**Relational Database**

Structure

* Tables
* Constraints
* Keys

Storage

* Concentrated (typically one node that contains entire copy. Not partitioned or segregated)

Scale

* Vertical (better machine)
* Horizontal (more machines). Master and read replicas.

Access

* Raw SQL with direct database connection
* ORM’s (object relational mappers)

**Pros**

* Data accuracy
* Easy access to data
* Data integrity
* Security

**Cons**

* Possible performance issues
* Set-up can be long

Harder to set up, easier to support.

**Non Relational Database**

Structure

* Tables
* Documents (json objects under the hood)
* Graphs (like social network)

Rely on key/value store. You need to know the key you’re looking for.

Storage

* Hash input (key given into a hash is a value. That value is distributed on one of multiple nodes.) If you give a key, the output would be physical location in which partition and where the data is stored.

Scale

* Adding more partitions that can scale individually.

Access

* Rest API’s
* CRUD in vendor specific language.

**Pros**

* Handling unstructured data
* Agility
* Readability (When need data on user, it’s enough to open one document)

**Cons**

* Limited functionality

**MongoDb**

* Support of various data types
* Ability to distribute data automatically between different servers.
* Fast performance.

--

* Requires more memory increasingly
* No control of duplication
* Lack of documentation

Easier to set up, harder to support.