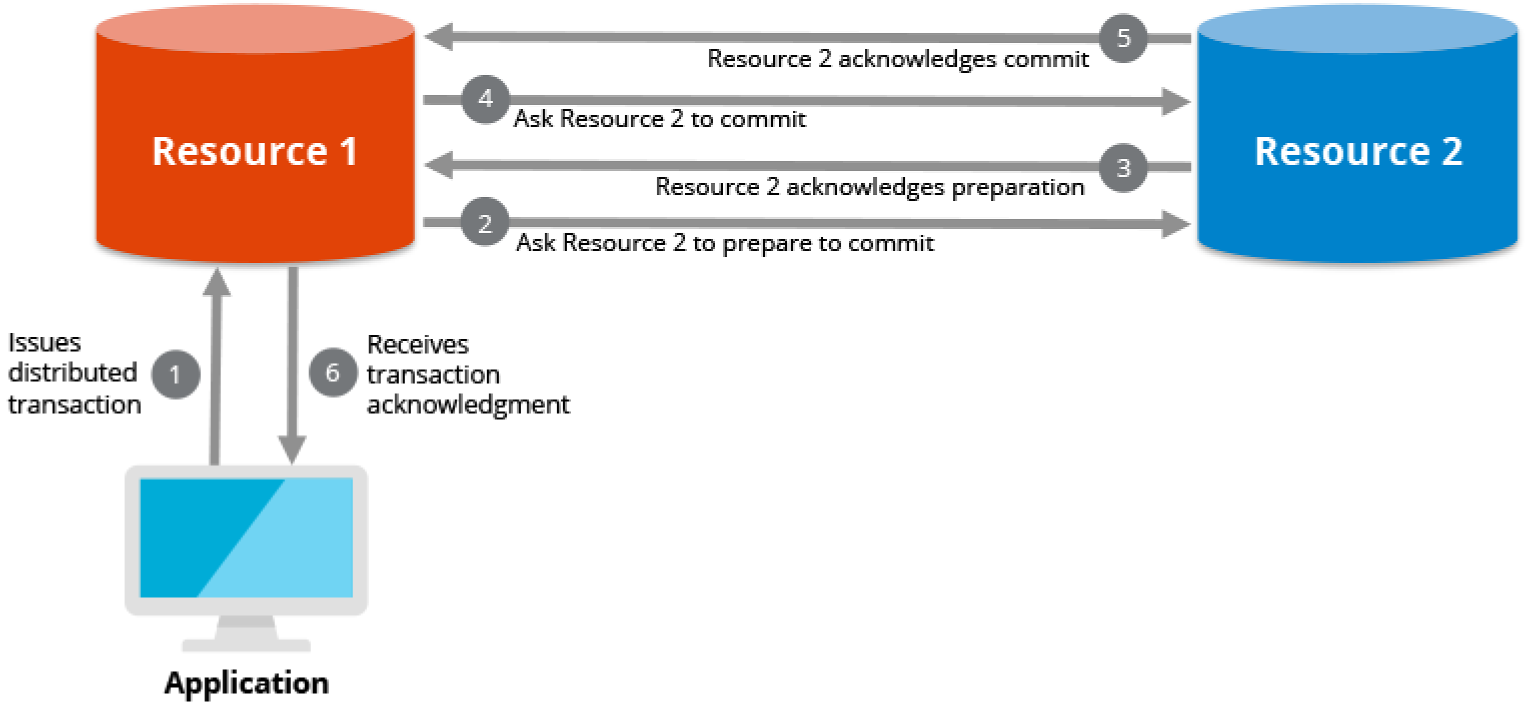
Distributed Transaction

- ✓A distributed transaction is a transaction that involves multiple independent components, each residing in a separate location or node within a distributed database
- ✓The main characteristic of a distributed transaction is that it spans across multiple nodes and must be executed atomically, ensuring that either all its component operations are completed successfully, or none of them are executed at all.
- ✓A distributed transaction is a set of operations on data that is performed across two or more databases. It is typically coordinated across separate nodes connected by a network, but may also span multiple databases on a single server.

How does it work?

✓There are two possible outcomes: 1) all operations successfully complete, or 2) none of the operations are performed at all due to a failure somewhere in the system.

✓The operation known as a “two-phase commit” (2PC) is a form of a distributed transaction.


Resource 1
Resource 2
5 Resource 2 acknowledges commit
4 Ask Resource 2 to commit
3 Resource 2 acknowledges preparation
2 Ask Resource 2 to prepare to commit
Issues distributed transaction 1
6 Receives transaction acknowledgment
Application

https://hazelcast.com/glossary/distributed-transaction/

Concurrency Control in Distributed Database

- ✓Concurrency control in a distributed database refers to the techniques and mechanisms used to manage multiple simultaneous transactions accessing and modifying data across different nodes in the distributed database system.
- ✓The goal of concurrency control is to ensure that transactions are executed in a correct and isolated manner, avoiding conflicts and maintaining data consistency.
- ✓In a distributed database, there are multiple copies of data distributed across different nodes. Transactions can be initiated at any node and may need to access and modify data from multiple nodes. This introduces the potential for conflicts and data inconsistencies when multiple transactions access the same data concurrently
- ✓Various approaches such as Lock-based and Timestamp-based concurrency control can be used.

Lock-based Concurrency Control

- ✓The lock-based concurrency control involves the use of locks to control access to shared resources (data items) so that only one transaction can access a particular resource at a time.
- ✓When a transaction wants to access a data item, it requests a lock on that item. Locks can be either shared locks (read locks) or exclusive locks (write locks).
- ✓A shared lock allows multiple transactions to read the data item simultaneously, while an exclusive lock ensures that only one transaction can modify the data item at a time
- ✓Two-Phase Locking (2PL) which is an extension of lock-based concurrency control is also used.

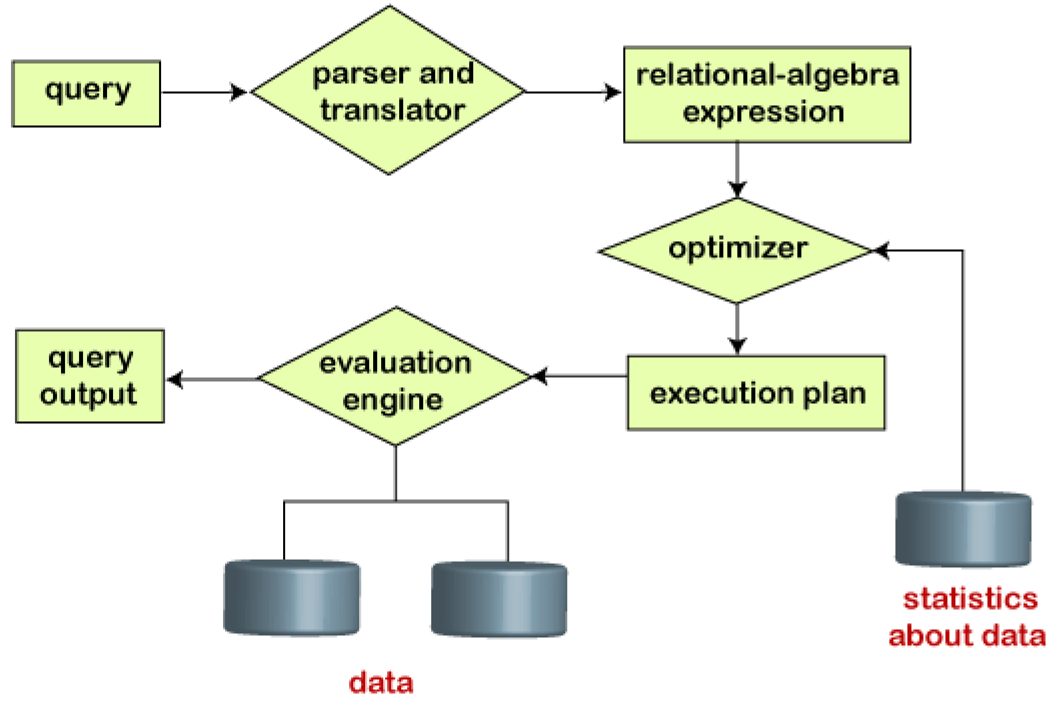
Timestamp-based Concurrency Control

In this approach, each transaction is assigned a unique timestamp. Whenever a transaction wants to read or write a data item, it must first obtain the timestamp of the last successful transaction that accessed that item. If the current transaction's timestamp is older than the previous one, it means the data is already modified, and the current transaction's request is rejected or delayed.

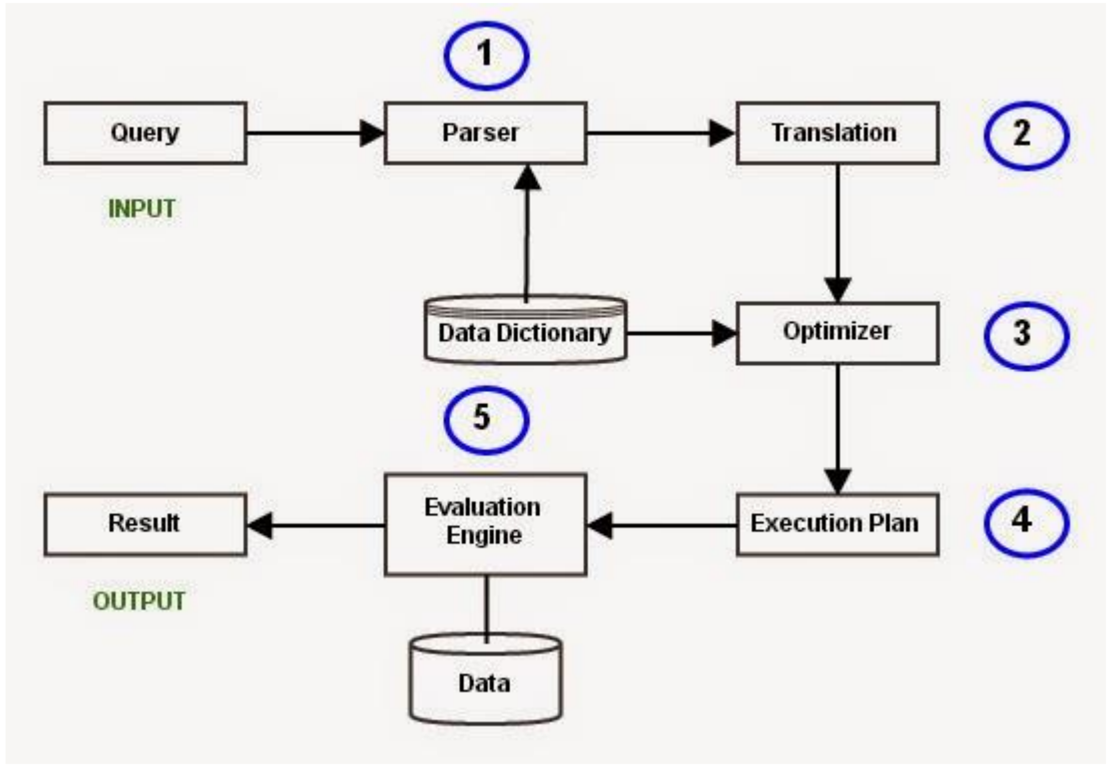
Query Processing in distributed database

- ✓Query processing in distributed databases involves the tasks and processes associated with handling user queries in a distributed environment, where data is stored across multiple interconnected nodes or servers,

Query Processing in Non-Distributed Environment



Steps in query processing



Steps in query processing (Distributed Database)

1. **Query Parsing and Optimization**: When a user submits a query, it first goes through parsing and optimization. This step is similar to centralized database.
2. **Query Distribution**: The query needs to be distributed to the appropriate nodes that store the relevant data. This involves determining which nodes need to be involved in processing the query and dividing the query into subqueries to be executed on different nodes.
3. **Local Processing**: Each distributed node processes its respective subquery using local data. This involves accessing local data and applying the necessary operations. Nodes operate independently and in parallel.

4. **Data Exchange and Communication:** In distributed databases, data might need to be exchanged between nodes to complete query processing. This involves transferring data between nodes over the network.
5. **Intermediate Result Integration:** After local processing, the intermediate results from different nodes need to be integrated to generate the final query result. This could involve additional processing, such as aggregating partial results, performing additional joins, or applying sorting and grouping.
6. **Result Presentation:** Once the final result is generated, it is presented to the user. This may involve formatting the data in a user-friendly way and returning the result to the user's application or interface.

Thanks