# NoSQL Databases: Document Stores

(https://github.com/learn-co-curriculum/dsc-nosql-document-stores-v2-1) (https://github.com/learn-co-curriculum/dsc-nosql-document-stores-v2-1/issues/new/choose)

## Introduction

In this lesson, we'll learn more about NoSQL databases, and a specific NoSQL category, *Document Stores*.

## Objectives

You will be able to:

- Describe why NoSQL is useful
- Describe situations when one would use NoSQL databases
- List different NoSQL databases

## Why NoSQL?

Relational databases are a cornerstone of a modern technology. They're reliable, dependable, and they seem to be pretty much everywhere. Since their invention at IBM by Edgar Codd in 1970, they've rapidly grown to be used all over the place. Their creation allowed companies to track, store, and analyze data in ways that simply couldn't be done before. For the majority of situations, they're a great choice. However, as technology has progressed into the era of internet and smartphones, we've run into many different sorts of data that aren't a natural fit for a relational format. Let's examine a few of these situations, and see why NoSQL might be a better choice.

Let's assume that we need to store chat logs between customer service and customers through a web interface. These chats could be very short, or very long -- some chats may only be 2 or 3 messages, while others may conceivably be in the hundreds or thousands. For each message in the chat, we want to store metadata associated with the message, so that we can know things like which party sent the message, the time it was sent, the time the other party read the message, etc. This is a great use case for a NoSQL database, because it would be a very poor fit for a relational database. For starters, each chat can be any size, meaning that we can't just clearly define a table that links which messages belong to which chat. If every chat had only five messages, we might be able to make it work, with a column for "message 1", "message 2", and so on -- but we can't, because a chat has no set size, and can always grow larger in the future. A relational database would also waste a lot of space storing redundant information, and getting all the messages in a chat could result in s‹ Help times for our SQL query if this data was stored in separate tables in something like th‹ format.

There are several variants of NoSQL databases. They can be categorized into:

- Document Stores
- Key-Value Stores
- Column Stores
- Graph Databases

In this lesson, we will explore *Document Stores*.

# Document Stores

A ***Document Store*** is a database that stores records as unique documents in the database. These documents can be arbitrarily long, and can even contain other documents inside of them! The chat log example we saw above is a prime use case for a document store. In a document store, we could store each message and its accompanying metadata as a document, and then embed each of those documents in order in a chat document. In this way, we can easily access the data as needed.

In these Document Stores, each document contains key-value pairs, with the actual data being stored in as the value. This makes Document Stores incredibly flexible, because each document can be unique. There is no constraint saying each document must have the same keys! This makes it great for working with data where we don't know what shape it will take (as we saw above, with chat logs that can be arbitrarily long or short), or perhaps when we don't know what data will be stored at all. This would be a problem in a relational database, because we would need to know what column the data belongs in before we could store it. With a Document Store, we can just create a key on the fly for the data that matters to us!

Note that while this flexibility makes it easy for us to store data on the fly, this also makes it harder for us to query data and get exactly what we need. Since each different document can potentially have its own ***Schema***, this means that we have to know what we are looking for. This also means that we have to be diligent in our naming conventions, because `chatLog` is different than `ChatLog`. This means that if we run a query across all documents to get all data with the key `chatLog`, we'll completely miss any data where they key is written as `ChatLog`!

## Popular Document Store Databases: MongoDB and Couchbase

Help


# Summary

In this lesson, we learned about the various sorts of NoSQL Databases in the market today. We dug into the similarities and differences between them all, and also looked at a few examples where a NoSQL Database is a more natural fit for storing data than a traditional relational database.

In this lesson, we looked at examples where a NoSQL database is a more natural fit for storing data than a traditional relational database. We also looked at *Document Stores*, one of the few types of NoSQL databases.

How do you feel about this lesson?

Have specific feedback?

**Tell us here! (https://github.com/learn-co-curriculum/dsc-nosql-document-stores-v2-1/issues/new/choose)**

Help