**NoSQL Databases**

**Aman Gour 120050030**

Over the last few years industry has seen the rise of a new type of databases, known as NoSQL databases, that are challenging the dominance of relational databases. Relational databases have dominated the software industry for a long time providing mechanisms to store data persistently, concurrency control, transactions, mostly standard interfaces and mechanisms to integrate application data, reporting. The dominance of relational databases, however, is cracking.

**NoSQL Meaning?**

A NoSQL (often interpreted as Not Only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. NoSQL databases provides an application specific way of storing data that further prevents the extra consistency checks. NoSQL databases relies on the principle of “*application specific databases*” in contrast to “*one database to rule them all*” philosophy behind the relational databases. Today, relational databases are not able to solve the needs of the current scope for storing gigantic data. NoSQL technique for storing data is a substitute for solving nowadays needs. NoSQL means Not Only SQL, implying that when designing a software solution or product, there are more than one storage mechanism that could be used based on the needs.

**NoSQL vs SQL**

There are many aspects in which the NoSQL databases are different from the traditional database design (RDBMS) but here we will try to compare the three main properties mentioned in the table below:

|  |  |  |
| --- | --- | --- |
| **Properties** | **SQL** | **NoSQL** |
| Transaction | ACID | BASE |
| Scalability | Vertical | Horizontal |
| Object-Relational Mapping | Necessary | Not Necessary |

* **Scalability** 
  + Relational database can be scaled just vertically, because entire database has to be hosted in a single server. This is necessary in order to ensure reliability and continuous availability of data. Vertical scaling is too expensive, places limits on scale and weak fault-tolerance.
  + In contrast to relational database management systems most NoSQL databases are designed to scale well in the horizontal direction and not rely highly on available hardware. NoSQL databases were designed to scale horizontally. Instead of increasing power of one single server you just need to add more server instances to get expected power. Usually it's too expensive and limited to buy more RAM or more powerful server instead of just add one small workstation into the cluster.
* **Object Relational Mapping**
  + Most of the NoSQL databases are designed to store data structures that are either simple or more similar to the ones of object-oriented programming languages compared to relational data structures. NoSQL does not make the expensive object relational mapping necessary.
  + This is particularly important for small applications that do not need such a mapping and are benefitted and operates fast when such a mapping is not present in the database.
* **Transaction**
  + **BASE** (**B**asically **a**vailable, **S**oft-state and **E**ventual-consistency)**:** The BASE approach forfeits the ACID properties of consistency and isolation in favor of “availability, graceful degradation, and performance”. e. For a growing number of applications and use-cases availability and partition tolerance are more important than strict consistency provided by **ACID** (**A**tomicity, **C**onsistency, **I**solation and **D**urability) property of RDBMS.
  + *Strict consistencies* advocates that “All read operations must return data from the latest completed write operation, regardless of which replica the operations went to” but the *eventual consistency* states that “In a steady state, the system will eventually return the last written value”

**Types/Classes of NoSQL Databases**

NoSQL can be broadly categorized into 4 types:

1. Key-Value Databases
2. Document Databases
3. Column-family Stores
4. Graph Databases

**Key-Value databases**

|  |  |
| --- | --- |
| Key-value stores are the simplest NoSQL data stores to use from an API perspective. The client can either get the value for the key, put a value for a key, or delete a key from the data store.  Since key-value stores always use primary-key access, they generally have great performance and can be easily scaled. | key-value.png |

**Document databases**

|  |  |
| --- | --- |
| document.png | Documents are the main concept in document databases. The database stores and retrieves documents, which can be XML, JSON, BSON, and so on.  Document databases store documents in the value part of the key-value store; think about document databases as key-value stores where the value is examinable. |

**Column-Family Stores:** Column-family databases store data in column families as rows that have many columns associated with a row key. Column families are groups of related data that is often accessed together.

**Graph Based:** Graph databases allow you to store entities and relationships between these entities. Entities are also known as nodes, which have properties.

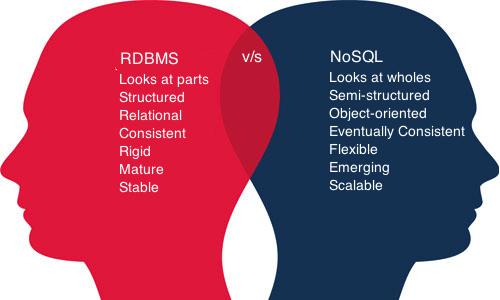
**Conclusions**

The availability of choice in NoSQL databases, is both good and bad at the same time. Good because now we have choice to design the system according to the requirements. Bad because now you have a choice and we have to make a good choice based on requirements and there is a chance where the same database product may be used properly or not used properly.

Broad reasons to consider the use of NoSQL databases :

* To improve programmer productivity by using a database that better matches an application's needs.
* To improve data access performance via some combination of handling larger data volumes, reducing latency, and improving throughput.

Picture worth a million words:



**References**

1. <http://sql-vs-nosql.blogspot.in/2013/10/the-base-difference-between-sql-and.html>
2. <http://en.wikipedia.org/wiki/NoSQL>
3. <http://www.thoughtworks.com/insights>

Along with the document on moodle.