NOSQL DATABASES

NoSQL databases (aka "not only SQL" or “Non SQL”) are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

Architecture Pattern is a logical way of categorizing data that will be stored on the Database. NoSQL is a type of database which helps to perform operations on big data and store it in a valid format. It is widely used because of its flexibility and a wide variety of services.

Architecture Patterns of NoSQL:

The data is stored in NoSQL in any of the following four data architecture patterns.

1. Key-Value Store Database
2. Column Store Database
3. Document Database
4. Graph Database

These are explained as following below.

1. **Key-Value Store Database:**

   This model is one of the most basic models of NoSQL databases. As the name suggests, the data is stored in form of Key-Value Pairs. The key is usually a sequence of strings, integers or characters but can also be a more advanced data type. The value is typically linked or co-related to the key. The key-value pair storage databases generally store data as a hash table where each key is unique. The value can be of any type (JSON, BLOB(Binary Large Object), strings, etc). This type of pattern is usually used in shopping websites or e-commerce applications.

   Advantages:

   - Can handle large amounts of data and heavy load,
   - Easy retrieval of data by keys.

   Limitations:

   - Complex queries may attempt to involve multiple key-value pairs which may delay performance.
   - Data can be involving many-to-many relationships which may collide.

   Examples:

   - DynamoDB

- Berkeley DB

Key:1	ID:210

Key:2	ID:411	Email: geeksforgeeks@gmail.com

Key:3	UID:219	Name: Geek	Age:20

2. **Column Store Database:**

   Rather than storing data in relational tuples, the data is stored in individual cells which are further grouped into columns. Column-oriented databases work only on columns. They store large amounts of data into columns together. Format and titles of the columns can diverge from one row to other. Every column is treated separately. But still, each individual column may contain multiple other columns like traditional databases.

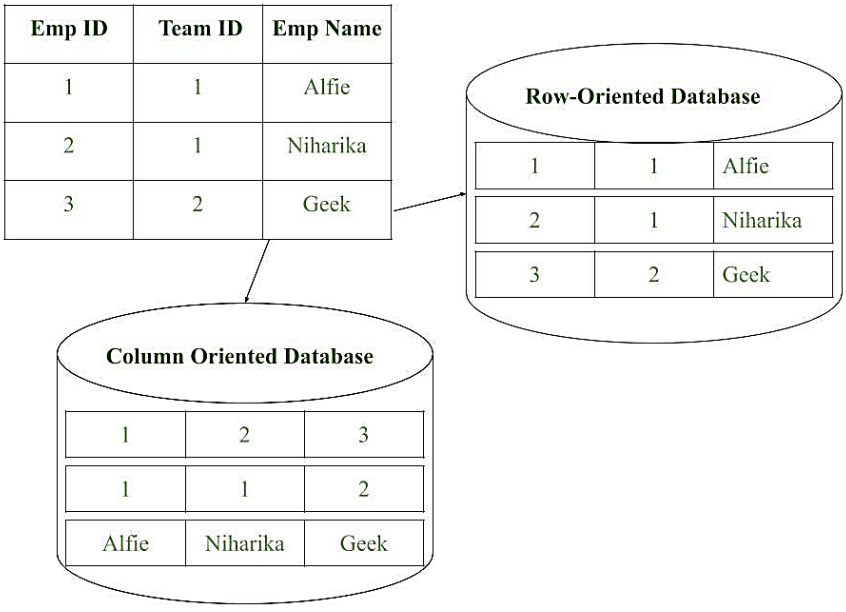
   Basically, columns are mode of storage in this type.

   Advantages:

   - Data is readily available
   - Queries like SUM, AVERAGE, COUNT can be easily performed on columns.

   Examples:

   - HBase
   - Bigtable by Google
   - Cassandra



3. **Document Database:**

   The document database fetches and accumulates data in form of key-value pairs but here, the values are called as Documents. Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format. The use of nested documents is also very common. It is very effective as most of the data created is usually in form of JSONs and is unstructured.

Advantages:

- This type of format is very useful and apt for semi-structured data.
- Storage retrieval and managing of documents is easy.

Limitations:

- Handling multiple documents is challenging
- Aggregation operations may not work accurately.

Examples:

- MongoDB
- CouchDB

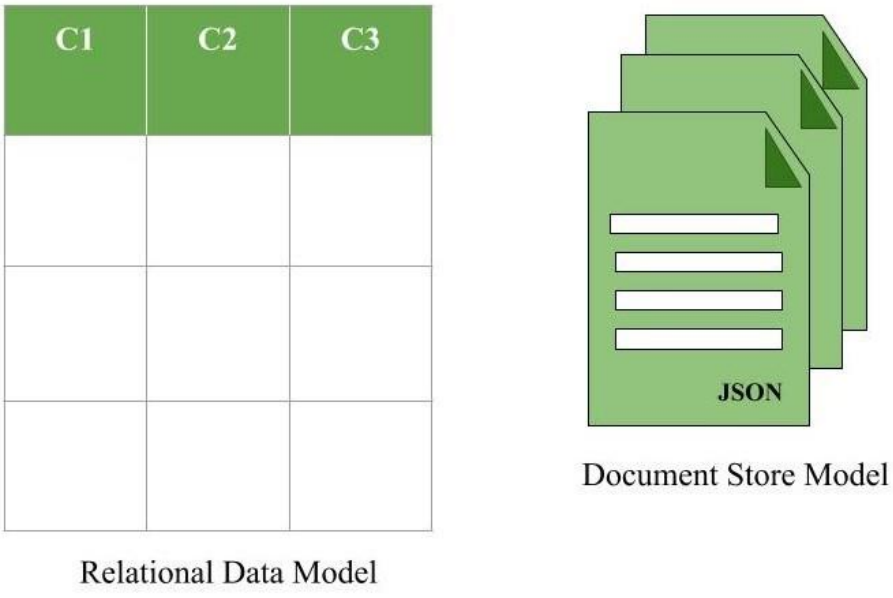


Figure – Document Store Model in form of JSON documents

4. **Graph Databases:**

   Clearly, this architecture pattern deals with the storage and management of data in graphs. Graphs are basically structures that depict connections between two or more objects in some data. The objects or entities are called as nodes and are joined together by relationships called Edges. Each edge has a unique identifier. Each node serves as a point of contact for the graph. This pattern is very commonly used in

social networks where there are a large number of entities and each entity has one or many characteristics which are connected by edges. The relational database pattern has tables that are loosely connected, whereas graphs are often very strong and rigid in nature.

Advantages:

- Fastest traversal because of connections.
- Spatial data can be easily handled.

Limitations:

Wrong connections may lead to infinite loops.

Examples:

- Neo4J
- FlockDB(Used by Twitter)

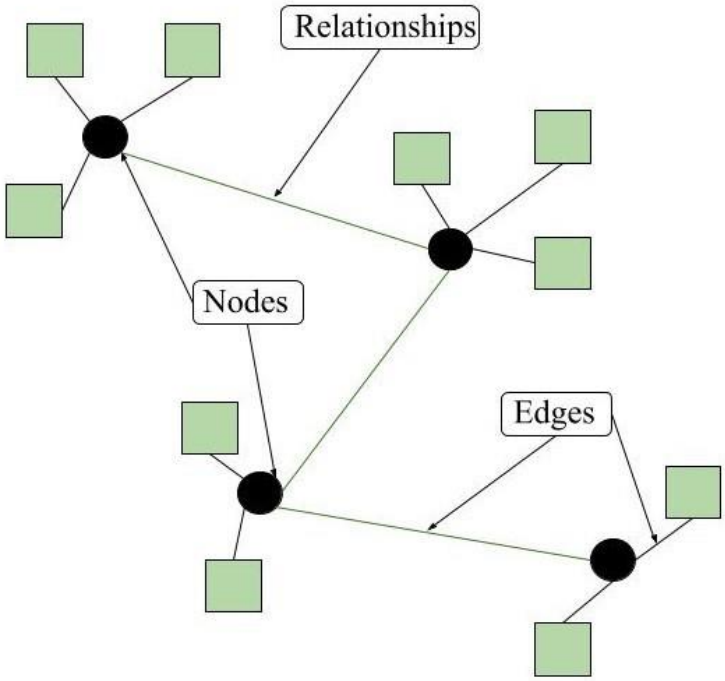


Figure – Graph model format of NoSQL Databases

BigData

Big data is a term that describes large, hard-to-manage volumes of data – both structured and unstructured – that inundate businesses on a day-to-day basis. But it's not just the type or amount of data that's important, it's what organizations do with the data that matters. Big data is a combination of **structured, semistructured and unstructured** data collected by organizations that can be **mined** for information and used in **machine learning projects**, **predictive modeling** and other **advanced analytics** applications.

An industry analyst Doug Laney articulated the now-mainstream definition of big data as the three V's:

- **Volume:** Organizations collect data from a variety of sources, including transactions, smart (IoT) devices, industrial equipment, videos, images, audio, social media and more. In the past, storing all that data would have been too costly – but cheaper storage using data lakes, Hadoop and the cloud have eased the burden.
- **Velocity:** With the growth in the Internet of Things, data streams into businesses at an unprecedented speed and must be handled in a timely manner. RFID tags, sensors and smart meters are driving the need to deal with these torrents of data in near-real time.
- **Variety:** Data comes in all types of formats – from structured, numeric data in traditional databases to unstructured text documents, emails, videos, audios, stock ticker data and financial transactions.

Later two additional dimensions (V's) are added when it comes to big data:

- **Variability:** In addition to the increasing velocities and varieties of data, data flows are unpredictable – changing often and varying greatly. It's challenging, but businesses need to know when something is trending in social media, and how to manage daily, seasonal and event-triggered peak data loads.
- **Veracity:** Veracity refers to the quality of data. Because data comes from so many different sources, it's difficult to link, match, cleanse and transform data across systems. Businesses need to connect and correlate relationships, hierarchies and multiple data linkages. Otherwise, their data can quickly spiral out of control.

Big Data is a concept of handling (storing and processing) large amount of data. Here, large mean gigabytes -> terabytes -> petabytes -> zetabytes and so on. With inception of this concept we found many emerging technologies which either facilitate the storage or processing of these data in a distributed fashion. Apache Hadoop being one of those early technologies/framework which started handling these data with it's sub-projects like HDFS, MapReduce, Hive, Pig, HBase etc. Later some advanced technologies like **Storm, Spark, Cassandra, MongoDB, CouchBase DB** etc. which very efficiently handled Big Data with much ease and efficiency.

Somtime people refers NoSQL technology as a part of BigData concept.

OBJECT ORIENTED DATABASE (OODB)

Object Database Definition

An object database is managed by an **o**bject-**o**riented **d**ata**b**ase **m**anagement **s**ystem (OODBMS). The database combines object-oriented programming concepts with relational database principles.

- **Objects** are the basic building block and an instance of a class, where the type is either built-in or user-defined.
- **Classes** provide a schema or blueprint for objects, defining the behavior.
- **Methods** determine the behavior of a class.
- **Pointers** help access elements of an object database and establish relations between objects.

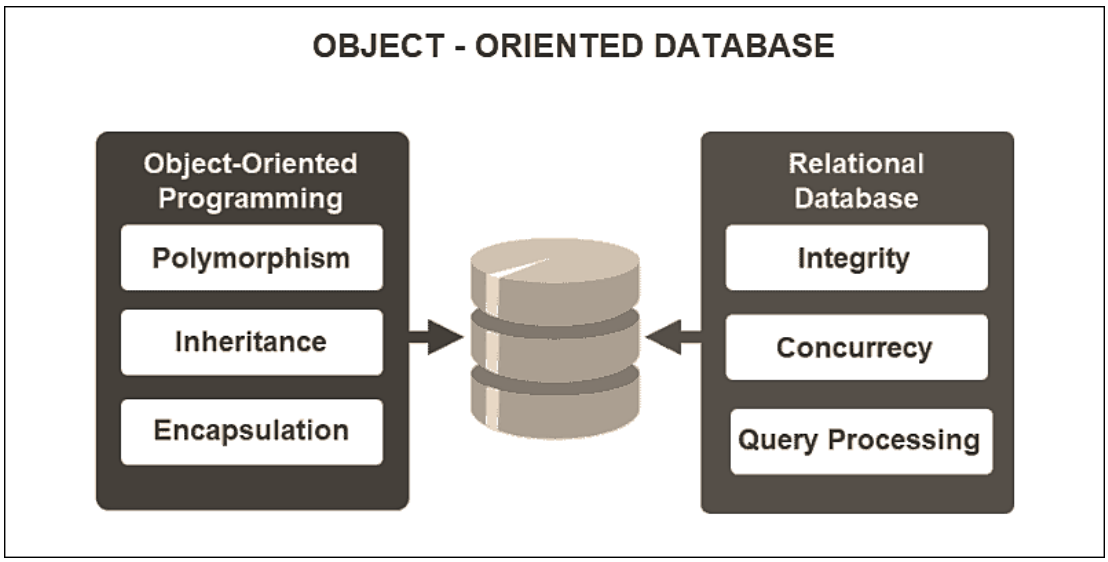


Figure: Features of OODB

The main characteristic of objects in OODBMS is the possibility of **user-constructed types**. An object created in a project or application saves into a database as is.

Object-oriented databases directly deal with data as complete objects. All the information comes in one instantly available object package instead of multiple tables.

In contrast, the basic building blocks of relational databases, such as PostgreSQL or MySQL, are tables with actions based on logical connections between the table data.

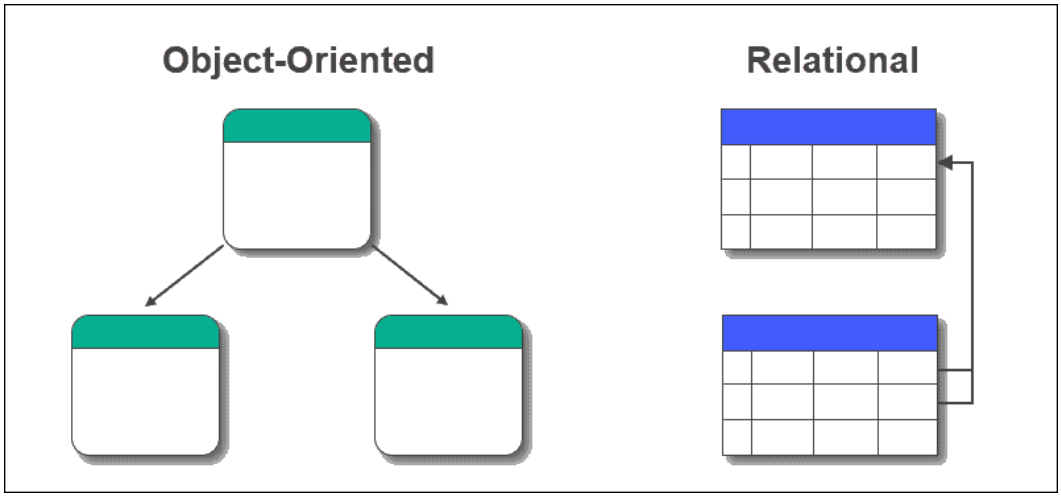


Figure: Conceptual schema of OODB

Object-Oriented Programming Concepts

Object-oriented databases closely relate to object-oriented programming concepts. The four main ideas of object-oriented programming are:

- **Polymorphism**
- **Inheritance**
- **Encapsulation**
- **Abstraction**

These four attributes describe the critical characteristics of object-oriented management systems.

Polymorphism

Polymorphism is the capability of an object to take multiple forms. This ability allows the same program code to work with different data types. Both a car and a bike are able to *break*, but the mechanism is different. In this example, the action break is a polymorphism. The defined action is **polymorphic** — the result changes depending on which vehicle performs.

Inheritance

Inheritance creates a hierarchical relationship between related classes while making parts of code reusable. Defining new types inherits all the existing class fields and methods plus further extends them. The existing class is the **parent** class, while the **child** class extends the parent.

For example, a parent class called *Vehicle* will have child classes *Car* and *Bike*. Both child classes **inherit** information from the parent class and **extend** the parent class with new information depending on the vehicle type.

Encapsulation

Encapsulation is the ability to group data and mechanisms into a single object to provide access protection. Through this process, pieces of information and details of how an object works are **hidden**, resulting in data and function security. Classes interact with each other through methods without the need to know how particular methods work.

These characteristics make object databases suitable for projects with complex data which require an object-oriented approach to programming. An object-oriented management system provides supported functionality catered to object-oriented programming where complex objects are central. This approach unifies attributes and behaviors of data into one entity.

Examples of OODB

- **GemStone/S :** GemStone/S is an object database system based on Smalltalk – an object-oriented programming language influenced by Java. Developers who write applications in Smalltalk adapt easily to this database. GemStone/S integrates seamlessly with existing Smalltalk applications, improving speed and productivity.

  Gemstone/S is best for high-availability projects. There are multiple options for licensing depending on the project size. The database server is available for various platforms, including Linux, Windows, macOS, Solaris, AIX, as well as Raspberry Pi.

- **ObjectDB : ObjectDB** is a NoSQL object database for the Java programming language. Compared to other NoSQL databases, ObjectDB is ACID compliant. ObjectDB does not provide an API and requires using one of the two built-in Java database APIs:

  JPA with **JPA Query Language** (JPQL) based on Java syntax.

  JDO with **JDO Query Language** (JDQL) based on SQL syntax.

  ObjectDB includes all basic data types in Java, user-defined classes, and standard Java collections. Every object has a unique ID. The number of elements is limited only by the maximum database size (128 TB). ObjectDB is available cross-platform and the benchmark performance is exceptional.
- **ObjectDatabase++ : ObjectDatabase++** is a real-time embeddable object database designed for server-side applications. The required external maintenance is minimal.

  ObjectDatabase++ supports:
  - Multi-process with multi-threaded server applications.
  - Full transaction control.
  - Real-time recovery.
  - C++ related languages, VB.NET as well as C#.

  The object database is C++ based. One of the main features is advanced auto-recovery from system crashes without compromising the database integrity.
- **Objectivity/DB : Objectivity/DB** utilizes the power of objects and satisfies the complex requirements within Big Data. The object database is flexible by supporting multiple languages:
  - C++
  - C#
  - Python
  - Java

  The schema changes happen dynamically without the need for downtime, allowing real-time queries against any data type. Objectivity/DB is available for multiple platforms, including macOS, Linux, Windows, or Unix.
- **ObjectStore : ObjectStore** integrates with C++ or Java and provides memory persistency to improve the performance of application logic. The object database is ACID-compliant. The responsiveness allows developers to build distributed applications cross-platform, whether on-premises or in the cloud.

  The main feature is cloud scalability, which allows database access from anywhere. ObjectStore simplifies the data creation and exchange process seamlessly.

- **Versant : Versant** provides primary transparent object persistence from C++, Java, and .NET. However, there is also support for Smalltalk and Python. Versant supports different APIs depending on the language used. Standard SQL queries are also available, making Versant a NoSQL database.

  The object database is a multi-user client-server database. Versant performs best when used for online transaction systems with large amounts of data and concurrent users.

Object-Oriented Database Advantages and Disadvantages

Every database modeling technique has advantages and disadvantages. Before opting in for object-oriented databases, you must know the available languages in addition to the application intent.

Advantages	Disadvantages
The main advantages are: • Complex data and a wider variety of data types compared to MySQL data types. • Easy to save and retrieve data quickly. • Seamless integration with object-oriented programming languages. • Easier to model the advanced real world problems. • Extensible with custom data types.	Some disadvantages include: • Not as widely adopted as relational databases. • No universal data model. Lacks theoretical foundations and standards. • Does not support views. • High complexity causes performance issues. • An adequate security mechanism and access rights to objects do not exist.

SPATIAL DATABASE

Spatial data is associated with geographic locations such as cities,towns etc. A spatial database is optimized to store and query data representing objects. These are the objects which are defined in a geometric space.

Characteristics of Spatial Database

A spatial database system has the following characteristics

- It is a database system
- It offers spatial data types (SDTs) in its data model and query language. Some of the functions of SDT models are listed bellow -
    - **Spatial Measurements:** Computes line length, polygon area, the distance between geometries, etc.
    - **Spatial Functions:** Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.
    - **Spatial Predicates:** Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a residence located within a mile of the area we are planning to build the landfill?'
    - **Geometry Constructors:** Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.
    - **Observer Functions:** Queries which return specific information about a feature such as the location of the center of a circle.
- It supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join.

In general, spatial data can be of two types −

- Vector data: This data is represented as discrete points, lines and polygons
- Rastor data: This data is represented as a matrix of square cells.

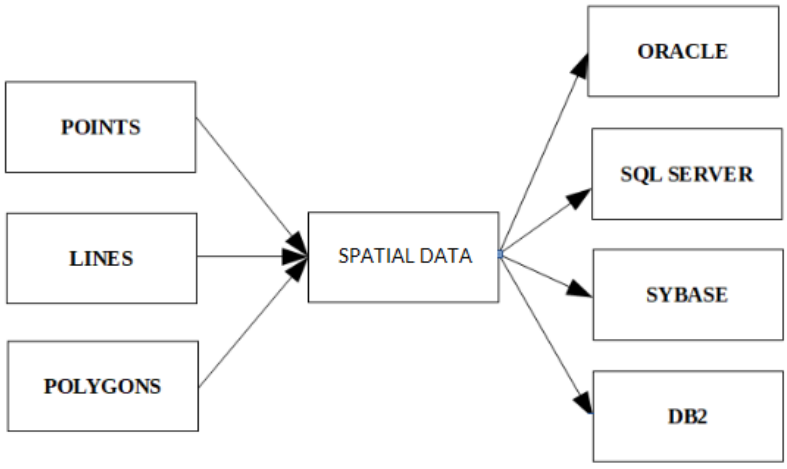


Figure : The spatial data in the form of points, lines, polygons etc. is used by many different databases as shown above.

Example

A road map is a visualization of geographic information. A road map is a 2-dimensional object which contains points, lines, and polygons that can represent cities, roads, and political boundaries such as states or provinces.

DATA WAREHOUSING

A data warehouse is **a central repository of information that can be analyzed to make more informed decisions or extraction of Knowledge**. Data flows into a data warehouse from transactional systems, relational databases, and other sources, typically on a regular cadence.

Data warehouses are characterized by being (By Willium H. Inmon):

1. **Subject-oriented:** A data warehouse typically provides information on a topic (such as a sales inventory or supply chain) rather than company operations.
2. **Time-variant:** Time variant keys (e.g., for the date, month, time) are typically present.
3. **Integrated:** A data warehouse combines data from various sources. These may include a cloud, relational databases, flat files, structured and semi-structured data, metadata, and master data. The sources are combined in a manner that's consistent, relatable, and ideally certifiable, providing a business with confidence in the data's quality.
4. **Persistent and non-volatile:** Prior data isn't deleted when new data is added. Historical data is preserved for comparisons, trends, and analytics.

Key attributes of most data warehouses are that they:

- Are often deployed as a **central database for the enterprise** and primarily focused on Online Analytical Processing (**OLTP**).
- **Provide ETL** (extract, transform, load) data processing capability. The ETL process requires a staging area where data is transformed before it enters the data warehouse for analysis. NOTE: An alternate pattern could exist wherein data can be transformed upstream in data lakes as well and fed into a data warehouse
- Store metadata.
- Stores the **aggregated data** in **Fact Tables** in a **multi-dimensional space** attributed by multiple **dimension tables**. Smallest unit of information is called a **data cube**.
- Different schema are used to design and implement Data warehouses namely **Star Schema, Snowflake Schema and Galaxy Schema.**
- Include access to reporting tools. BI tools such as PowerBI, QlikSense, and Tableau may connect to the Warehouse through built-for-purpose drivers or they may leverage SQL for queries—albeit it is important to guard against unbounded queries that could impact performance. Often, there are a set of certified reports that are frequently and automatically refreshed.