Very Large Databases: Challenges and Strategies

Ogbonna J. C.[1], Nwokoma F. O. [2], Alala A. G. [3]

[1]Reseacher, Dept. of Computer Science, Clifford University Owerrinta, Abia State Nigeria
[2]Reseacher, Dept. of Computer Science, Federal University of Technology Owerri, Imo State Nigeria
[3]Reseacher, Dept. of Library and Information Science, Clifford University Owerrinta, Abia State Nigeria

---

Abstract - *The world of IT has grown to the point that the rate at which data is being generated calls for appropriate techniques and strategies to handle such large data set called Big Data through an appropriate database management system beyond the traditional DBMS. As at 30 years ago, data of size 100GB could have been regarded as a very large data, but in today's environment, a new database deployment in a large IT industry might start at 1TB meaning that 100GB may be regarded today as a small database size. The transition from desktop computing to mobile computing also has increase the rate of the usage of mobile devices since these devices are mobile, cheap, easily purchased and easily programmed, though having relatively low processing capabilities. As 100GB of database size could be regarded as a small database size for desktop applications, 50GB as well could be regarded as a very large database size for mobile applications. Hence, "Large" is a relative term that changes with time and the nature of the device in question. This paper aims at reviewing the challenges associated with very large databases and recommends the best techniques/strategies to those challenges.*

Key Words: Very Large Database, Data Warehouse, Data Mining, OLAP, OLTP, Big Data, Database Partitioning.

1. INTRODUCTION

Very Large Database (VLDB) is typically a database that contains an extremely high number of tuples (database rows or records), or occupies an extremely large physical file system storage space due to wide tables with large number of columns or due to multimedia objects (such as videos, audios, images). However, the most common definition of VLDB is a database that occupies more than 1TB or contains several billion rows. Data Warehouses, Decision Support Systems (DSS) and On-Line Transaction Processing (OLTP) systems serving large numbers of users would fall into this category (Wikramanayake & Goonetillake, 2006).

Over the years, the definition of a Very Large Database (VLDB) has radically changed. "Large" is a relative term that changes with time, what was large ten or twenty years ago could be small by today's standards, and what is large today will not be so in a few years from now. It is not uncommon to find databases in the tens to hundreds of TB's and even PB's today serving traditional data warehouse and increasingly On-Line Transaction Processing (OLTP) activities (Tim & Greg, 2008).

1.1 Very Large Database Challenges

The Very Large Database challenges include the following:

- There is a steady growth in the size of the database.
- There is no minimum absolute size for a very large database.
- It is not cost effective to perform operations against a system of such size.
- What are the best ways to capture, manage, backup and recover data in a very large database systems?

1.2 Trends Responsible for the Steady Growth in Database Size

According to Belden et al. (2016), there are several trends responsible for the steady growth in database size:

- For a long time, systems have been developed in isolation. The benefits of combining these systems to enable cross-departmental analysis (i.e. consolidation of databases and applications) while reducing system maintenance costs is a key factor in the ongoing growth of database size (Belden et al., 2016).
- Many companies face regulations that set specific requirements for storing data for a minimum amount of time. The regulations generally result in more data being stored for longer periods (Belden et al., 2016).
- Companies grow organically and through mergers and acquisitions, causing the amount of generated and processed data to increase. At the same time, the user population that relies on the database for daily activities increases (Belden et al., 2016).

To manage the steady growth in size of very large databases we employ partitioning strategies. Growth is the basic challenge that partitioning strategy addresses for very large databases and partitioning enables a "divide and conquer" technique for managing the tables and indexes in the database, especially as those tables and indexes grow.

1.3 Categories of Very Large Databases

The categories of Very Large Databases include the following:

- Data Warehouses and OLAP (On-Line Analytical Processing) Systems

- Operational Databases such as OLTP (On-Line Transaction Processing) systems
- Big Data in a Very Large Database

2. DATA WAREHOUSE

Data warehouse is a database that stores current and historical data of potential interest to managers through the organization. This data originates from operational and historical systems (internal data sources) as well as external systems (external data sources). Data from internal data sources and external data sources are extracted and transformed into the data warehouse. External data sources here could be transactions from websites, data from structured database model (such as relational database model, object-oriented database model, distributed database model, Hierarchical database model, Network database model etc), or data from unstructured data model (such as HTML, XML, text file, flat file, spreadsheets etc.). Data from these divers sources are extracted, transformed, and are standardized into a common data model and consolidated so that they can be used across the enterprise for management analysis and decision making purposes.

The data in a data warehouse are made read only data and are available for anyone to access. It does not require frequent update and it helps executives to organize, understand, and use their data during their decision-making process.

A data warehouse is designed to support analysis and decision-making. It separates analysis workload from transaction workload and enables an organization to consolidate data from several sources. Data warehouse environment can include an Extraction, Transformation, and Loading (ETL) process, On-Line Analytical Processing (OLAP), Data Mining capabilities, Queries and Reports, and other applications that manage the process of gathering data and delivering it to business users as shown in Figure 1.

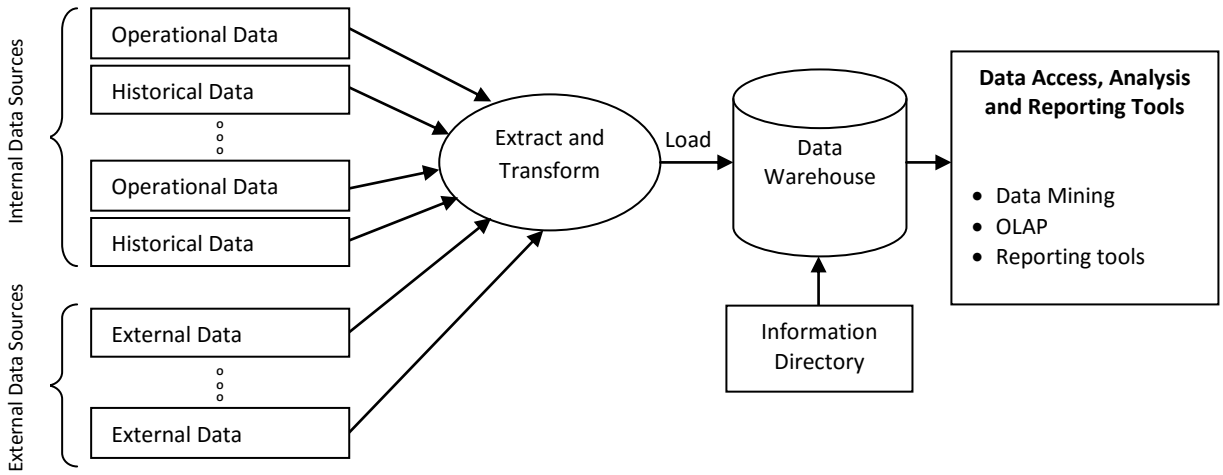

Fig -1: Components of a Data Warehouse

2.1 Data warehouse Extract, Transform, and Load (ETL) Operations

- Data Extraction: This involves gathering data from heterogeneous sources.
- Data Transformation: This means converting data from its source format to data warehouse format.
- Data Load: This involves sorting, summarizing, consolidating, checking integrity, building indexes and creating partitions.

2.2 On-Line Analytical Processing (OLAP) System

OLAP system is used to analyze an existing data in a data warehouse, generate summaries, calculate sum/aggregates, and exposes the data pattern because the data in an OLAP system rarely changes. Therefore, the data in an OLAP system can be optimized to perform complex operations such as calculating summaries and aggregates.

OLAP requires special data organization, access methods, and implementation methods, not generally provided by commercial DBMSs targeted for OLTP. It is for all these reasons that data warehouses are implemented separately from operational databases (Chaudhuri & Dayal, 1998). OLAP system also supports data analysis technique called Data Mining.

2.3 Data Mining

Data mining is the use of statistical tools to extract information from a very large database. Data mining software finds hidden patterns and relations in a large pool of large data and generates rules that would be used to predict future behaviour and guide the managers for their decision-making.

Database mining activities differ from traditional database interrogation in that data mining seeks to identify previously

unknown patterns as opposed to traditional database inquiries that merely ask for the retrieval of stored facts. Moreover, data mining is practiced on static data collections, called data warehouses, rather than "online" operational databases that are subject to frequent updates (Brookshear, Smith, & Brylow, 2013). This is because finding patterns in a static data collection is easier than in a dynamic data collection. At the same time, the explosive growth of databases makes the scalability of data-mining techniques increasingly important (Venkatesh, Johannes, & Raghu, 1999).

Data-mining tools use advanced techniques from knowledge discovery, artificial intelligence, neural networks etc to obtain "knowledge" and apply it to business needs. The discovered knowledge is then used to predict/forecast the outcome of an event. Data mining describes a new breed of specialized decision support tools that automate data analysis. In short, data mining tools initiate analysis to create knowledge. Such knowledge can be used to address any number of business problems. For example, banks and credit card companies use knowledge-based analysis to detect fraud, thereby decreasing fraudulent transactions (Rob & Coronel, n.d.). The explosive growth of databases makes the scalability of data-mining techniques increasingly important.

2.4 Benefits of Data Warehouse

The following are the basic benefits of a data warehouse:

- It does not only provide important information, but it provides improved information that can be used by decision makers.
- It has the ability to model and remodel data.
- It enables decision makers to access data as often as possible without affecting the overall performance of the operational system in question.
- Data extracted from a data warehouse through data mining and OLAP systems help organizations refocus on their business.

2.5 Scalability Requirements of Data Warehouses

Data Warehouses often contain large tables and require techniques both for managing these large tables and for providing good query performance across these large tables. Database Partitioning helps scaling a data warehouse by dividing database objects into smaller pieces, enabling access to smaller and more manageable objects. Having direct access to smaller objects addresses the scalability requirements of data warehouses (Belden et al., 2016; Morales et al., 2007):

- **Bigger Databases:** The ability to split a large database object into smaller pieces transparently provides benefits to manage a larger total database size. You can identify and manipulate individual partitions and sub-partitions in order to cope with large database objects.
- **Bigger Individual tables**: It takes longer to scan a big table than it takes to scan a small table. Queries against partitioned tables may access one or more partitions that are small compared with the total size of the table.
- **More Users Querying the System**: With partitioning, users are more likely to hit isolated and smaller data sets. As a result, the database will be able to return results faster than if all users hit the same and much larger data sets.
- **More Complex Queries**: Smaller data sets help perform complex queries faster. If smaller data sets are being accessed, then complex calculations are more likely to be processed in memory which is beneficial from a performance perspective and which reduces the application's I/O requirements.

3. ONLINE TRANSACTION PROCESSING SYSTEMS

Online Transaction Processing (OLTP) systems are one of the most common data processing systems in today's enterprises. Classical examples of OLTP systems are order entry, retail sales, and financial transaction systems. OLTP applications typically automate clerical data processing tasks (such as banking transactions, telecom transactions etc) that are the fundamental day-to-day operations of an organization. These tasks are structured and recurring, and consist of short, atomic, isolated transactions. The transactions require detailed up-to-date data, and retrieve or update few records using their primary keys. Operational databases tend to be hundreds of megabytes to gigabytes in size (Chaudhuri & Dayal, 1998).

OLTP systems are primarily characterized through a specific data usage that is different from data warehousing environments, yet some of the characteristics, such as having large volumes of data and lifecycle-related data usage and importance, are identical (Belden et al., 2016; Morales et al., 2007).

3.1 Characteristics of an OLTP System

The basic characteristics of an OLTP system Belden et al. (2016); Morales et al. (2007) include:

- **Short response time**: The nature of OLTP environments is predominantly any kind of interactive ad hoc usage, such as telemarketers entering telephone survey results. OLTP systems require short response times in order for users to remain productive (Belden et al., 2016; Morales et al., 2007).
- **Small transactions**: OLTP systems normally read and manipulate highly selective, small amounts of data; the data processing is mostly simple and complex joins are relatively rare. There is always a mix of queries and DML workload (Belden et al., 2016; Morales et al., 2007).
- **Data maintenance operations**: It is not uncommon to have reporting programs and data updating programs that need to run either periodically or on an ad hoc basis. These programs, which run in the background while users continue to work on other tasks, may require a large number of data-intensive

computations (Belden et al., 2016; Morales et al., 2007).

- **Large user populations**: OLTP systems can have immeasurably large user populations where many users are trying to access the same data at once. For example, an online auction web site can have hundreds of thousands (if not millions) of users accessing data on its web site at the same time (Belden et al., 2016; Morales et al., 2007).
- **High concurrency**: Due to the large user population, the short response times, and small transactions, the concurrency in OLTP environments is very high. A requirement for thousands of concurrent users is not uncommon (Belden et al., 2016; Morales et al., 2007).
- **Large data volumes**: Depending on the application type, the user population, and the data retention time, OLTP systems can become very large. For example, every customer of a bank could have access to the online banking system which shows all their transactions for the last 12 months (Belden et al., 2016; Morales et al., 2007).
- **High availability**: The availability requirements for OLTP systems are often extremely high. An unavailable OLTP system can impact a very large user population, and organizations can suffer major losses if OLTP systems are unavailable. For example, a stock exchange system has extremely high availability requirements during trading hours (Belden et al., 2016; Morales et al., 2007).
- **Lifecycle related data usage**: Similar to data warehousing environments, OLTP systems often experience different data access patterns over time. For example, at the end of the month, monthly interest is calculated for every active account (Belden et al., 2016; Morales et al., 2007).

4. VERY LARGE DATABASE AND BIG DATA

Big Data simply means complex and very large data sets from new data sources, voluminous enough that traditional data processing software tools are inadequate to process and manage them. Big Data makes it possible for one to gain more and complete answers to unharnessed problems in a business environment. More and complete answers simply mean having more confidence in the data which involves having a complete different approach to problems solving. The emergence of machine learning has produced more and voluminous data. Similarly, with the advent of the Internet of Things (IoT), more devices are connected to the internet, gathering data on customer usage patterns and product performance (Oracle Big Data, n.d.-a).

For instance, let us consider the case of a road traffic automation monitoring and control system that controls traffic flows and reduces traffic jams. If 10000+ GPS devices are placed on city buses that collect GPS location information per minute along the route, at the same time 2000+ CTTV cameras are mounted at some strategic points along the road to collect real-time video at those points for security reasons. Then, if the information is collected into a database, definitely the database will grow instantaneously to a point where a traditional data processing application may not manage such database. In this case, we regard this geo-special data as Big Data.

According to Berardi et al. (2015), there are a number of different data sources in a typical Big Data system which include:

- **Machine and Sensor Data**: Machine and sensor data are data generated by devices, sensors, networks and related automated systems including Internet of Things (IoT).
- **Image, Audio and Video Data**: This type of data source captures any form of media (pictures, audios, videos, etc.) which can be annotated with tags, keywords and other metadata.
- **Social Data**: Social data source are data/information created/generated in virtual networks and virtual communities.
- **Internet Data**: These are data stored on websites, mobile devices and other internet-connected systems.
- **Third Party Data**: Data from this type of data source are used to augment and enhance existing data with new attributes like demographics, geospatial etc.

Big Data brings together data from many disparate sources and applications (Oracle Big Data, n.d.-b) just like Data Warehouse, but traditional data integration mechanisms such as ETL used to extract, transform, and load data into a Data Warehouse are generally not suitable for the management and analysis of Big Data. It requires new strategies and technologies to analyze Big Data sets at Terabyte, Petabyte, or even Exabyte scale.

The best strategy for Big Data is to keep our data in the cloud than the traditional approach, as the cloud supports more computing environment and resources necessary to carry out Big Data Integration, Management and Analytics.

As Big Data analytics is becoming popular, many professionals are predicting that the era of data warehousing has come to an end. However, Big Data is not a replacement of Data Warehouse. Big Data is a technology but data warehouse is an architecture. Big data is capable of storing large data in inexpensive manner, whereas, data warehouse is capable of organizing data to reach a decision (Mukherjee & Kar, 2017). Some Big Data vendors including Oracle Corporation built their Big Data systems on top of their existing Data Warehouse platforms.

4.1 Characteristics of Big Data

- **Volume**: The quantity of generated and stored data is very large.
- **Velocity**: The speed at which the data is being generated is very fast.
- **Variety**: The nature of data varies (video, audio, geospatial data, 3D data, text files etc).

- **Veracity**: The quality of data can vary (data inconsistency), affecting the accuracy of analysis.

4.2 Data Warehouse versus Operational Database

The information in Table 1 shows the comparison between Data Warehouse and Operational Database systems.

Table -1: Data Warehouse versus Operational Database

Comparison Between Data Warehouse (OLAP) and Operational Database (OLTP)	
Warehouse (OLAP) System	**Operational Database (OLTP) System**
Data Warehouse contains historical data.	OLTP contains current/operational data.
OLAP deals with historical data	OLTP mainly handles current data
Data Warehouse is typically larger than OLTP.	OLTP is typically smaller than Data Warehouse.
It is used to analyze a business.	It is used to run a business
OLAP systems are highly flexible	OLTP systems provides high performance
Data analyst, managers, executives and decision makers use OLAP systems.	Clients, DBAs and DB professionals use OLTP systems.
OLAP system provides summarized and consolidated data.	OLTP system provides primitive and high detailed data.
Data Warehouse often has a lower availability requirement than an OLTP system.	OLTP system often has a higher availability requirement than a Data Warehouse.
Significant portion of the old data in a Data Warehouse are static/read only.	There is no such thing as static data in an OLTP system.
The advantage of static data is that it does not require regular backup.	Data in an OLTP system can be updated at any giving time; and so, the system does not require any data to be static.
Data Warehouse static data can be compressed to further reduce the size of the database.	Data in an OLTP system need not to be compressed, as compressing them will require frequent compressing and decompressing process during data update.
Data Warehouses support analytical tools such as data mining and OLAP systems.	Operational databases do not support data mining tools and OLAP systems.
In Data Warehouse, data are processed using ETL (Extract, Transform, and Load) process.	In an Operational Database, end users modify data themselves using DDL and DML statements.
Data Warehouse provides a unified view of all data elements with a common definition and representation for all business units.	In an Operational Database, similar data can have different representations or meanings.
Data are stored with subject orientation that facilitates multiple views of data, which facilitates decision-making. For example, books can be organized/arranged by title, by author etc.	Data are stored with a functional orientation or process orientation. For example, data may be stored as tuitions, fees, payments, amounts, bills etc.
Data are added only periodically from historical system. Once data commits, no changes are allowed. For example, the outcome of past elections does not need to be modified.	Data updates are frequent and common. For example, bank transaction updates are frequent and common.
It provides summarized and multidimensional view of data.	It provides detailed and flat relational view of data.

4.3 Very Large Databases Big Players

According to Edjlali and Beyer (2016); Edjlali et al. (2017); Feinberg and Beyer (2010); and Ronthal, Edjlali, and Greenwald (2018), there are a number of big players in very large database management solutions for analytics. These vendors are categorized into four major categories (leaders, challengers, visionaries, and niche players) based on the services they provide.

- **Leaders**: The Leaders category contains those vendors that demonstrate the greatest degree of support for data warehouses of all sizes, with large numbers of concurrent users and the management of mixed data warehousing workloads. These vendors lead the market in data warehousing by consistently demonstrating customer satisfaction and strong support, as well as longevity in the data warehouse DBMS market, with strong hardware alliances (Feinberg & Beyer, 2010). They maintain a strong vision with respect to emerging architectures and functionalities over the years.
- **Challengers**: These are vendors having strong offerings for the client, having a highly capable execution model, and have demonstrated corporate stability and proven products toward their customers. Challengers show a wide variety of data warehousing implementations across different sizes of data warehouses with mixed workloads (Feinberg & Beyer, 2010); and the ease of implementation, clarity of message and end-client engagement all contribute to making these vendors successful challengers.

- **Visionaries**: This category of vendors comprises of new/existing vendors having new architectures and functionality that are unproven in the market. They must demonstrate that they have customers in production proving the value of the new architecture and functionality. According to (Berardi et al., 2015) these vendors must move from providing good ideas to driving other vendors and products in this market toward new concepts and engineering enhancements.
- **Niche Players**: Niche players typically offer smaller, specialized solutions that are used for specific data warehouse applications, depending on the needs of the client (Feinberg & Beyer, 2010). They have low market share or low market appeal and provide exceptional data warehouse DBMS products, usually isolated or limited to a specific customers. This category comprises of vendors that lack strong or large customers, lack general customer acceptance, lack proven functionality to move them beyond niche category, or lack the functionalities of those of the leaders.

The information in Table 2 describes the capability of different vendors.

Table -2: Very Large Database Big Players

	The Categories of Very Large Database Big Players			
Year	**Leaders**	**Challengers**	**Visionaries**	**Niche Players**
2018	Oracle; Amazon Web Services; Microsoft; Teradata; IBM; SAP	Snowflake; MemSQL	Google; MarkLogic	Cloudera; MapR Technologies; Micro Focus; Huawei; Pivotal; Treasure Data; Alibaba Cloud; Qubole; GBase; Neo4j; Actian
2017	Oracle; Teradata; Microsoft; IBM; Amazon Web Services; SAP	Google; Cloudera; HP; MemSQL	MarkLogic	Snowfake Computing; Pivotal; MongoDB; MapR Technology; 1010data; Hortonworks; EnterpriseDB; Huawei; Transwarp Technology
2016	Oracle; Teradata; Microsoft; IBM; SAP	Amazon Web Services; HPE; 1010data; Infobright; MarkLogic	Cloudera; MapR Technology; Transwarp; Hortonworks; Pivotal; MemSQL	Exasol; Actian; MongoDB; Kognitio; Hitachi
2010	Teradata; Oracle; IBM; Netezza; Microsoft; Sybase		Greenplum; Aster Data	Vertica; HP; ParAccel; Kongnitio; Infobright; 1010data; Ingress; Sun Microsystems; SAND Technology; Illuminate

Source: (Edjlali & Beyer, 2016; Edjlali et al., 2017; Feinberg & Beyer, 2010; Ronthal et al., 2018)

5. VERY LARGE DATABASE PARTITIONING

The key component for solving some of the challenges associated with very large databases is partitioning.

Partitioning addresses key issues in supporting very large tables and indexes by letting you decompose them into smaller and more manageable pieces called partitions, which are entirely transparent to an application. SQL queries and Data Manipulation Language (DML) statements do not need to be modified in order to access partitioned tables. However, after partitions are defined, Data Definition Language (DDL) statements can access and manipulate individual partitions rather than entire tables or indexes. This is how partitioning can simplify the manageability of large database objects (Belden et al., 2016).

Each partition of a table or index must have the same logical attributes, such as column names, data types, and constraints, but each partition can have separate physical attributes such as compression enabled or disabled, physical storage settings, and table spaces (Belden et al., 2016).

In a Very Large Database, partitioning enables faster data access. Partitioning improves data access by orders of magnitude whether a database has 10GB or 100TB of data. Database partitioning is possible without requiring any modification(s) to applications that use it. I.e. DML statements such as SELECT, INSERT, UPDATE, DELETE, LOCK etc, need not to be modified because of database partitioning. Therefore, any application that accesses a given database takes the advantage of partitioning without the need to rewrite the application code.

5.1 Fundamentals of Partitioning

Partitioning enables a table or index to be sub-divided into smaller unit, where each unit of such a database object is called a partition. Different names are giving to different partitions which can be stored in the same or different storage locations.

Any table can be partitioned into a million separate partitions, and each row in a partitioned table is unambiguously assigned to a single partition. The

partitioning key is comprised of one or more columns that determine the partition where each row will be stored (Belden et al., 2016). With the use of partitioning key; insert, update, and delete operations are automatically directed to the appropriate partition.

The main advantage of partitioning is that we can store different parts of our data on different physical units, using file groups. We can also switch data in and out of a partitioned table, which is very desirable in many data warehouse applications. For instance, we can store old data that isn't referenced very often on slower drives, or perhaps on drives that have better read performance and less write performance (since this data doesn't change), whereas we can keep our current data on drives with good overall read/write performance.

Partitioning enhances the performance, manageability, and availability of a wide variety of applications and helps reduce the total cost of ownership for storing large amounts of data (Belden et al., 2016; Morales et al., 2007).

5.2 When to Consider a Database as a Candidate for Partitioning

According to (Belden et al., 2016; Morales et al., 2007), a database should be partitioned based on the following suggested conditions:

- Tables greater than 2 GB should always be considered as candidates for partitioning.
- Tables containing historical data, in which new data is added into the newest partition and have read-write access where as old data should be assigned read only access.
- When the contents of a tables need to be distributed across different types of storage devices.

5.3 Benefits of Partitioning

- **Performance**: Partitioning provides a number of performance benefits by limiting the amount of data to be examined or operated on, and by providing data distribution for parallel execution.
- **Manageability**: Partitioning allows tables and indexes to be partitioned into smaller, more manageable units, providing database administrators with the ability to pursue a "divide and conquer" approach to data management. It enables database designers and administrators to tackle some of the toughest problems posed by cutting-edge applications.
- **Availability**: Storing different partitions in different table-spaces allows the database administrator to do backup and recovery operations on each individual partition, independent of the other partitions in the table. Partitioning is a key tool for building multi-terabyte systems or systems with extremely high availability requirements.

5.4 Partitioning Strategies

Using data distribution methods such as Range, Harsh, and List, a table can be partitioned either as a single list (Single-Level Partitioning) or as a composite partitioned table (Composite Partitioning).

A. Single-Level Partitioning

In a single-level partitioning, a table is defined by specifying one of the data distribution methodologies (Range Partitioning, List Partitioning, and Hash Partitioning), using one or more columns as the partitioning key. For example, the diagram in Figure 2 shows a graphical representation of these three data distribution methodologies.

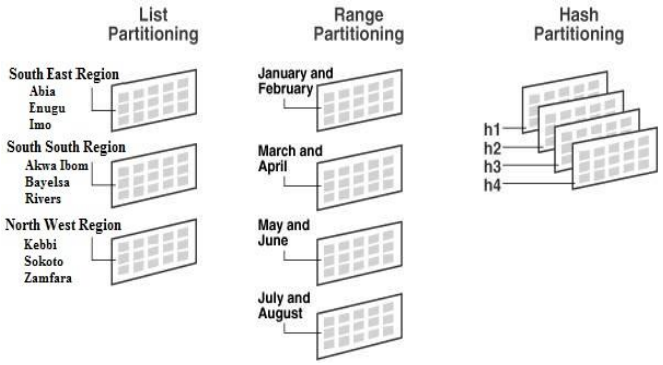

Fig -2: List, Range, and Hash Partitioning

- **Range Partitioning**: Range partitioning divides a table into partitions based on a range of values. You can use one or more columns to define the range. It is the most common type of partitioning and is often used with dates.
- **List Partitioning**: List partitioning according to Belden et al. (2016) and Morales et al. (2007) enable you to explicitly control how rows map to partitions by specifying a list of discrete values for the partitioning key in the description for each partition. The advantage of list partitioning is that you can group and organize unordered and unrelated sets of data in a natural way. In Nigeria for example, if the Federal Civil Service database table is to be partitioned using the region column as the partitioning key, then the South East partition might contain values such as Abia, Anambra, Ebonyi, Enugu, and Imo. For example, in Oracle Database 12c release, list partitioning can be handled as shown in Figure 3.
- **Hash Partitioning**: Hash partitioning is the ideal method for distributing data evenly across devices. Hash partitioning is also an easy-to-use alternative to range partitioning, especially when the data to be partitioned is not historical or has no obvious partitioning key (Belden et al., 2016; Morales et al., 2007). Hash partitioning maps data to partitions based on a hashing algorithm to the specified partitioning key. The hashing algorithm evenly

distributes rows among partitions, giving partitions approximately the same size.

```
CREATE TABLE Monthly_Payroll_By_Region
(
National_ID NUMBER,
Dept_ID NUMBER,
Dept_Name VARCHAR (25),
Monthly _Salary NUMBER (10, 2),
State VARCHAR (10)
)
PARTITION BY LIST (State)
(
PARTITION North_Central VALUES ('Benue', 'Kogi', 'Kwara',
'Nasarawa', 'Niger', 'Plateau', 'FCT'),

PARTITION North_East VALUES ('Adamawa', 'Bauchi', 'Borno',
'Gombe', 'Taraba', 'Yobe'),

PARTITION North_West VALUES ('Jigawa', 'Kaduna', 'Kano',
'Katsina', 'Kebbi', 'Sokoto', 'Zamfara'),

PARTITION South_East VALUES ('Abia', 'Anambra', 'Ebonyi',
'Enugu', 'Imo'),

PARTITION South_South VALUES ('Akwa Ibom', 'Cross River',
'Bayelsa', 'Rivers', 'Delta', 'Edo'),

PARTITION South_West VALUES ('Ekiti', 'Lagos', 'Ogun', 'Ondo',
'Osun', 'Oyo'),

PARTITION Region_Unknown VALUES (DEFAULT)
);
```

Fig -3: List Partitioning Example using Oracle Database

B. Composite Partitioning

Composite partitioning is a combination of the basic data distribution methods. A table is partitioned by one data distribution method and then each partition is further subdivided into sub-partitions using a second data distribution method. All sub-partitions for a given partition together represent a logical subset of the data (Belden et al., 2016; Morales et al., 2007). For example, Figure 4 shows a graphical representation of Range-Hash and Range-List composite partitioning.

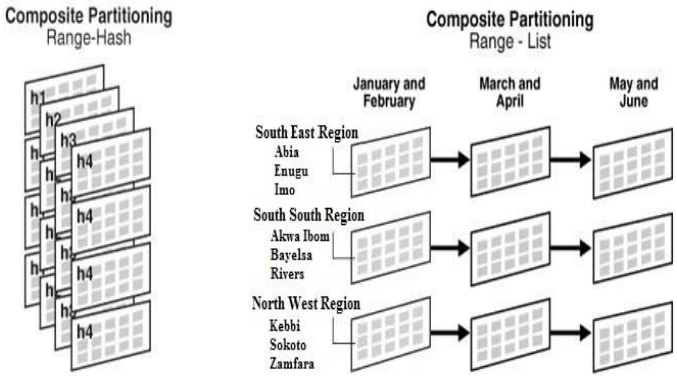

Fig -4: Composite Partitioning

6. THE VERY LARGE DATABASES PROBLEM

One problem with Very Large Databases is 'How do we backup and recover a VLDB of 50TB – 100TB in size?' This section will focus specifically on the backup and recovery challenges, and a look at how these challenges can be resolved.

Today's typical approach to database backup requires the IT administrators to have backup copies of their applications data onsite and offsite in order to recover from disruption of local services, application failure, user error, virus or any other disaster, from a prior point in time (Actifio, n.d.).

The best backup and recovery strategy for very large databases will have to answer the following questions:

- What is the best way to capture data?
- What is the best way to manage data?
- What is the best way to recover data?

6.1 What Is The Best Way To Capture Data?

Once a full database backup operation has been performed successfully for the first time, the best solution is never to backup the entire database again. In this case, it is recommended to perform incremental forever backup on the database and each backup should be database-consistent. This means that only the new data and the modified data will be copied during incremental forever backups, and this operation is performed based on a specified time interval. Figure 5 shows the diagram of Incremental Forever Backups, where full backups are performed once and incremental backups are performed on daily, weekly or yearly basis.

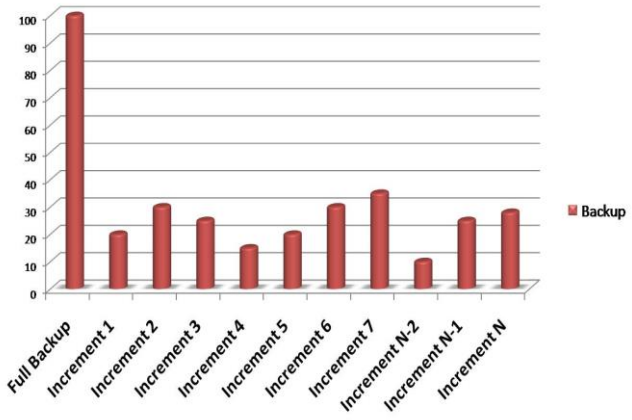

Fig -5: Incremental Forever Backups

According to (Actifio, n.d.), one of the compromises to 'Incremental Forever Backups' approach says, in a situation where a traditional backup product do not have incremental forever backup for a VLDB, the solution is to perform full backup over the weekend, and perform incremental backup daily. Figure 6 shows the diagram of Weekly Full and Daily Incremental Backups, where full backups are performed on Sundays and incremental backups are performed on Mondays through Saturdays.

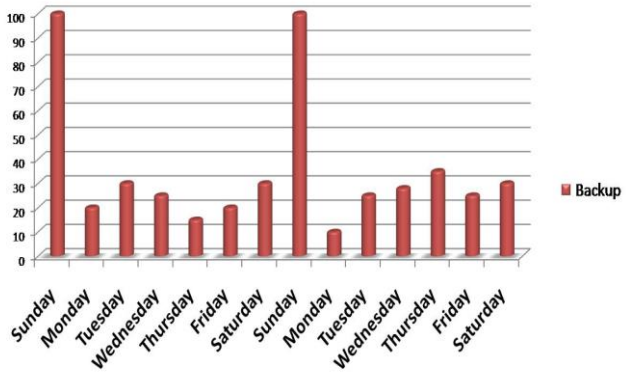

Fig -6: Weekly Full, and Daily Incremental Backups

According to (Actifio, n.d.), the advantages of incremental forever backup include:

- It significantly reduces the storage I/O, network traffic, CPU and memory utilization on production database environments up to 10 times.
- It reduces backup windows and increases the probability of success for backup operations.
- Smaller backup window opens up the possibility of many backups per day, thus reducing the RPO (Recovery Point Objective), even for a 100+ TB database.

6.2 What Is The Best Way To Manage Data?

One of the best solutions to managing VLDB data according to Actifio (n.d.) is to:

- Store data in its native format so that the recoveries are instant.
- Keep daily copies for a few days, weekly copies for a month, and monthly copies for a few months. This ensures a minimum of storage consumption with the incremental forever strategy to satisfy retention needs (cost effectively).
- Compress static data to further reduce storage consumption
- Replicate data to remote clouds or sites with only changed blocks and compression to reduce bandwidth consumption.

6.3 What Is The Best Way To Recover Data?

The best solution to recover data for VLDB according to Actifio (n.d.) is to:

- Restore a full VLDB from a deduplication appliance (an appliance that eliminates redundant duplicate data).
- Select the point in time usually an incremental backup from wherever it resides, and the backup server will have to restore that incremental, as well as the prior full backup.
- Then both the full and incremental database backup images will be rehydrated from their deduplicated state so the backup software can read the data in its proprietary backup format, combine them, and then converting and writing the database back to the target host in the native application format.

7. CONCLUSIONS

We have reviewed Very Large Databases, and examined the challenges associated with them. We also, discussed strategies for solving those challenges; and why we believe those strategies such as partitioning will continue to enhance the performance, manageability, and availability of a wide variety of database applications and help in reducing the total cost of ownership for storing large amounts of data for many years to come. Partitioning, indeed, has emerged as a new methodology and a critical feature for solving and managing problems associated with very large databases.

We also discussed the trends responsible for the steady growth in database size as well as the categories of very large databases including Data Warehouse and On-Line Analytical Processing (OLAP) Systems, Operational Databases such as On-Line Transaction Processing (OLTP) Systems, and the concept of Big Data with respect to Very Large Databases (VLDBs). We went further to recommend that the best approach to complex and very large data sets from new data sources, voluminous enough that traditional data processing software tools are inadequate to process and manage them is to keep such data in the cloud than using the traditional approach.

For the purpose of selecting the right tools for the pool of our big data sets, we analyzed and compared the very large database big players and their degree of support for data warehouses of all sizes. At the same time, we analyzed the nature of the architecture and the level of functionality they have provided in the market and how consistent they have demonstrated customer satisfaction, strong support and strong vision with respect to VLDs emerging architectures and functionalities over the years.

Finally, we concluded by answering the big questions with respect to very large databases such as the best way to capture data, the best way to manage data, and the best way to recover data.

REFERENCES

Actifio. (n.d.). *The Very Large Database Problem: How to Backup & Recover 30–100 TB Databases*. Retrieved from http://cdn2.hubspot.net/hubfs/214442/Actifio_For_Very_Large_Databases_White_Paper.pdf

Belden, E., Avril, P., Baer, H., Dijcks, J.-P., Fogel, S., Ganesh, A., … Wie, M. Van de. (2016). Oracle ® Database

VLDB and Partitioning Guide, *1*(July).

Berardi, T., Chessell, M., Gupta, M., Kak, A., Kreger, H., Statchuk, C., & Schalk, K. (2015). Customer Cloud Architecture for Big Data. *Cloud Standards Customer Council*, 1–20.

Brookshear, J. G., Smith, D. T., & Brylow, D. (2013). *Computer Science: An Overview* (11th ed.). Pearson Education.

Chaudhuri, S., & Dayal, U. (1998). An Overview of Data Warehousing and OLAP Technology. *ACM Sigmod Record*, *26*(1), 65–74.

Edjlali, R., & Beyer, M. A. (2016). Magic Quadrant for Data Warehouse and Data Management Solutions for Analytics. *Gartner, Inc. ID: G00275472*, (February), 1–34.

Edjlali, R., Ronthal, A. M., Greenwald, R., Beyer, M. A., & Feinberg, D. (2017). Magic Quadrant for Data Management Solutions for Analytics. *Gartner, Inc. ID: G00302535*, (February), 1–56.

Feinberg, D., & Beyer, M. A. (2010). Magic Quadrant for Data Warehouse Database Management Systems. *Gartner, Inc. ID: G00173535*, (January), 1–38.

Morales, T., Baer, H., Fogel, S., Hobbs, L., Lane, P., Lorentz, D., & Moore, V. (2007). Oracle ® Database VLDB and Partitioning Guide. *Oracle Corporation*, *1*(July).

Mukherjee, R., & Kar, P. (2017). A Comparative Review Of Data Warehousing ETL Tools With New Trends And Industry Insight. In *Advance Computing Conference (IACC), 2017 IEEE 7th International* (pp. 943–948). IEEE.

Oracle Big Data. (n.d.-a). Capitalize on the Data Explosion. Retrieved September 19, 2018, from https://www.oracle.com/big-data/index.html

Oracle Big Data. (n.d.-b). What Is Big Data? Retrieved September 19, 2018, from https://www.oracle.com/big-data/guide/what-is-big-data.html

Rob, P., & Coronel, C. (n.d.). *Database Systems: Design,Implementation,and management* (7th ed.). Thomson Course Technology.

Ronthal, A. M., Edjlali, R., & Greenwald, R. (2018). Magic Quadrant for Data Management Solutions for Analytics. *Gartner, Inc. ID: G00326691*, (February), 1–39.

Tim, C., & Greg, G. (2008). Very Large Database (VLDB) Backup & Recovery Best Practices: An Oracle White Paper. *Oracle Corporation*.

Venkatesh, G., Johannes, G., & Raghu, R. (1999). Very Large Databases. *IEEE*.

Wikramanayake, G. N., & Goonetillake, J. S. (2006). Managing Very Large Databases and Data Warehousing. *Sri Lankan Journal of Librarianship and Information Management*, *2*(1), 22–29.

BIOGRAPHIES

Ogbonna James Chinyere obtained his B.Tech. and M.Sc. degrees in Computer Science from Federal University of Technology Owerri (FUTO) in 2009 and 2016, respectively. He is an academic staff of Computer Science Dept., Clifford University Owerrinta, Abia State. His research interests include Mobile Computing, Information Security, Database Management Systems, and Software Engineering. He is a member of Nigeria Computer Society (NCS).

Nwokoma Francisca Onyinyechi obtained a National Diploma in Computer Science from Abia State Polytechnic Aba in 2005; B.Tech. and M.Sc. in Computer Science from Federal University of Technology Owerri in 2010 and 2016 respectively. She is an academic Staff of Computer Science Depts., FUTO. Her research interest includes Data Communication and Networking, Software Engineering, and Human Computer Interaction. She is a member of IEEE.

Alala Amarachi Glory obtained a National Diploma (OND) and a Higher National Diploma (HND) in Library and Information Science from Federal Polytechnic Nekede Owerri in 1997 and 2000 respectively. She also obtained a PGDE degree from University of Calabar in 20005 and MLS degree from Imo State University Owerri in 2012 both in Library and Information Science. She is presently an academic staff and a University Librarian, Clifford University Owerrinta; and a Ph.D. candidate in Library and Information Science, Imo State University Owerri.