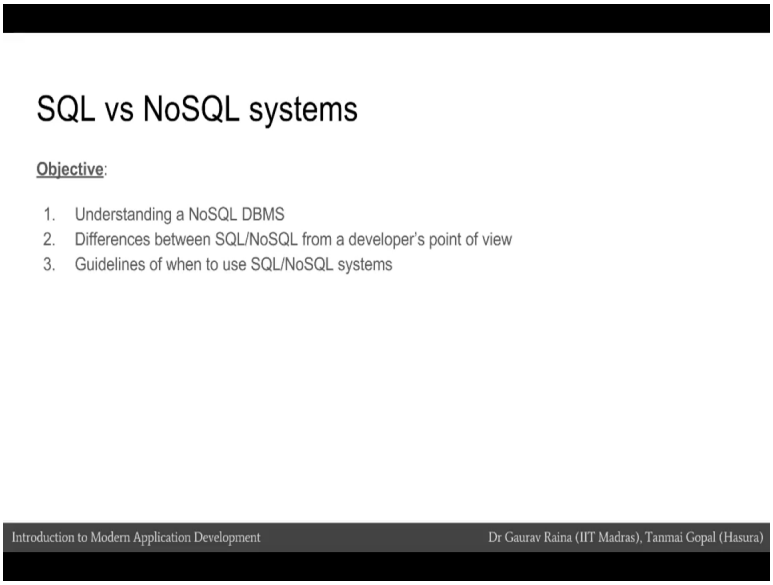
Introduction to Morden Application Development

Dr. Gaurav Raina
Prof. Tanmai Gopal
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

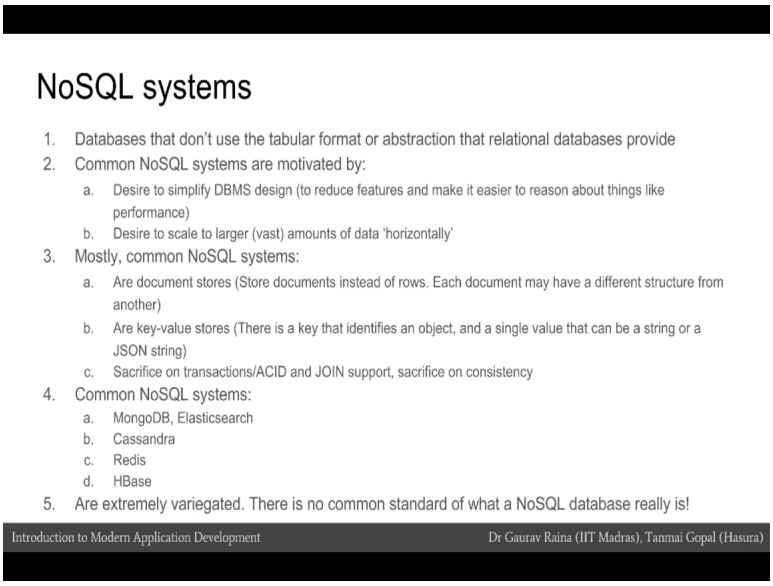
Module - 17
Lecture - 23
SQL and NoSQL systems

(Refer Slide Time: 00:04)



In this module we will be talking about SQL versus NoSQL systems. Our main objective is to understand what a NoSQL DBMS is and what is the differences between SQL and NoSQL systems are and what the key takeaways for us as developers are and how we should make the choice of using a particular database for the applications that we are building.

(Refer Slide Time: 00:20)



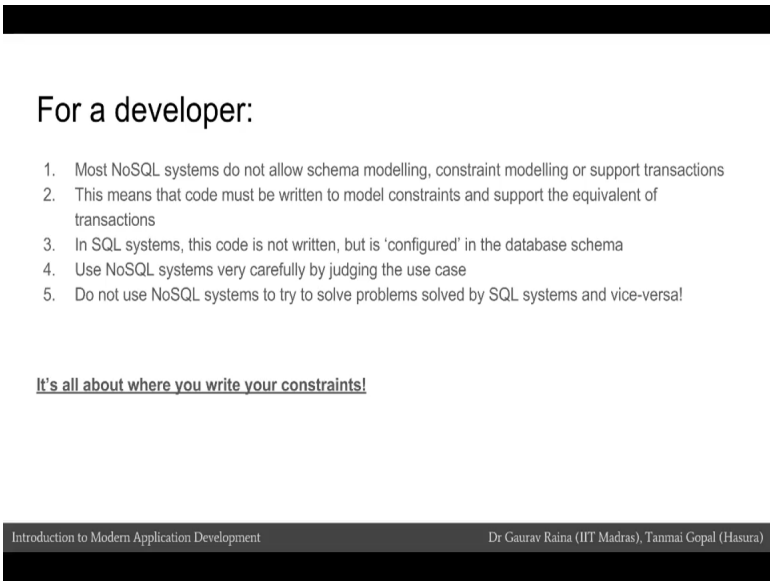
So, NoSQL systems are systems that typically do not use the tabular format or the extraction of having a table columns put together in a schema that kind of abstraction in the relational database provide. Mostly NoSQL systems are one of two times. So, they are either document stores where they are storing documents instead of rows. So, two or more documents inside a collection or inside a table may have nothing to do with each other. This is very different to relational model where multiple rows in the table all have the same schema and the same columns and each column has the same type; however, in a database like MongoDB you can have documents that are stored in a collection which is the equivalent of rows in a table and each of these documents might actually have different columns and they might even have the same column names, but with different types.

The other kind of common NoSQL system is key value store and you can think of key value stores as tables that are just two columns a table and a value key value stores are very useful for caching like applications and Redis is one of the best examples of a key value NoSQL system. Most NoSQL system sacrifice on support for transaction acid and joins. So, they also sometimes sacrifice on the kind of consistency that you expect in a relational database because of the lack of a schema. NoSQL systems in modern times has been motivated by two main factors, the first main factor is the desire to simplify the DBMS design and by simplifying DBMS design I mean that these NoSQL systems try to reduce the number of features that a database can provide for example, foreign key,

constraint checking or transaction support and they do this. So, that it is easier to reason about things like performance because the database does much lesser work when you are trying to read data or write data a very good example of this kind of a system is MongoDB.

The other main motivation behind NoSQL systems has is the desire to be able to scale to vast amount of data very very easily. So, if you are willing to sacrifice on certain features of a database then NoSQL systems are able to provide a very easy way to scale to very very large amounts of data and by very very large amount of data I am talking about the scale of a few terabytes to a few petabytes. Some common NoSQL systems are MongoDB, Elasticsearch, Cassandra, Redis, HBase, most all SQL systems are extremely diverse and there is no actual common standard of what a NoSQL database is or how a NoSQL database is structured it is not really two categories of databases is called NoSQL databases and SQL databases its rather SQL databases and other databases which are all grouped together and called NoSQL databases.

(Refer Slide Time: 03:17)

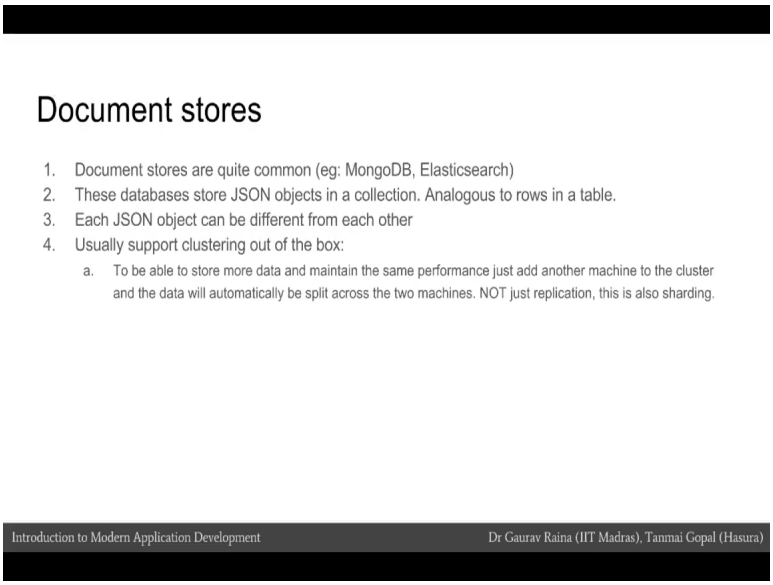


From a developers point of view the main difference a NoSQL database and an SQL database is about how we think about are constraints. So, as a developer if we want the database to do our work to do the work of checking our constraints and making sure that constraints in our data model are validated then SQL is a good fit. However, as developers if we do not want the database to model our constraint willing to do the extra

work to gain on features like scaling to hundreds of terabytes of data then the no then the NoSQL system is a good fit. For example, NoSQL systems do not allow modeling of schema constraint modeling or support transactions most NoSQL systems some SQL systems some NoSQL systems do support a subset of these features. So, if we use a NoSQL database; that means, that we must write code in our application to actually model these constraints for example, if we look at the application that we were building if we want to enforce that in an article the user id or the author id of a particular article must actually exist in a user table.

It is not possible to enforce this kind of a constraint in a NoSQL system. This kind of a constraint must be validated by us at the application; that means, that if somebody goes and modifies the database directly there is no guarantee that the database will have that the data models will have a certain amount of consistency. It is very important to remember that we should not try to take a NoSQL system to try to solve the problems that are already solved by SQL systems and vice versa.

(Refer Slide Time: 04:52)

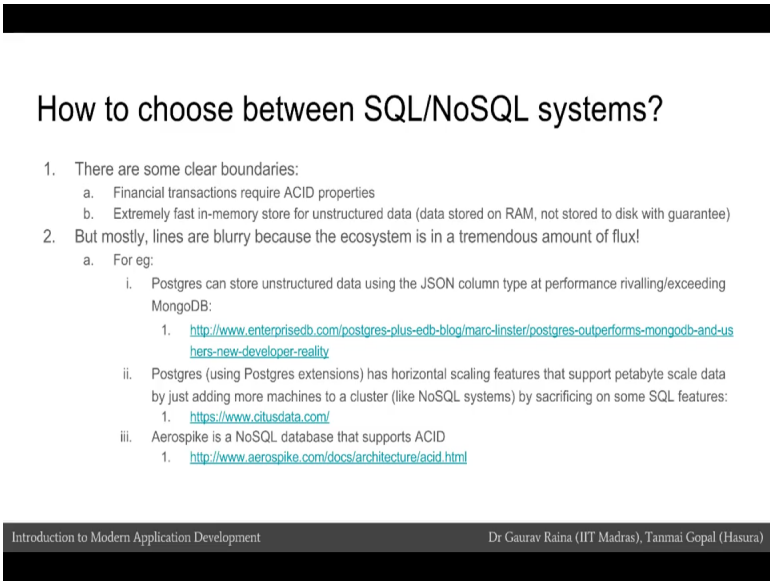


Let us talk a little bit about document stores which are a category of a NoSQL database. So, MongoDB and Elasticsearch and the two most common examples of document stores these databases store JSON like objects in a selection and these objects or documents are analogous to rows. Both MongoDB and Elasticsearch try to support clustering out of the box which means that if you want to store more data and we want to have the same

performance then we just have to add another machine to our cluster and the database system will automatically replicate and short the data between two instances of the database in the cluster.

So, we do not have to worry about how to short our data, but we do not have to worry about how to do the replication and this happens automatically.

(Refer Slide Time: 05:38)

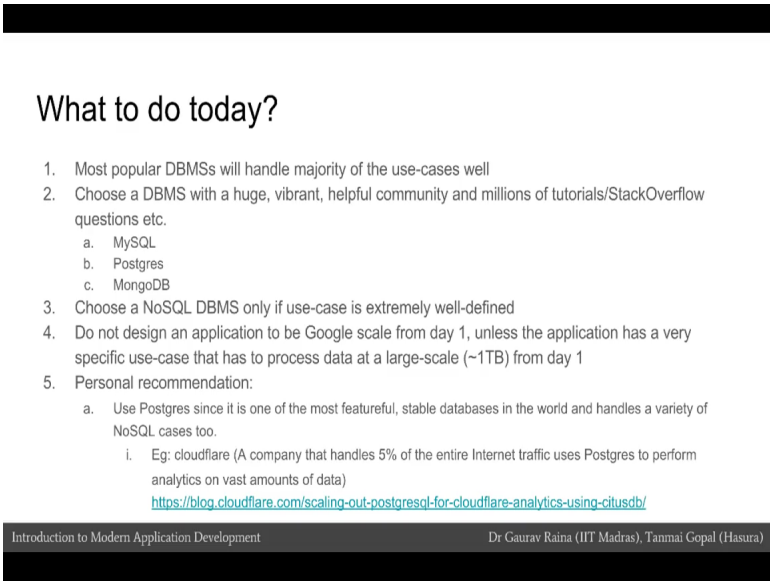


So, if we have to choose between SQL and NoSQL systems what are the factors that we should use to decide? Sometimes there are very very clear cut use cases for SQL, NoSQL systems for example, if you want to support financial transactions that require acid properties then in such a situation as a developer one should blindly choose a mature SQL system which will support transactions and ACID be the primary database.

Another extreme example is if we need a very fast in memory store which is typically used cased for caching and we want to store our data on RAM, we do not care about writing a detect (Refer Time: 06:21) and we want read and write access to be extremely fast this is a perfect use case for a NoSQL system like Retis. But most of the times the lines a blurry it is not a clear cut choice between a SQL and a NoSQL systems let us look at a few example. Postgress is technically an SQL database and relational database; however, Postgres can store unstructured data using the JSON column type and in fact, Postgress performance in reading and writing JSON data rivals or exceeds that of MongoDB in a few cases.

In another case there are extensions to the Postgres system that allow for horizontal scaling to petabyte scale data and just by adding more machine to cluster very similar to NoSQL systems and they do this by sacrificing on some SQL features. Similarly Aerospike is a NoSQL database that actually supports acid properties. A lot of things are continuously changing month on month year or year in this environment and so it is very hard and so is very hard to choose a database that is one size fits or that will work in all scenarios.

(Refer Slide Time: 07:32)



So, when we are in a scenario like this how do we make right decisions? The first important thing to remember the developer especially if we are starting to build a application is at almost any popular DBMS is good enough if you building a simple application the difference is not going to be between choosing MongoDB or MySQL Postgres. So, the difference is actually going to be in building the application till it reaches a scale of data or data usage that actually brings out the differences between these different database management systems.

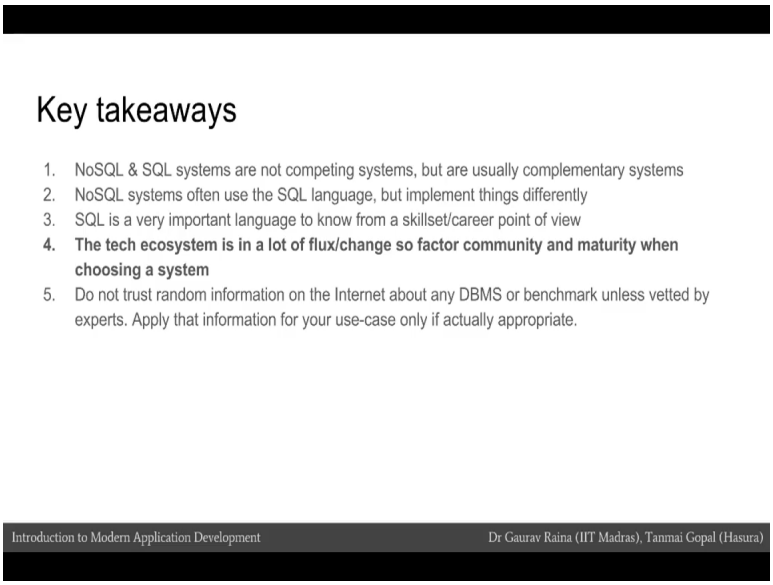
If you have a database size which is less than one terabyte and your typical web application then there will be almost no difference between you using MongoDB MySQL or PostgreSQL. When you are choosing a DBMS the most important thing is to choose a DBMS with a huge and a vibrant community possibly one with several tutorials and stack overflow questions and answers so that all of the different use cases that you

want to try to do and that you want to achieve are already solved and answered and there is an open community around it. Data bases with some of the best communities are mysql Postgress and MongoDB.

Unless you have an extremely a set application do not use a NoSQL DBMS this particular point is a little opinionated and a personal judgment; however, it is something that would be acquired by most experienced developers. If you are choosing a NoSQL DBMS make sure that the use case is extremely well defined because as a developer there is going to be a lot of extra work that you will have to do if you go for a NoSQL system, which also is closed related to the point that do not design the application to be Google scaled from day one unless the application is working in a very very specific domain that has to process data at the scale of a few terabytes right from day one.

Unless your application is processing a few terabytes of data right from day one do not even think about any fancy database and just go with something that is popular and that has a huge community around it. My personal recommendation would be to use Postgres since it is one of the most feature full and stable databases in the world there are several examples of people running the Postgres database for months and even years without any amount of down time. Postgres is a very rich community and a very rich ecosystem of extensions that allows us to handle several NoSQL use cases like scale out. So, for example, Cloudflare uses a Postgres extension called Citus which allows it to handle hundreds of terabytes of data and Cloudflare uses that system to perform analytics.

(Refer Slide Time: 10:07)



Key takeaways are that NoSQL systems and SQL systems are not competing systems and you should not be looking at them as a versus b they are usually complementary systems that you are going to use in your application for achieving different objectives. SQL is a very important language to know from a skill set and a career point of view because it is a stepping stone to most specialized use cases for NoSQL systems or SQL systems. The tech ecosystem regarding database is in a fair amount of flux. So, factor and community and maturity when you are choosing a system and that is in fact, the most important point here.

Do not trust any random information that you find on the internet about any particular DBMS or do not trust random benchmarks without understanding how those benchmarks been achieved or unless those benchmarks have been rated by experts. Apply the information that you have learnt that you find in the community that you find online about DBMSs and apply that information for your use case only if it is actually appropriate.