Week 6 – Managing large databases – Part 2

Partitioning

- Dividing the storage of very large tables or indexes into database partitions.
- Horizontal Partitioning: Distributes the rows of the tables across multiple partitions. For example, partitioning by date separates recent data from old data. The recent data partitions is used for insert/update/deletes/index rebuilds while the old data is used primarily for selects.
    - Partitioning column – the column that a partition function uses to partition the data in the table.
    - Partitioning techniques:
        - Range partitioning – range of values are grouped in the partitions. For example, a retailer's sales data table partitioned by date.
        - List partitioning – range of pre-selected values are grouped in partitions, e.g. East region sales, West region sales, etc.
        - Hash partitioning – key that hash to the same values end up in same partitions, beneficial for equal distribution of data across partitions, typically used when there's no obvious key, like date, as an obvious partitioning key.
    - Indexes may be partitioned using a separate technique than the table or linked to the table's partitioning method.
- Vertical Partitioning: Distributes the columns of a table across partitions. The primary key ends up being duplicated to go with the groups of columns so they can be joined back together.
- When should you partition a table:
    - Table size is very large, e.g. greater than 2GB
    - Table containing historical data as well as recent data. New data can be added to the newest partition. For example, a retailer's sales data.
- When should you partition an index:
    - When you have a very large index on a very large table.
    - When you want to avoid rebuilding the entire index every time data is modified/deleted, i.e. the DBMS would rebuild the index for the partition that is impacted.
    - When you want to perform maintenance on parts of the data without invalidating the entire index.
- Partitioning Advantages:
    - Enhanced performance:
        - Of range queries. For example, queries or stored procedure operating on sales data could prune the data subject for scans to one particular partition.
        - By allowing for parallel execution, i.e. allowing the DBMS to split and parallel the query across multiple partitions then union the result.
        - By providing more options for the query optimizer, e.g. partition join
        - By allowing the capability of distributing the partitioned data onto different disks that can be accessed in parallel.

        - By enhancing the bulk load speeds of data, e.g. from an OLTP into a DW. Loading data into a particular partition of a smaller size is faster than operating on a very large single table. The load speed could improve into seconds vs minutes and hours.
    - Improved availability – applicable only if partitions are spread across different secondary storage. If the storage containing one partition fails, the others can keep going.
    - Improved manageability/maintenance - DBAs can manage smaller partitions instead of one very large table. The maintenance tasks are faster because they are more targeted.
        - Can backup/restore specific partitions
        - Can bulk load new data into one partition or a brand new partition. For example, a DW data load could load every batch (e.g. a week worth) into a new partition.
        - When data compression is used, can compress specific partitions.
        - Can rebuild indexes faster.
- Partitioning Disadvantages:
    - Could impact performance if the user queries end up joining data across partitions often
    - Vertical partitioning involves the duplication of the primary key column(s).

Parallel Database

Parallel databases run across multiple processes and/or multiple disks – possibly across one or multiple machines- in order to parallel operations (such as CPU cycles and/or disk reads) *for the purpose of improving performance*. It seeks to improve the database performance through parallelization of various operations such as data loading, index building, and query evaluation. There are 3 types of Parallel Databases:

1. Shared Memory / Shared everything (shares the memory only so multiple disks pooled together for sharing and multiple processors pooled together for sharing) – also known as SMP: Symmetric Multiprocessing. This architecture does not involve multiple nodes typically so one copy of the OS controls the system, but multiple SMPs could be joined together.
2. Shared disk (meaning sharing the disk only so multiple CPU/Memory nodes). It involves multiple machines/servers, each with its own memory and CPU. A load balancer is required to spread the load across the nodes. Since the nodes are sharing the same disk, no need to partition the data. It is typically used for database cluster implementations.
3. Shared nothing (each unit has its own CPU/Memory/Disk) – also known as MPP (Massively Parallel Processing). The data is partitioned across the disks. It is typically used for Data Warehouses.

Database Cluster

- Various database servers are physically close to each other in the same data center. *Oracle clustering allows for the nodes to be separated across floors, buildings, or even the city.*
- Provides location transparency. Database looks like one instance to the application.
- Not intended to improve performance, but instead the goal is to achieve high availability.
- Could be considered a form of DDBMS because the data is distributed across multiple nodes.

Distributed Databases

- Distributed databases (DDBMS) – a database system that consists of:
    - Databases (DBMS & data) that are physically stored across multiple hardware servers.
    - The hardware servers are connected by a network which may or may not be a high speed network.
    - Possibly on different hardware platforms.
    - They could be separated geographically
    - Could be on different database architectures, and/or from different vendors and different types of databases
    - The data on the various hardware components is managed by a separate DBMS
    - The logically related data is distributed across various number of database fragments.
    - Local apps operate independently using the local DBMS, but must have at least one global application. A global app may not necessarily have a local DBMS. A site may host a global application only without any local DBMS underneath.
    - The distribution of the data across multiple databases should be transparent to the DDBMS user. The DDBMS should handle all the complexities associated with the transparency.
- A homogeneous DDBMS consist of DBMS systems that are the same while a Heterogeneous DDBMS consist of sites that have different schemas and different software.
- Distributed Transactions – a bundle of operations that manipulate data on two or multiple networked computers.
- Two-phase commit – a standardized protocol that ensures the integrity of the transactions in a distributed database environment. A coordinator process performs the following steps:
    - Phase 1: requests that each database perform the write operation. If any of the databases return an error, the transaction rolls back.
    - Phase 2: requests that each database commit its transaction. If any of the databases return an error, the coordinator requests a rollback from each database.
- By definition, a DDBMS has to deal with the challenges of distributed transactions, i.e. the data is spread across nodes necessitating the need for two-phase commit.
- Not to be confused with database replication which involves mirroring all content of a master server onto another.
- Not to be confused with distributed processing. Distributed processing is a term used to describe an enterprise system architecture that distributes the processing across multiple computers, e.g. Client Server, 3-tier, n-tier, …
- The implementations vary across vendors. The DBA must have a good understanding of the various capabilities of a vendor's implementation to ensure it meets the business needs.
- DDBMS Advantages:
    - Local autonomy by having DBMS at the local level. Local business applications may use the locally stored data only while global applications may take advantage of the globally distributed data.
    - Performance by having the pertinent data closer to the user. For example, an international company would have each country's data closer to the local corporation.

    - Reliability by having multiple copies of the data. In some implementations, each location will have a copy of other locations. Failure of one node doesn't mean the data is not available.
    - High Availability because the DDBMS consists of several nodes with redundancy, so failure of a node or a communication failure does not mean the DDBMS will cease to function.
    - Scalability by adding nodes when needed in various geographic locations. For example, an international corporation can add a node whenever they open an office in a new country.
- Disadvantages
    - Data replication overhead.
    - Managing potential data anomalies due to difficulty in synchronizing updates.
    - Security due to difficulty in managing access across multiple nodes.
- Challenges
    - Performing distributed transactions which necessitate a more complex transaction commit protocol.
    - Managing user access and various privileges from one central location based on the same company policies.
    - Maintaining location transparency to the clients (users, applications) where applicable.
    - Specialized query optimization process to break up queries into multiple sub-queries that can be distributed and executed in parallel then the ability to join the results together and present as one query result to the client.
    - Data replication across the physical locations, e.g. replicating across geographical locations to place the information closer to the clients.

Federated Database

- Data sources are not necessarily DBMS, i.e. could be files. Therefore, maybe homogenous or heterogeneous.
- Provides location transparency
- Typically, geographically separated with low bandwidth connections.

Data/Information Warehouse

- A centralized repository of an Enterprise's data used for mining in order gain business insights.
- Typically, the data is a copy of the transactional data optimized for data mining, analysis, and reporting.
- Typically, is implemented on a DBMS that is optimized for fast access through some form of a Parallel Database technique. It is not necessarily a distributed database.
- Scalable to support a business' growth as more and more data is added daily and demand grows for more granular data, e.g. sales information by 15 minutes intervals.
- Have some form of Reliability and Availability as it's crucial to the enterprise.