

# NoSQL Databases

This lesson will introduce NoSQL databases and go over some of their common characteristics.

## WE'LL COVER THE FOLLOWING ^

- Introduction
- Common NoSQL Characteristics



## Introduction #

The term **NoSQL** originated as a Twitter hashtag for a meet-up back in 2009. It is sometimes translated as an acronym for *Not Only SQL*, or short-hand for *No SQL*.

This term is pretty loose and is used to cover a wide range of databases that try to tackle problems that relational databases have with newer applications. These include:

- *Flexibility*
- *Scalability*
- *Performance*

- *Performance*

Nevertheless, in order to do so, these databases have sacrificed some of the good things that relational databases provided, such as:

- *Expressive query language*
- *Secondary indexes*
- *Transactional mechanisms*
- *Strong consistency*

Here are some examples of *NoSQL* databases:



Some examples of NoSQL databases

## Common NoSQL Characteristics #

NoSQL databases have these common characteristics:

- They usually don't use *SQL*. However, they do have their own querying languages which are often similar to *SQL* since *SQL* is easy to learn. For example, *Cassandra's* querying language is called *CQL*, or *Couchbase's* *N1QL*, and actually extends *SQL* for *JSON*.
- They are not *relational databases*; meaning that they don't provide a set of formally described tables in which data should fit. Essentially

or formally described tables in which data should fit. Essentially, *relational* databases are named as they are because they are structured

around *tables* and *relations* between those tables. *NoSQL* databases **don't** have tables (at least not in the way that *relational* databases do) and, therefore, **no relations**.

- Most of them are *cluster-friendly*. The initial idea was to store databases on multiple machines; but, some *NoSQL* databases are *Graph oriented* just like many applications today (e.g, online social networks where people are nodes, who are connected to other nodes, with edges in a friendship, relationship, etc.).
- They don't enforce a fixed *scheme* as strongly as relational databases do. Hence, it is possible to add a field into "record" without first making changes to the structure itself. Since, in the relational databases, you have to know in advance what you want to store, *NoSQL* databases make this process easier.

All the above are common characteristics, but certainly, by no means are they the definition of *NoSQL*. At this point, I don't think we'll ever have a full, proper definition of *NoSQL* databases. However, this may be for the best as it goes hand-in-hand with *NoSQL*'s "free spirit" identity.

---

Now that you're familiar with *NoSQL*, we will learn about different types of *NoSQL* databases.