## Explain, generally, what is meant by a NoSQL database

A NoSQL database is a database where it’s possible to store and retrieve data without the usage of SQL statements. This is in contrast to a relational database where every query made is done by using SQL query calls. The need for NoSQL databases came due to the fact that normal relational databases where not designed to cope with scale and agility challenges that many developers face today.

Some of the different kind of NoSQL databases that exist are:

* **Document-oriented databases:** These pair each key with a complex data structure also known as a document. Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.
* **Graph stores:** These are used to store information about networks of data, such as social connections.
* **Key-value stores:** These are the simplest NoSQL databases. Every item is stored as an attribute name (a key), together with its value.
* **Wide-column stores:** These are used for large datasets, and store columns of data together instead of in rows.

Another difference between NoSQL databases and relational databases is that the latter uses schemas to be defined before data is added. A NoSQL database allow the insertion of data without a schema which makes it easy to make application changes in real-time without worrying about service interruptions. That said it’s also possible to make schemas for NoSQL databases with the use of node modules. A popular node module for this is mongoose.

## Explain Pros & Cons in using a NoSQL database like MongoDB as your data store, compared to a traditional Relational SQL Database like MySQL

**Pros:**

* **Flexible Data Model:** Can store and combine any type of data.
* **Dynamically updated schemas:** As NoSQL databases don’t require schemas, this means that if requirements for the database changes the “schema” can be updated without any downtime (taking the database offline as would be needed with a relational database)
* **Elastic Scalability:** NoSQL databases scale out on low cost, commodity hardware, allowing for almost unlimited growth
* **High Performance:** Faster queries, inserts and updates

**Cons:**

* **Might lose referential integrity guarantees:** “Eventual consistency” is a term used in the NoSQL universe, meaning that an update or insert on one server may not be visible on other servers immediately.
* **No universal language:** There is no universal language for NoSQL databases like SQL

## Explain how databases like MongoDB and redis would be classified in the NoSQL world

MongoDB is a document-oriented database, and redis is a key-value store database.

A document-oriented database (document store) is designed for storing, retrieving and managing document-oriented information. As the name says, a document store is about documents that can be encoded in some standard format. Example of encodings: XML, JSON, or BSON that MongoDB uses. Documents are retrieved, edited, or deleted via a unique key that represents a specific document. The key is an identifier that is typically a string.

A key-value store database is as mentioned earlier the simplest NoSQL databases. This kind of database as a key to every item in the database, together with the keys value.

## Explain, using relevant examples, the strategy for querying MongoDB (all CRUD operations)

I’ll do a step-by-step guide here:

* Open a command prompt
* Write “*mongod*” so mongoDB is accessible
* Write “*mongo*” to open op the mongo shell
* To create a database type “*use league*”. This will create a database called league.
* Write “*db.createCollection(“Teams”)”* to create a collection of teams within the database.
* Create a variable called a by writing “*var a = “name”: “Sønderjyske”, “points”: 55}”*
* Save the variable into our teams collection by writing “*db.Teams.save(a)*”. This is the First CRUD operation we use (Create).
* We can now use the next CRUD operation (READ) by writing “*db.Teams.find()*”.
* We can also find a specific object within our database by writing “*db.Teans.find({“name”: “Sønderjyske”});*”. This will find teams in the collection named Sønderjyske.
* We are now at the next step in the CRUD operations (UPDATE) and we can do that by writing “*db.Teams.update({“name”:”Sønderjyske”},{“name”:”FC København”});*”. This will update teams with the name of Sønderjyske into being called FC København.
* This brings us to the last CRUD operation (DELETE) which we can do by writing “*db.Teams.remove({“name”: “FC København”});*”. This will delete our object from our collection.

## Demonstrate, using a REST-API, how to perform all CRUD operations on a MongoDB

The project RestAPI shows all CRUD operations in use, getting data from MongoDB. It has a simple index site on path: “localhost:3000”, where everything can be seen in action.

The project SimpleRestfullAPI is the simplest way possible to create a REST API. It uses the node module node-restful to auto generate all the CRUD operations (post, get, put, delete). The post and get operations work on the path: “localhost:3000/api/products, and the put and delete methods works on the same path with an id “localhost:3000/api/products/:id. MongoDB is not put into this, but this project is mainly to show of the node-restful module.