Relational and Non Relational Databases: A Review

Nimesh Thakur, Nishi Gupta

Department of Mathematics, University Institute of Sciences, Chandigarh University, Gharuan, Mohali, Punjab-140413, India
Email: 20msm3054@cuchd.in

Relational and non-relational databases are the two types of databases that are used to store data and perform different operations on it. For data storage, they use a variety of formats. In this paper, we'll try to figure out what they're all about and what the main differences are. Databases serve as a data centre from which information is collected and processed. Data science is a multidisciplinary field that combines mathematics, statistics, and programming to research data. For a data scientist, a basic understanding of databases is a must-have ability. We'll look at how a data scientist can make the most of different database types.

Keywords: Database; Big Data; Relational Database Systems; Internet Technologies; Non Relational Database Systems;

1. INTRODUCTION

To put it another way, a database is a repository of data where data is stored, accessed to perform desired operations on data, and then unadulterated or corrupted or new data is stored back.

As the name implies, a relational database establishes relationships between data. Tables are used to store data, and these tables are linked in some way. Such that you can picture one table being connected to another table, which is related to another table, and so on. This paper will look at these connections and how they are connected. Database management systems are software for application-independently describing, storing, and querying data. All database management systems contain a storage and a management component[1].

Non-relational databases, also known as No-sql databases, take a different approach to data organisation than relational databases, modelling data in ways other than tabular relations. Non-relational databases were created to address the limitations of relational databases.

Data science is highly reliant on data, and a data scientist must be able to perform desired data operations. For relational databases, the Structured Query Language (SQL) is the traditional user and application programme interface, while mongo-db is common for non-relational databases. As great volume of data is generated over the last few years the demand in storing, processing and analysing data efficiently has increased and can't be handled by traditional systems. Therefore the term big data has been coined. The term "big data" refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods. The act of accessing and storing large amounts of information for analytics has been around a long time[2]. As with the traditional technologies, big data technologies are used for many tasks, including data engineering[3].

2. BACKGROUND

From the mid-1960s on-wards, the availability of direct-access storage (discs and drums) coincided with the emergence of the term database.

A relational database is a tabular database that was introduced by E.F. Codd at IBM in 1970 and allows data to be reorganised and viewed in a variety of ways[4].

Carlo Strozzi used the term NoSQL in 1998 to describe his open-source relational database Strozzi NoSQL, which did not expose the SQL interface but was still relational. [two] His NoSQL RDBMS is distinct from the general definition of NoSQL databases that emerged around 2009. Since the new NoSQL trend "departs from the relational paradigm entirely," according to Strozzi, it should have been dubbed - The name attempted to label the emergence of an increasing number of non-relational, distributed data stores, including open source clones of Google's Bigtable/MapReduce and Amazon's DynamoDB. While object-oriented languages succeeded in becoming the major force in programming, object-oriented databases faded into obscurity[5].Object relational mapping is a way to connect object oriented paradigm with the data base . Object relational mapping came in play which and this exposed loopholes in traditional database system

3. REALTIONAL DATABASE

A relational database is a list of data items that are linked together in some way. Tables with rows and columns are used to store information in these objects. A database table is made up of rows and columns. The data is defined in the first column, which is called the header column. Except for the header column, each column has a datatype. Columns can't have values that aren't part of the data type. Consider the following example.

ID	salary	Name	city	age
001	20000	nim	Mumbai	22
002	30000	tin	Chennai	30

Each column header has an unique type and each column contains similar values corresponding to thier column type. A value that describes the object is stored in the row. In this case, row values are often referred to as entities. Column headers are referred to as attributes. As you can see, each column contains a similar value.

Kinds of data-type in Relational databases are- Numeric data types such as int, tinyint, bigint, float, real etc. Date and Time data types such as Date, Time, Datetime etc. Character and String data types such as char, varchar, text etc. Unicode character string data types, for example nchar, nvarchar, ntext etc. Binary data types such as binary, varbinary etc. Miscellaneous data types – clob, blob, xml, cursor, table etc.

It would be best if we look at an example query for creating a database -

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

A table with the attributes PersonID, LastName, FirstName, City, and Address is created using the keywords build and table. When we insert values and build a row, we're making a person entity. As a result, a row is often referred to as an entity. A semicolon often appears at the end of a sentence to indicate that it has come to an end.

It's critical that we have distinct values to separate this row from the others in the table. As a result, we'll need a column that distinguishes each row from the others. This is accomplished in the Persons table by the PersonId sector, which is special for each record. This is referred to as the primary key. There should be only one primary key in any relational database. Since a primary key has to be unique it cannot be null and the value is incremented by the database if specified.

The international Key is another important term. The international key in one table is the primary key in another. Since it binds two columns, a foreign key is often known as a referential restriction. After the primary key, a foreign key is typically produced. It is possible for it to be null, and it is not immediately auto generated. In your table, you may have several international keys. Also there are different types of relations among data entities.

Foreign and primary keys are two essential concepts to consider when working with relational databases.

Databases consists of many tables and these table are defined using entity-relation diagram. A completed entity-relation diagram represents the overall, logical plan of the database[6].

Types of relational ships

1. One-to-One Relationship - When each record in one table is compared to just one record in the other table, this is known as a one-to-one relationship.
2. One-to-Many or Many-to-One Relationship - When each record in one table can be linked to one or more records in the other table, this is known as a one-to-many or many-to-one relationship. The most popular type of partnership discovered is this one. Depending on how we look at it, a one-to-many relationship can be defined as a many-to-one relationship.
3. Relationship of Many-to-Many - When each record of the first table can be related to one or more records of the second table, and a single record of the second table can be related to one or more records of the first table, such a relationship occurs.

In the following example we will get an intuitive idea of how these relationships work

Husband <——— one to one ———> wife

team <——— one to many ———> Players

student<———many to many ———> subject

Drawbacks of relational databases:

1. Impedance mismatch between the object-oriented and the relational world.
2. The relational data model doesn't fit in with every domain.
3. Difficult schema evolution due to an inflexible data model.
4. Weak distributed availability due to poor horizontal scalability.
5. Performance hit due to joins, ACID transactions and strict consistency constraints (especially in distributed environments).

4. NON-RELATIONAL DATABASE OR NOSQL DATABASE

As previously mentioned, NoSql does not organise data using a tabular schema of rows and columns. Non-relational databases, on the other hand, use a storage model that is specific to the database form. The term "NoSQL" refers to data stores that don't use SQL to query their data and instead rely on a different construct. Despite the fact that they support SQL-compatible requests, NoSQL refers to a non-relational database. On the other hand, query execution varies greatly from that of the RDBMS.

Non-relational databases are often used when large quantities of complex and diverse data need to be organized. For example, a large store might have a database in which each customer has their own document containing all of their information, from name and address to order history and credit card information. Despite their differing formats, each of these pieces of information can be stored in the same document.

Non-relational databases are also quicker than relational databases because a question doesn't have to look at several tables to get a response. Non-relational databases are thus suitable for storing constantly changing data or for applications that deal with a wide range of data types. They can handle large volumes of complex, unstructured data and help rapidly evolving applications that include a dynamic database that can change quickly.

The parts that follow explain the various types of non-relational or NoSQL databases.

4.1. Document data stores

In an organisation known as a log, a document data store maintains a collection of named string fields and object data values. Data is usually stored in the form of JSON documents in these data stores.

key	Document
001921	{ name: {first: "Alan", last: "Turing" }, birth: new Date('Jun 23, 1912'), death: new Date('Jun 07, 1954'), contribs: ["Turing machine", "Turing test", "Turingery"], views : NumberLong(1250000)

Data is organised into columns and rows in a columnar or column-family data store. In its most basic form, a column-family data store may resemble, at least conceptually, a relational database. An example of two column families, Identity and Contact Details, is shown in the diagram below.

customer ID	column Family : Identity
001	First Name : nim Last Name :tin
002	First Name : kyo Last Name :pan

customer ID	column Family : contact Info
001	phone number : 9167519 email: nim@gmail.com
002	phone number : kyo Last Name :pan

4.2. Key/value data stores

The key-value store is the simplest of the NoSQL stores, and it is simply a list of key-value pairs stored within an entity, as the name implies. Key-value databases are highly partitionable and allow horizontal scaling at scales that other types of databases cannot achieve[7].

4.3. Graph data stores

A graph data store keeps track of two types of data: nodes and edges. Nodes represent entities, while edges define the connections between them. Similar to columns in a table, nodes and edges may have properties that provide details about them. The essence of the relationship can also be shown by the orientation of the edges.

The goal of a graph data store is to allow an application to run queries that traverse the network of nodes and edges quickly and analyse the relationships between entities.

4.4. Time series data store

A time series data store is designed specifically for this form of data, which is a collection of values ordered by time. Since they usually receive vast quantities of data in real time from a wide variety of sources, time series data stores must be able to handle a large number of writes.Having access to detailed, feature rich time-series data has become one of the most valuable commodities in our information-hungry world[8].

4.5. Object data store

Large binary objects or blobs, such as images, text files, video and audio sources, large device data objects and

records, and virtual machine disc images, are stored and retrieved using object data stores. The stored data, some metadata, and a specific ID for accessing the object make up an object. Object stores are designed to handle large files independently while still providing a large volume of overall capacity to accommodate all files.

Object storage is a relatively new option for data storage, optimized for general binary or unstructured data, often multimedia[9].

5. CASE STUDY

Apache HBase is a NoSQL database that runs as a distributed and flexible large data store on top of Hadoop. This means that HBase will use the Hadoop Distributed File System (HDFSdistributed)'s computing paradigm and learn from Hadoop's MapReduce programming model. It's designed to run on a cluster of commodity hardware and host massive tables with billions of rows and theoretically millions of columns. However, HBase is a powerful database in and of itself, combining real-time query functionality with the speed of a key/value store and offline or batch processing through MapReduce.

HBase is not a relational database and requires a different approach to modeling your data. HBase actually defines a four-dimensional data model

1. Row key
2. Column qualifier
3. Column family
4. Version

A single row can be accessed using the row key and is made up of one or more column families. Each column family has one or more column qualifiers (referred to as "columns" in Figure 1), and each column may have several copies. You'll need to know the row key, column kin, column qualifier, and version to get to a specific piece of info.

When creating an HBase data model, it's a good idea to consider how the data will be used. HBase data can be accessed in two ways:

1. Through their row key or via a table scan for a range of row keys
2. In a batch manner using map-reduce

HBase's dual-approach to data access is one of its most powerful features. Often, saving data in Hadoop ensures that it is suitable for offline or batch processing (which it excels at), although not inherently for real-time access. HBase solves this by serving as both a key/value store for real-time analysis and a map-reduce engine for batch processing.

The meaning is the set of column families, and the key is the row key we discussed earlier (that have their associated columns that have versions of the data). You can get the value associated with a key; in other words, you can "get" the row associated with a row key, or you can "get" a series of rows by providing the beginning row key and ending row key, which is known as a table check. In a real-time question, you can't look up values in columns, which brings up a significant point: design of a row key.

6. CONCLUSIONS

Data is stored in both relational and non-relational databases, and neither can do something that the other can't. When comparing relational and non-relational databases, it's important to remember that these two kinds of databases are both useful in their own right—but for various purposes and usage cases. There is no one-size-fits-all database, and both relational and non-relational databases have their uses.

Instead of the Structure Query Language (SQL) used by relational databases, the NoSQL database uses Object-relational-mapping (ORM). The concept of ORM is the ability to write queries using your preferred programming language. Some of the more popular ORMs are Java, Javascript, .NET and PHP.

Relational databases are good for structured data, whereas NoSQL databases are good for unstructured data. NoSQL is capable of storing vast volumes of data with minimal structure. NoSQL databases are more scalable than relational databases.

Most application development today is done in object-oriented programming languages, which leads to a common criticism of the SQL data model: if data is stored in relational tables, an awkward translation layer is required between the objects in the application code and the database model of tables, rows, and columns. The disconnect between the models is sometimes called an impedance mismatch[10].

As a result, Nosql has the scalability and versatility to adapt to evolving market needs. MySql, Oracle, Sqlite, Postgres, and MS-SQL are examples of SQL databases. MongoDB, BigTable, Redis, RavenDb, Cassandra, Hbase, Neo4j, and CouchDb are examples of NoSQL databases.

It is said that RDBMS is vertical scalable whereas NoSQL is both horizontally and vertically scalable. SQL databases are best fit for heavy duty transactional type applications and are secure.

Depending upon the type of data you are working with, you can pick one over the other.

REFERENCES

[1] Andreas Meier, Michael Kaufmann. *Models, Languages, Consistency Options and Architectures for Big Data Management*. 2019.

[2] SAS, *Big Data What it is and why it matters*. https://www.sas.com/en_in/insights/big-data/what-is-big-data.html

[3] Foster Provost and Tom Fawcett,"Data Science and its Relationship to Big Data and Data-Driven

Decision Making" *Big DataVol. 1, No. 1* (2013) *https://doi.org/10.1089/big.2013.1508* 20 July 2021

[4] wikipedia, *Relational Database.* https://en.wikipedia.org/wiki/Relational_database.

[5] Fowler, M., Sadalage, P. J. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence.* (n.p.): Pearson Education.

[6] Harrington, J. L. (2002). *Relational Database Design Clearly Explained, Second Edition (The Morgan Kaufmann Series in Data Management Systems).* Netherlands: Morgan Kaufmann Publishers.

[7] Amazon, *What Is a Key-Value Database.* https://aws.amazon.com/nosql/key-value/

[8] Timescale, *What the heck is time-series data (and why do I need a time-series database)?* https://blog.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/

[9] IBM, *Object storage benefits, myths and options.*https://www.ibm.com/blogs/cloud-computing/2017/02/01/object-storage-benefits-myths-and-options/

[10] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.* United States: O'Reilly Media.

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my mentor Nishi Gupta for her exemplary guidance, monitoring and constant encouragement throughout the process.