GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

Unit 2. ACCESS TO DATABASES

Part 2. Working with Non-Relational Databases

Acceso a Datos (ADA) **(a distancia en inglés)**

CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martínez

Year 2024-2025

Credits

- Notes made by Abelardo Martínez.
- Based and modified from Sergio Badal (www.sergiobadal.com).
- The images and icons used are protected by the LGPL licence and have been obtained from:
  - https://commons.wikimedia.org/wiki/Crystal_Clear
  - https://www.openclipart.org

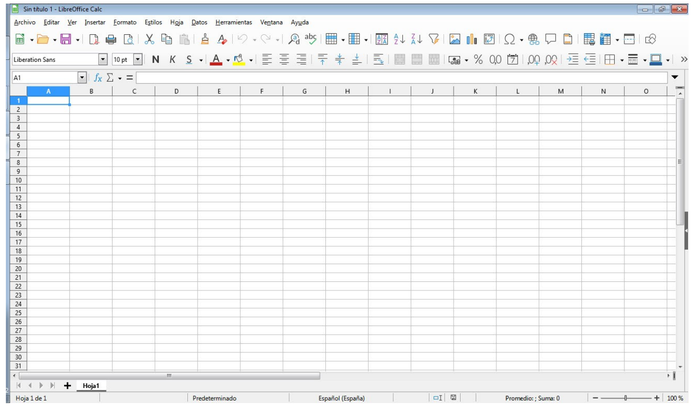
Contents

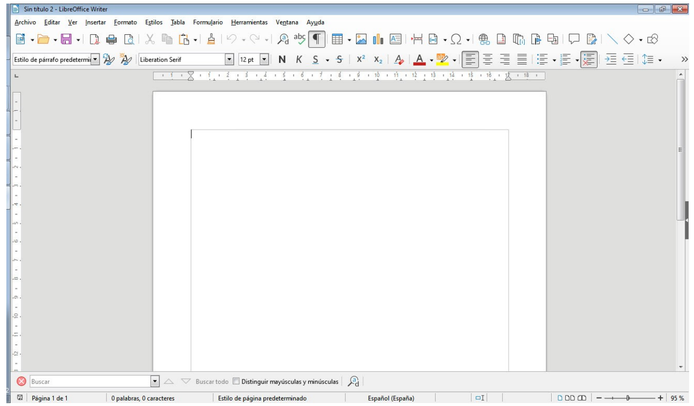
1. WHAT IS A NON-RELATIONAL DATABASE?

What is a non-relational database?

Imagine your data is a dog. In front of it, you place a **Spreadsheet** and a **Text processor document**. Which one will the dog go to?

- It may be a little silly, but it’s a good way to understand exactly what kind of data works for the two main types of databases: relational and non-relational.
- Let’s go over the difference between these two types of databases, as well as list some key questions every business should answer before choosing a database.



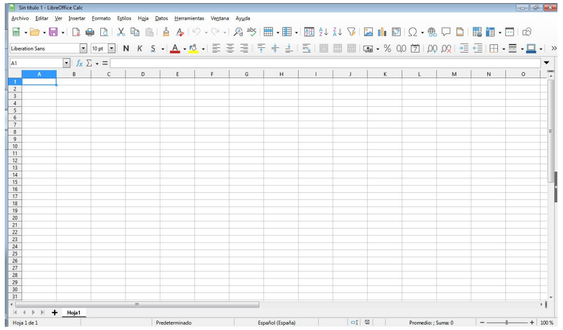


More information: https://www.logianalytics.com/relational-vs-non-relational-databases/

1.1 Relational databases

Relational Databases

Maybe the dog prefers the spreadsheet. Why? Because it fits nicely into rows and columns. A **relational database** is one that stores data in tables.



- The relationship between each data point is clear and searching through those relationships is relatively easy.
- The relationship between tables and field types is called a schema.
- For relational databases, **the schema must be clearly defined**.

Let's look at an example:

Name	Dry/Wet Food	Good Boy (Y/N)
Fido	Dry	Y
Rex	Wet	N
Bubbles	Dry	Y
Cujo	Wet	N

Tag #	Height (in)	Weight (lbs)
1573	15	21
2684	9	7
3795	27	130
4806	6	5

Tag #	Name	Breed	Color	Age
1573	Fido	Beagle	Brown/White	1.5
2684	Rex	Pekingese	White	9
3795	Bubbles	Rottweiler	Black	5
4806	Cujo	Chihuahua	Gold	4

SQL language. RDBMSs

Relational databases are also called SQL databases.

- **SQL** stands for **S**tructured **Q**uery **L**anguage and it's the language relational databases are written in.
- SQL is used to execute queries, retrieve data, and edit data by updating, deleting or creating new records.

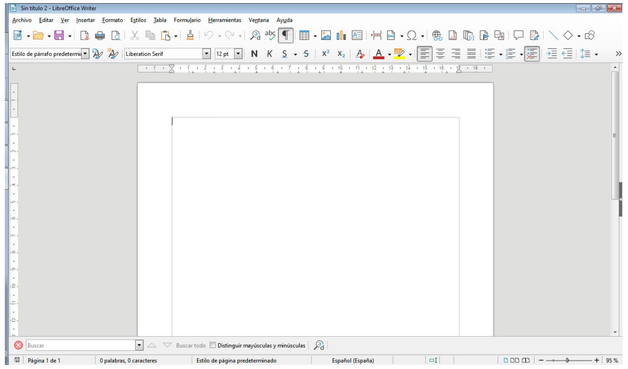
On the right you can see an infogram with the main RDBMSs.

1.2 Non-Relational databases

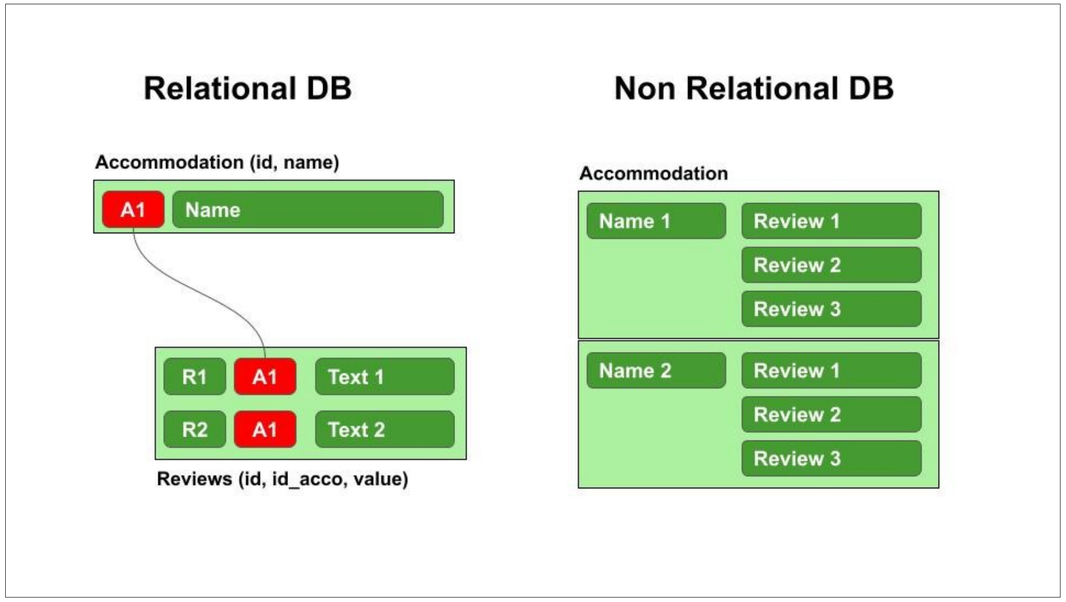
Non-Relational Databases

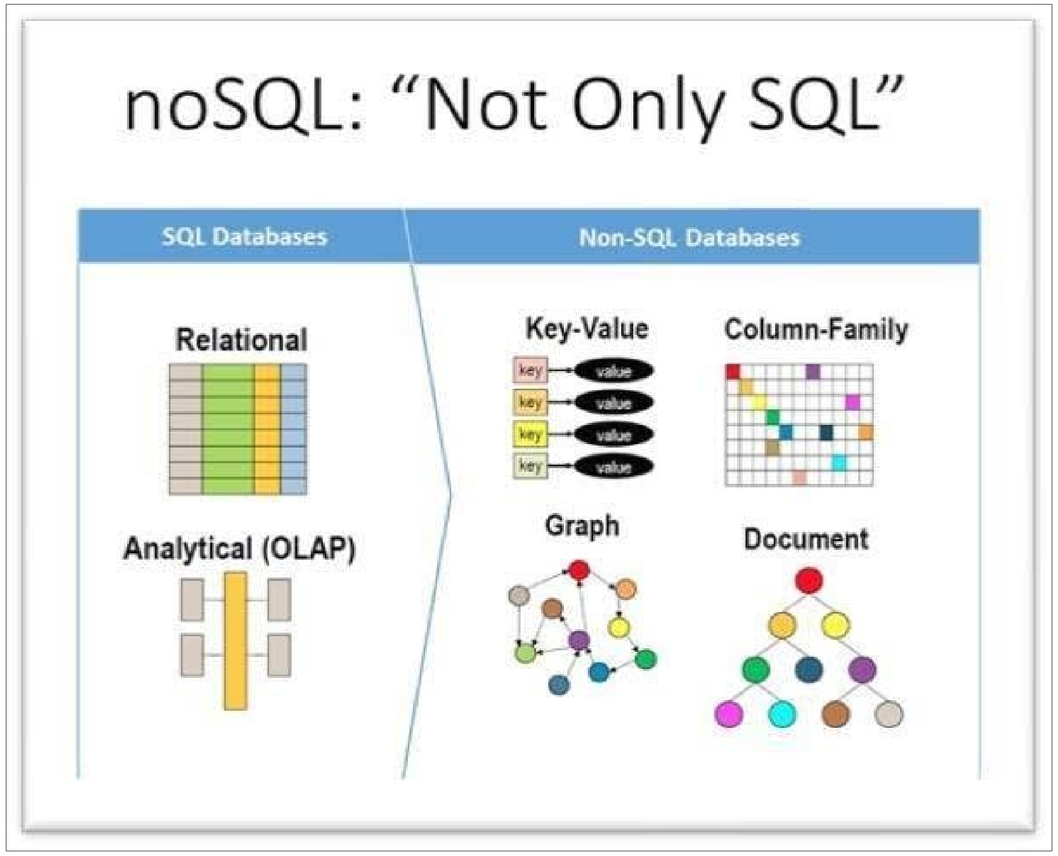
Back to your “data dog.” This time, it went over to the **Text processor doc**. Why? All the open space! The data comes in all different shapes and sizes. It needs room to spread out.

A non-relational database is any database that **does not use the tabular schema of rows and columns like in relational databases**. Rather, its storage model is optimized for the type of data it’s storing.

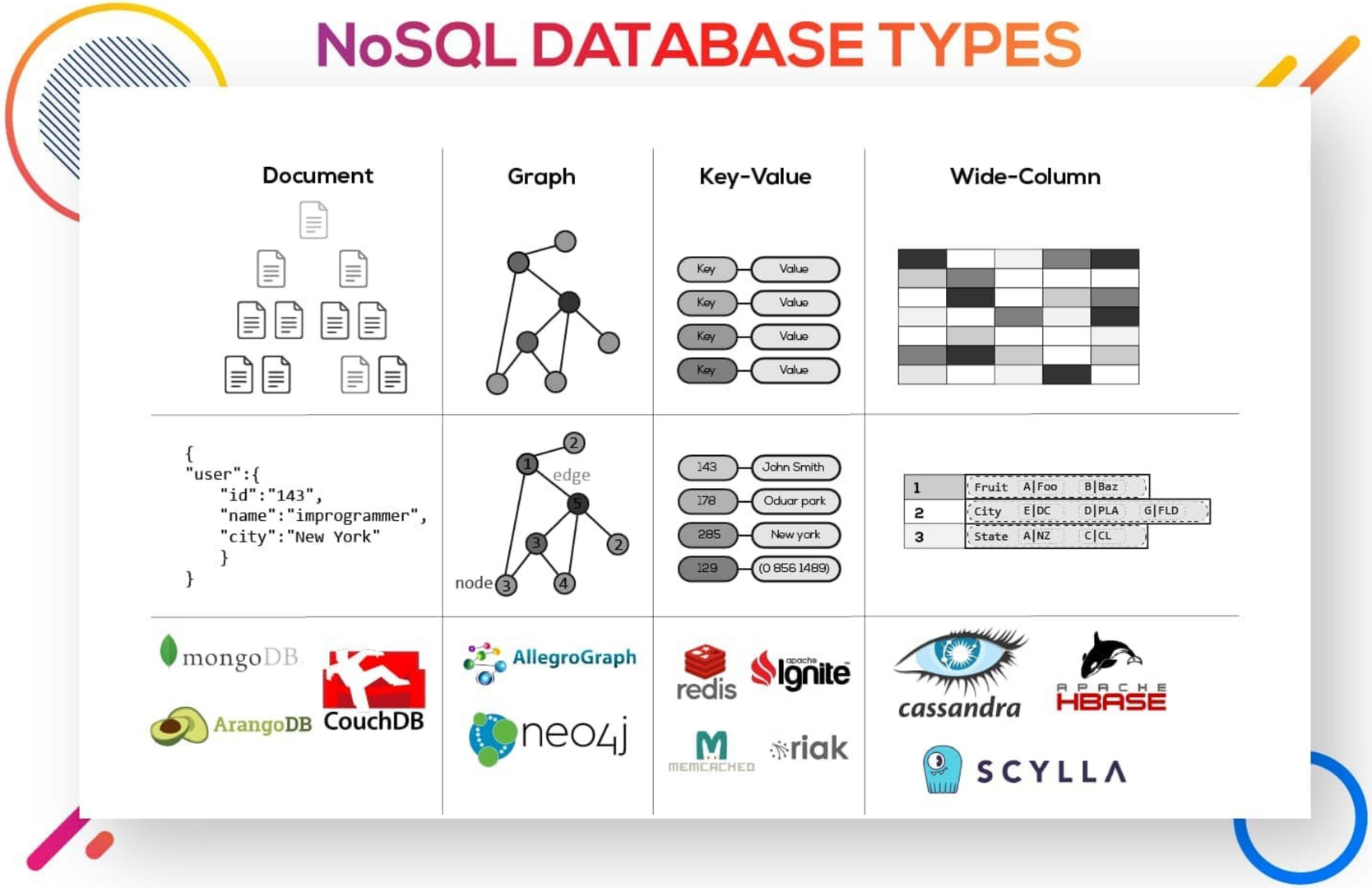


Relational and Non-Relational databases. Differences

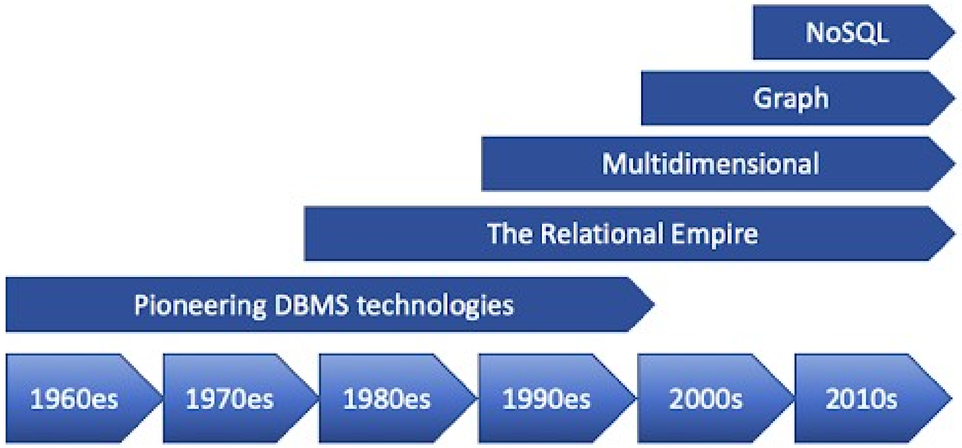


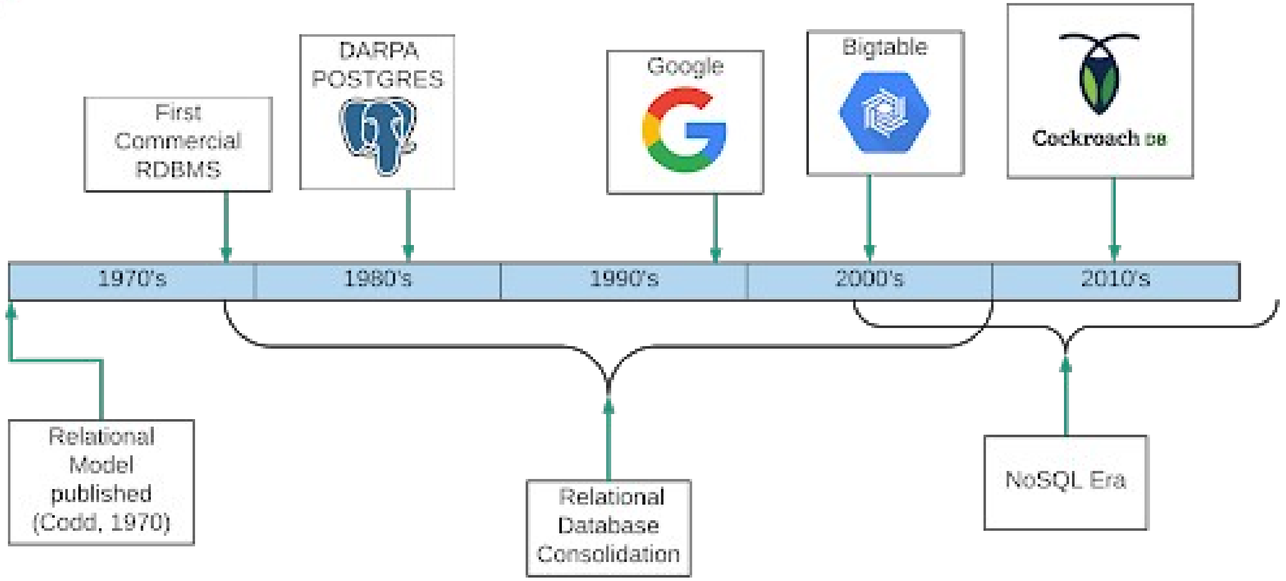


Types of Non-Relational databases



Databases timeline





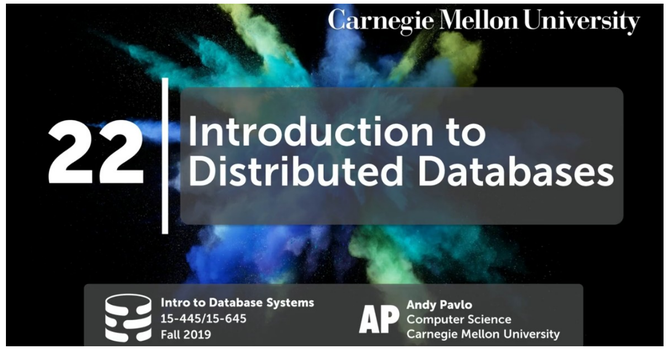
Recommended resources

We're not going to go deeper into those databases in this module; instead we will be accessing MongoDB from Java, one non-relational DBMS which its growing has been exponential last decade.

For a deeper dive into non-relational databases, here are a few recommended resources:

- Introduction to No-SQL Databases. https://www.youtube.com/watch?v=uD3p_rZPBUQ
- Introduction to MongoDB (video tutorial). https://www.youtube.com/watch?v=pWbMrx5rVBE
- Introduction to MongoDB (tutorial). https://www.tutorialspoint.com/mongodb/index.htm
- No-SQL Database Guide for Beginners. https://hostingdata.co.uk/how-to-use-nosql-databases-guide/
- Seminar on Distributed Databases (90 min). https://www.youtube.com/watch?v=0_m5gPfzEYQ

CARNEGIE MELLON DATABASE GROUP



2. WHAT IS MONGODB?

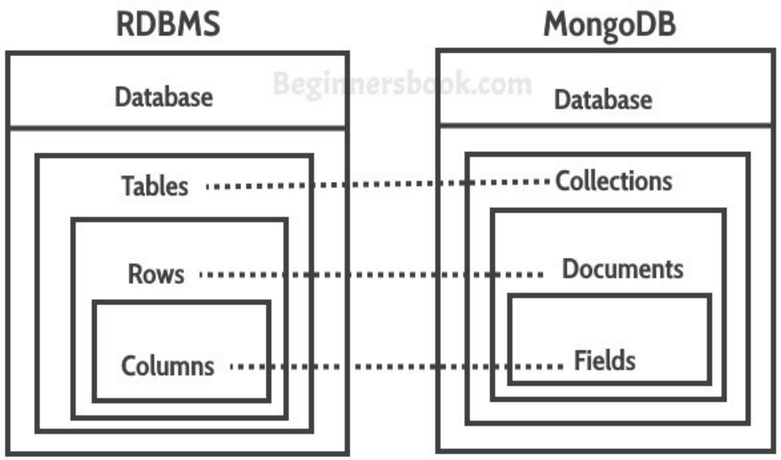
MongoDB

MongoDB is a **document-oriented No-SQL database** used for high volume data storage.

Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.

- **Documents** consist of key-value pairs which are the basic unit of data in MongoDB.
- **Collections** contain sets of documents and function which is the equivalent of relational database tables.

MongoDB is a database which came into light around the mid-2000s.



Document example

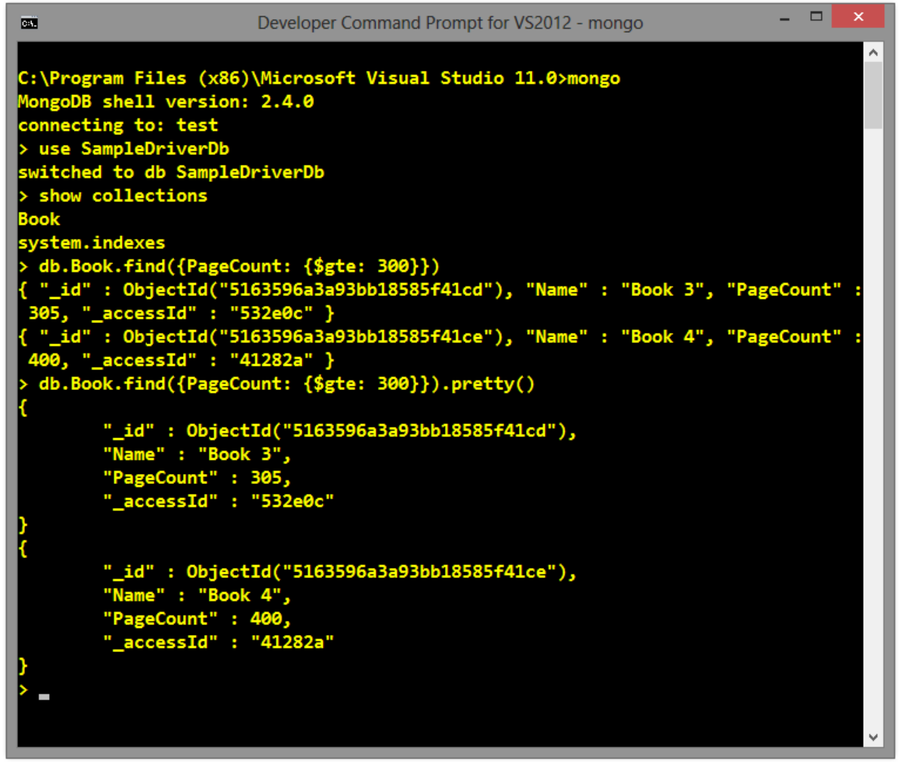
MongoDB. Installation

Everything you know changes in non-relational databases.

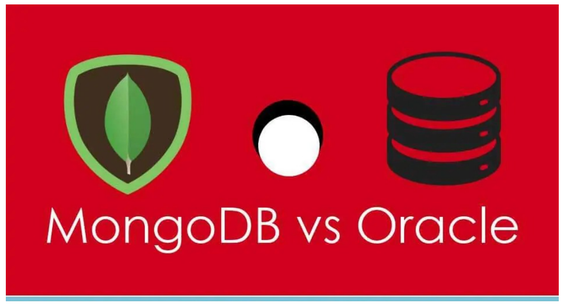
To install MongoDB you can use these resources:

- Official website: Install MongoDB Community Edition on Ubuntu
- Easy tutorial: How To Install MongoDB In 2 Minutes

Then, you can use Compass, Atlas or the console.



MongoDB vs Oracle



#1. About & Description

MongoDB

MongoDB is one of the most famous stores of documents.

Oracle

Oracle Database is multi-model database management system and it is highly used RDBMS to build enterprise applications.

#2. Secondary Database Models

MongoDB

In MongoDB, it uses Secondary database models is Key-value store:From an API perspective,
Key-value stores are the very easiest NoSQL data stores to use and these are the simplest form of DBMS. Key-value stores are always will have very high performance and can be easily scaled; this is because it always uses primary-key access.

Oracle

In Oracle DB, it uses Secondary database models are Document store, Graph DBMS info, Key-value store and RDF store info.Document store: Document stores are characterized by its schema-free organization of data. Records in it need not have a uniform structure and those records can also have nested structure.Graph DBMS: It is also known as graph-oriented DBMS. In this type, data can be represented in graphical structures as nodes and edges.RDF store: The RDF (Resource Description Framework) is a methodology to describe the information, and it is exclusively developed to describe the metadata of IT resources.

#3. Implementation language

MongoDB

A MongoDB is written in C++, C and JavaScript programming language.

Oracle

An Oracle database is written in Assembly language, C, and C++ programming language.

#4. Server-side scripts

MongoDB

In MangoDB, JavaScript is the programming language used in Server-side scripting.

Oracle

In Oracle DB, PL/SQL is the programming language used in Server-side scripting.Also uses java in developing Stored procedures.

#5. Server Operating Systems

MongoDB

MangoDB can be operated in the following Operating systems: Windows Vista and later, Linux, OS X 10.7 and later, Solaris, and FreeBSD.

Oracle

MangoDB can be operated on all major platforms / operating systems including Windows, UNIX, Linux and Mac OS.

#6. Specific Characteristics

MongoDB

MongoDB is considered as the next-generation database which helps in businesses transform their industries by taking a control over the power of data.

Oracle

Oracle database is a multi-model and world's most popular database.It is commonly used for running online transaction processing (OLTP), data warehousing (DW) applications and mixed (OLTP & DW) database workloads.

www.educba.com

3. CONNECTING TO MONGODB

Step 1. Install MongoDB

Let's see an example of a MongoDB database connection and query.

- Install MongoDB: https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/

Procedure

Install MongoDB

Windows | macOS | Linux

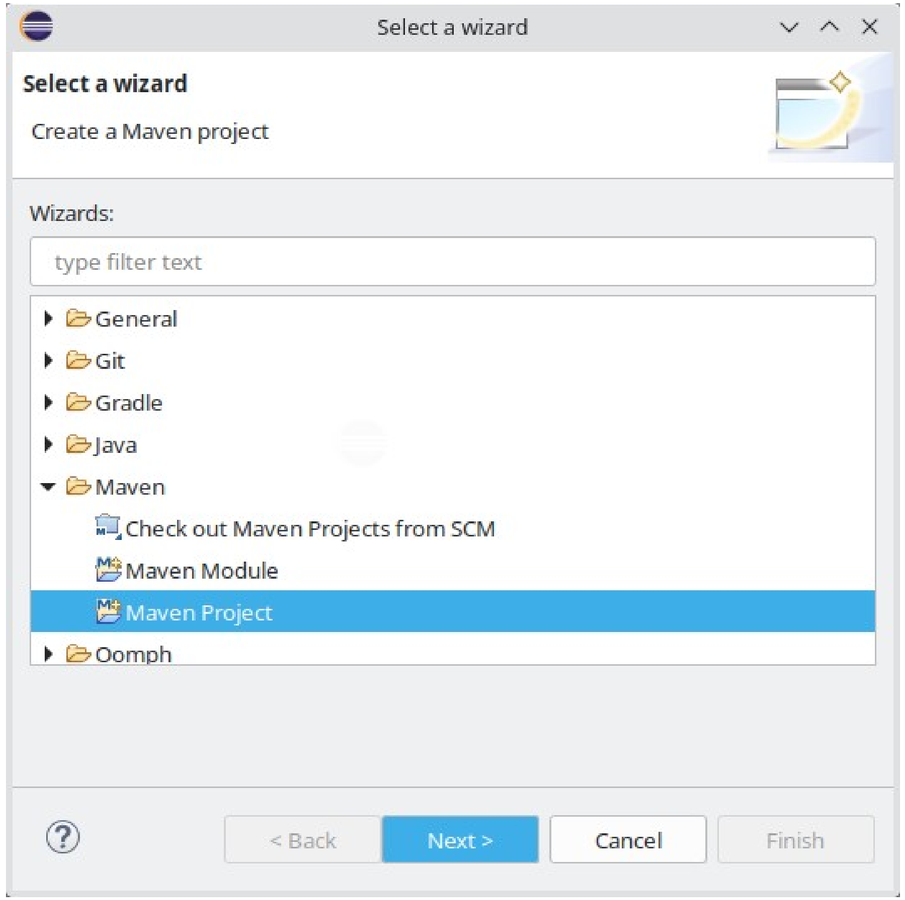
Download the binaries from the MongoDB Download Center.

1. Open Windows Explorer/File Explorer.
2. Change the directory path to where you downloaded the MongoDB `.msi` file. By default, this is `%HOMEPATH%\Downloads`.
3. Double-click the `.msi` file.
4. The Windows Installer guides you through the installation process.
   If you choose the **Custom** installation option, you may specify an installation directory.
   MongoDB does not have any other system dependencies. You can install and run MongoDB from any folder you choose.

Step 2. Create the project within the IDE

The first step is to open Eclipse, which comes with the integrated Maven environment.

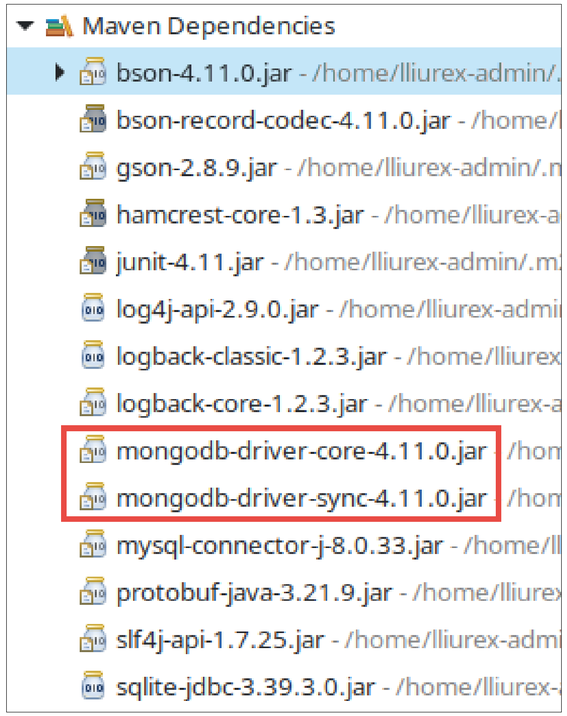
- Go to the File menu, option **New → Project**.
- Select the **Maven Project** option.
- We follow the rest of the steps explained in the extended notes.
- Finally, we select **Project → Clean** on our project so the necessary libraries and files have been downloaded correctly.



Step 3. Add the dependency to the POM file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project                                xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>ceed.ada</groupId>
  <artifactId>U2JDBCExample</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>U2JDBCExample</name>
      [...]
  <dependencies>
   <!-- https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-sync -->
   <dependency>
       <groupId>org.mongodb</groupId>
        <artifactId>mongodb-driver-sync</artifactId>
        <version>4.11.0</version>
   </dependency>
   <!--  Gson: Java to Json conversion -->
   <dependency>
       <groupId>com.google.code.gson</groupId>
       <artifactId>gson</artifactId>
       <version>2.8.9</version>
       <scope>compile</scope>
   </dependency>
  </dependencies>
```

Go here and click on your MongoDB version number to get the code:
https://mvnrepository.com/artifact/org.mongodb/mongodb-driver-sync



Step 4. Create a class, the imports and connection methods

- We create a class called DBMongoDB.
- We create the necessary imports.
- We create the connection methods to the database

```java
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoIterable;
import com.mongodb.client.model.Projections;

import org.bson.Document;
```

For further information "**Connect to MongoDB**":
https://www.mongodb.com/docs/drivers/java/sync/upcoming/fundamentals/connection/connect/

```java
//DB constants
static final String URL = "mongodb://localhost:27017";

/*
 * Connect to the database
 */
public static MongoClient connectToDB() {

    try {
        // https://www.mongodb.com/languages/java
        // You can instantiate a MongoClient object without any parameters to connect to
        // a MongoDB instance running on localhost on port 27017:
        MongoClient cnDB = MongoClients.create(URL);
        System.out.println("Connection to database has been established.");
        return cnDB;
    } catch (Exception exe) {
        System.out.println("Something was wrong while trying to connect to the
database!");
        exe.printStackTrace(System.out);
    }
    return null;
}

/*
 * Just try to disconnect to the database
 */
public static void closeDB(MongoClient cnDB) {

    try {
        cnDB.close(); //close connection to the DB
    } catch (Exception exe) {
        System.out.println("Something was wrong while closing the database!");
        exe.printStackTrace(System.out);
    }
}
```

Step 5. Connect to the database

Establish the connection

The connection to the DB is established using the getDatabase method passing as parameter DBNAME, which is the name of the DB.

```
//DB constants
static final String DBNAME = "ADAU2MDBCompany";

public static void main(String[] stArgs) {
   MongoClient cnDB = connectToDB();

   try {
      //Instruction getDB is deprecated!
      //establish the connection to DB
      MongoDatabase mDB = cnDB.getDatabase(DBNAME);

   } catch (Exception exe) {
      System.out.println("Something went wrong!");
      exe.printStackTrace(System.out);
   }
   closeDB(cnDB);

}
```

4. DDL QUERIES

Execute DDL sentences

We can check if the collection exists or not and then create it.

```java
public static void main(String[] stArgs) {
   MongoClient cnDB = connectToDB();

   try {
      //Instruction getDB is deprecated!
      //establish the connection to DBCompany
      MongoDatabase mDB = cnDB.getDatabase(DBNAME);
      createCollectionIfNotExists(mDB);

   } catch (Exception exe) {
      System.out.println("Something went wrong!");
      exe.printStackTrace(System.out);
   }
   closeDB(cnDB);

}
```

```java
//DB constants
static final String COLLECTIONNAME = "Employees";


/*
 * Check if collection exists
 * https://stackoverflow.com/questions/53810753/how-to-check-collection-
mongo-db-in-java
 */
public static boolean collectionExists(String stCollection, MongoDatabase
mDB) {

   MongoIterable<String> mitCollection = mDB.listCollectionNames();
   for (String stIterCollection : mitCollection) {
      if (stIterCollection.equals(stCollection)) {
         return true;
      }
   }
   return false;
}

/*
 * Create collection if not exists
 */
public static void createCollectionIfNotExists(MongoDatabase mDB) {

   try {
      if (!(CollectionExists(COLLECTIONNAME, mDB))) {
         System.out.println("Collection does not exist");
         mDB.createCollection(COLLECTIONNAME);
         System.out.println("Created collection " + COLLECTIONNAME + "
in given database...");
      }
   } catch (Exception exe) {
      System.out.println("Something was wrong when creating the
collection!");
      exe.printStackTrace(System.out);
   }
}
```

5. DQL QUERIES

Reading a collection

Here you can see how we would do this using the console (with and without _id):

```
ADAU2MDBCompany> db.Employees.find()
[
  {
    _id: ObjectId('66c62aeed10857367299f38d'),
    taxID: '11111111A',
    firstname: 'José',
    lastname: 'Salcedo López',
    salary: '1279.90'
  },
  {
    _id: ObjectId('66c62aeed10857367299f38e'),
    taxID: '22222222B',
    firstname: 'Juan',
    lastname: 'De la Fuente Arqueros',
    salary: '1100.73'
  },
  {
    _id: ObjectId('66c62aeed10857367299f38f'),
    taxID: '33333333C',
    firstname: 'Antonio',
    lastname: 'Bosch Jericó',
    salary: '1051.45'
  },
  {
    _id: ObjectId('66c62aeed10857367299f390'),
    taxID: '44444444D',
    firstname: 'Ana',
    lastname: 'Sanchís Torres',
    salary: '1300.02'
  },
  {
    _id: ObjectId('66c62aeed10857367299f391'),
    taxID: '55555555E',
    firstname: 'Isabel',
    lastname: 'Martí Navarro',
    salary: '1051.45'
  }
]
```

```
ADAU2MDBCompany> db.Employees.find({}, {_id:0})
[
  {
    taxID: '11111111A',
    firstname: 'José',
    lastname: 'Salcedo López',
    salary: '1279.90'
  },
  {
    taxID: '22222222B',
    firstname: 'Juan',
    lastname: 'De la Fuente Arqueros',
    salary: '1100.73'
  },
  {
    taxID: '33333333C',
    firstname: 'Antonio',
    lastname: 'Bosch Jericó',
    salary: '1051.45'
  },
  {
    taxID: '44444444D',
    firstname: 'Ana',
    lastname: 'Sanchís Torres',
    salary: '1300.02'
  },
  {
    taxID: '55555555E',
    firstname: 'Isabel',
    lastname: 'Martí Navarro',
    salary: '1051.45'
  }
]
```

Reading a collection

And our code in Java:

```
MongoCollection<Document> mCollection = mDB.getCollection(COLLECTIONNAME);

// Retrieving the documents
MongoCursor<Document> mCursor =
mCollection.find().projection(Projections.excludeId()).iterator();
int iNumItems = 0;
while (mCursor.hasNext()) {
   iNumItems++;
   Document docEmployee = mCursor.next();
   System.out.println(DOCUMENTNAME);
   System.out.println("Tax ID: " + (String) docEmployee.get("taxID"));
   System.out.println("First name: " + (String) docEmployee.get("firstname"));
   System.out.println("Last name: " + (String) docEmployee.get("lastname"));
   System.out.println("Salary: " + (String) docEmployee.get("salary"));
   System.out.println("");
}
if (iNumItems == 0)
   System.out.println("No items found on collection " + COLLECTIONNAME);
mCursor.close(); //close cursor
```

We need to remove the self-generated _id field

Output:

```
Employee Tax ID: 11111111A
Employee First name: José
Employee Last name: Salcedo López
Employee Salary: 1279.90

Employee Tax ID: 22222222B
Employee First name: Juan
Employee Last name: De la Fuente
Arqueros
Employee Salary: 1100.73

Employee Tax ID: 33333333C
Employee First name: Antonio
Employee Last name: Bosch Jericó
Employee Salary: 1051.45

Employee Tax ID: 44444444D
Employee First name: Ana
Employee Last name: Sanchís Torres
Employee Salary: 1300.02

Employee Tax ID: 55555555E
Employee First name: Isabel
Employee Last name: Martí Navarro
Employee Salary: 1051.45
```

6. DML QUERIES

Inserting data

Here you can see how we would do this using the console:

```
ADAU2MDBCompany> db.createCollection("Employees")
{ ok: 1 }
ADAU2MDBCompany> db.Employees.insertOne({taxID: "11111111A", firstname: "José", lastname: "Salcedo López", sal
ary: "1279.90"});
{
  acknowledged: true,
  insertedId: ObjectId('66c628cec7113e9c685e739d')
}
ADAU2MDBCompany> db.Employees.find()
[
  {
    _id: ObjectId('66c628cec7113e9c685e739d'),
    taxID: '11111111A',
    firstname: 'José',
    lastname: 'Salcedo López',
    salary: '1279.90'
  }
]
```

```java
//create collection Employees
mDB.createCollection(COLLECTIONNAME);
System.out.println("Created collection " + COLLECTIONNAME + " in
given database...");
//insert employees
Document docEmployee;
docEmployee = new Document("taxID",
String.valueOf("11111111A")).append(
                "firstname", String.valueOf("José")).append(
                "lastname", String.valueOf("Salcedo López")).append(
                "salary", String.valueOf("1279.90"));

mDB.getCollection(COLLECTIONNAME).insertOne(docEmployee);
```

We can do several times **insertOne** or do just one **insertMany**

Inserting data (arrayList)

Here you can see how we would do this using the console. In this case it refers to a database of books.

```
> db.createCollection("books")
{ "ok" : 1 }
> db.books.insertMany([{bookId:"123112312", bookName: "HARRY POTTER"}, {bookId: "34556346", bookName: "ETERNALS"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("61870da1ce53823a1e3fa27e"),
                ObjectId("61870da1ce53823a1e3fa27f")
        ]
}
> db.books.find({},{_id:0})
{ "bookId" : "123112312", "bookName" : "HARRY POTTER" }
{ "bookId" : "34556346", "bookName" : "ETERNALS" }
>
```

```java
MongoClient cnDB = connectToDB();
try {
   MongoDatabase mDB = cnDB.getDatabase("ADAU2MDBLibrary");

   Iterator<Book> itBook = arlBook.iterator();
   Document docBook;
   while (itBook.hasNext()) {
      itemFound = (Book) (itBook.next());
      bookId = String.valueOf(itemFound.getISBN());
      bookName = String.valueOf(itemFound.getName());
      docBook = new Document("bookId", bookId).append("bookName",
bookName);
      System.out.println("Element about to be inserted...");
      mDB.getCollection("books").insertOne(docBook);
   }
} catch (Exception exe) {
   System.out.println("Something was wrong while populating the
collection!");
   exe.printStackTrace(System.out);
}
closeDB(cnDB);
```

Assume that we have an arrayList with the books.

We can do several times **insertOne** or do just one **insertMany**

7. PATCHES IN JAVA

Useful resources

By default, Mongo driver shows a text message with every operation. Use this patch (imports + POM lines + method) to remove this annoyning feature.

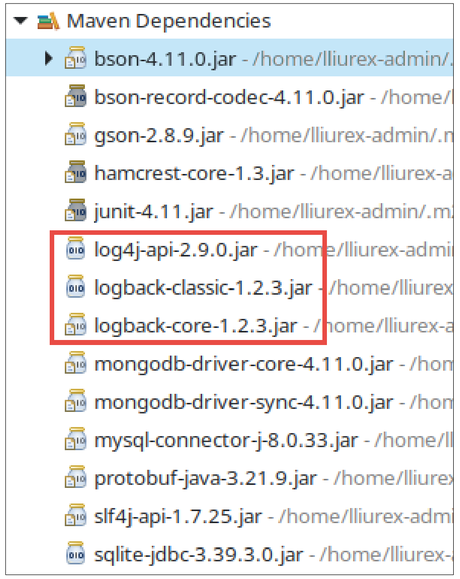
```
<!--  prevent log to print unwanted messages -->
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.9.0</version>
</dependency>
<!--  prevent log to print unwanted messages -->
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.3</version>
</dependency>
```

```
import org.slf4j.LoggerFactory;

import ch.qos.logback.classic.Level;
import ch.qos.logback.classic.LoggerContext;

[…]

/*
 * Static method: Disable annoying mongoDB log messages This method require add some code
to POM
 * https://stackoverflow.com/questions/30137564/how-to-disable-mongodb-java-driver-logging
 */
public static void DisableMongoLogging() {
   ((LoggerContext)
LoggerFactory.getILoggerFactory()).getLogger("org.mongodb.driver").setLevel(Level.ERROR);
}
```



Source: https://stackoverflow.com/questions/30137564/how-to-disable-mongodb-java-driver-logging

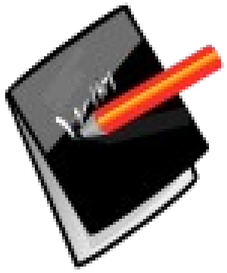
8. ACTIVITIES FOR NEXT WEEK

Proposed activities

Check the suggested exercises you will find at the “Aula Virtual”. **These activities are optional and non-assessable but** understanding these non-assessable activities is essential to solve the assessable task ahead.

Shortly you will find the proposed solutions.

9. BIBLIOGRAPHY



Resources

- MongoDB Documentation. https://www.mongodb.com/docs/
- W3 Schools.MongoDB Tutorial. https://www.w3schools.com/mongodb/
- Geeks for Geeks. MongoDB Tutorial. https://www.geeksforgeeks.org/mongodb-tutorial/?ref=lbp

// EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDU

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

ceedcv CENTRE ESPECÍFIC D'EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

CC BY NC ND

A // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A DISTANCIA // EDUCACIÓN PÚBLICA Y GRATUITA // EDUCACIÓN DE CALIDAD // EDUCACIÓN AUTONÓMICA // EDUCACIÓN A