Introduction to non-centralized DBMSs

ACID Properties

Atomicity

A transaction is an atomic unit of work or collection of separate operations. So, a transaction succeeds and is committed to the database only when all the separate operations succed. If any single operations fail during the transaction, everything will be considered as failed and must berolled back if it is already taken place. Atomicity helps to avoid data inconsistencies in database by eliminating the chance of processing a part of operations only.

Consistency

A transaction must leave the database into a consistent state whether or not it is completed successfully. The data modified by the transaction must comply with all the constraints in order to mantain integrity.

Isolation

Every transaction has a well defined boundary. One transaction will never affect another transaction running at the same time. Data modifications made by one transaction must be isolated from the data modification made by all other transactions. Under any circumstance a transaction can not be in any intermediate state.

Durability

If a transaction succeds, the updates are stored in permanent media even if the database crashes immediately after the application performs a commit operation. Logs are mantained so the database can be restored to its original position before failure took place.

DB2 Memory Architectures

Symmetric Multiprocessor system (SMP)

Several equally powerful processors within the same machine, resources, such as disk space and memory, are shared. Executing a query, loading data, backing up and restoring table spaces, and creating indexes on existing data, can take advantage of multiple processors.

Intensive write/update workload. I/O can scale increasing the number of disks.

Shared Nothing Clusters (MPP)

Many database partitions. Each database partition resides on its own machine, and has its own processor, memory, and disks. This environment is referred to by many different names, including: cluster, cluster of uniprocessors, massively parallel processing (MPP) environment, and shared-nothing configuration.

A partitioned database environment allows a database to remain a logical whole, despite being physically divided across more than one database partition. The fact that data is distributed remains transparent to most users.

Intensive read workload. Scales up and out.

Shared Disk Cluster

Highest level on SQL execution concurrency. Continuos availability and scalability as main goal.

80% read, 20% write workload. A shared disk, multiple DB2 instances. Scales up and out.

Non distributed databases

Federated database system

Practitioners define a Federated Database as a *collection of cooperating component systems which are autonomous and are possibly heterogeneous.*

A federated database system is a type of meta-database management system (DBMS), which transparently maps multiple autonomous database systems into a single federated database. The constituent databases are interconnected via a computer network and may be geographically decentralized. Since the constituent database systems remain autonomous, a federated database system is a contrastable alternative to the (sometimes daunting) task of merging several disparate databases. A federated database, or virtual database, is a composite of all constituent databases in a federated database system. There is no actual data integration in the constituent disparate databases as a result of data federation.

Through data abstraction, federated database systems can provide a uniform user interface, enabling users and clients to store and retrieve data from multiple noncontiguous databases with a single query—**even if the constituent databases are heterogeneous**. To this end, a federated database system must be able to decompose the query into subqueries for submission to the relevant constituent DBMSs, after which the system must composite the result sets of the subqueries. Because various database management systems employ different query languages, federated database systems can apply wrappers to the subqueries to translate them into the appropriate query languages.

The three important components of an FDBS are:

- autonomy,
- heterogeneity,
- distribution.

Can be extended to allow access to non-relational data sources. Has performance impact and query optimization is needed.

Applications connect to a single "virtualized database" and the federation is transparent.

Database replication

Database replication is a technique through which an instance of a database is exactly copied to, transferred to or integrated with another location. Database replication enables the copying of a database file from a master database management system (DBMS) and its exact deployment on a slave DBMS.

Database replication is primarily used in distributed DBMS environments where a single database is deployed, used and updated at several simultaneous locations. Database replication is generally performed frequently in a transactional database that is routinely and dynamically updated. Typically, database replication is done to provide a consistent copy of data across all the database nodes. It also removes any data redundancy, merging of two databases into one and updating slave databases with outdated or incomplete data.

Database replication has three distinct types:

- Transactional replication;

- Snapshot replication;
- Merge replication.

Often between heterogeneous DBMS. Topological variants: 1-1, 1-many, many-1, fan-out.

Has perfomance impact on currently running applications. Data can be replicated as-is or transformed in flight. Can be scheduled or real time. The event can be triggered by events or log-watching.

Event publishing

A *variant* of replication, events are triggered to send the recently changed data to external applications (new, updated or deleted records).

Can be in real time, maybe a component of the Enterprise Application Integration.

The events/data are emitted in a standard data format (e.g. JSON, XML).

ETL

An *extension* of the replication model:

- *Extract* data from data sources
- *Transform* it in a format suitable to be used in the DW environment
- *Load* data into the DW DBMS..

Variants: Extract - Load - Transform, Transform - Extract - Load.

Sources:

- IBM - Database partition and processor environments