Unit - 5

Distributed Database Management Systems

DDB - a collection of multiply logically related databases distributed over a computer network.

D-DBMS - a software system that manages a distributed database, while making the distribution transparent to the user.

* Conditions to be a Distributed Database

(i) connection of database nodes over a computer network

(ii) logical interrelation of the connected databases

(iii) possible absence of homogenity among connected nodes

Note that the sites may have physical proximity or may be geographically distributed

* Characteristics :

(1) Transparency

A. Distribution and Network Transparency

→ Users do not have to worry about operational details and placement of data in the network

Location Transparency - freedom of issuing command from any location

Naming Transparency - allows access to any names object from any location without additional specification

B. Replication Transparency

→ store copies of data at multiple sites, minimizes access time

→ better availability, performance & reliability

C. Fragmentation Transparency

→ allows to fragment a relation horizontally or vertically

→ fragmentation transparency makes the user unaware of the existence of fragments.

D. Design Transparency & Execution Transparency

→ how the db is designed and where a transaction executes

② Increased Reliability and Availability

Reliability - the probability that a system will fail in a given period

→ a distributed system is considered reliable if it keeps delivering its services even when one or several of its software / hw components fail.

Availability – probability that the system is continuously available during a time interval

Fault Tolerance: D-DBMS must recognize that faults will occur & design mechanisms to detect or remove faults before they can result in an error.

(3) Scalability and Partition Tolerance

Scalability – determines the extent to which a system can expand its capacity while continuing to operate without interruption.

A. Horizontal Scalability – expanding the no. of nodes in the distributed system, should distribute some data & loads from existing nodes to the new nodes

B. Vertical Scalability – expand the capacity of the individual nodes in the system, such as expanding the storage/processing power of a node.

Partition Tolerance – Faults cause nodes to be partitioned into groups of nodes.

→ Partition Tolerance states that the system should have the capacity to continue operating while the network is partitioned.

④ Improved Performance

→ fragments database to keep data closer to where it is needed most

→ reduces data management time significantly

→ easier expansion

→ can add new nodes anytime

* Distributed Data Storage

2 approaches - replication or fragmentation

1. Replication - system maintains multiple copies of data, stored in different sites for faster retrieval & fault tolerance

2. Fragmentation - partition relation into several fragments and store in distinct sites

3. Combination of replication & fragmentation - system maintains several identical replicas of each such fragment

Data Allocation - each fragment - or each copy of a fragment must be assigned to a particular location in the distributed system.

choice of sites and degree of replication depends on the performance & availability goals of the system & the types & frequencies of transactions submitted at each site.

* Data Replication

Adv

(i) Availability
(ii) Parallelism
(iii) Reduced data transfer

Disadv

(i) increased cost of updates
(ii) increased complexity of concurrency control

* Data Fragmentation

→ can be horizontal or vertical

Reasons for fragmentation

→ Every relation is not a suitable unit for distribution, since each application views a subset of its relations.

→ Fragmentation ⇒ parallel execution of a single query by dividing it into a set of subqueries that operate on fragments

→ increases the level of concurrency (intraquery concurrency) and thereby, the throughput

A. Horizontal Fragmentation

horizontal subset of a relation satisfying selection conditions

derived horizontal fragmentation - partitioning of a primary key relation to other secondary relations related w/ foreign keys

→ specified by σ in relational algebra
→ reconstruct using ∪

B. Vertical Fragmentation

→ a subset of columns
→ no selection condition used
→ specified by π in relational algebra
→ reconstruct using natural join.

* Correctness Rule for Fragmentation

(i) Reconstruction - If a relation R is decomposed into fragments $R_1, R_2 \ldots R_n$, it should be possible to define a relational operator Δ such that $R = \Delta R_i$

(ii) Disjointness -

Horizontal fragmentation - If a relation R is horizontally decomposed into $R_1, R_2 \ldots R_n$ and a data item d_i is in R_j, it is not in any other fragment R_k.

Vertical fragmentation - If a relation R is vertically decomposed, its primary key attributes are typically repeated in all the fragments

* Types of Distributed Database Systems

(i) Homogenous - all sites of database system have identical setup

→ underlying OS may be different

(may be a mixture of Linux, Windows)

(ii) Heterogenous

Federated - each site runs a different database system, but the access is managed through a single conceptual schema.

Multidatabase - there is no conceptual global schema, the schema is constructed dynamically as needed by the application software.

Unit-5

NoSQL

* NoSQL → Not Only SQL or NO to SQL

→ focuses on:

(i) semistructured data storage

(ii) high performance

(iii) availability

(iv) data replication

(v) scalability

→ NOSQL systems are used for social media, posts and tweets, weblinks, spatial data etc

* When should NoSQL be used?

→ huge amount of data

→ relationship between data is not that important

→ data changes over time and is not structured

→ constraints and joins not required at the database level

→ database is growing continuously

* Characteristics of NoSQL databases

① High Availability

② Scalability : → determines the extent to which the system can expand its capacity without interruption

(write about horizontal & vertical scalability)

(3) Replication

→ data is replicated over 2 or more nodes, if one node fails, the data is still available on other nodes

→ improves data availability & can improve read performance

→ Eventual consistency - if no updates take place for a long time, all replicas will gradually and eventually become consistent.

Replication models

(i) Master-Slave Replication: one copy is the master or the primary copy, all write operations must be applied to the master copy and propagated to the slave copies, usually using eventual consistency.

→ Master handles writes and slaves synchronize with master.

Adv

(i) more read requests
(ii) if master fails slaves can still handle read requests
(iii) good for datasets w/ a read-intensive dataset

Disadv

→ master is a bottleneck
→ failure eliminates ability to write
→ inconsistency due to slow propagation of changes to slaves
→ bad for datasets with heavy write traffic.

(ii) Master - Master Replication:

→ allows reads and writes at any replicas

→ all members are responsive to client data queries

→ replication allows write to any node

Adv

→ node failures would not affect access to data

→ can easily add nodes

Disadv

→ Inconsistency

→ slow propagation of changes to copies on different nodes

→ Two people can update different copies of the same record at the same time (write - write conflict)

→ Inconsistent writes are forever.

(4) Sharding of Files

→ files can have millions of records

→ cannot store whole file in one node

→ deploy horizontal partitioning

(5) High - Performance Data Access

- (i) hashing
- (ii) range partitioning - location determined via a range of key values

⑥ Does not require a schema

→ semi structured and self-describing data of NOSQL systems does not require a schema.

→ partial schema used to improve storage efficiency - JSON

⑦ less Powerful Query Languages

→ functions and operations as an API

→ reading and writing accomplished by calling appropriate operations

CRUD operations

↙ ↓ ↘ →

create read update delete

SCRUD

↓

search

⑧ Versioning

→ provides storage of multiple versions of data items w/ timestamps

* Types of NOSQL Systems

(i) Document - based

(ii) NOSQL - key-value stores

(iii) Column based

(iv) Graph based

* CAP Theorem

→ refers to 3 desirable properties

(i) Consistency - nodes have same copies of replicated data

(ii) Availability - guarantees that every request receives a response (successful or failed)

(iii) Partition Tolerance - system continues to operate despite communication breakages.

→ The CAP theorem states that it is not possible to guarantee all 3 of the desirable properties at the same time

Classification based on CAP characteristics

(i) CP database - consistency & partition tolerance at the cost of availability.

→ if partition occurs, system shuts down the non-consistent node until the partition is resolved. eg. BigTable
MongoDB
Berkeley DB

(ii) AP database

→ if partition occurs, nodes remain available, but those at the wrong end of a partition might get an older version
eg. Voldemort, Kai

(iii) CA database - if there is partition, cannot deliver fault tolerance
eg. Vertica
Greenplum

Document-based Database Systems – MongoDB

→ A mongodb database may have one or more **collections**

group of MongoDB documents
equiv to a RDBMS' table
all documents in a collection are related.

→ A collection may have zero or more **documents**

a set of key value pairs
have a dynamic schema
equivalent to a row/tuple in an RDBMS

→ need not have the same fields

→ documents are addressed by a unique key → provided by MongoDB itself
16-bit field automatically indexed

→ Documents are stored in JSON format

Functions

db.createCollection

Insertion: db.mycol.insert ( )

db.<collectionname>.remove (<condn>)
say: {title:}

Characteristics

(i) uses 2-phase commit protocol

(ii) replication - master-slave approach
arbiter

(iii) Sharding

(iv) Accessing: hashing / range partitioning