**NoSQL Databases**
**Begin your course today**

Welcome to NoSQL Databases. My name is Dr. Casas and during the next six weeks we are going to learn basic concepts of NoSQL databases. Each week has a series of activities to complete. If you are a verified student you an take quizzes in weeks 1 to 5, and a final exam in week 6. You all are encouraged to participate in the weekly discussions. Each week there is also a Q&A Forum. Post any questions your have regarding the course material in this forum. I hope you all find the course a good learning experience. Dr. Casas.

1. Welcome to NoSQL Databases, Incomplete section
    1. About this course and Getting Started, Incomplete
2. Week 1 Introduction, Incomplete section
    1. Week 1 Introduction, Incomplete
    2. Quiz Week 1, Incomplete

    Quiz due Sep 10, 2023, 11:39 PM GMT+8

3. Week 2 What is No SQL?, Incomplete section
    1. What is NoSQL?, Incomplete
    2. Quiz Week 2, Incomplete

    Quiz due Sep 16, 2023, 11:39 PM GMT+8

4. Week 3 Types of NoSQL Databases, Incomplete section
    1. Types, Incomplete
    2. Quiz Week 3, Incomplete

    Quiz due Sep 22, 2023, 11:39 PM GMT+8

5. Week 4 Introduction to MongoDB, Incomplete section
    1. What Is MongoDB?, Incomplete
    2. Quiz Week 4, Incomplete

    Quiz due Sep 28, 2023, 11:39 PM GMT+8

6. Week 5 - Exploring Documents, Incomplete section
    1. Subsection, Incomplete
    2. Week 5 Quiz, Incomplete

    Quiz due Oct 4, 2023, 11:39 PM GMT+8

7. NoSQL Final Project/Exam, Incomplete section
    1. Final Project, Incomplete
    2. Final Exam, Incomplete

    Final Exam due Oct 10, 2023, 11:39 PM GMT+8

Syllabus:
*Instructors:*
Professor: Dr. Augusto Casas, acasas@umbc.edu
Graduate Assistant: Maliha Momtaz, mmomtaz1@umbc.edu
*Weekly Schedule:*
Week 1: Introduction
Week 2: What is No SQL?
Week 3: Types of NoSQL Databases
Week 4: Introduction to MongoDB
Week 5: Exploring Documents
Week 6. Final Project
*Grading*
The grading for this course will be composed of a weekly quizzes. These are worth 80% of your final grade. In week 6 there is a Final Exam. It is worth 20% of your final grade. A passing score of 80% is required to pass the course.

**Get to know the class and staff**
Dr. Augusto Casas, acasas@umbc.edu, is a professor at the University of Maryland Baltimore County.

Maliha Momtaz, a graduate teaching assistant for this course at University of Maryland Baltimore County. Any doubts regarding the course or curriculum, you can reach out to me via email or discussions. email id: **mmomtaz1@umbc.edu** Feel free to ask any queries.
This is a self-paced course, but our instructors and teaching assistants are active in the class to help you. Also other students will be in the class. Take a moment to instruce yourself to the class.

**Getting help with the course and EdX**

Learners that are new to edX should consider viewing the edX DemoX. To get help with a technical problem, click Help to send a message to edX Student Support.

edX HONOR CODE
Recall that you agreed to the edX terms of service when you started your edX account.
**COLLABORATION POLICY**
By enrolling in a course on edX, you are joining a special worldwide community of learners. The aspiration of edX is to provide anyone with an internet connection access to courses from the best universities and institutions in the world and to provide our learners with the best educational experience internet technology enables. You are a part of the community that will help edX achieve this goal. EdX depends upon your motivation to learn the material and to do so with honesty and academic integrity. In order to participate in edX, you must agree to the Honor Code below and any additional terms specific to the class. This Honor Code and any additional terms will be posted on each classes' website.
**EDX HONOR CODE PLEDGE**
**By enrolling in an edX course, I agree that I will:**

- Complete all tests and assignments on my own, unless collaboration on an assignment is explicitly permitted.
- Maintain only one user account and not let anyone else use my username and/or password.
- Not engage in any activity that would dishonestly improve my results, or improve or hurt the results of others.
- Not post answers to problems that are being used to assess student performance.

**VIOLATIONS**
If you are found in violation of the Terms of Service or Honor Code, you may be subject to one or more of the following actions:

- Receiving a zero or no credit for an assignment;
- Having any certificate earned in the course withheld or revoked;
- Being unenrolled from a course;
- Termination of your use of the Site
- Additional actions may be taken at the sole discretion of edX and edX Member course providers.
- No refunds will be issued in the case of any corrective action for such violations.

Honor code violations will be determined at the sole discretion of edX Members. You will be notified if a determination has been made that you have violated this honor code, and you will be informed of the corresponding action to be taken as a result of the violation.
CHANGING THE HONOR CODE
Please note that we review and may make changes to this Honor Code from time to time. Any changes to this Honor Code will be effective immediately upon posting on this page, with an updated effective date. By accessing the Site after any changes have been made, you signify your agreement on a prospective basis to the modified Honor Code and any changes contained therein. Be sure to return to this page periodically to ensure familiarity with the most current version of this Honor Code.

**GETTING STARTED**
*Introduction*

Welcome to NoSQL Databases. In this course, we are going to learn about databases that are well suited to store data that doesn't conform to the tabular form we see in relational databases. 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.

*Brief Course Outline*

This course is divided into six units. Each unit will be covered in one week. The six units are:

1. Introduction
2. What is a NoSQL Database
3. Types of NoSQL Databases
4. Introduction to MongoDB
5. Exploring Documents
6. Final Project/Exam

*Unit 1 Introduction*

In this unit, we will get to know each other, review the course syllabus, and start a discussion about NoSQL databases. This unit has the following sections:

1. Let's meet each other
2. Course Syllabus
3. History of NoSQL
4. Discussion Forum
5. Quiz

Syllabus:

*Instructors:*

Professor: Dr. Augusto Casas

Graduate Assistant: Maliha Momtaz, mmomtaz1@umbc.edu

*Weekly Schedule:*

Week 1: Introduction and History of NoSQL
Week 2: What is NoSQL?
Week 3: Types of NoSQL Databases
Week 4: Introduction to MongoDB
Week 5: Exploring Documents
Week 6: Final Project/Exam

*Grading*

The grading for this course will be composed of five quizzes and a final exam. The quizzes cover material from that week. The quizzes are worth 80% of your grade. The final exam, 20%. A passing score of 80% is required to pass the course.

This course uses a variety of materials. This includes:

- Some of the material was written by my UMBC Professors and staff. In other places, where the material was public domain or creative commons, the material was modified by UMBC Professors and staff
- Some of the course material has been taken from the MogoDB documentation.

**MongoDB Database Server and Tools**

- MongoDB, Inc.'s Server Side Public License (for all versions released after October 16, 2018, including patch fixes for prior versions).
- Free Software Foundation's GNU AGPL v3.0 (for all versions released prior to October 16, 2018).
- Commercial licenses are also available from MongoDB, Inc.

**Documentation**

- Documentation: Creative Commons.

Licensing Policy

Our goal in selecting the Server Side Public License (SSPL) v1.0, a license introduced by MongoDB, as our license is to require that enhancements to MongoDB be released to the community. We also make our drivers available under the Apache License v2.0.

If the use of the database under the SSPL v1.0 does not satisfy your organization's legal department, commercial licenses are available with MongoDB Enterprise Advanced. Feel free to contact us for more details.

MongoDB Trademark Guidelines

MongoDB, Mongo, and the leaf logo are registered trademarks of MongoDB, Inc. Your use of these trademarks is subject to the MongoDB Trademark Standards for Use. For trademark use approval or any questions you have about using these trademarks, please email trademark@mongodb.com.

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.

Reference:

Vaish, G. (2013). Getting Started with NoSQL : Your Guide to the World and Technology of NoSQL. Packt Publishing.

# *Week 2: 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**
DBMS stands for Database Management System, which consists of tables that have no relations among themselves
The Relationship between tables in DBMS is Physical
DBMS does not support Data Integrity
DBMS does not support Data Integrity

**RDBMS**
RDMBS stands for Relational Database Management System, in which there are relationships between the tables.
The relationship in RDBMS is Logical.
RDBMS supports Data integrity
RDBMS supports Structural Independence and Advanced Query Capabilities
RDBMS supports Structural Independence and Advanced Query Capabilities

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.

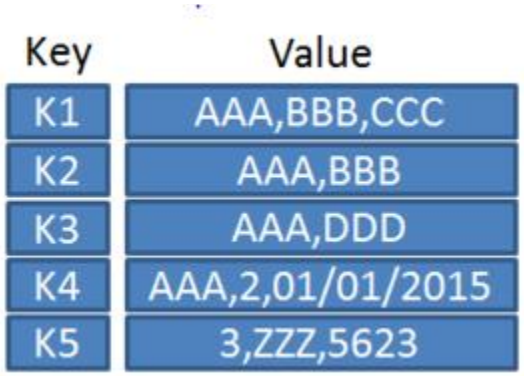
An important characteristic of a document database is that documents can be nested. Furthermore, elements in the database can be indexed for faster querying. Because the data is stored in a way that resembles the physical application, it can be stored and retrieved with less overhead and translation 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*

This 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*

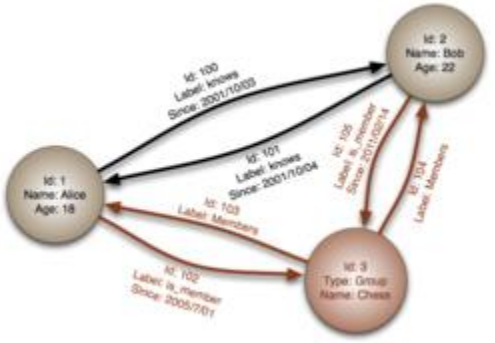
A column-oriented database is organized as a set of columns. The result is that when you run a query on a small number of columns, the query returns just those columns directly without consuming memory with unwanted data. When multiple columns are of the same type compressing them is more efficient and 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.



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.
- `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** to create your MongoDB account. To register for a new MongoDB account, you must do the following:

**1**

*Provide the following information about yourself:*

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

**2**

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

**3**

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

Access your MongoDB services.

**1**

Log in to your account at **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 page for your project.
  - Open the Build a Database dialog.

Click the **Build a Database** button to display the **Deploy a cloud database** dialog.

Select a Cluster Type.
Select the **Shared** cluster.

Select a Cluster Tier.
If 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.

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

Click Create User.

Now, you see your user under the Create User button.

Configure a Network Connection
After creating your cluster, Atlas directs you to a page called Security Quickstart. In the previous guide, we completed the first section, which added a user to the database.

Select My Local Environment.

In the Where would you like to connect from? section, select the box labeled My Local Environment.**2**

Add your IP address
In the box labeled **Add entries to your IP Access List**, click **Add My Current IP Address**.

Click Finish and Close.
After clicking the Finish and Create button at the bottom of the page, you see the following modal:

*Load Sample Data*
Navigate to the Database Deployments page for your project.
Click on your cluster name.

Click on your cluster name.
Click the *Collections* tab.
Click *Load Sample Dataset*.

**In the ensuing dialog, click *Load Sample Dataset* confirm.**
Once the load completes, the **Collections** tab refreshes to show your sample data.

You see the following databases in your cluster:

| Dataset Name | Description |
|---|---|
| `sample-airbnb` | Contains details on AirBnB listings. |
| `sample-analytics` | Contains training data for a mock financial services application. |
| `sample-geospatial` | Contains shipwreck data. |
| `sample-guides` | Contains planet data. |
| `sample-mflix` | Contains movie data. |
| `sample-restaurants` | Contains restaurant data. |
| `sample-supplies` | Contains data from a mock office supply store. |
| `sample-training` | Contains MongoDB training services dataset. |
| `sample-weather` | Contains detailed weather reports. |

Navigate to the Database Deployments page for your project.

Click the *Connect* button.
Click the **Connect** button on the cluster management panel. The following Atlas screenshot shows the **Connect** button:Copy the

connection string
In the **Choose your connection method** step in the modal, select the button marked **Connect Your Application**.
The following Atlas screenshot shows the connection option buttons:

# Week 5: Manage Documents in Data Explorer

Now, we are ready to query our documents database. The easiest and most user-friendly method is by accessing the MongoDB Data Explorer. Review the tutorial below, which explains how to access the data explorer and query collections and documents:

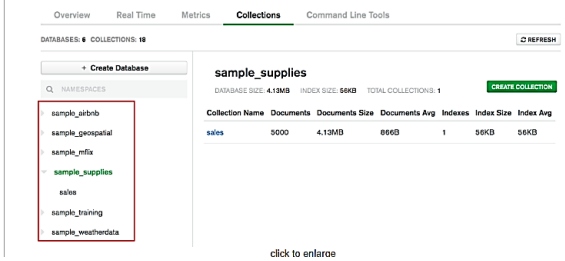
Required Roles
To insert, edit, or delete documents, you must have been granted access through one of the following roles:

Project Owner or Organization Owner
Project Data Access Admin
Project Data Access Read/Write

View Documents
From the Collections tab, you can view documents in a collection. To view documents for a collection:

**1 Select the database for the collection.**

The main panel and **Namespaces** on the left side list the collections in the database.




click to enlarge

**2 Select the collection on the left-hand side or in the main panel.**

The main panel displays the **Find** view and the **Indexes** view.

**3 Select the *Find* view.**

The panel displays the documents in the collection. Each page displays up to 20 documents.

**4 Optional: Specify a query to find specific documents.**

You can use the query bar to search for specific documents in your collection. You can specify one or more of the following in the query bar:

- A filter condition
- A project document to include and exclude specific fields in the results
- A sort order for the documents in the results
- A collation document for language-specific rules.

Filter | Project | Sort | Collation

To specify a filter condition, type in a query filter document in the **Filter** field. For example, to specify equality condition, use a filter document of the form:

copy

```
{ <field1>: <value1>, ... }
```

To use query operators to specify a filter condition, use a filter document of the form:

copy

```
{ <field1>: { <queryoperator>: <value1> }, ... }
```

NOTE
**Data Explorer** does not support date queries that use the `IsoDate()` function. Instead, use the MongoDB Extended JSON (v2) `$date` data type for date queries.

For example, the following query returns all documents where the date added to a `created_at` field is equal to or more recent than midnight on January 1, 2019, UTC time:

```
{ created_at: { $gte: { $date: "2019-01-01T00:00-00:00" } } }
```

For more information on specifying query filters, including compound conditions, see Query Documents.

NOTE
As you type, the **Apply** button is disabled and the field name in the User Interface turns red until a valid query is entered.

**5 Click *Apply* to run your query.**

Number of Documents Displayed per Page

Cloud Manager limits the *total byte size* of documents shown per page in the **Data Explorer**. As a result, you may see varying numbers of documents per page, especially if your documents vary significantly in size.

### Insert Documents

To add one or more documents to a collection through the **Data Explorer**, you can specify the document(s) to insert from scratch or you can clone an existing document and modify its fields and values as needed.

Insert One Document

1. **Go to the *Find* tab in Data Explorer.**
   Select the collection and go to the **Find** tab.
2. **Click *Insert Document*.**
   The document editor appears with the _id field with an ObjectId value that reflects the time of its generation and not the insertion time of the document. As such, the ObjectId does not represent a strict insertion order.
3. **Modify the document.**
   - To add a new field after an existing field, hover over the field and click on the plus sign that appears over the field's line number.
   - To delete a field, hover over the field and click on the x sign that appears to the left of the field's line number. You cannot delete the _id field.
   - To edit a field name, value, or type, click on the field name, value, or type.
4. **Click *Insert*.**

**Insert Multiple Documents**

1. **Go to the *Find* tab in Data Explorer.**
   Select the collection and go to the **Find** tab.
2. **Click *Insert Document*.**
   Data Explorer opens the **Insert to Collection** dialog.
3. **Select the JSON View.**
4. **Type or paste an array of documents to insert.**

EXAMPLE

The following array of documents inserts three documents into the collection:

```
[
  {
    "name": "Alice",
    "age": 26,
    "email": "alice@abc.com"
  },
  {
    "name": "Bob",
    "age": 43,
    "email": "bob@def.com"
  },
  {
    "name": "Carol",
    "age": 19,
    "email": "carol@xyz.com"
  }
]
```

# SERIES of VIDEOS to WATCH

# Week 6: Final Project

We will conclude this course with a short project where you will apply the lessons learned in the past five weeks. The project consists of querying one of the databases in the MongoDB Atlas dataset. Instructions to complete the project are shown below. As you complete each step, write down the answer to each of the questions. You will need those answers to complete the final exam, which is due this week.
*Project Instructions:*

1. Log in to your MongoDB Atlas Account.
2. Go to Database Deployments
3. Click on Browse Collections
4. Expand the sample_analytics collection
5. Find the email address of Geoffrey Ball **(Question 1)**
6. Find the address of Michelle Phillips **(Questions 2)**
7. Find the total number of documents in the customers collection **(Question 3)**
8. Find the total number of documents in the accounts collection **(Question 4)**
9. Find the type of product(s) in the accounts whose ID is 358133 **(Question 5)**
10. Find the total number of documents in the transactions collection **(Question 6)**
11. In the transactions collection, what is the data type of the transaction_count field? **(Question 7)**
12. How many transactions documents have 100 transactions? **(Question 8)**
13. What is the username of Yolanda Harris? **(Question 9)**
14. How many customers are named Christopher Watson? **(Question 10)**
15. Once you have the answers to the above questions, you may proceed to the final exam. Make sure you have saved all the answers.

**NoSQL and DBaaS**
**Modules 1:- Introducing NoSQL**
**Question 1: Scalability is a reason for employing a NoSQL database. True or false?**

- **True**
- False

**Question 2: Which of the following is a category of a NoSQL database?**

- KeyValue
- Document
- Graph
- BigTable
- **All of the above**

**Question 3: A NoSQL database is slightly more costly than a standard RDBMS. True or false?**

- True
- **False**

**Modules 2. Defining NoSQL Database Types, Options, and Use**
**Question 1. Which type of NoSQL database is the simplest?**

- Graph
- BigTable/Column-Family
- **Key-Value**
- Document

**Question 2. In a document database, documents have a flexible schema where more than one document can share the same information. True or false?**

- **True**
- False

**Question 3. Column-Family databases are worse than Graph databases when dealing with large amounts of sparse data. True or false?**

- True
- **False**

**Modules 3. Choosing a Data Layer for Your Application**
**Question 1. Since NoSQL databases offer horizontal scalability, they generally work well within a cloud architecture. True or false?**

- **True**
- False

**Question 2. Since many NoSQL databases primarily operate within a cluster, they cannot meet high availability requirements. True or false?**

- True
- **False**

**Question 3. Due to the flexibility and ease of integration of many NoSQL databases, it is often unnecessary to assess the skill sets of those who are developing applications. True or false?**

- True
- **False**

**Modules 4. Introducing Cloudant – a NoSQLDBaaS**
**Question 1.Cloudant scales very well to large amounts of**

- **Data and Concurrent Users**
- Concurrent Users
- Data
- None of the above

**Question 2. What file format does Cloudant use to store documents?**

- TXT
- ASCII
- **JSON**
- XML

**Question 3. Data can be replicated within clusters to keep them synchronized without administration intervention. True or false?**

- **True**
- False

**Modules 5. Getting Started With IBM Cloudant: A Hands-On Introduction**
**Question 1. Database names should contain lowercase alphanumeric characters, and they are allowed to contain spaces. True or false?**

- True
- **False**

**Question 2. For each database, Cloudant creates a primary index and stores it in a T-Tree. True or false?**

- True
- **False**

**Question 3. Which of the following is a component of the Cloudant HTTP API hierarchy?**

- Attachment

- Database
- Document
- Account
- **All of the above**

**NoSQL and DBaaS**

**Question 1. The term NoSQL refers to a family of databases that vary widely in style and technology. True or false?**

- **True**
- False

**Question 2. Which of the following is a characteristic of NoSQL databases?**

- Fixed schemas are supported
- **Datastores are non-relational**
- The data model is always a set of tables
- All of the above

**Question 3. What prompted the need for NoSQL databases?**

- Companies could no longer afford to develop and host databases
- **The dotcom boom in the late 90's and early 2000's**
- Relational databases became obsolete
- All of the above

**Question 4. Which of the following database solutions is NOT a NoSQL database?**

- **Oracle**
- Mongo
- Apache CouchDB
- Riak

**Question 5. In which of the following situations should you consider using a Key-Value NoSQL database?**

- **The database only needs to perform basic CREATE-READ-UPDATE-DELETE operations. The data to be housed is not highly related.**
- The database will be housing sparse data sets or Big Data sets. There will be a need for compression or versioning control.
- The data is highly interconnected and business use-cases need to be able to trace relationships between data points.
- The database schema for insertions is not well-defined. The schema is likely to change over time and your solution needs to remain flexible.

**Question 6. In which of the following situations should you consider using a Column-family NoSQL database?**

- The database only needs to perform basic CREATE-READ-UPDATE-DELETE operations. The data to be housed is not highly related.
- **The database will be housing sparse data sets or Big Data sets. There will be a need for compression or versioning control.**
- The data is highly interconnected and business use-cases need to be able to trace relationships between data points.
- The database schema for insertions is not well defined. The schema is likely to change over time and your solution needs to remain flexible.

**Question 7. In which of the following situations should you consider using a Document NoSQL database?**

- The database only needs to perform basic CREATE-READ-UPDATE-DELETE operations. The data to be housed is not highly related.
- The database will be housing sparse data sets or Big Data sets. There will be a need for compression or versioning control.
- The data is highly interconnected and business use-cases need to be able to trace relationships between data points.
- **The database schema for insertions is not well defined. The schema is likely to change over time and your solution needs to remain flexible.**

**Question 8. In which of the following situations should you consider using a Graph NoSQL database?**

- The database only needs to perform basic CREATE-READ-UPDATE-DELETE operations. The data to be housed is not highly related.
- The database will be housing sparse data sets or Big Data sets. There will be a need for compression or versioning control.
- **The data is highly interconnected and business use-cases need to be able to trace relationships between data points.**
- The database schema for insertions is not well defined. The schema is likely to change over time and your solution needs to remain flexible.

**Question 9. When using database-as-a-service, which aspect of development is left to the database team?**

- **Database Design**
- Server Administration
- Server Software
- Hardware Maintenance

**Question 10. If you require strong consistency and transactional rollback capabilities, which type of datastore will you most likely choose?**

- Search Engine
- Document Database
- **Relational Database**

**Question 11. When using a hosted database solution, the provider is responsible for the underlying infrastructure, and your database administrators maintain the software and databases running on top of the infrastructure. True or false?**

- **True**
- False

**Question 12.Cloudant is based on which open-source NoSQL database technology?**

- Riak
- **Apache CouchDB**
- DB2
- Apache HBase

**Question 13.Cloudant stores all of your data on a single database server node. True or false?**

- True
- **False**

**Question 14.Cloudant's replication can be effectively applied to which of the following customer use cases?**

- Avoidance of vendor lock-in (e.g. replication to a local CouchDB instance)
- Client offline access for mobile devices or web browsers
- Hot backups for continuous replication to another cluster
- A data delivery network requiring continuous replication to geographically dispersed clusters
- **All of the above**

**Question 15. Where can Cloudant Database-as-a-Service be deployed?**

- Cloud environments based on virtual machines, such as Amazon Web Services
- Non-IBM bare metal cloud environments, such as Rackspace or Windows Azure
- IBM infrastructure, such as IBM SoftLayer
- **All of the above**

**Question 16. What file format does Cloudant use to store data?**

- Objects
- **JSON documents**
- Tables

**Question 17.Cloudant writes each document on three separate nodes. True or false?**

- **True**
- False

**Question 18. Which field can be used to look up a document in a Cloudant database by using the primary index?**

- _rev
- **_id**
- primary
- key

**Question 19. Which value types are supported by a Cloudant document?**

- nested objects
- booleans
- arrays
- strings
- **All of the above**

**Question 20. Which type of Cloudant index leverages Apache Lucene libraries?**

- Geospatial Index
- Secondary Index with Map Reduce
- **Search Index**
- Primary Index

**Question 21. Which type of deployment does Cloudant offer?**

- Hybrid Cloud
- Private – Locally hosted
- Public – Database-as-a-service
- **All of the above**

**Question 22. Which tools can you use to work with the HTTP API?**

- POSTMan
- RESTClient
- cURL
- **All of the above**