**WEEK 1: INTRODUCTION**

In the 1970s when relational databases came into the picture, the data schemas to be worked upon were reasonably elemental and simple wherein the data items were to be arranged as a set of formally described tables with rows and columns. But with the need to store volumes and a variety of data (unstructured) in recent years, non-relational database technologies have emerged to address the requirement that allows data to be grouped together more naturally and logically. One of the most popular ways of storing data is with a document-oriented database, employed for storage, management, and retrieval of semi-structured data where each record and its associated data is considered a “document”. A document-oriented database is also termed a document store or simple document.

The NoSQL movement began in the early years of the 21st century when the world started its deep focus on creating web-scale databases. By web-scale, we mean scale to cater to hundreds of millions of users and now growing to billions of connected devices including but not limited to mobiles, smartphones, internet TV, in-car devices, and many more. While some authors refer to "not only SQL", NoSQL originally started off as a simple combination of two words—No and SQL—clearly and completely visible in the new term.

NoSQL databases responded to the feeling of "I don't want to use SQL" (Vaish, 2013, page 8). After many years of relational database prevalence, new types of data emerged, which are not well suited to be stored in tables. Think for example of pictures, voice messages, songs, and videos just to name a few. How will you fit the content of those files in a relation?

The term NoSQL refers to any databases that do not follow the relational database management system (RDBMS) principle. Furthermore, NoSQL databases are built to handle the speed and growth of the likes of Google, Facebook, Twitter, and other social media websites. NoSQL databases handle both structured and unstructured data, and one of their main benefits is their ability to process Big Data quickly. Big Data refers to “huge, overwhelming, and uncontrollable amounts of information.” With the emergence of the Web, Social Media, and other technologies, huge, overwhelming, and uncontrollable amounts of data can be and are being stored by business organizations. Traditional relational databases cannot handle these volumes and types of data efficiently, which drove computer scientists to create what we now know as NoSQL databases.

**WEEK 2: WHAT IS NoSQL?**

**I. WHAT IS NoSQL?**

**Databases**

Before diving into NoSQL, we should review the concept of relational databases and their structured query language or SQL. A database is a collection of related data.  A Database Management System (DBMS) is a software package/system that facilitates the definition, construction, manipulation, and sharing functions of a computerized database. A DBMS has the following functionality:

* Define a particular database in terms of its data types, structures, and constraints.
* Construct or load the initial database contents on a secondary storage medium.
* Manipulate the database:
  + Retrieval: Querying, generating reports
  + Modification: Insertions, deletions, and updates to its content
  + Accessing the database through Web applications
* Share a database allows multiple users and programs to access the database simultaneously.

Data can be stored in a conventional file or in a database. A database offers significant advantages. In the database approach, a single repository of data is maintained that is defined once and then accessed by various users.

The major differences between a Database and File are:

* The database is self-described.
* In the database, there is insulation between programs and data.
* The database supports multiple views of the data.
* The database supports sharing of data and multiuser transaction processing.

##### **Self-describing nature of a database system**

* The Database system contains not**only the** database itself but**also a** complete definition of the database structure and constraints.
* The information stored in the catalog is called meta**-data (data about data)**, and it describes the structure of the primary database.

##### **Insulation between programs and data**

* In file processing, any changes to the structure of a file may require changing all programs that access the file.
* In a database system, the structure of data files is stored in the DBMS catalog separately from the access program.
* This is called program-data independence.

##### **Support of multiple views of the data**

* Each user may see a different view of the database, which describes only the data of interest to that user.
* It may also contain some virtual data that is derived from the database files but is not explicitly stored.

##### **Sharing of data and multi-user transaction processing**

* The DBMS allows a set of concurrent users to retrieve from and update the database.
* Concurrency control within the DBMS guarantees that each transaction is correctly executed or aborted. For example, when several reservation clerks try to assign a seat on an airplane flight.
* These types of applications are generally called online**transaction processing (OLTP)**

##### **Controlling Redundancy**

* Controlling Redundancy is one of the most important features of a DBMS.
* In the traditional file approach, each group independently keeps its own file. For example, the accounting office keeps data on registration and billing info; whereas the registration office keeps track of registration, student courses, and grades.

##### **Other Advantages of using the DBMS approach**

* It restricts unauthorized access to data.
* It provides Storage Structures (e.g., indexes) for efficient Query Processing
* It provides backup and recovery services.
* It provides multiple interfaces to different classes of users.
* It can reflect complex relationships among data.

##### **Disadvantages:**

• It increases the opportunity for persons or groups outside the organization to gain access to information about the firm's operation.

• It increases the opportunity for a fully trained person within the organization to misuse the data resources intentionally.

• The database approach is costly due to higher hardware and software requirements.

• Database systems are complex (due to data independence), difficult, and time-consuming to design.

• Damage to the database affects virtually all application programs.

• Extensive conversion costs in moving from a file-based system to a database system.

• Initial training is required for all programmers and users.

#### The Relational Database Management System (RDBMS)

In an RDBMS data is structured in database tables, fields, and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields. RDBMS store the data into a collection of tables, which might be related by common fields (database table columns). RDBMS also provides relational operators to manipulate the data stored in the database tables. Most RDBMS use SQL as a database query language.

Edgar Codd introduced the relational database model. Many modern DBMS do not conform to Codd’s definition of an RDBMS, but nonetheless, they are still considered to be RDBMS. The most popular RDBMS are Microsoft SQL Server, Oracle, and MySQL.

##### **Features of an RDBMS:**

* It stores data in tables.
* Tables have rows and columns.
* These tables are created using SQL.
* And data from these tables are also retrieved using SQL.

##### **Difference between DBMS and RDBMS**

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS stands for Database Management System, which consists of tables that have no relations among themselves | RDMBS stands for Relational Database Management System, in which there are relationships between the tables. |
| The Relationship between tables in DBMS is Physical | The relationship in RDBMS is Logical. |
| DBMS does not support Data Integrity | RDBMS supports Data integrity |
| DBMS does not support Structural Independence and Advanced Query Capabilities | RDBMS supports Structural Independence and Advanced Query Capabilities |
| DBMS does not support Security | RDBMS supports Security on Databases. |

An RDBMS must provide appropriate languages and interfaces for each category of users to express database queries and updates. Database Languages are used to create and maintain databases. The most common database language for relational databases is the **Structured Query Language or SQL**. SQL is a standard query language certified by ANSI and ISO. SQL is used to access different databases like Microsoft SQL Server, MySQL, MS Access, Oracle, and more.

**SQL** statements can be categorized as data definition language (DDL), data control language (DCL), and data manipulation language (DML).

##### **Data Definition Language**

This is the language that allows the users to define data and their relationship to other types of data. It is mainly used to create files, databases, data dictionaries, and tables within databases.

##### **Data Manipulation Language**

This is the language that provides a set of operations to support the basic data manipulation operations on the data held in the databases. It allows users to insert, update, delete and retrieve data from the database. The part of DML that involves data retrieval is called a query language.

##### **Data Control Language**

DCL statements control access to data and the database using statements such as GRANT and REVOKE. A privilege can either be granted to a User with the help of a GRANT statement or it can also be revoked (taken back) by using the REVOKE command.

###### **INSTANCES, SCHEMAS, AND SUBSCHEMA in DBMS**

The collection of information stored in the database at a particular moment is called an **instance** of the database. The overall design of the database is called the database **schema**. The schema will remain the same while the values filled into it change from instant to instant. The data in the database at a particular moment of time is called a database state or snapshot, which is also called the current set of occurrences or instances in the database. A subschema is a subset of the schema and inherits the same property that a schema has. The plan (or scheme) for a view is often called subschema.

##### **Data Independence**

Data Independence refers to the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. A major objective of a database's three-level architecture is to provide data independence, which means that upper levels are unaffected by changes in lower levels. There are two kinds of data independence: Logical data independence and physical data independence. Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemas. Physical data independence indicates that the physical storage structures or devices could be changed without affecting conceptual schema.

In the 1970s when relational databases came into the picture, data schemas to be worked upon were reasonably elemental and simple wherein the data items were to be arranged as a set of formally defined tables with rows and columns. However, in recent years we have faced the need to store large volumes and a wide variety of data types. To address these needs, non-relational database technologies such as document-oriented, graph-based, column-based, key-value, and hybrid have emerged, which allow data to be grouped together more naturally and logically as opposed to in tables. One of the most popular ways of storing data is in a document-oriented database. A document-oriented database is employed for the storage, management, and retrieval of semi-structured data where each record and its associated data is considered to be a “document”. A document-oriented database also termed a document store or simple document is one of the most common types of NoSQL databases.

When we need to store attributes of objects such as employees, students, products, services, suppliers, and customers, the relational database model fits our data very well. However, when we want to store and manage images, recordings, text messages, video messages, and other types of unstructured data, the relational database model becomes inefficient and ineffective. These new types of unstructured data can be best managed in NoSQL databases:

A **document-oriented database**, or document store, is a computer program and data storage system designed for storing, retrieving, and managing document-oriented information, also known as semi-structured data. Document-oriented databases are the most popular NoSQL databases, and this course focuses on this type of database.

A graph **database (GDB)**, on the other hand, is a database that uses graph structures for semantic queries with nodes, edges, and properties to represent and store data. A key concept of the system is the graph (or edge or relationship). The graph relates the data items in the store to a collection of nodes and edges, the edges representing the relationships between the nodes.

A column**-oriented DBMS** or columnar DBMS is a database management system (DBMS) that stores data tables by column rather than by row. In a relational database, querying by column is a common task. For applications requiring frequent retrieval of columns, a column-oriented database proves to be more efficient. It is important to note;l however, that column-oriented databases are less efficient when inserting new data.

A **key–value database**, or key–value store, stores, retrieves, and manages associative arrays, structured as a dictionary or hash table. Dictionaries contain a collection of objects, or records, which in turn have many different fields within them, each containing data. These records are stored and retrieved using a key that uniquely identifies the record and is used to find the data within the database.

### **What is a NoSQL Database?**

When people use the term “NoSQL database,” they typically use it to refer to any non-relational database. Some say the term “NoSQL” stands for “non-SQL” while others say it stands for “not only SQL.” Either way, most agree that NoSQL databases are databases that store data in a format other than relational tables. As discussed earlier, NoSQL databases are the response to the need to store unstructured data. Unstructured data is not arranged according to a pre-set data model or schema like a table or collection of tables, and therefore cannot be stored in a traditional relational database. Text and multimedia are two common types of unstructured content. Many business documents are unstructured, as are email messages, videos, photos, webpages, and audio files.

##### **Features of NoSQL Databases**

Each type of NoSQL database has its own unique features. The following are attributes common to all types:

###### **Flexible schemas**

Unlike SQL databases, where you must determine and declare a table's schema before inserting data, a NoSQL database does not require its documents to have the same schema. That is:

* The documents in a single collection do not need to have the same set of fields and the data type for a field can differ across documents within a collection.
* To change the structure of the documents in a collection, such as adding new fields, removing existing fields, or changing the field values to a new type, update the documents to the new structure.
* This flexibility facilitates the mapping of documents to an entity or an object. Each document can match the data fields of the represented entity, even if the document has substantial variation from other documents in the collection.

###### **Horizontal Scaling**

Horizontal scaling refers to bringing on additional nodes to share the load. This is difficult with relational databases due to the difficulty in spreading out related data across nodes. With NoSQL databases, this is made simpler since collections are self-contained and not coupled relationally. This allows them to be distributed across nodes more simply, as queries do not have to “join” them together across nodes.

###### **Fast queries due to the data model**

Queries in NoSQL databases can be faster than in SQL databases. Why? Data in SQL databases are typically normalized, so queries for a single object or entity require you to join data from multiple tables. As your tables grow in size, the joins can become expensive. However, data in NoSQL databases are typically stored in a way that is optimized for queries. Queries typically do not require joins, so the queries are very fast.

###### **Ease of use for developers**

Some NoSQL databases map their data structures to those of popular programming languages. This mapping allows developers to store their data in the same way that they use it in their application code. While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.

**WEEK 3: TYPES OF NoSQL DATABASES**

NoSQL databases exist in four different types:

1. Document databases
2. Key-value stores
3. Column-oriented databases
4. Graph databases

While they share some common characteristics, they also have difference and each type is best suited for specific applications. The following paragraphs describe each type.

**Document Databases**

Document databases store data in JSON, BSON , or XML documents. JSON or JavaScript Object Notation is an open data interchange format that is both human and machine-readable.  JSON is independent of any programming language and many applications produce their output in JSON documents. BSON or Binary Javascript Object Notation is a binary-encoded serialization of JSON documents. BSON provides support for data types not supported by JSON. XML or Extensible Markup Language is a simple text-based format for representing structured information: documents, data, configuration, books, transactions, invoices, and much more.

An important characteristic of a document database is that documents can be nested. Furthermore, elements in the database can be indexed for faster querying. Because the data is stored in a way that resembles the physical application, it can be stored and retrieved with less overhead and translation compared to a relational database. Document databases have gained acceptance among developers because of their flexibility to modify their document structures when an application requires it. Application requirements may change over time and the ability of a document database to adapt to the application is a big plus. Document databases are also easily scalable which makes it easier to add or remove data and or traffic. Document databases are used in eCommerce, trading platforms, and mobile applications in a wide variety of industries.

Commercial Document Databases:

* MongoDB
* DynamoDB
* CosmosDB

**Key-Value Stores**

A blue rectangular box with white text

Description automatically generatedThis is the most basic type of NoSQL. Every data element in the database is stored as a key value pair that consists of an attribute name (the "key") and a value. A key-value store is similar to a relational database but the tables have only two columns: the key or attribute name and the value.

Key-value stores are used in shopping carts, user preferences, user profiles, etc. Commercial key-value stores:

* Amazon DynamoDB
* Aerospike
* Berkeley DB
* Couchbase
* Memcached
* Riak
* Redis

**Column-Oriented Databases**

A column-oriented database is organized as a set of columns. The result is that when you run a query on a small number of columns, the query returns just those columns directly without consuming memory with unwanted data. When multiple columns are of the same type compressing them is more efficient and they can be read faster. They are widely used in business analytics. Calculating the total of a column can be done very quickly. On the negative side, they are inefficient when writing data to a disk.

Commercial Column Databases:

* Apache Cassandra
* DataStax
* Microsoft Azure Cosmos DB
* ScyllaDB

**Graph Databases**

A graph database stores nodes and relationships instead of tables, or documents. Each element is stored as a node (such as a person in a family tree). The connections among elements are called links or relationships. The connections between the elements of the database are considered to be first-class that are stored directly.  A graph database is more efficient than a relational database in the search for connections among elements as these connections are also elements stored in the database.

A diagram of a group

Description automatically generated

Graph databases are used for fraud detection, social networks, knowledge graphs, and much more. Commercial graph databases:

* Nebula Graph
* Neo4j
* Ontotext GraphDB

**WEEK 4: INTRODUCTION TO MongoDB**

MongoDB is a database, meaning a structured way to store and access data. More specifically, it is a NoSQL database and it is categorized as a document database. As discussed earlier, this means that we store data in an organized way, but not in tables that have rows and columns. Instead, data is stored in documents, and these documents are stored in collections of documents.

A document is a way to organize and store data as a set of field-value pairs. MongoDB documents are composed of field-and-value pairs and have the following structure:

{

   field1: value1,

   field2: value2,

   field3: value3,

   ...

   fieldN: valueN

}

The field is a unique identifier for some data point, and the value is data related to a given identifier. The value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON. JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. For example, the following document contains values of varying types:

var mydoc = {

               \_id: ObjectId("5099803df3f4948bd2f98391"),

               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)

            }

The above fields have the following data types:

* \_id holds an [ObjectId](https://www.mongodb.com/docs/manual/reference/bson-types/" \l "std-label-objectid).
* name holds an embedded document that contains the fields first and last.
* birth and death hold values of the Date type.
* contribs holds an array of strings.
* views holds a value of the NumberLong type.

MongoDB is perhaps the most commonly used document database. MongoDB has a free, cloud-based version, which makes it ideal for this course. In this course, I suggest you work with MongoDB Atlas, the cloud version. However, you can also download the Community version to your own machine. I will be using Atlas in this course. The first step to work with Atlas is to create a MongoDB account. Follow the instructions below:

### **Create your MongoDB account.**

Go to [**account.mongodb.com/account/register**](https://account.mongodb.com/account/register) to create your MongoDB account. To register for a new MongoDB account, you must do the following:

#### Provide the following information about yourself:

* Email Address
* First Name and Last Name
* Password
* Phone Number
* Company
* Job Function
* Country

1. Review and select the checkbox to accept the Privacy Policy and the Terms of Service.

#### Click *Sign up* to create the account.

### **Access your MongoDB services.**

1. Log in to your account at [**account.mongodb.com/account/login**](https://account.mongodb.com/account/login) and select **Overview** from the left navigation menu to access the MongoDB services.
2. **Click:**

* **Visit MongoDB Atlas** to go to MongoDB Atlas or MongoDB Cloud Manager UI.

### Navigate to the [Database Deployments](https://cloud.mongodb.com/v2) page for your project.

### Open the Build a Database dialog.

### cluster-db-deploymentClick the **Build a Database** button to display the **Deploy a cloud database** dialog.

### Select a Cluster Type.

### Select the **Shared** cluster.

#### cluster-shared-type

### Select a Cluster Tier.

### cluster-m0-tierIf it is not already selected, select **M0 Sandbox**.

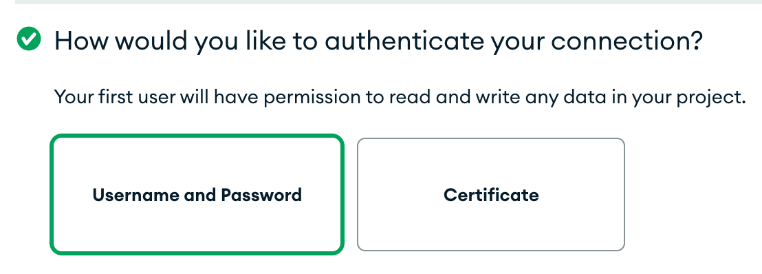
### Create the cluster.

### Click **Create Cluster**. This creates a cluster with the default values for the **Cloud Provider and Region**, **Additional Settings,** and **Cluster Name**.

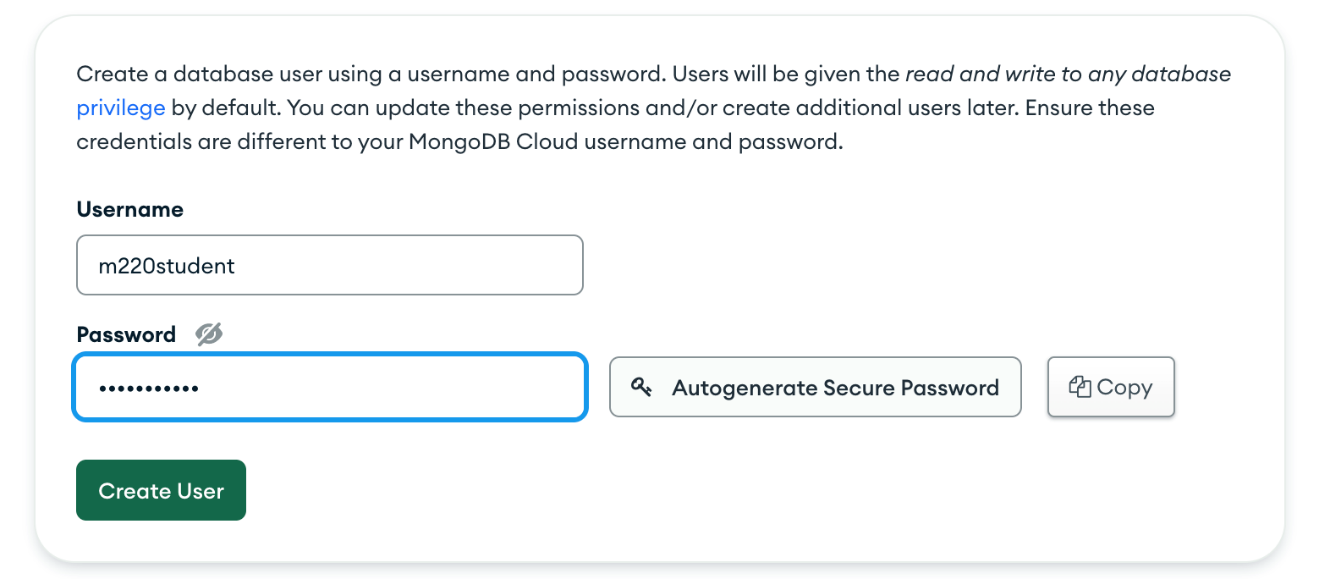
### **Add a Database User**

### **Select *Username and Password*.**

### In **How would you like to authenticate your connection?** section, select the box labeled **Username and Password**.



1. **Enter the users information.**
   * + In the box underneath, there are two text fields.
       1. Enter a username for the new user in the top text field.
       2. Enter a password for the new user in the lower text field.
          - Enter a password of your choice, or use an Atlas auto-generated one, by clicking the Autogenerate Secure Password button.



1. **Click *Create User*.**
   * + Now, you see your user under the **Create User** button.

