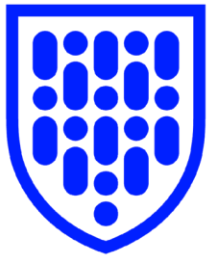


Institute of Data

2022

Software Engineering

Module 8

Databases

2

Institute of Data

Agenda

Section 1 : Introduction to Databases

Section 2 : Databases Designs

Section 3 : Introduction to MongoDB

Section 4 : Introduction to MySQL

Section 5 : Introduction to Redis

Institute of Data

Section 1 : Introduction to Databases

You'll be tempted to improvise and hope for the best when you first start out as a web developer. While this is a frequent habit among newcomers, you will quickly realise that planning is the most crucial thing you can do in this business.

Whether you are working on your side hustle or a client project, you should always have a good plan. In web development, the plan start with the **Design and Architecture of the Application**.

What is a Database ?

It's an organised method for storing your data that applies rules to it, and the rules are up to you because the severity of these issues varies depending on your demands. Perhaps your issue is one of size, whereas someone else has a smaller amount of data with a high level of sensitivity.

It's the things you can't see that are going on in the background: data security, enforced integrity, the ability to get to it quickly and reliably, and robustness; serving a large number of users at once and even surviving crashes and hardware issues without data corruption.

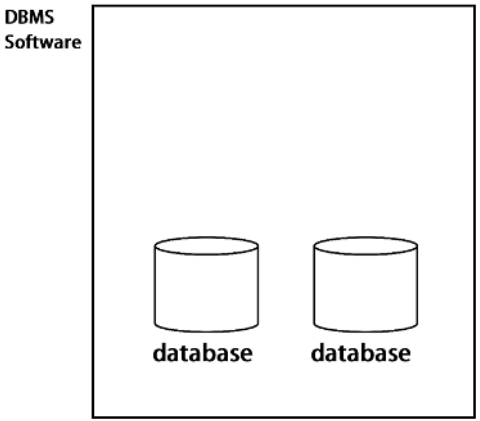
And that's exactly what we need to do here: figure out how to explain our structure and create those rules so that all of these unseen events can take place.

Database Management Systems (DBMS)

We frequently make the mistake of referring to our database as Oracle, MySQL, SQL Server, or MongoDB. However, they are database management systems (DBMS), not databases.

The database management system (DBMS) is software that you instal on your computer or on a server and then use to administer one or more databases.

The database contains your actual data as well as the rules that govern it, whereas the database management system (DBMS) is the programme that surrounds and manages your actual data while also enforcing the rules you set for it. The type of data, such as integer or string, or the relationship between them, are examples of rules.



Database Management Systems (DBMS)

Multiple databases are extremely prevalent in practise. Your order and customer information database could be completely separate from your human resource information database. In many organisations, there are not only several databases, but also different database management systems (DBMS). It's sometimes because one DBMS is superior than the other at something.

There are different DBMS, and they are categorised under:

- Relational Database Management Systems
- Hierarchical Database Systems
- Network Database Systems
- Object-Oriented Database Systems
- NoSQL Database Systems

Database Management Systems (DBMS)

Multiple databases are extremely prevalent in practise. Your order and customer information database could be completely separate from your human resource information database. In many organisations, there are not only several databases, but also different database management systems (DBMS). It's sometimes because one DBMS is superior than the other at something.

There are different DBMS, and they are categorised under:

- Relational Database Management Systems
- Hierarchical Database Systems
- Network Database Systems
- Object-Oriented Database Systems
- NoSQL Database Systems

Section 2: Database Design

Database design is a collection of actions or processes that help with the planning, development, deployment, and maintenance of enterprise data management systems. The cost of maintaining a database is reduced, which improves data consistency, and the cost-effective methods are highly influenced in terms of disc storage space. As a result, a fantastic database design concept is required. The designer must adhere to the limitations and determine how the elements relate to one another and what data must be saved.

The primary goals of database design are to create physical and logical models of the database system under consideration. To clarify, the logical model focuses primarily on data requirements, and decisions must be made in terms of monolithic considerations; hence, stored physical data must be stored independently of physical conditions. The physical database design model, on the other hand, involves a translation of the database's logical design model by maintaining physical media control using hardware resources and software systems such as Database Management System (DBMS).

Importance of Database Design

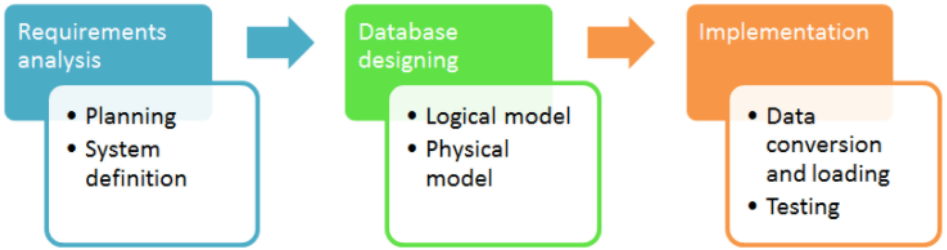
The following points can be used to describe the significant considerations that should be taken into account when emphasising the importance of database design.

- Database designs are plans for how information will be stored in a system. The total performance of any programme is heavily influenced by the database design.
- A database's design principles provide a clear picture of how any application will behave and how requests will be handled.
- Another reason to stress database design is that a good database design satisfies all of the user's requirements.
- Finally, if the restrictions of creating a highly efficient database are effectively implemented, an application's processing time is considerably decreased.

Database Design Lifecycle

The database design lifecycle basically contains three major steps:

- Requirement Analysis
- Database Designing
- Implementation



Database Design Lifecycle: Requirement Analysis

First and foremost, a strategy must be developed to determine what the project's essential requirements are and how the database design should proceed.

Planning - This stage is involved with the overall DDLC planning (Database Development Life Cycle). Before starting, strategic considerations are taken into account.

System Definition - After planning, this step covers the boundaries and scopes of the correct database.

Database Design Lifecycle: Database Designing

The next phase is to design the database around the user's requirements, breaking them up into different models to avoid imposing too much burden or significant reliance on a single feature. As a result, a model-centric approach has emerged, with logical and physical models playing a critical role.

Physical Model - The physical model is concerned with the logical model's practises and implementations.

Logical Model - The primary goal of this stage is to create a model based on the specified requirements. The whole model is created on paper with no implementation or database management system considerations.

Database Design Lifecycle: Implementation

The next step is to look at the implementation techniques and see if they satisfy our criteria. Continuous integration testing of the database with various data sets and data conversion into machine understandable language are used to ensure this. Data manipulation is primarily focused on these phases, which include running queries and determining whether or not the programme is constructed satisfactorily.

Data Conversion and loading - This section is for importing and converting data from the old system to the new.

Testing - This stage focuses on identifying errors in the newly implemented system. Testing is critical since it directly examines the database and compares the requirement specifications.

Important Terms

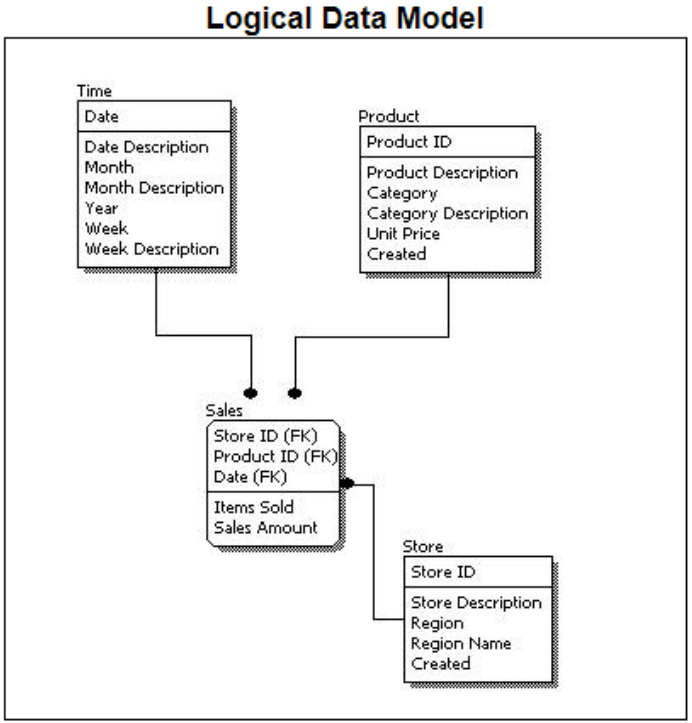
When talking about database design there are a few terminologies that are very important to understand.

- Entity - The abstract data that we save in our database can be defined as an entity in the database. Consider a consumer and their merchandise.
- Attributes - An attribute is a type of data that contains entities such as length, name, price, and so on.
- Relationship - The link between two entities or figures is known as a relationship. A person can, for example, relate to many members of a family.
- Foreign Key - It serves as a link to another table's Primary Key. A foreign key is a collection of columns whose values are unique to the primary key column to which they relate.
- Primary Key - A primary key is a unique and non-null record pointer that is used to uniquely identify table properties.
- Normalisation - A flexible data model must adhere to certain guidelines. Normalising is the process of applying these standards.

Data Models: Logical Model

A logical data model explains the data in as much detail as possible without having to worry about the database's physical implementations. The following are some characteristics of a logical data model:

- All of the entities and their connections.
- Each entity has well-defined characteristics.
- The primary key for each entity is specified.
- Foreign keys are defined as identifiers for relationships between various entities.
- At this stage, normalisation takes place.



Data Models: Logical Model

A logical model can be designed using the following approach:

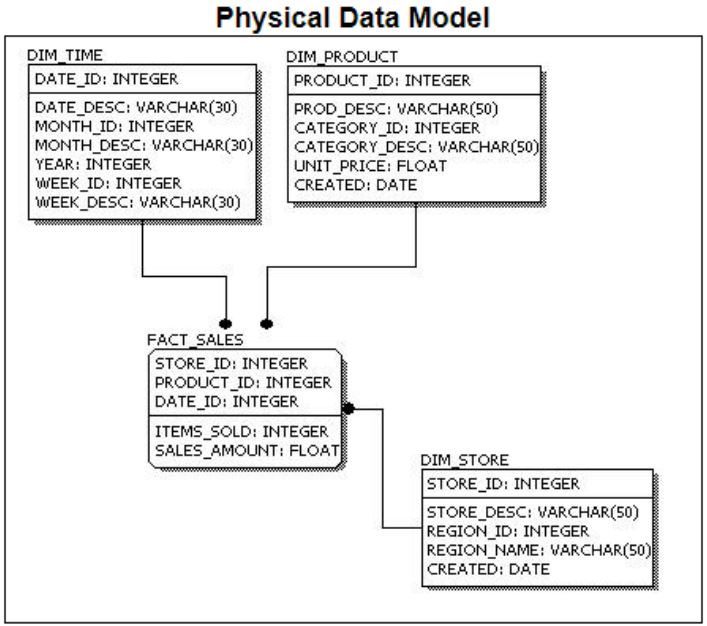
- Make a list of all the entities that have primary keys.
- Concurrent relationships between distinct entities should be specified.
- Determine the properties of each entity.
- Many-to-many relationships must be resolved.
- Carry out the normalising procedure.

A critical examination of the design based on requirement gathering is also a crucial component after using the aforesaid approach. There is a probability of constructing a highly efficient database architecture that follows the native method if the above stages are rigorously followed.

Data Models: Physical Model

A physical data model describes the approach or concept used to construct the database. The fundamental goal of the physical data model is to display all of the table's structures, including column names, column data types, constraints, primary and foreign keys, and table relationships. The following are some characteristics of a physical data model:

- All columns and tables are specified.
- Foreign keys are used to specify the relationship between tables.
- De-normalisation may develop as a result of user requirements.
- Because the physical considerations are taken into account, there will be obvious reasons for differences when compared to a logical model.
- Distinct databases may have different physical models.



Data Models: Physical Model

While designing a physical data model, the following points should be taken into consideration:

- Convert the entities into tables.
- Create foreign keys from the defined relationships.
- Convert the attributes of the data into columns.
- Change the limitations in the data model based on the physical requirements.

Exercise 1

Design a Logical and Physical Model for a general Blogging application.

Requirements:

- The system should have a User Model (should store basic user related keys)
- The users should be able to create multiple posts (Post should be very basic with Title, description and image)
- Other users should be able to like the posts and comment on the posts.

Section 3: Introduction to MongoDB

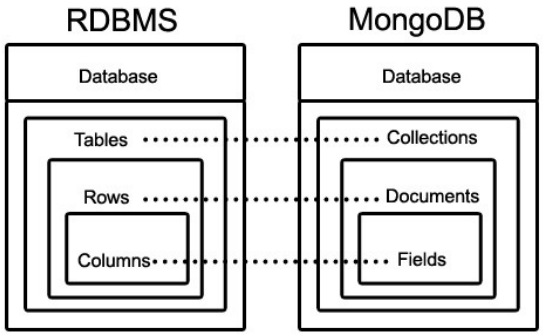
MongoDB is a document-oriented NoSQL database for storing large amounts of data. It contains the data model, which permits hierarchical connections to be represented. Instead of using tables and rows as in standard relational databases, it employs JSON-like documents with optional schema. The basic units of data in MongoDB are documents that include key-value pairs.

A document can have a number of fields (values) that contain information about the record. Because two separate documents can store a different amount of fields, one document's size may differ from the other. Every field (value) has a key that corresponds to it. Any data type, such as common text, number, and so on, can be used in the field. A field can either be an array or a link to another document. MongoDB employs BSON (a binary encoding variant of JSON) to incorporate data types that aren't compatible with JSON, such as dates.

Although no required relationship exists between the fields, linkages and data models can be formed throughout the creation process. Data models make it easier to store arrays, construct hierarchical relationships, and so forth.

Unlike RDBMS, data is stored in a stream rather than a schema.

The Mongo shell, a JavaScript interface provided by MongoDB, is used to query and create documents.



MongoDB - Features

MongoDB's entry to the market has resulted in an increase in the number of developers adopting it in specific applications due to its numerous perks and advantages.

Scalability - MongoDB is a Big Data database designed specifically for processing large amounts of data. It can store massive amounts of data, making it extremely scalable. It also enables horizontal scaling, which allows data to be dispersed across numerous database instances. This aids in data load balancing as well as data duplication for backup purposes.

Flexibility - MongodB is schema-less, which means it doesn't enforce field relationships and instead allows the storage of any data type values in a stream. Because data is stored in a document/collection kind of model, it allows for on-the-fly data modelling, data change, and updation without the headache of schema updates that an RDBMS requires.

Sharding - In MongoDB, sharding is a fascinating and powerful technique. MongoDB, as opposed to a single server, permits data to be distributed over multiple servers. Load can be balanced in applications with large volumes of data by storing data on multiple servers. This also aids in boosting database capacity as well as concurrent and parallel task processing.

MongoDB - Features

Data Replication and Recovery - In the event of a system breakdown, MongoDB provides specific tools for data replication as a backup. A main server, which serves as the principal server, keeps all of the data and handles read/write and other transactions. The secondary server is a replica of the primary server that can act as the primary server in the event that the first server crashes or fails, ensuring continuous high availability.

High Performance and Speed - MongoDB has a variety of capabilities that aid in faster processing, including as dynamic ad-hoc querying, indexing for faster search functionality, and utilities like Aggregation pipeline for aggregation queries. In comparison to an RDBMS, querying is also faster because relevant data is kept together in documents. Even in high-volume, crucial circumstances, this ensures great performance.

MongoDB - Installation

MongoDB can be installed very easily on the 3 most used platforms Ubuntu, MacOS and Windows. In order to install on these platforms please follow the steps :

- Ubuntu
- MacOS
- Windows

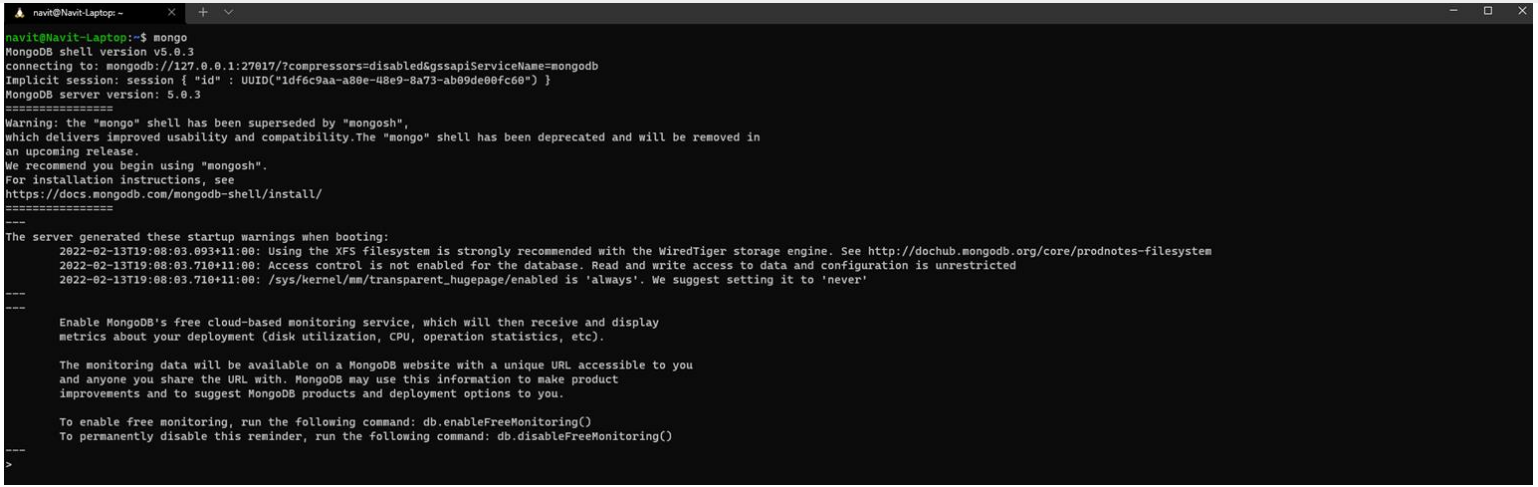
If you follow the steps it should be pretty easy to install MongoDB on your systems.

Tip: Please prefer to install the latest community version of MongoDB

MongoDB - Basic Access and CRUD operations

After the complete installation you should be having mongodb service running in one of your terminal. You would want to open a new terminal. In that terminal just type:

```
mongo
```

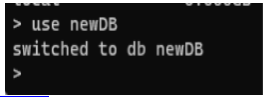


MongoDB - Basic Access and CRUD operations

Now let's try having a look at what all databases do we have in our mongo. By default mongo create three databases admin, config and local. These databases are for mongo’s internal use so please don't touch them or delete them. You can see all the databases by running the command

```
> show databases
```

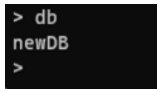

```
> use newDB
```



MongoDB - Basic Access and CRUD operations

This command will also come in handy if you wish to move across databases. So, if there is already a database named newDB, we can use it instead of creating it with this command. It creates a new database if there isn't one called newDB. Now that we have created the newDB and the db is in use we can verify that by typing just the command

```
> db
```



The outcome should be newDB

MongoDB - Basic Access and CRUD operations

What is CRUD ?

Crud is a succinct way of describing four essential storage operations functions. Create, read, update, and delete, to be specific. You should be familiar with these features if you want to play with databases. You must employ CRUD operations in both SQL and NoSQL environments. However, there are some minor application variations.

Create Operations

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne( )
- db.collection.insertMany( )

In MongoDB, collections are equivalent to tables in the SQL world. They are document collections. Documents are essentially objects, similar to rows in a SQL database.

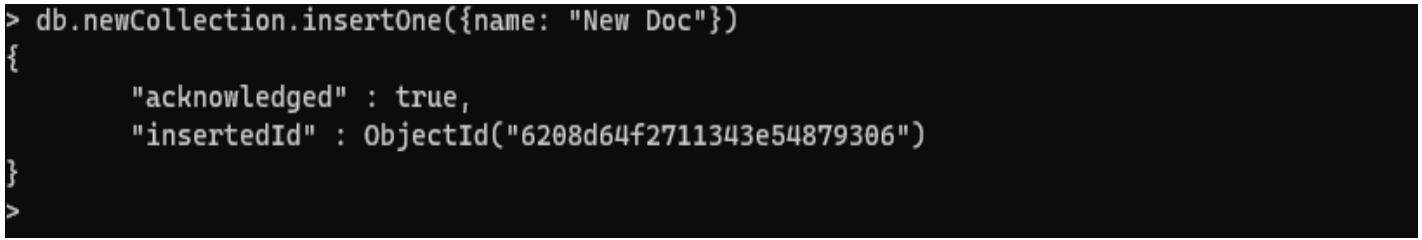
MongoDB - Basic Access and CRUD operations

Let’s see the create operations in real cases

```
> db.newCollection.insertOne({name: "New Doc"})
```

Whatever you enter in there will be instantly created as a new collection, just like we did with newDB above. So now we have our first collection: newCollection, thanks to this command.

You can only insert one document at a time with the insertOne() method. The document we inserted in this example is a basic JSON object. There is only one field name.



MongoDB - Basic Access and CRUD operations

With same logic you can insert more than one document at a time with insertMany( ) method into the relevant collection. You can also see all the collections by using the command

```
> show collections
```

```
> show collections
newCollection
>
```

Read Operations

The document is retrieved from a collection using read operations.

- db.collection.find( )

That find( ) method has 2 optional parameters.

- db.collection.find(query,projection)

MongoDB - Basic Access and CRUD operations

Find Operations

```
> db.newCollection.find()
```

```
> db.newCollection.find()
{ "_id" : ObjectId("6208d64f2711343e54879306"), "name" : "New Doc" }
>
```

Since we inserted only one document yet, we will get just that document as output. You can also see that there is another field called _id there. MongoDB creates a _id field for every document inserted which is a unique field and can be used in future for a lot of manipulations for that document.

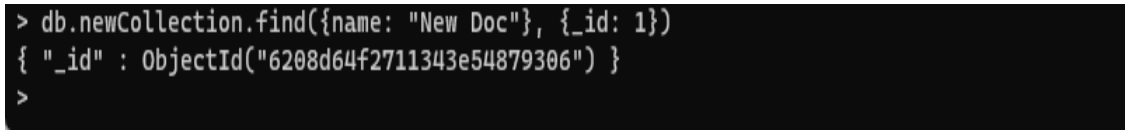
Find Operations with Query

```
> db.newCollection.find({name: “New Doc”})
```

MongoDB - Basic Access and CRUD operations

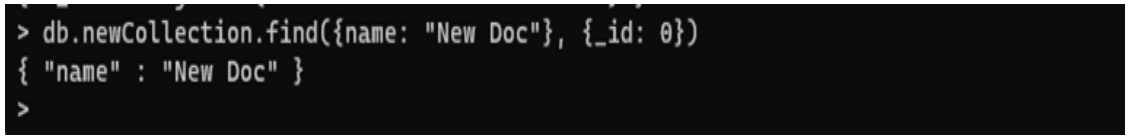
Now we are using query parameter and looking for a document that has a name “New Doc”.

```
> db.newCollection.find({name: "New Doc"}, {_id: 1})
```



You can see the returned document only returns the field we set as 1. You can also use 0 instead of 1 and then mongodb would return the document with all the fields expect the field we set as 0.

```
> db.newCollection.find({name: "New Doc"}, {_id: 0})
```



MongoDB - Basic Access and CRUD operations

Update Operations

You can update the documents in your database with following commands.

- db.collection.updateOne( )
- db.collection.updateMany( )

With parameters:

- db.collection.updateOne(filter, update)

This method can take more optional parameters. You can see all of them here.

MongoDB - Basic Access and CRUD operations

Update One Operations

```
> db.newCollection.updateOne({name: "New Doc"}, {$set: {number: 5}})
```

```
> db.newCollection.updateOne({name:"New Doc"},{$set:{number:5}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

First parameter is the filter parameter that we are looking for documents with name of “New Doc” and then we use the command $set to tell what needs to be updated. What $set would do is that if the key we are trying to update exists it would update it or if it doesn't exist it would create it. You can verify that with the find command again.

```
> db.newCollection.find()
{ "_id" : ObjectId("6208d64f2711343e54879306"), "name" : "New Doc", "number" : 5 }
>
```

MongoDB - Basic Access and CRUD operations

Delete Operations

We can remove documents from a collection using these operations. We have two options, as usual:

- db.collection.deleteOne( )
- db.collection.deleteMany( )

Delete One Operations

```
> db.newCollection.deleteOne({name: "New Doc"})
```

As you might expect, deleteMany() will eliminate all documents that meet the query.

MongoDB - Basic Access and CRUD operations

Drop Database

Finally, there is a fast command that we can use to delete the database we just established. Make sure you're in the appropriate database before you delete. To check the current database, type db, and to switch, type newDB.

Drop Database Operations

```
> db.dropDatabase()
```

We no longer have newDB database.

Exercise 2

Try Designing your own mongoDB database for a general Blogging application. You can use the physical and logical model created before as a reference for you to create this task.

Requirements:

- The system should have a User Model (should store basic user related keys)
- The users should be able to create multiple posts (Post should be very basic with Title, description and image)
- Other users should be able to like the posts and comment on the posts.

Section 4: Introduction to MySQL

MySQL is a database management system for relational databases (RDBMS). MySQL is a database engine that implements SQL. SQL is a query language that is used to manage data in databases. MySQL is an open-source database with a large community and worldwide support. MySQL is a relational database management system (RDBMS), which means it creates a database with tables, columns, rows, and indexes.

MySQL - Structured Query language (SQL)

A relational database's basic language is SQL, which is used to access and manage the database. SQL allows you to add, edit, and delete rows of data, get subsets of data, modify databases, and conduct a variety of other tasks. The different subsets of SQL are as follows:

- DDL (Data Definition Language) - It allows you to do things like CREATE, ALTER, and DELETE objects on the database.
- DML (Data Manipulation Language) — This language allows you to manipulate and access data. It aids in the inserting, updating, deleting, and retrieval of data from a database.
- DCL (Data Control Language) — This language allows you to control database access. Grant or revoke access permissions, for example.
- TCL (Transaction Control Language) — This language allows you to work with database transactions. Commit, Rollback, Savepoint, and Set Transaction are some examples.

MySQL - Features

Some of the feature of MySQL include:

- Ease of Management – The software is simple to instal and employs an event planner to automatically schedule jobs.
- Robust Transactional Support — Maintains the ACID (Atomicity, Consistency, Isolation, and Durability) properties while also supporting distributed multi-version support.
- Integrated Application Development — MySQL comes with plugin libraries that allow you to integrate the database into any application. For application development, it also provides stored procedures, triggers, functions, views, and many more features.
- High Performance — With different memory caches and table index partitioning, it provides quick load utility.
- Low Total Cost Of Ownership (TCO) - This lowers licence and hardware costs.

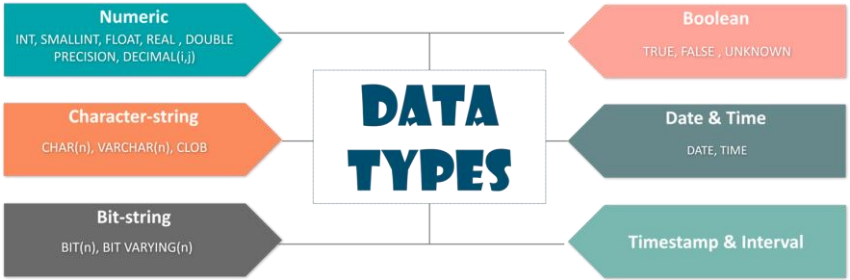
MySQL - Features

- Open Source & Round-the-Clock Assistance — This RDBMS is open source and offers round-the-clock support for both open source and enterprise editions.
- Secure Data Protection - MySQL has a number of sophisticated procedures in place to ensure that only authorised users have access to databases.
- High Availability — MySQL supports high-performance master/slave replication as well as cluster servers.
- Scalability and Flexibility — MySQL allows you to run deeply integrated applications as well as develop massive data warehouses.

MySQL - Data Types

Lets see some of the data types supported by MySQL:

- Numeric — Integers of various sizes, floating-point(real) numbers of varying precisions, and formatted numbers are all included in this data type.
- Character-string — These data types have a set amount of characters or a variable number of characters. CHARACTER Huge OBJECT (CLOB) is a variable-length string that is used to describe columns with large text values in this data type.
- Bit-string — These data types can be either fixed or variable in length. BINARY Big OBJECT(BLOB) is a variable-length bit string data type that can be used to express columns that have large binary values, such as photos.



MySQL - Data Types

- Boolean — This data type has two possible values: TRUE or FALSE. Because SQL has NULL values, UNKNOWN is used as a three-valued logic.
- Date & Time – The DATE data type has the values YYYY-MM-DD for YEAR, MONTH, and DAY. Similarly, the HOUR, MINUTE, and SECOND components of the TIME data type are in the format HH:MM:SS. Depending on the situation, these formats may alter.
- Timestamp & Interval — In addition to the DATE and TIME fields, the TIMESTAMP data type provides a minimum of six places for decimal fractions of seconds and an optional WITH TIME ZONE qualifier. A relative value can be used to increment or decrement an absolute value of a date, time, or timestamp, as shown by the INTERVAL data type.

MySQL - Installation

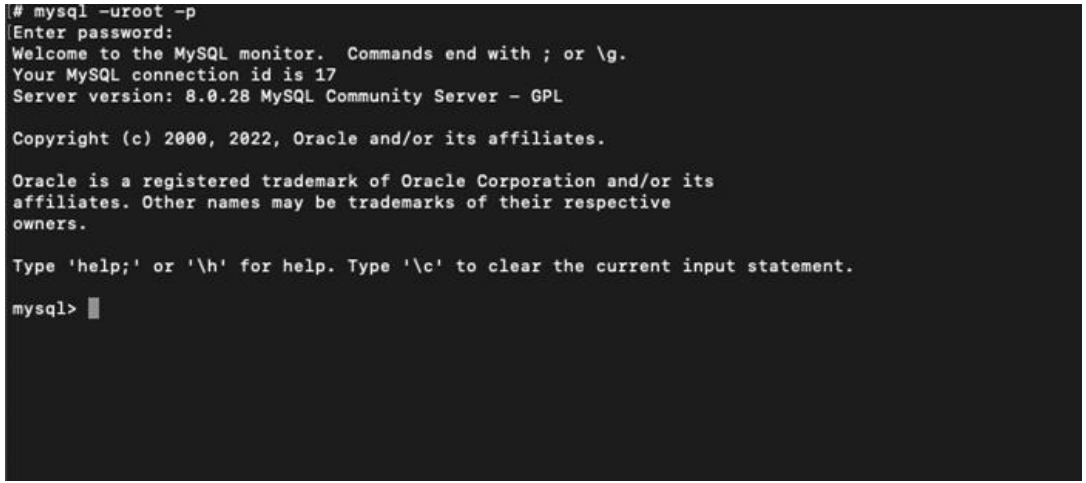
MySQL can be installed very easily on the 3 most used platforms Ubuntu, MacOS and Windows. In order to install on these platforms please follow the steps :

- Ubuntu
- MacOS
- Windows

If you follow the steps it should be pretty easy to install MySQL on your systems.

MySQL - Basic Access and CRUD operations

After the complete installation you should be having mysql service running in one of your terminal. It Would look something like this:



MySQL - Basic Access and CRUD operations

A MySQL server can host several databases, each with its own set of tables. By default mysql create four databases information_schema, mysql, performance_schema and sys. These databases are for MySQL's internal use so please don't touch them or delete them. You can see all the databases by running the command

```
> show databases;
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.03 sec)
```

MySQL - Basic Access and CRUD operations

```
> CREATE DATABASE new;
```

```
mysql> CREATE DATABASE new;
Query OK, 1 row affected (0.04 sec)
```

This would have successfully created a new database for use with the name new.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| new                |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.02 sec)
```

MySQL - Basic Access and CRUD operations

In order to work in a database, we have to type in USE <database_name>;, where ‘database_name’ is the name of the database we intend to work on. This would change us to the current database we want to use.

```
> USE new;
```

```
mysql> USE new;
Database changed
```

Now lets create a table in our current database. We can do this by using the command ‘CREATE TABLE <table_name> (column1 datatype, column2 datatype2, ...);’. Let’s create a table name ‘first_table’ with five columns — ‘id’, ‘movie’, ‘director’, ‘genre’ as follows.

```
> CREATE TABLE first_table (id INT NOT NULL, movie VARCHAR(50) NOT NULL, director VARCHAR(50), genre VARCHAR(50), PRIMARY KEY(id));
```

```
mysql> CREATE TABLE first_table (id INT NOT NULL, movie VARCHAR(50) NOT NULL, director VARCHAR(50), genre VARCHAR(50), PRIMARY KEY(id));
Query OK, 0 rows affected (0.12 sec)
```

MySQL - Basic Access and CRUD operations

Now that we have created out first table in the database. We can check if the table was created or not by using the command ‘SHOW TABLES;’

```
> SHOW TABLES;
```

```
mysql> SHOW TABLES;
+----------------+
| Tables_in_new |
+----------------+
| first_table    |
+----------------+
1 row in set (0.04 sec)
```

You can also see details of the table by using the command ‘DESC <table name>’;

```
> DESC first_table;
```

```
mysql> DESC first_table;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int         | NO   | PRI | NULL    |       |
| movie    | varchar(50) | NO   |     | NULL    |       |
| director | varchar(50) | YES  |     | NULL    |       |
| genre    | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.06 sec)
```

MySQL - Basic Access and CRUD operations

Let's have a look at some of the CRUD operations with MySQL.

Create Operations

In MySQL, we use the following format to add a new record to a table, 'INSERT INTO table_name (column1, column2, column3,....) VALUES (value1, value2, value3,....);'. At the same time, if all of the values are in the same order as the table's columns, we won't need to write down the column names.

```
> INSERT INTO first_table VALUES (1, "Troy", "Wolfgang Peterson", "Historical Fiction");
```

We can also add multiple rows at the same time

```
> INSERT INTO first_table VALUES (2, "Inception", "Christopher Nolan", "Science Fiction"), (3, "The Greatest Showman", "Michael Gracey", "Musical Drama");
```

MySQL - Basic Access and CRUD operations

Create Operations

On running both the commands we should get this output for a successful insert.

```
mysql> INSERT INTO first_table VALUES (1, "Troy", "Wolfgang Peterson", "Historical Fiction");
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO first_table VALUES (2, "Inception", "Christopher Nolan", "Science Fiction"), (3, "The Greatest Showman", "Michael Gracey", "Musical Drama");
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql>
```

MySQL - Basic Access and CRUD operations

Read Operations

The 'SELECT' clause is used for retrieving records from a table. There are two ways to use a SELECT clause

- SELECT column1, colum2, column3... FROM table_name;
- SELECT * FROM table_name;

The data will be selected from the table mentioned after the ‘FROM’ clause. The first syntax retrieves only the columns we want, while the second uses the wildcard '*' to get all columns.

```
mysql> SELECT movie, director FROM first_table;
+---------------------+-------------------+
| movie               | director          |
+---------------------+-------------------+
| Troy                | Wolfgang Peterson |
| Inception           | Christopher Nolan |
| The Greatest Showman | Michael Gracey    |
+---------------------+-------------------+
3 rows in set (0.02 sec)

mysql> SELECT * FROM first_table;
+----+---------------------+-------------------+-------------------+
| id | movie               | director          | genre             |
+----+---------------------+-------------------+-------------------+
|  1 | Troy                | Wolfgang Peterson | Historical Fiction |
|  2 | Inception           | Christopher Nolan | Science Fiction   |
|  3 | The Greatest Showman | Michael Gracey    | Musical Drama     |
+----+---------------------+-------------------+-------------------+
3 rows in set (0.01 sec)
```

MySQL - Basic Access and CRUD operations

Update Operations

When we need to make changes to existing records, we use the UPDATE clause. A typical update command would look something like this 'UPDATE table_name SET column1 = value1, column2 = value2... WHERE condition'. So lets try updating a field

```
> UPDATE first_table SET genre='War Film' WHERE id=1;
```

```
mysql> UPDATE first_table SET genre='War Film' WHERE id=1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

So it returns to us telling it was successfully updated. We can also test if it was successfully updated or not by simply running the read command again.

```
mysql> SELECT * FROM first_table;
+----+----------------------+-------------------+-----------------+
| id | movie                | director          | genre           |
+----+----------------------+-------------------+-----------------+
|  1 | Troy                 | Wolfgang Peterson | War Film        |
|  2 | Inception            | Christopher Nolan | Science Fiction |
|  3 | The Greatest Showman | Michael Gracey    | Musical Drama   |
+----+----------------------+-------------------+-----------------+
3 rows in set (0.03 sec)
```

MySQL - Basic Access and CRUD operations

Delete Operations

The DELETE clause is used in conjunction with the WHERE clause to delete one or more records. The general syntax for a delete command would look like 'DELETE FROM table_name WHERE condition;'

```
> DELETE FROM first_table WHERE id=2;
```

```
mysql> DELETE FROM first_table WHERE id=2;
Query OK, 1 row affected (0.04 sec)
```

We can also test by simply running the read command again

```
mysql> SELECT * FROM first_table;
+----+---------------------+------------------+---------------+
| id | movie               | director         | genre         |
+----+---------------------+------------------+---------------+
|  1 | Troy                | Wolfgang Peterson | War Film      |
|  3 | The Greatest Showman | Michael Gracey   | Musical Drama |
+----+---------------------+------------------+---------------+
2 rows in set (0.02 sec)
```

MySQL - Basic Access and CRUD operations

Truncate a Table

You can actually delete all the rows of a table by just using the truncate command like this 'TRUNCATE <table name>'

```
> TRUNCATE first_table;
```

```
mysql> TRUNCATE first_table;
Query OK, 0 rows affected (0.08 sec)

mysql> SELECT * FROM first_table;
Empty set (0.02 sec)
```

MySQL - Basic Access and CRUD operations

Drop a Table or Database

You can also drop a table or even the whole database using the drop command ‘DROP TABLE <table name>’ or ‘DROP <database name>’

```
> DROP TABLE first_table;
```

```
mysql> DROP TABLE first_table;
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW tables;
Empty set (0.02 sec)
```

```
> DROP DATABASE new;
```

```
mysql> DROP DATABASE new;
Query OK, 0 rows affected (0.06 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.02 sec)
```

Exercise 3

Try Designing your own MySQL database for a general Blogging application. You can use the physical and logical model created before as a reference for you to create this task.

Requirements:

- The system should have a User Model (should store basic user related keys)
- The users should be able to create multiple posts (Post should be very basic with Title, description and image)
- Other users should be able to like the posts and comment on the posts.

Section 5: Introduction to Redis

Redis is an in-memory data structure store that may be used as a database, cache, and message broker. It is open-source (BSD licenced). Strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geographical indexes with radius searches, and streams are all supported data structures.

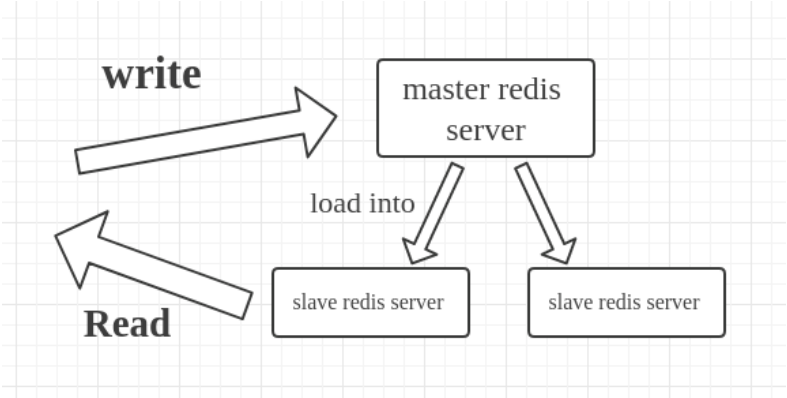
redis

Redis - Key-Value Store

Data is stored as a key-value pair in the Redis server/store. For instance, "Name"="John" The key is "Name," and the value is "John" Redis may be used as a database because it can read and write data.

We can WRITE data to the Redis server using SET “name” “John” and also we can READ data from the server using GET “name”.

It's a non-relational database. This means that, unlike MySQL and Oracle databases, it lacks table, row, column, function, and procedure structures. It also doesn't employ SELECT, UPDATE, INSERT, or UPDATE statements.



Redis - Key-Value Store

Data is stored in Redis using data structures. Strings, Lists, Sets, Sorted Sets, and Hashes are examples of basic data structures. Bitmaps, HyperLogLogs, and Geospatial Indexes are examples of additional data structures.

It's a persistence-enabled in-memory database. This means that Redis maintains data in memory (cache) and does not write to disc, making it extremely fast. Redis, on the other hand, has the ability to write data to disc.

Redis - Features

Some of the feature of Redis include:

- Simple and Flexible – Because Redis is a NoSQL data store, we don't need to specify tables, rows, or columns. There's no need for declarations. The READ and WRITE data to the Redis server is quite straightforward and simple.
- Durability — Redis, on the other hand, operates on data in memory/cache. It does, however, have the ability to write to the disc. This option is also customisable. We can also utilise Redis as a full-fledged database or as a caching system.
- Compatibility — One of the most common uses of Redis is as a supplementary database for your apps to speed up transactions. A cache, like Redis, provides a high-speed data store, whereas a database is slower but more dependable and has more functionality.
- Scaling — The master-slave replication feature in Redis is excellent. Redis can be used in a variety of ways. A write-only data storage can be used as Master. One of the slaves can also be utilised as a read-only slave. Other slaves can write the data to the Disk as well. While the master and one of the slaves read and write, it can improve performance.

Redis - Installation

Redis can be installed very easily on the 3 most used platforms Ubuntu, MacOS and Windows. In order to install on these platforms please follow the steps :

- Ubuntu
- MacOS
- Windows

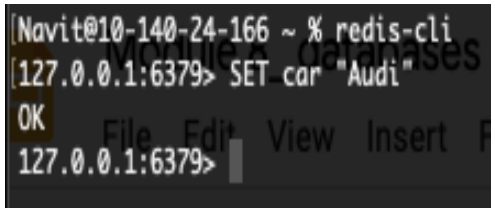
If you follow the steps it should be pretty easy to install Redis on your systems.

Redis - CRUD Operations

Compared to all other databases redis has a much simpler implementation as it doesn't not have a pure CRUD operations. So, go ahead and start your terminal and start the Redis server using the redis-cli command

Create Operations

```
> SET car "Audi"
```

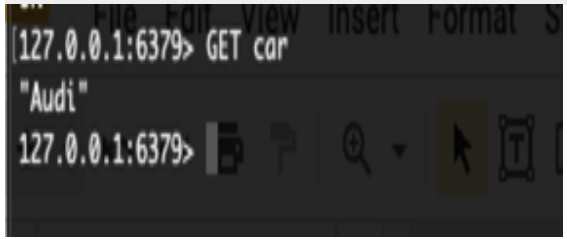


Redis - CRUD Operations

Read Operations

GET in Redis: It gets a value using a key

```
> GET car
```

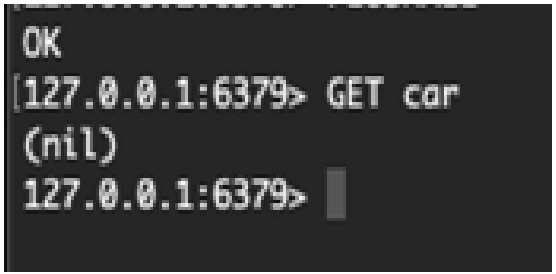


Redis - CRUD Operations

Drop Database

FLUSHALL in Redis: It removes all the data from redis

```
> FLUSHALL
```



Exercise 4

Using the databases you created in exercise 1, 2 and 3 try incorporating redis into your system.

Requirements:

- Try modifying the logical and physical model to incorporate redis
- Try connecting redis with one of the two databases MongoDb/MySQL (using a diagram)

End of Presentation