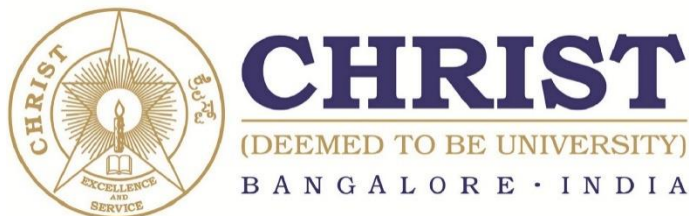


BDM CIA III
ON
INTRODUCTION TO DATABASE, TYPES AND
IMPLEMENTED APPLICATIONS FOR THE DATABASES -
CYIENT

By
GROUP III

2028609 SADDAM HUSSAIN
2028617 SHIVANI KOLLURI
2028619 CHARMI KANANI
2028625 SUCHITA J
2028626 DEEKSHYA BIDYANTA

Submitted To
PROF. MEERA SURESH



MBA PROGRAMME
SCHOOL OF BUSINESS AND MANAGEMENT
CHRIST (DEEMED TO BE UNIVERSITY), BANGALORE

APRIL 2021

Acknowledgment

We are indebted to all the people who helped us accomplish the study on the database domain, different applications and capabilities of other databases in various fields, and different solutions.

We thank Prof. Meera Suresh for allowing us to work and explore this technology (Database and implemented domains).

We wish to express our sincere thanks to Gowthaman S, Global Delivery Head – Digital Services & Solutions at Cyient Ltd., for allowing us to have a brief discussion on digital solutions and get the required details on these technologies' issues.

GROUP 3

Chapter No.	TABLE OF CONTENTS	Page No.
1	INTRODUCTION	4 - 10
2	TYPES OF DB AND COMPARISON	11 - 15
3	APPLICATION	16 - 25
4	PROS AND CONS	26
5	ETHICAL ISSUES	27 - 29
6	CONCLUSION	30
	REFERENCES	31

1. INTRODUCTION

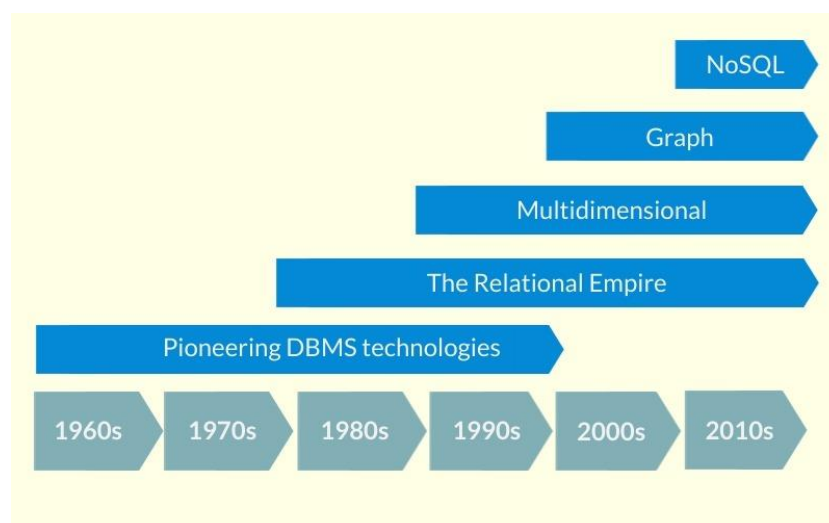
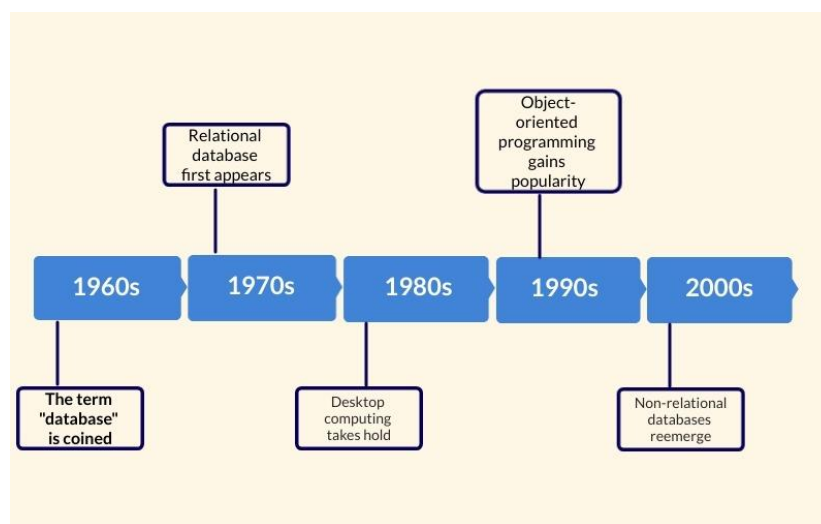
The advancements in technology have taken the whole world by storm. This has led to an exponential increase in the generation of data. Various sources create and generate embedded devices, creating data like files, servers, messages, images, videos, transactions, and enormous amounts of metadata. The embedded devices comprise chip cards, RFID readers, machine-to-machine technologies, and cellular networks. The data that is being created, consumed, and used is increasing rapidly, and it is estimated to reach 149 zettabytes by 2024.

With the availability of humongous data, it becomes essential to handle it efficiently. The companies must be aware of the significant data growth areas and leverage them in an organized way. The onus is now on the organizations/companies across the globe to handle the data diligently by having efficient storage systems in place to store the data and retrieve the data as and when the need arises. Data is now being considered a strategic asset by most companies. They are also using virtual servers, databases, computer networks, and cloud computing to compute and store the data. Databases have become widely popular for firms to store vast volumes of data as they can be retrieved instantaneously with minimal effort. The databases serve as repositories of information for the organizations where every form of data can be stored. The required information can be extracted easily at the click of a button.

These databases also have different storage tiers, and with changes in the application's performance needs, the need to be re-architected is brought down considerably. The databases also have some robust reporting features that make them the key and essential resources for analyzing the data and predicting trends. Databases have become much more efficient and business-friendly; they help store, track, and update the data regularly, simplifying the firms' data handling and storage. Another essential feature of the databases is that crucial importance is given to securing the stored information. Databases also aid in creating centralized systems, which facilitates the management of critical and vital business-related data.

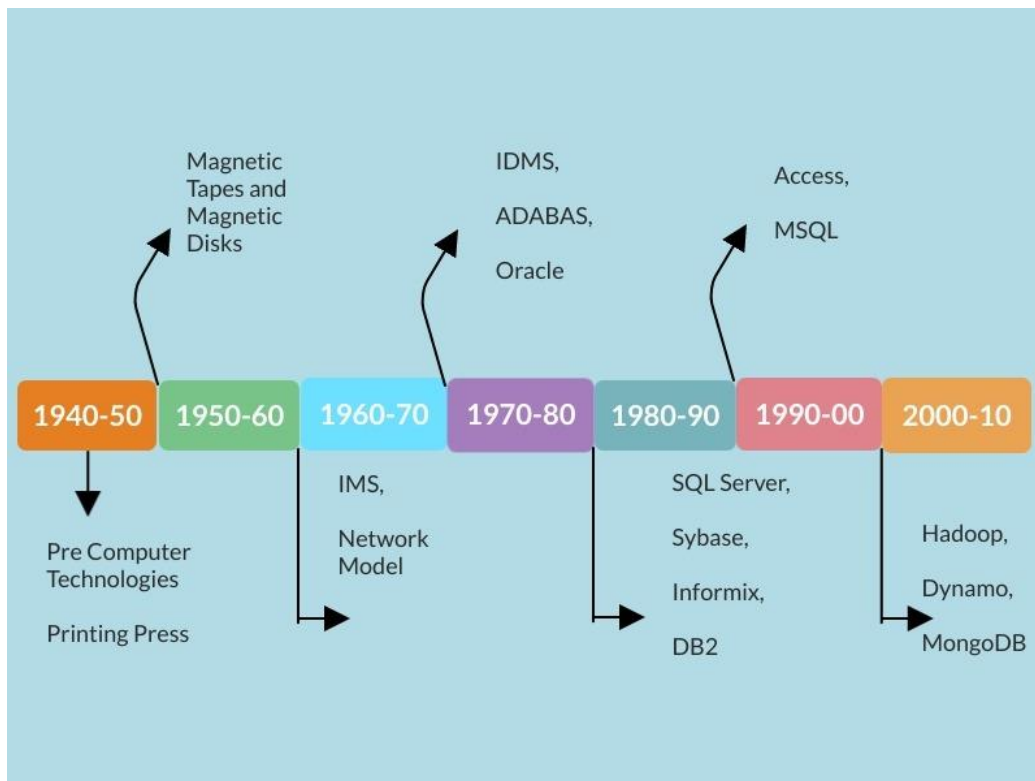
HISTORY AND EVOLUTION OF DB

Computerized databases started way back in the 1960s when the development of technology was still in the nascent stages. Back then, the two popular data models were CODASYL, and a hierarchical model named IMS. In the 1970s, the concept of relational databases had originated. Two primary relational database system prototypes were also created during this time. The structured Query language (SQL) was developed in the next decade, and the relational database systems turned out to be a commercial success. During this time, certain database companies were set up, and new products such as DbaseIII, PARADOX, and RIM were created.

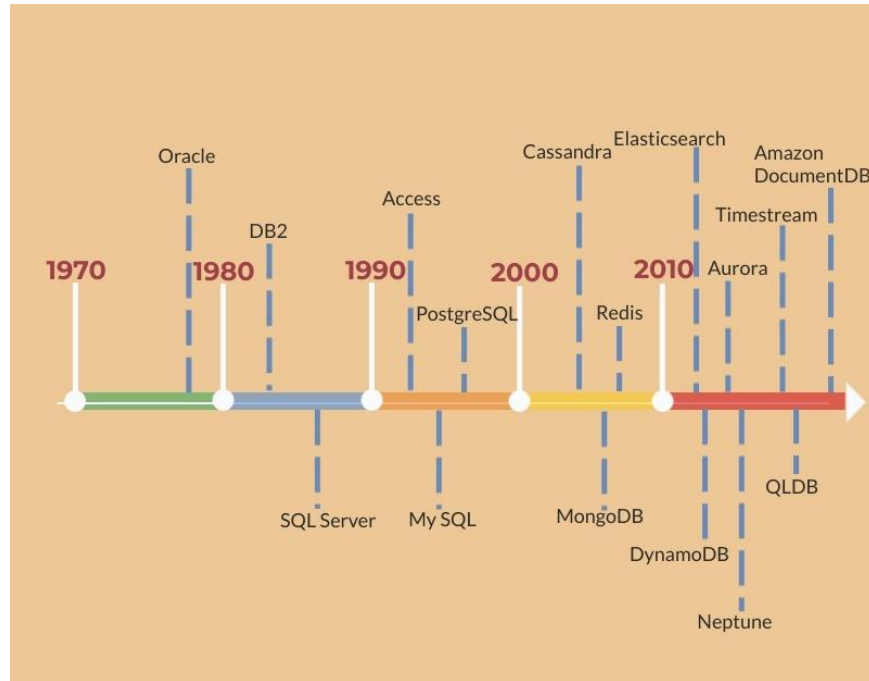


The early 1990s saw the widespread usage of database systems to store data in organizations, and complex database products were also created. New client tools were also developed to support the

applications, and the prototypes for Object Database Management Systems were also created. The late 1990s saw an exponential increase in the demand for Internet database connectors such as Java Servlets ColdFusion and Active Server Pages. The open-source license gave more freedom to the website developers to explore further in this domain. This went on to form the foundation of the websites that are being used currently.



New and interactive database applications were developed in the 2000s to facilitate vendors' compilation and promote point-of-sale transactions. At present, databases are extensively being used by companies of all sizes to store crucial business data. The advent of databases has made it possible to come up with personalized cloud storage services. It has also made it possible to develop and offer unique services and solutions to clients. The current relational databases comprise tech giants Oracle, DB2, and MySQL.



ABOUT COMPANY

Cyient is a global engineering and technology solutions company. They partner with customers along their value chain, supporting them to design, build, operate and maintain the products and services that have helped them become market leaders and respected brands in their industries and markets. Customers rely on Cyient's infrastructure, manufacturing, and emerging technology experience to build and sustain next-generation solutions that satisfy the highest safety, reliability, and efficiency standards.

Infotech Enterprises Ltd. was established in 1991 in Hyderabad, India. InfoTech Enterprises was re-branded Cyient in 2014. Cyient delivers its services and solutions to a diversified base of over 300 customers, including a few of the Fortune 500 companies, across multiple industries. The company became a publicly-traded organization in March 1997, with its equity shares listed in India, the National Stock Exchange, and the Bombay Stock Exchange. Cyient has a hold on various verticals or domains like Aerospace & Defense, Industry & Equipment, Healthcare, Semiconductor, Utilities, etc. It develops and delivers various end-to-end solutions for each vertical for its customers. Based on the customer requirement and the solution's need, they try to implement the best-suited DB to apply. They consider various aspects while developing the application like feasibility, easier integration, and most importantly, must be highly scalable for

future new implementations. They work on both SQL and NoSQL databases. Cyient has got comprehensive expertise for each database. Under SQL/RDBMS, they work on MsSQL, MySQL, PostgreSQL, Oracle, etc. similarly, for the NoSQL, they have covered almost most of the databases under it, some of them are Cassandra, DynamoDB (cloud-based solution), Neo4j, Redis (Logging based implementation), etc. Being a multi-national company, they have a good hold on all the required databases as needed for all the end customers.

DynamoDB

ABOUT DYNAMODB

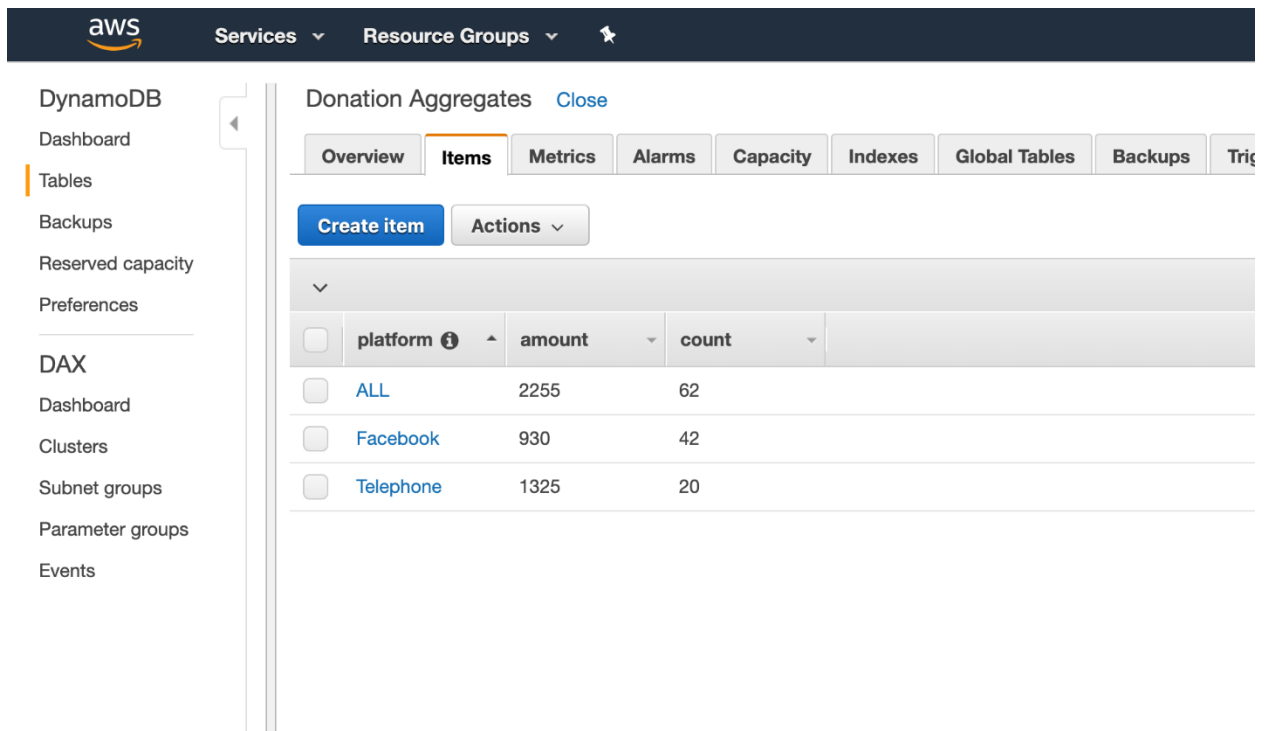
DynamoDB is a non-relational database developed by Amazon in 2012 and offered as a licensed database service as a part of Amazon Web Services. Amazon DynamoDB primarily supports key-value and document data structures. It is a wholly managed database with in-built security that can handle approximately ten trillion requests per day for designing several applications like mobile, gaming, web, IoT, etc. AWS customers have been using DynamoDB for handling the low-latency data. DynamoDB does not require any server or software to operate. It can adjust according to the capacity automatically. Moreover, DynamoDB has certain in-built features that eliminate the need for its inclusion in different AWS customers' applications and make it enterprise-ready.

HISTORY OF DYNAMODB

The e-commerce platform developed by Amazon had some limitations. For instance, the services had to be handled individually and required a separate API for each one of them. Amazon required a database that could address all its requirements like scalability, availability, and durability needs, which were more than any other vendors. Initially, all the services were handled by a relational database. However, they later realized that they did not need such a complex set of queries to fetch the data, and they decided to build a NoSQL database to overcome the issues mentioned above. Engineering teams were interested in adopting other databases compared to DynamoDB as it was expensive at that time. However, after making improvements, Amazon released DynamoDB to a public cloud in 2012.

DATA MODEL OF DYNAMODB

In contrast to the relational databases, Amazon's DynamoDB is a schema-less database. This means that no other attributes or data types must be defined except for the primary key while creating the table. The data model of DynamoDB consists of three major components: 1. Table 2. Item 3. Attribute. The table of this database is like any other relational database. However, the table can have an infinite number of items and does not require to have the exact attributes or the same number of attributes. The items are composed of primary or composite vital attributes. DynamoDB is unique as there is no limitation of restricting the number of attributes associated with a single item. The attribute associated with a data item consists of an attribute name and a set of values. These attributes do not have any size limit; however, an item's total value should not exceed 400KB, including the attribute names.



The screenshot displays the AWS Management Console interface for a DynamoDB table named 'Donation Aggregates'. The left-hand navigation pane shows the 'DynamoDB' section with options like Dashboard, Tables, Backups, and Reserved capacity. The 'Tables' option is currently selected. The main content area shows the 'Donation Aggregates' table with the 'Items' tab active. A 'Create item' button and an 'Actions' dropdown are visible above the table. The table itself has columns for 'platform', 'amount', and 'count'. The data rows are as follows:

	platform	amount	count
<input type="checkbox"/>	ALL	2255	62
<input type="checkbox"/>	Facebook	930	42
<input type="checkbox"/>	Telephone	1325	20

Scan: [Table] device_data: serial, timestamp ^

Scan [Table] device_data: serial, timestamp ^

Filter payload.lock String = L1

+ Add filter

Start search Cancel changes

How can I filter payload?

	serial	timestamp	payload
<input type="checkbox"/>	SX001	1472784505406	{ "lock": { "S": "L1" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX001" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input checked="" type="checkbox"/>	SX001	1472784542467	{ "lock": { "S": "L1" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX001" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX001	1472784545397	{ "lock": { "S": "L1" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX001" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX002	1472784437522	{ "lock": { "S": "L2" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX002" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX002	1472784440078	{ "lock": { "S": "L2" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX002" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX002	1472784532899	{ "lock": { "S": "L2" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX002" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX002	1472784536931	{ "lock": { "S": "L2" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX002" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }
<input type="checkbox"/>	SX002	1472784539466	{ "lock": { "S": "L2" }, "maxtmpr": { "N": "50" }, "serial": { "S": "SX002" }, "stat": { "N": "255" }, "timeover": { "N": "0.17" }, "tmpr": { "N": "30" } }

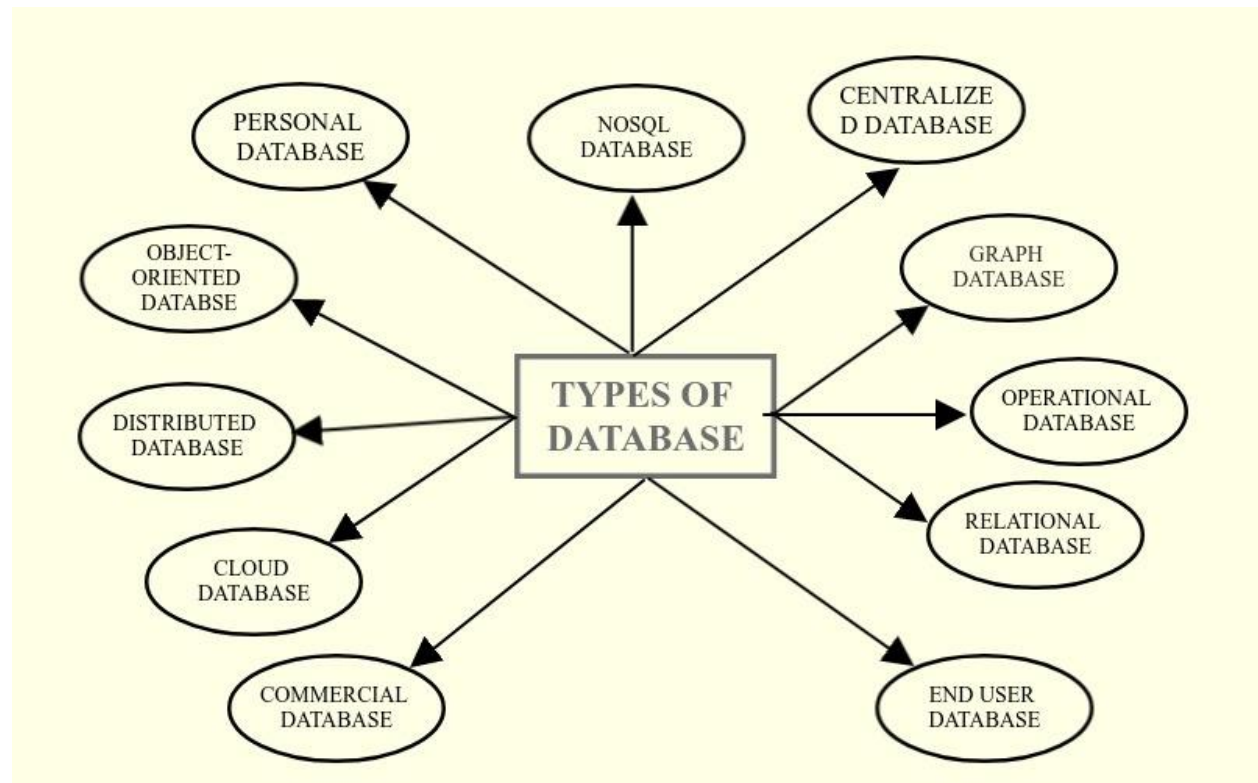
DATA EXPLANATION IN DYNAMODB

The NoSQL database's primary function is to retrieve the data according to predefined queries as fast as possible. For this, DynamoDB has a specific set of instructions that are to be followed. For instance, to create a DynamoDB table, it is mandatory to choose a partition key among the attributes. DynamoDB splits the data into smaller partitions based on the key. This happens in multiple steps. This is done through the hashing algorithm. Data is split into multiple "Hash Ranges," which corresponds to one or more partition keys.

Similarly, querying also has specific steps. When a client requests some information, DynamoDB identifies the partition key responsible for this chunk of data. It routes the query to the path that leads to the specific partition. This partition returns the data to the client.

2. DATABASE AND COMPARISON

Everyone talks about data in today's world, but one of the major aspects of data is having the best provision to store them. There are various types of databases available at the current time. Below are the categories of databases based on the usage requirement:



We will elaborate more on the SQL and NoSQL types. SQL is a structured and relational-based database, whereas NoSQL is an un-structured and non-relational type of database. There are various providers or databases under each type.

SQL: It is a relational-based database. It uses structured query language for data manipulation or any other functionality related to it. It follows a predefined schema for the structure of work to be achieved.

- ✓ **MySQL:** It is a free and open-source database available for everyone, and it is managed and owned by ORACLE. A huge community for support. It is supported for most of the OS, be Windows, Linux, and Mac, etc. This DB can be replicated across any other system. Sharding can be done quickly in this DB, whereas it cannot be done on most SQL databases.
- ✓ **Oracle:** Licensed-based database as it is managed professionally, some of the version has been made accessible for the users. It is developed and managed by Oracle entirely, as managed by them, to get good updates and customer support. It uses PL/SQL dialect for the DB. It is one of the expensive DBs available for any OS. Can handle large database size. Every connection is separate; hence there is a provision of rollback and commit, and it is easily upgradeable. Also, integration with other systems is effortless.
- ✓ **MSSQL:** Developed and entirely managed by Microsoft. It provides faster updates and excellent customer support available. It also follows a unique SQL dialect which is called T-SQL. It is available and supported for Windows and Linux systems. It runs separate execution commands; hence it is tough to make any changes in between if something goes wrong. The tables, views, etc., are managed by an SQL server. Ease while using this DB. It provides various other features with the package like Management studio, SQL server management.
- ✓ **PostgreSQL:** It is open source and free to use. It was developed by PostgreSQL global development group. It is available for any other OS. It is a hybrid type of database and is ORDBMS type. There is free community service or support available. It provides the highest offering for ACID property.

NoSQL: Non-relational database follows an unstructured way of handling the data or the data manipulation. It follows dynamic schemas can be stored in many other ways. It can be key-Value, document-oriented, graph-based, or column-oriented.

- ✓ **MongoDB:** It has been made accessible after 2018. It follows dynamic schema type as we can easily change the structure of the data stored. It can be horizontally scalable and swift for the queries. It is very flexible with the table data. Recently added cloud-based services called MongoDB Atlas.
- ✓ **Apache Cassandra:** Managed initially by Facebook, but Apache started working on it after making free and open source. It is a highly scalable type of DB. It can be accessed using any nodes. It does the data manipulation very fast as the data are distributed. It does not follow the ACID property. When it comes to updating and deletion of data, it is poor. Data are highly available.
- ✓ **Apache HBase:** It is also free and open-source managed by Apache. It was created after the Google cloud Bigtable. It was created to manage the vast amount of data. It can be easily scaled across the clusters. Both structured and unstructured data types can be used. It is automatic and easily configurable for Sharding. It manages its data quickly as the data increases; then, it splits to another server for better management. As it offers replication, hence no runtime can be seen.
- ✓ **Amazon DynamoDB:** Developed and managed by Amazon. Does not have to worry about the hardware and highly configurable and scalable. Highly secure as follows encryption and decryption by default eliminating the work separately for protecting the data. Cloud-based service. Available to be used for most languages.

RELATIONAL VS NON-RELATIONAL

	SQL/RDBMS	NoSQL
Type	Relational	Non-Relational
Lanaguge	Strucutred query language	un-strucutred query language
Schema	Static	Dynamic
Scalability	Vertically	Horizontally
Data	Tabular data with structured format	Data stored as Unstructured JSON file but graph database supports relationships
Strucuture	De-centralized structure	Centralized structure
OLTP	recommended and best suited for OLTP	Less likely to be used for OLTP system
Joins	Useful while designing complex queries	No joins, Doesn't have powerful interface to prepare complex query
Integrated Caching	Supports inline memory (SQL 2014 and SQL 2016)	Supports Integrated caching
Support	Great support	Community dependent, expanding the support model
Transaction	ACID	CAP theorem
flexible	rigid schema bound to relationship	Non-reigid schema and flexible
Insert & Updates	Synchronous insert and updates	Asynchronous insert and updates
Auto elasticity	Requires downtime in most cases	Automatic, No outage required
Top Companies Using	Hootsuite, CircleCI, Gauges	Airbnb, Uber, Kickstarter
Best features	Cross-platform support, Secure and free	Easy to use, High performance, and Flexible tool
Network	Highly available network (Infiniband, Fabric Path, etc.)	Commodity network (Ethernet, etc.)
Importance	It should be used when data validity is super important	Use when it's more important to have fast data than correct data
Development Year	It was developed in the 1970s to deal with issues with flat file storage	Developed in the late 2000s to overcome issues and limitations of SQL databases.
Examples	Oracle, Postgres, and MS-SQL.	MongoDB, Redis, Neo4j, Cassandra, Hbase.

AMONG DIFFERENT NON-RELATIONAL DATABASES

	MongoDB	Cassandra	Apache Hbase	DynamoDB
Language	C++	Java	Java	Java
Version	Free & Limited	Free & Open Source	Free & Open Source	Licensed database service of AWS
Architecture	Schema-less database, storing data as JSON-like documents (Document store)	BigTable and DynamoDB (Wide column store)	Apache Hadoop and BigTable (Wide column store)	Hosted by Amazon, (Document store, Key-value store)
Data Storage	Memory, File, Framework	DynamoDB	HDFS	Amazon Cloud
Developer	MongoDB, Inc	Apache Software Foundation	Apache Software Foundation	Amazon
Query Language	Read-only SQL queries via the MongoDB Connector for BI	CQL (Cassandra Query Language)	XML API calls	PartiQL
Server Operating Systems	Linux, OS X, Solaris, Windows	BSD, Linux, OSX, Windows	Linux, Unix, Windows	Hosted
Supported Programming Languages	Actionscript, C, C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Rust, Scala, Smalltalk, Swift	C#, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala	C, C#, C++, Groovy, Java, PHP, Python, Scala	.Net, ColdFusion, Erlang, Groovy, Java, JavaScript, Perl, PHP, Python, Ruby

ORACLE VS DYNAMODB

	Oracle	DynamoDB
Language	C and C++	Java
Initial Release	1980	2012
Data Scheme	Schema based	Schema-free
Architecture	Relational DBMS	Document store, Key-value store
Data Storage	Installed Server (on-premises)	Amazon Cloud
Developer	Oracle	Amazon
Server Operating Systems	AIX, HP-UX, Linux, OS X, Solaris, Windows, z/OS	Hosted
Query Language	SQL	PartiQL
APIs and other access methods	JDBC, ODBC, ODP.NET, Oracle Call Interface (OCI)	RESTful HTTP API
Supported Programming Languages	C, C#, C++, Clojure, Cobol, Delphi, Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp, Objective C, OCaml, Perl, PHP, Python, R, Ruby, Scala, Tcl, Visual Basic	.Net, ColdFusion, Erlang, Groovy, Java, JavaScript, Perl, PHP, Python, Ruby
User concepts	Fine grained access rights according to SQL-standard	Access rights for users and roles can be defined via the AWS Identity and Access Management (IAM)

3. APPLICATIONS

Social Media: Social Media is one of the significant contributors to data in today's world. It generates a massive volume of data every second. Almost every person uses one or the other social media platform. As per one of the studies, 3 billion people use social networking sites. Think about the volume of data being generated from these platforms, and it will grow exponentially in the coming days as everyone is adopting digital media. Different databases can be used in the domain as most social media generates unstructured data every second; it is essential to use the database to handle this much size of data and handle these data with ease. Different NoSQL databases can be used for these purposes. This data leads to Big Data; hence the best databases suited for these types of NoSQL (Preferred). Most of the Social media platforms started with the relational database, MySQL as it was open-course, but now all of these platforms' companies prefer NoSQL DB on a larger scale due to the type of data.

Banking: In today's fast-moving world, most processes have become swift due to easy accessibility. Online transactions and net banking form a part of such day-to-day processes. Through these transactions, much data is generated every single day. To track all the transactions, personal details, balances, and deposits, banks use a database management system. This enables the bank to store and retrieve data whenever required. Most of the banks use a combination of databases like Oracle, MySQL, Sybase, and MongoDB. These are the typical databases that the banks are using. However, in recent years, the databases like Cassandra and Postgres are the banks' novel approaches.

E-Commerce: As technology has progressed, we as individuals acquiring digital media for our day-to-day life have gone up exponentially. E-Commerce is one such area that has arisen due to the same. It helps the entire business process, i.e., sell, buy, and payment been implemented on a single platform to make our life easier. It applies to B2B and B2C. All these processes or different process under it has got a considerable amount of data under it. So, there comes the need for a database to store all the information. Due to the current AI world, it is essential to have a massive chunk of data where NoSQL plays a very important.

Healthcare: It is one of the major fields which requires new technologies it and implementing them to improve the domains leads to saving a tremendous amount of money. Healthcare is one domain that cannot be ignored, and the amount of people visiting a hospital daily is going up like

anything. It is essential to store all the patient's data and the other required data. Depending on the requirement of current technologies, most hospitals are trying to migrate to new ways and faster ways of using these data, like migrating to cloud-based databases or big data, which can be used for further innovation improvements.

Retail: In this sector, databases are used for a better customer experience. Databases store the store's inventory, customer details, purchase, and transaction history of a customer. As a result of these data available to the businesses, they can provide quick and easy service to their customers, leading to customer satisfaction. Storing the business's facts or figures can enhance the business in terms of revenue generated through sales. The businesses use databases like MongoDB, CouchDB, and Redis.

VERTICALS/DOMAINS OF CYIENT

Cyient being a multi-national tech company working in various domains or verticals for their large customer base. It has got a stronghold of its tech in all the fields and trying to innovate in various fields. Working on various verticals and their data handling process is fundamental as any business 70% of the process gets completed if the data are correct and managed or meaningful data. As per the discussion, different domains/applications require different databases based on the requirement. Based on the customer's requirement for any specific solution, they work on the selected database, be it relational or non-relational.

It covers a wide range of applications in different verticals. Some of them are explained below as per the discussion:

Aerospace: Cyient is into offering aerospace manufacturing engineering solutions. These solutions are developed right from the conceptual design to its implementation and maintenance. Most of the solutions include PCB assembly, aerospace CNC machining, wire cable harness, precision tooling, in-circuit testing, first article impression, vibration testing, and many more. These solutions enable automated decision-making and facilitate aerospace supply chain management and the deployment of aircraft predictive maintenance solutions. Cyient uses databases like Cassandra and Elastic Search to offer these engineering solutions in this domain; it also incorporates other databases based on the requirement.

Manufacturing: Cyient also works extensively in the manufacturing domains for various customers; it also works with GE, Schneider, and other manufacturing domains to build better and high-quality products. It has also developed the industry 4.0 capability in their premise and one of their customers where integration of level 0 to level 4 is being shown. Moreover, all these integrations of different solutions or building the Industry 4.0 had a large amount of data. So, different databases were used while building the on-prem solution. MSSQL was used to store the data from PLC, MongoDB for sensor data. Also, MySQL was used to store project management data; Oracle was used for the MES and SAP systems; Cloud was also integrated to better efficiency of the solution. It comprises SQL as well as a NoSQL database.

Healthcare: Cyient has been planning to incorporate several technology-enabled healthcare solutions primarily focusing on diagnostic space. Cyient mainly focuses on additive manufacturing. Moreover, 3D printing is adapted for almost all the solutions of the healthcare systems. Cyient also partnered with various initiatives to bring MedTech stakeholders to address “the last-mile challenge” in this sector.

Energy: The energy sector is in suitable transition mode. Different modes of the medium are tried to achieve in this domain. Their main aim is to reduce the energy sector’s cost. The various solutions developed under it are like having a plant’s simulation process, improving efficiency using the 3D Model solution, and the image analytics-based solution in this domain. These solutions require different databases depending on the customer and the software used, and most importantly, they require requirements. The DB used for the 3D model solution is Oracle, Image analytics used on MongoDB, and Cassandra.

Defense: Cyient addresses various defense requirements by data management, innovative engineering, and several other analytics solutions to help the Original Equipment Manufacturers get the products on time. The defense market has to evolve along with developments in and around this field. Cyient has its expertise on military platforms. In addition to this, Cyient has been a qualified offset partner for Military Airborne applications and defense platforms. In addition to this, Cyient has been a qualified offset partner for Military Airborne applications and defense platforms. These services are provided using data management and analytics.

Navigation: It uses its experience in the navigational data mapping and navigation DB system by providing more accessible and precise solutions for better safety and driving value-added

experience. It utilizes the data generated from the devices installed in the transportations like navigation data to provide a better navigation system for the user, and It also uses an aerial image with high resolution to process the status of the transportation, satellite imagery for the security mainly in the defense and better navigation scheduling.

Choosing various databases and then implanting them into various domains was not an easy task for Cyient. It will help them after implementing various DB to the system was a thing to be discussed. Cyient has divided its solutions into five different broad categories under their Engineering services:

Digital Manufacturing: CAD or the digital copy of the plan or the solutions that will help them check the solution before actually implanting it using tools like TechnoMatrix (provided by Siemens, CATIA, etc.). It has its native way to store the data, whereas to have a better and improved system, Cyient had built a standard communication layer to integrate them to get these data to other databases (to and fro flow of data) like integrating with the Neo4j available DB in the system also with the MSSQL Db.

Machine to Machine Connectivity: Connecting the factory devices (from a small device to the large machines or the tool being used inside the factory). It helped them to have a standard layer where each device was able to communicate with one other. The different systems had their DB used, and the integration between them was also a significant challenge while building the framework layer. Majorly used DB under these were PostgreSQL, MSSQL, Cloud Databases (Dynamo) or processing these data for the third-party system to better improve the application.

Robot and Automation: implementing the Automation by also using the commonly achieved layer from the Machine-to-Machine connectivity. Cross communications, like having a standard layer, were all kinds of Robots or PLC (ABB, Siemens, Allen-Bradley, etc.) were connected, and each was able to understand the data from one other. It helped them to have a highly connected solution. One of the other solutions added for the aerospace industry was CATS (cognitive asset tracking solution); it was mainly to track and monitor the tools being used in the aero factory while doing the production. Used databases were MSSQL, Cassandra, Excel integration as well.

Analytics/ML: It is one of the significant and vital technologies that helps the various customer understand their machines or the devices and help them make confident decisions based on their

behavior. A solution built on the analytics is with a US-based Nidec Motor company to have an analytics-based solution to analyze the behavior and usage of the motors and then provide the analytics on these motors (analytics implemented was predictive maintenance). The predictive maintenance model helped them find the expected no of usage. It will fail, which will help the end-user have a backup plan based on this provided analysis and predictive solution. This solution was provided to their end customer as a subscription-based model (extra feature), where they can take corrective action depending on this feedback. The cloud used for the solution was Microsoft Azure. It was majorly the combination of descriptive and predictive analysis where all the data needs to be available. The various DB was used again for this category like MongoDB, Cassandra, and MSSQL, depending on the type of solution.

AR/VR: It is a kind of domain that is still improving. AR helps the user to be in the physical world as well as the virtual world. Some of the real-life examples of AR are the Pokémon Go game, Google glass lens, wholly based on AR. At the same time, VR helps individuals to be in the virtual world ultimately. E.g., for the same as a Virtual Tour of the Car showroom. Cyient has implemented various solutions on both these categories and one of them being a virtual based training built for their manufacturing sector customer and a virtual solution of the complete remote monitoring of the manufacturing plant was also built as an AR solution and also having the feature of remote diagnostics in this virtual solution. Again, these all required using a high volume of data it and the used databases were MongoDB and PostgreSQL.

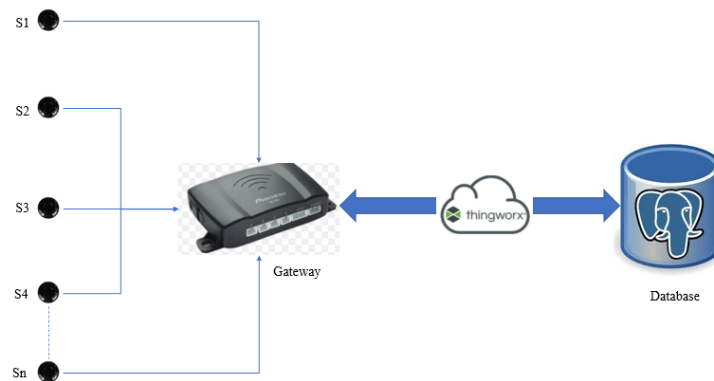
Below are some of the solutions:

- **Smart City:** Built an entire smart city solution for one of their end customers where the real solutions were divided into various use cases like smart homes, bright lighting, smart bin, smart parking, etc. As per the discussion on intelligent lighting and parking solution, the entire flow was having a sensor being installed on the premise then these sensors talking to IoT gateway where the data the cumulative data were sent to the IoT platforms or the backend server and then complete solutions were handled as per the requirement. As the requirement was not to store all the time series data as they wanted to track only when the sensors got activated and deactivated or wanted to control them from the back end. The RDBS used for the tabular form of data storage, and the DB used was PostgreSQL for the

entire process. This DB was integrated with the other system like other DB for further analysis like implementing analytics-based solutions. Below is the sample JSON data:

```
{
  "time":1617356307,
  "payload":{
    "device_id":"SID00B102",
    "battery":80,
    "parking_status":1
  }
}
```

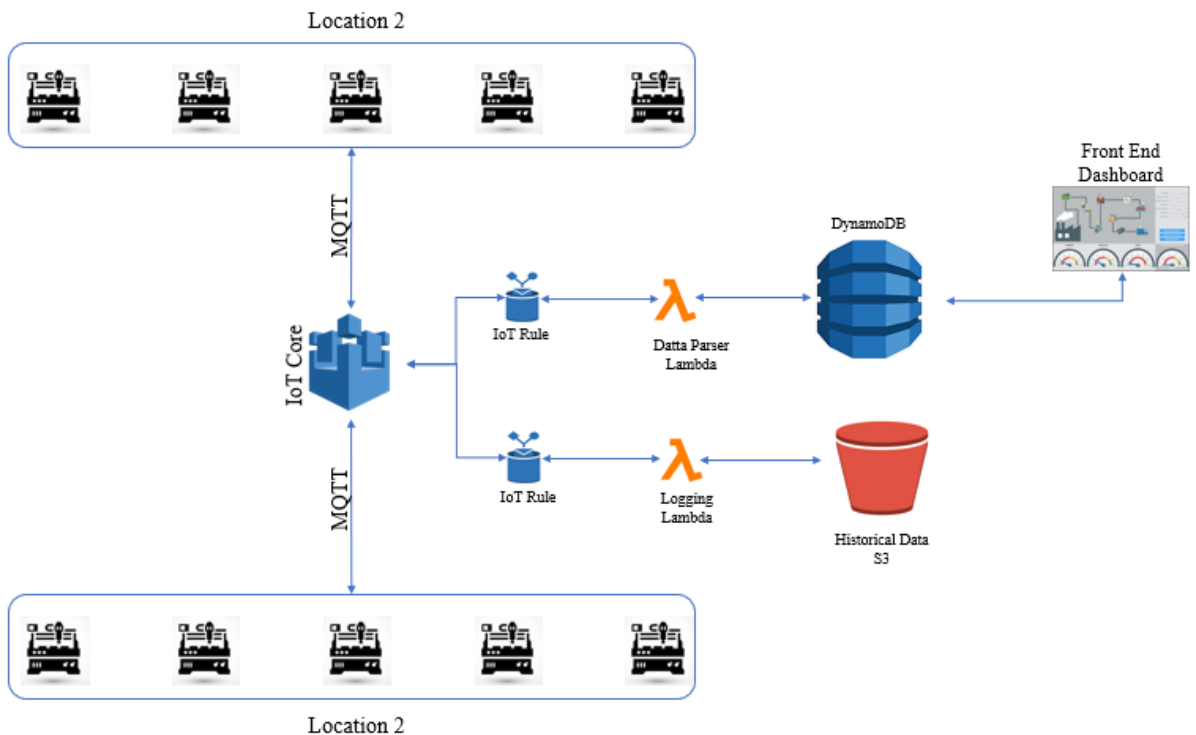
The basic architecture for the solution:



- **Factory Remote Monitoring:** The customer wanted to have real-time monitoring of the machines installed in their premise with the analytics-based solution to predict the motors' failure in advance. Real-time monitoring required time-series data, and for analytics, all data were required. They did not want any loss of data in any form, so the system was required to be tightly coupled to achieve this and, in such case, to meet all these requirements, NoSQL was used as it can store the enormous amount of data as per the requirement and losses of data will not be there. The DB used was DynamoDB for the same. The different schema was created for the different requirements like a separate table for the available machines, user-based table, plants table, analytics data table current data table where the last data were received. An IoT device was installed on the plant, and these were directly sending the data to DynamoDB using the RESTful services. All these data

were stored for six months. So, a separate function was running to clear the older data as per the requirement. Below is the JSON sample data (4 tags of PLC) of one of the tables:

```
{
  "time":1617356765,
  "payload":{
    "plc_id":"PLC0068",
    "ip":"10.76.54.89",
    "plc_status":2
  },
  "messageinfo":{
    "actualspeed1":"78",
    "actualspeed2":"104.5",
    "setspeed1":"80",
    "setspeed2":"105"
  }
}
```



- **Integration for the ERP, SAP to other systems like MES and PLC:** Built an Industry 4.0 for their Japanese customer to have all the intelligent factory capabilities, also called the future factory. Level 0 to Level 4 integration was made for a solution. The customer's main aim was not having complete visibility of the entire factory in one go. They had to visit the premise for any deviation or the updates individually. Also, they had to be physically present in the factory for the essential operation, which could have skipped if a better solution or the framework was there. So, they built a standard layer for the different devices used in the factory for the same, and as well as other solutions were integrated with this entire application for the same. They had an essential requirement to scale the application and then have a quick and powerful way to get or retrieve the data. So, to achieve the same, they had used MongoDB in the system. As the system was to be built or extended to the cloud, the mongo DB had a provision to be used on-premises or on the cloud. Below is the sample JSON data:

```
{
  "tool":"BOSCH",
  "freq":902.5,
  "size":11,
  "seq":0,
  "port":2,
  "tmstp":1617356765,
  "ack":false,
  "info":{
    "deviceId":"35-sd-86-56-78-90-3a-df",
    "frame_counter":0,
    "frame_type":4
  },
  "messageinfo":{
    "tighteningID":"L0005689745",
    "torque":10.8,
    "speed":20.8,
```

```

        "angle":45,
        "tighteningstatus":true
    }
}

```

- **Navigation-based tracking system:** Built a tracking system for one of the most significant construction handling or building product company. Various delivery was made based on the raised request as per the MES system, and these tasks were used to assign the truck/driver to deliver the product. The main aim was to smoothen the entire process, track the truck in real-time, and track the actions based on any issues. There were various categories in the delivery type, and at least 20-25 primary used cases under it. All these were divided into various modules and then worked on. Data were received from the trucks using the OBD, a basic mobile app giving the location of trucks where these phone/tabs were installed and sent to the IoT core via MQTT or RESTful services depending on the device types. The entire application was built on AWS cloud to better scalability, enhanced, available system, serverless based system. The data received on IoT core were sent to the DB table for the sensor data using a function; there was a provision to store the data to the DB directly. They wanted to push only the specific data, so it was done using a Lambda function. Various systems or modules were built on the cloud for the application, like using S3, Lambda, Appsync, DynamoDB, IoT core, etc. The DB used was DynamoDB, and the entire application was created using Java and ReactJS. There were different tables to store the data like a sensor data table, devices tables, user management table, historical truck data. All these data were time series based, and as per the requirement, the data were received every sec. There was one requirement: all these data had to be sent to GCP to store the historical data and study these data separately. Hence all the historical data older than three months were sent to GCP. Below is the sample JSON data:

```

{
  "timestamp":"1617357441",
  "metadata":{
    "lat":"PLC0068",
    "long":"10.76.54.89",

```



```
        "type": "ISO8605"  
    }  
}
```

4. PROS & CONS

Pros	Cons
The database is easily accessible, and its implementation is simple.	DynamoDB has a weak querying model when compared to that of the relational databases.
The implementation of DynamoDB can reduce the hosting costs for applications to a great extent.	DynamoDB works on Pay per method and therefore, predicting the usage might be difficult. When the requirement increases unexpectedly it might be difficult to incorporate the additional cost to the regular budget.
Streams are supported by the database. Due to this, the other systems data will automatically change. Having this feature in the database be proven useful when the functions like view, update or aggregate are rendered.	For this database, there is a lack of server-side updates. This means if there is any change in the data model, all the records have to be updated individually which could be a tedious task.
There is an in-built feature called TTL (Time-to-live). This feature enables to replace a lot of use cases directly which requires a database like Redis normally.	DynamoDB can not process a large amount of read/writes in a short time. There are chances to write errors which will require rework on the data.
The need of backend servers is completely eliminated as there is a one-click solution in the dashboard of the database.	DynamoDB does not support the backup region wise. The backups will be in the same regions which might be a potential problem.
High scalability & reliability while supporting ACID transactions.	The way how the data is partitioned does not have any control. This can be hard from the perspective of compliance.

5. ETHICAL DATA MANAGEMENT

Today, everything is about data, and the amount of data being generated from different sources is enormous. With the devices being used these days and other sources like social media, it is known that it will generate a massive amount of data. Using millions of devices that produce data transmitted over the Internet or other communication protocol poses various security issues, which raises social and ethical implications. When technology advances faster than legal and moral structures, future governance for managing the increasing data and its consequences should be developed. If an attacker gets into the system, they will quickly get the organization's confidential data. They will have various such data, and the organization does not want to share these data to anyone and poses a threat to them.

DATA ETHICS POLICIES

- To maintain transparency, the methods for handling data must be mentioned during data collection.
- To establish efficient, ethical review practices both at the internal and external levels of the organizations.
- Data governance practices and policies must be explicitly mentioned and communicated to stakeholders at the local and global levels.
- The use of data obtained should be following the understanding and intentions of the disclosing party.
- The policies must consider the expectations of privacy and security of the data provided by the data subjects.
- Emphasis should be laid on having policies that mitigate the disparity caused to the subjects based on the data provided by them.
- To incorporate practices that promote accountability, transparency, fairness, and suitability.
- To minimize the storage, collection, and handling of personally identifiable data.
- Formulation of strict policies that would prevent the exploitation of data to target vulnerable individuals.

ETHICS WHILE HANDLING DATA

- Giving utmost importance to the fair handling of data will promote trust among data subjects.
- To minimize the storage, collection, and handling of personally identifiable data.
- To prevent the monetizing of any form of data that is collected for personal benefits.
- Promoting a compliance culture to ensure that the goals of data protection are met.
- By handling the data as per the guidelines /data ethics framework.
- By not engaging in any misuse of data so obtained.

ETHICAL/LEGAL PART OF DATA COLLECTION/STORAGE/SECURITY

- Preserving the privacy and authenticity of data without making the data readily available to third parties.
- Ensuring that the privacy policies are communicated to the stakeholders.
- Providing reassurance to the customers, public, and other beneficiaries about an organization's integrity while handling data.
- To retain multiple copies of the essential data for backup. At the same time, these copies should not be shared with any other party.
- Ensuring that unintentional sharing of data does not occur at any point in time.
- By clearly communicating the purpose of data collection to the subjects and obtaining the same only after getting their consent.
- To remain unbiased and neutral, especially during the collection of data.
- Ensuring careful anonymization of data.
- The access to the data storage must be controlled/limited to key personnel in the organizations.
- By changing the passwords at regular intervals to ensure that the data is secure.

Steps to minimize the issues related to data:

- ✓ Beneficial: is the use of the knowledge support our clients/customers as much it benefits us?
- ✓ Progressive: should we have a culture of constant change and minimization of information?
- ✓ Sustainable: Overtime, are the perspectives we associate with knowledge sustainable?
- ✓ Respectful: Were we open and inclusive?
- ✓ Fair: Have we thought about the possible impacts on all stakeholders of our data use?

6. CONCLUSION

Data have remained constant across the growing technologies, and it will always be unique and required in the coming time. It is said that data is one of the important pillars for any solution, and without data, nothing is possible in the current world. As per the discussion and study made, the data will be growing exponentially, and by 2024 the estimated data will be around 149 zettabytes, which is a huge no. There are different types of data, like structure and unstructured, required to be stored based on the requirement. There are different types of databases available based on the data structure it will be chosen. We did explain various databases and their uses and the domains where they can be applied. Various databases available are SQL: MSSQL, PostgreSQL, MySQL, Oracle, etc., and under NoSQL: MongoDB, Cassandra, HBase, DynamoDB, etc.

After storing the data, the next important thing is to have a secure way to store them, which is considered the next concern for everyone. To protect the data and beneficiary, different security protocol has to be added while building the system. Different security protocols are HTTPS, enabling key-based encryption and decryption of data. Encryption and decryption can be one of the principal securities added to add the extra layer of security. All the data stored will be encrypted and then stored; once it requires further use, it will be decrypted back to make it feasible for usage. The complete server can be used inside a secure firewall to have a secure system. If we are using the cloud, different securities protocols can be added: VPN, KMS, and IAM users. The data collected will be sent to the server, inside the firewall or VPN, and the data will be encrypted.

Further during the analysis, the data will be decrypted and will be used. In the coming time, there will be many opportunities to give our best to improve the entire system from the devices, data collection, data filtering, storage, security, application, and then analytics. Improving all these will help us to improve the world to a better place.

REFERENCES

- <https://www.quickbase.com/articles/timeline-of-database-history>
- <https://www.internetsociety.org/policybriefs/responsible-data-handling/>
- <https://www.cyient.com/about-us>
- <https://www.cedrus.digital/post/introduction-to-dynamodb-and-modeling-relational-data-part-3>
- <https://db-engines.com/en/system/Amazon+DynamoDB%3BOracle>
- https://www.mongodb.com/cloud/atlas/lp/compare-mongodb-vs-dynamodb?utm_source=google&utm_campaign=gs_apac_india_search_nbnoncompetitor_atlas_desktop&utm_term=%2Bdynamodb&utm_medium=cpc_paid_search&utm_ad=b&utm_ad_campaign_id=12314592873&gclid=CjwKCAjwgZuDBhBTEiwAXNofRPiK_n--2R8JDUhrqxQSScCduNLZIBbKQJbdUCRg-g00Ok53fuE_WRoC6GMQAvD_BwE
- <https://medium.com/@rpolding/databases-evolution-and-change-29b8abe9df3e>
- <https://www.izenda.com/relational-vs-non-relational-databases/#:~:text=To%20summarize%20the%20difference%20between,type%20of%20data%20it's%20storing>
- <https://aws.amazon.com/dynamodb/>
- <https://logz.io/blog/nosql-database-comparison/>
- <https://db-engines.com/en/system/Cassandra%3BHBase%3BMongoDB>
- <https://db-engines.com/en/system/Amazon+DynamoDB%3BCassandra%3BHBase>