

NoSQL

NOT ONLY SQL

The term NoSQL was coined by Carlo Strozzi in the year 1998.

NoSQL is a non-relational database management systems, different from traditional relational database management systems in some significant ways.

It is designed for distributed data stores where very large scale of data storing needs (for example Google or Facebook which collects terabits of data every day for their users).

These type of data storing may not require fixed schema, avoid join operations and typically scale horizontally.

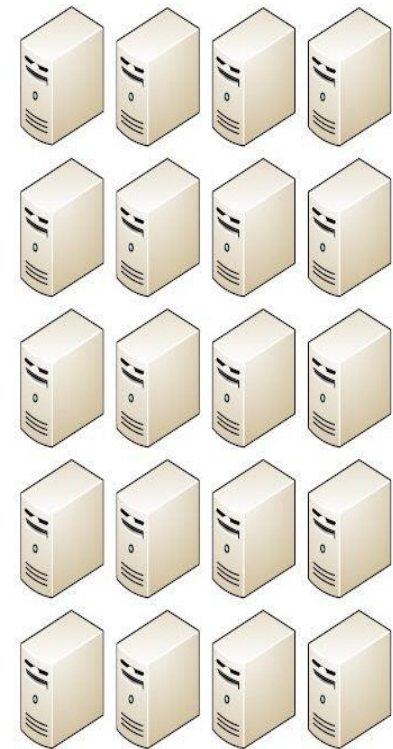
NoSQL why, what and

Era of Distributed Computing

- ❑ Relational databases were not built for **distributed applications**.

Rdbms is not preferred because

- ❑ Joins are expensive
- ❑ Hard to scale horizontally
- ❑ Impedance mismatch occurs (RIGID SCHEMA)
- ❑ Expensive (product cost, hardware, Maintenance)

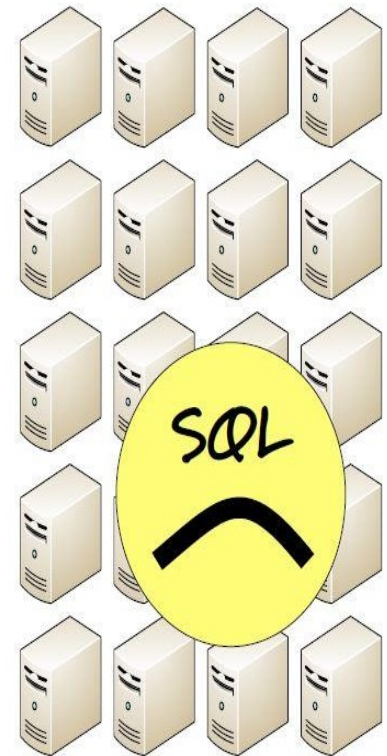


NoSQL why, what and

And....

It's weak in:

- ☐ Speed
(performance)
- ☐ High
availability
- ☐ Partition
tolerance



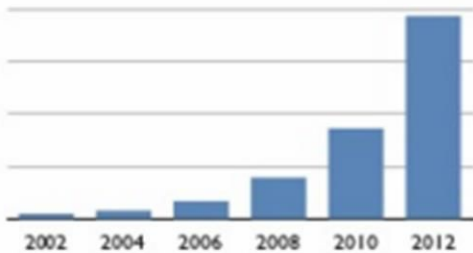
Why NOSQL

Ans. Driving

Why??

Trends

New Trends



Big data



Connectivity



P2P Knowledge



Concurrency

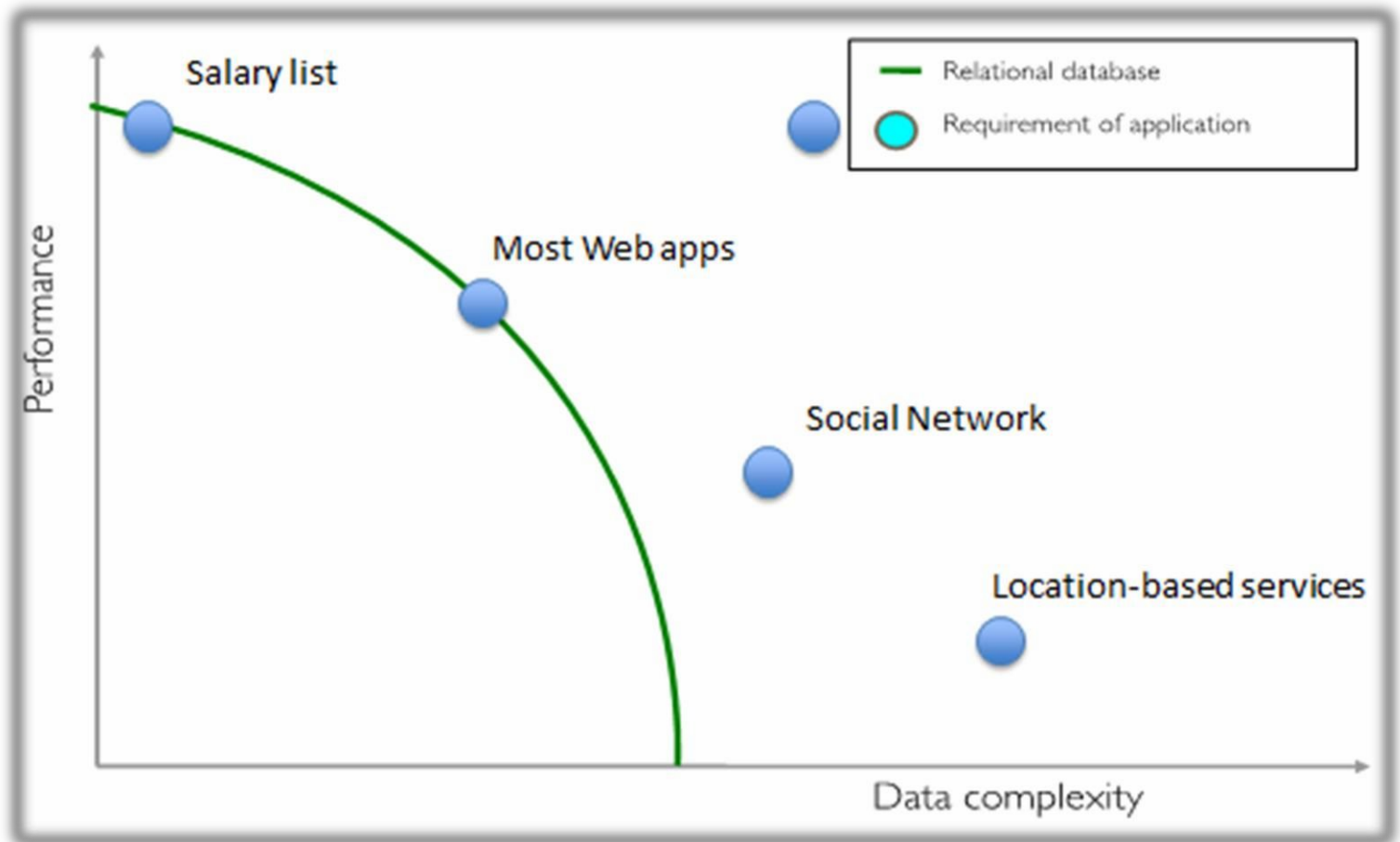


Diversity



Cloud-Grid

Side note: RDBMS



- ❑ A No SQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational database

Not only SQL

- ❑ No SQL systems are also referred to as "NotonlySQL" to emphasize that they do in fact allow SQL-like query languages to be used.



Characteristics of NoSQL

NoSQL avoids:

- ❑ Overhead of ACID transactions
- ❑ Complexity of SQL query
- ❑ Burden of up-front schema design
- ❑ DBA presence
- ❑ Transactions (It should be handled at application layer)

Provides:

- ❑ Easy and frequent changes to DB
- ❑ Fast development
- ❑ Large data volumes(eg.Google)



NoSQL why, what and

When and when not to use it?

WHEN / WHY ?

- When traditional RDBMS model is too restrictive (flexible schema)
- When ACID support is not "really" needed
- Object-to-Relational (O/R) impedance
- Because RDBMS is neither distributed nor scalable by nature
- Logging data from distributed sources
- Storing Events / temporal data
- Temporary Data (Shopping Carts / Wish lists / Session Data)
- Data which requires flexible schema
- **Polyglot Persistence** i.e. best data store depending on nature of data.

WHEN NOT ?

- Financial Data
- Data requiring strict ACID compliance
- Business Critical Data

NoSQL is getting more & more

Google

ebay™

LinkedIn™

YAHOO!®

NETFLIX

amazon

theguardian



facebook®

What is a schema-less

In relational Databases:

- ❑ You can't add a record which does not fit the schema
- ❑ You need to add NULLs to unused items in a row
- ❑ We should consider the datatypes.
i.e : you can't add a string to an integer field
- ❑ You can't add multiple items in a field (You should create another table: primary-key, foreign key, joins, normalization, ... !!!)

```
create table customers (id int, firstname text, lastname text)  
insert into customers (firstname, middlename, lastname) values (...
```

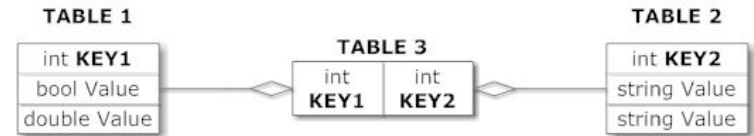


What is a schema-less

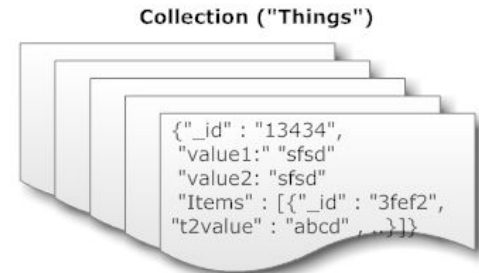
In NoSQL Databases:

- There is no schema to consider
- There is no unused cell
- There is no datatype (implicit)
- Most of considerations are done in application layer
- We gather all items in an aggregate (document)

Relational Model



Document Model

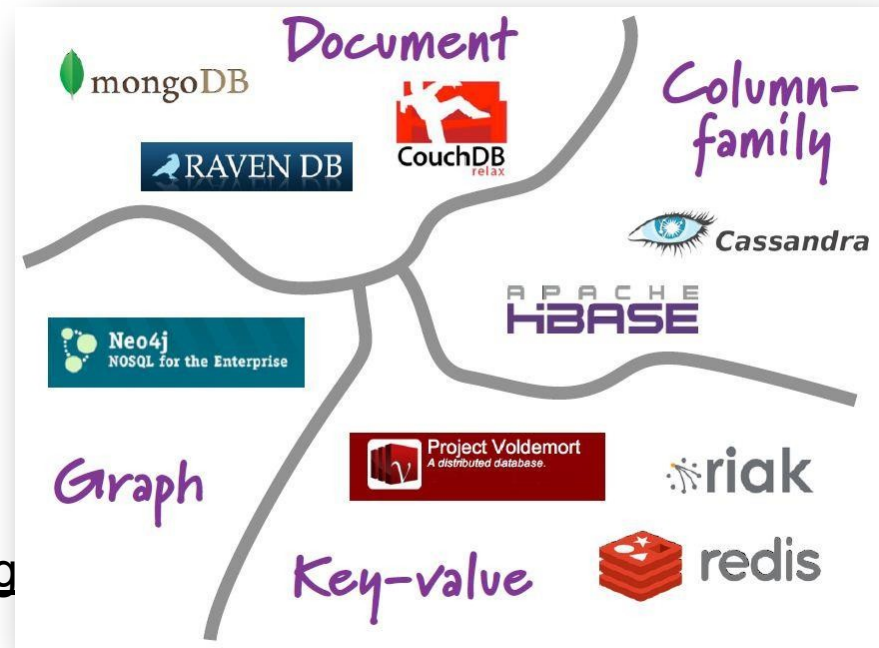


Aggregate Data

NoSQL databases are classified in four major datamodels:

- Key-value
- Document
- Column family
- Graph

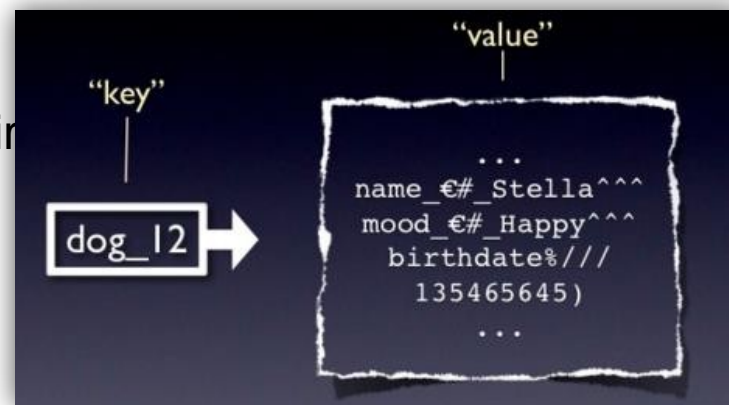
Each DB has its own query language



Key-value data

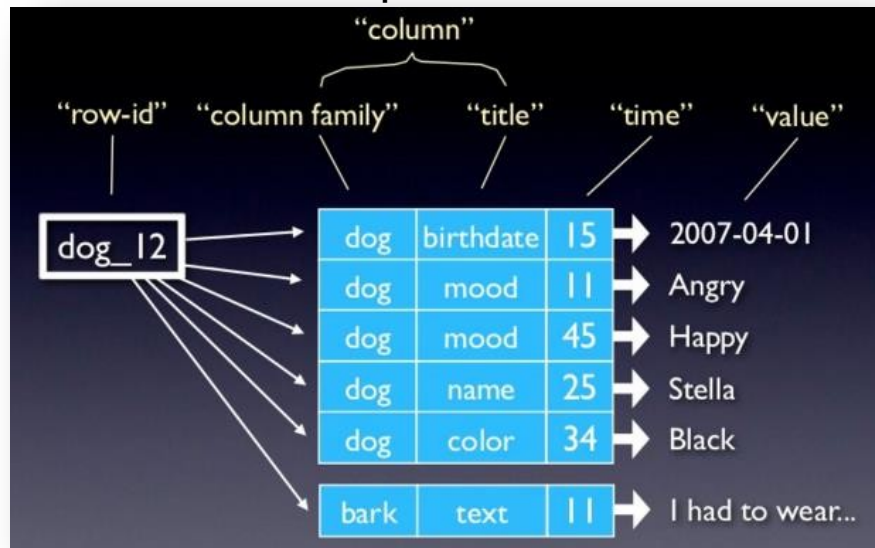
- Simplest NOSQL databases
- The main idea is the use of a hash table
- Access data (values) by strings called keys
- Data has no required format
data may have any format
- Data model: (key, value) pair
- Basic Operations:
Insert(key, value),
Fetch(key),
Update(key),
Delete(key)

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year: 2005 Transmission: Auto



Column family data

- The column is lowest/smallest instance of data.
- It is a tuple that contains a name, a value and a timestamp



ColumnFamily: Authors		
Key	Value	
"Eric Long"	Columns	
	Name	Value
	"email"	"eric (at) long.com"
	"country"	"United Kingdom"
	"registeredSince"	"01/01/2002"
"John Steward"	Columns	
	Name	Value
	"email"	"john.steward (at) somedomain.com"
	"country"	"Australia"
	"registeredSince"	"01/01/2009"
"Ronald Mathies"	Columns	
	Name	Value
	"email"	"ronald (at) sodeso.nl"
	"country"	"Netherlands, The"
	"registeredSince"	"01/01/2010"

Column family data

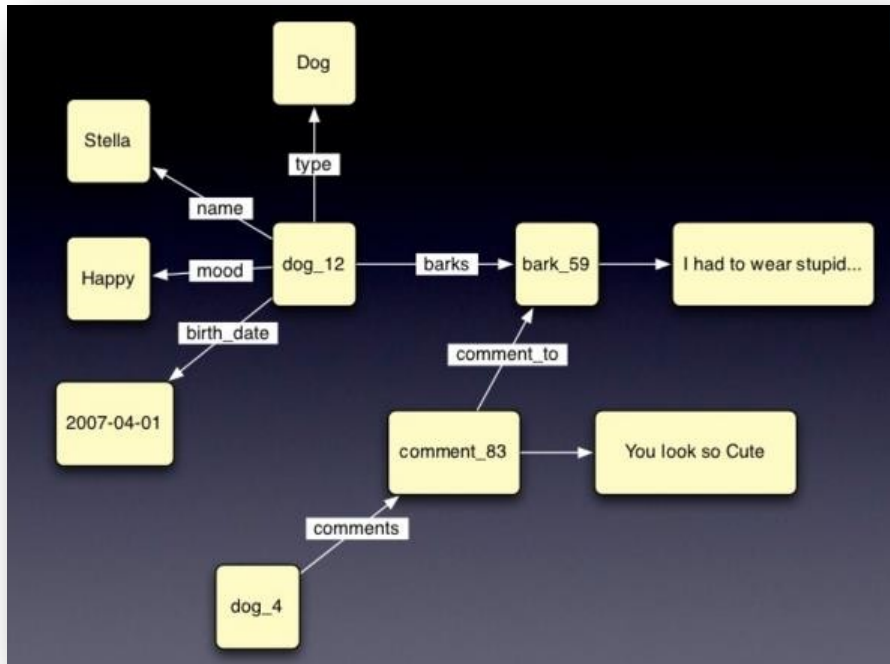
Some statistics about Facebook Search (using Cassandra)

- ❖ MySQL > 50 GB Data
 - Writes Average : ~300 ms
 - Reads Average : ~350 ms
- ❖ Rewritten with Cassandra > 50 GB Data
 - Writes Average : 0.12 ms
 - Reads Average : 15 ms



Graph data

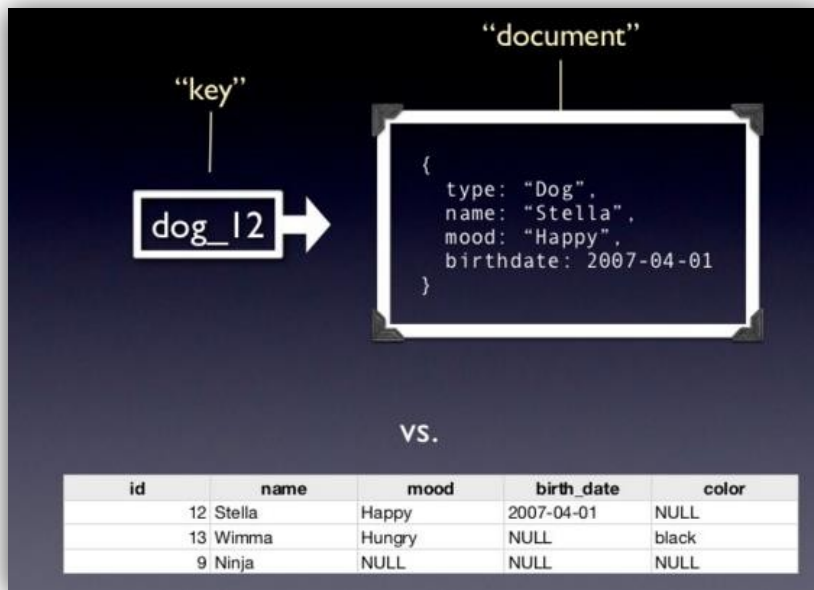
- Based on Graph Theory.
- Scale vertically, no clustering.
- You can use graph algorithms easily
- Transactions
- ACID



Document based data

- Pair each key with complex data structure.
- Indexes are done via B-Trees.
- Documents can contain many different key-value pairs, or key-array pairs, or even nested documents.

```
{  
  person: {  
    first_name: "Peter",  
    last_name: "Peterson",  
    addresses: [  
      {street: "123 Peter St"},  
      {street: "504 Not Peter St"}  
    ],  
  },  
}
```



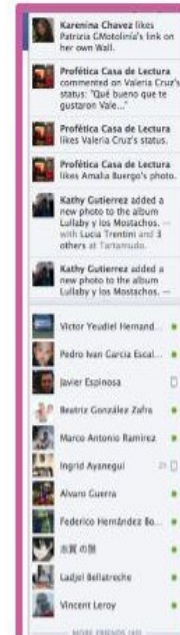
Document based data

```
SELECT name, pic, profile_url
FROM user
WHERE uid = me()
```



```
SELECT message, attachment
FROM stream
WHERE source_id = me() AND type = 80
```

```
SELECT name
FROM friendlist
WHERE owner = me()
```



```
SELECT name
FROM group
WHERE gid IN ( SELECT gid
                FROM group_member
                WHERE uid = me() )
```



```
SELECT name, pic
FROM user
WHERE online_presence = "active"
AND uid IN ( SELECT uid2
              FROM friend
              WHERE uid1 = me() )
```

<https://developers.facebook.com/docs/reference/fql/>

SQL vs

Differences

	SQL Databases	No SQL Database
Example	Oracle , mysql	Mondo DB, CouchDB, Neo4J
Storage Model	Rows and tables	Key-value. Data stored as single document in JSON, XML
Schemas	Static	Dynamic
Scaling	Vertical & Horizontal	Horizontal
Transactions	Yes	Certain levels
Data Manipulation	Select, Insert , Update	Through Object Oriented API's

What we need

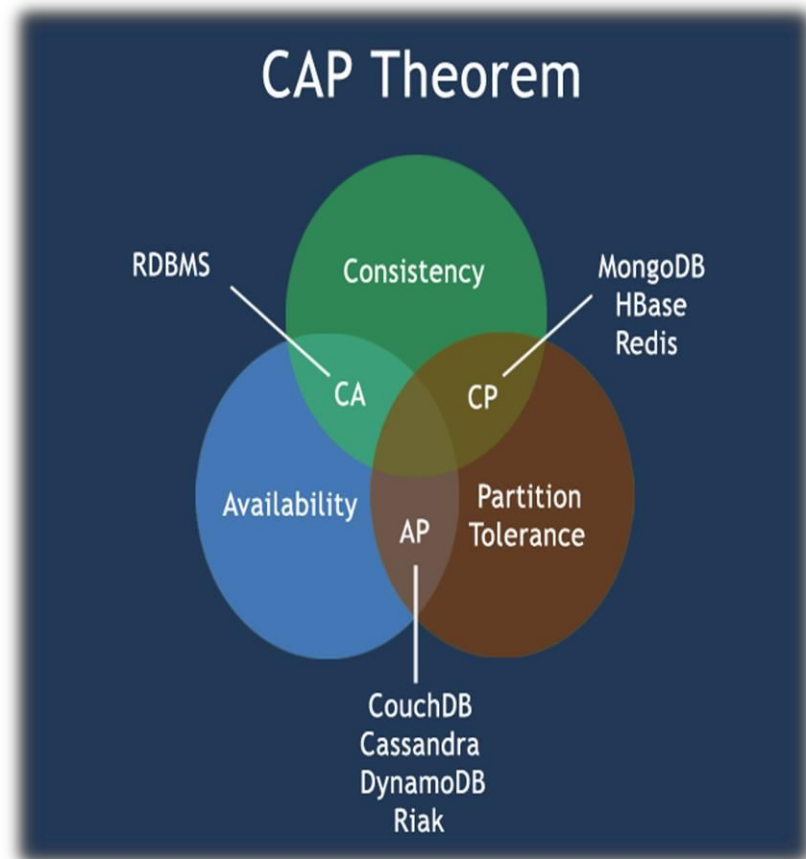
- We need a distributed database system having such features:
 - Fault tolerance
 - High availability
 - Consistency
 - Scalability

**Which is
impossible!!!
According to CAP
theorem**

CAP

The CAP Theorem

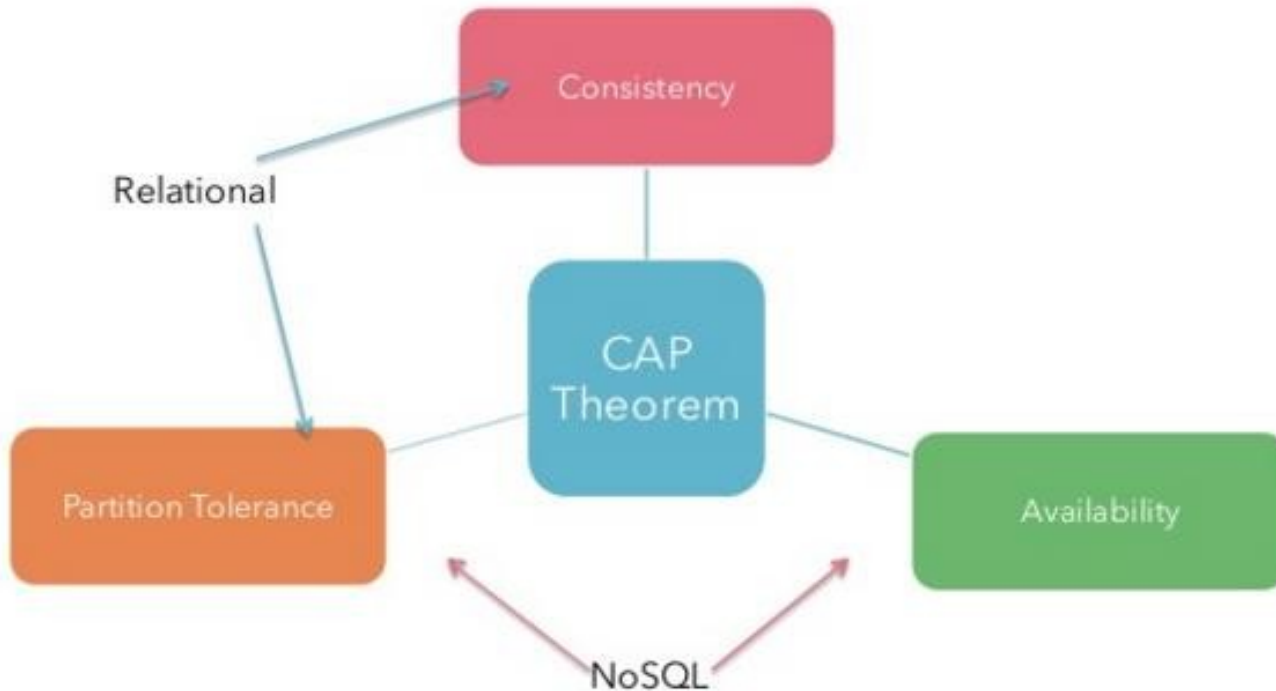
- Impossible for any shared data-system to guarantee simultaneously all of the following three properties:
 - Consistency – once data is written, all future read requests will contain that data
 - Availability – the database is always available and responsive
 - Partition Tolerance – if part of the database is unavailable, other parts are unaffected



We can not achieve all the three items In distributed database systems (center)

CAP

Traditional vs. NoSQL



Conclusion

In Conclusion!

- RDBMS is a great tool for solving ACID problems
 - When data validity is super important
 - When you need to support dynamic queries
- NoSQL is a great tool for solving data availability problems
 - When it's more important to have fast data than right data
 - When you need to scale based on changing requirements
- Pick the right tool for the job

Reference

- ❑ nosql-database.org/
- ❑ <https://www.mongodb.com/nosql-explained>
- ❑ www.couchbase.com/nosql-resources/what-is-no-sql
- ❑ <http://nosql-database.org/> "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open- source and horizontally scalable"
- ❑ NoSQL distilled, Martin Fowler