NoSQL

Contents

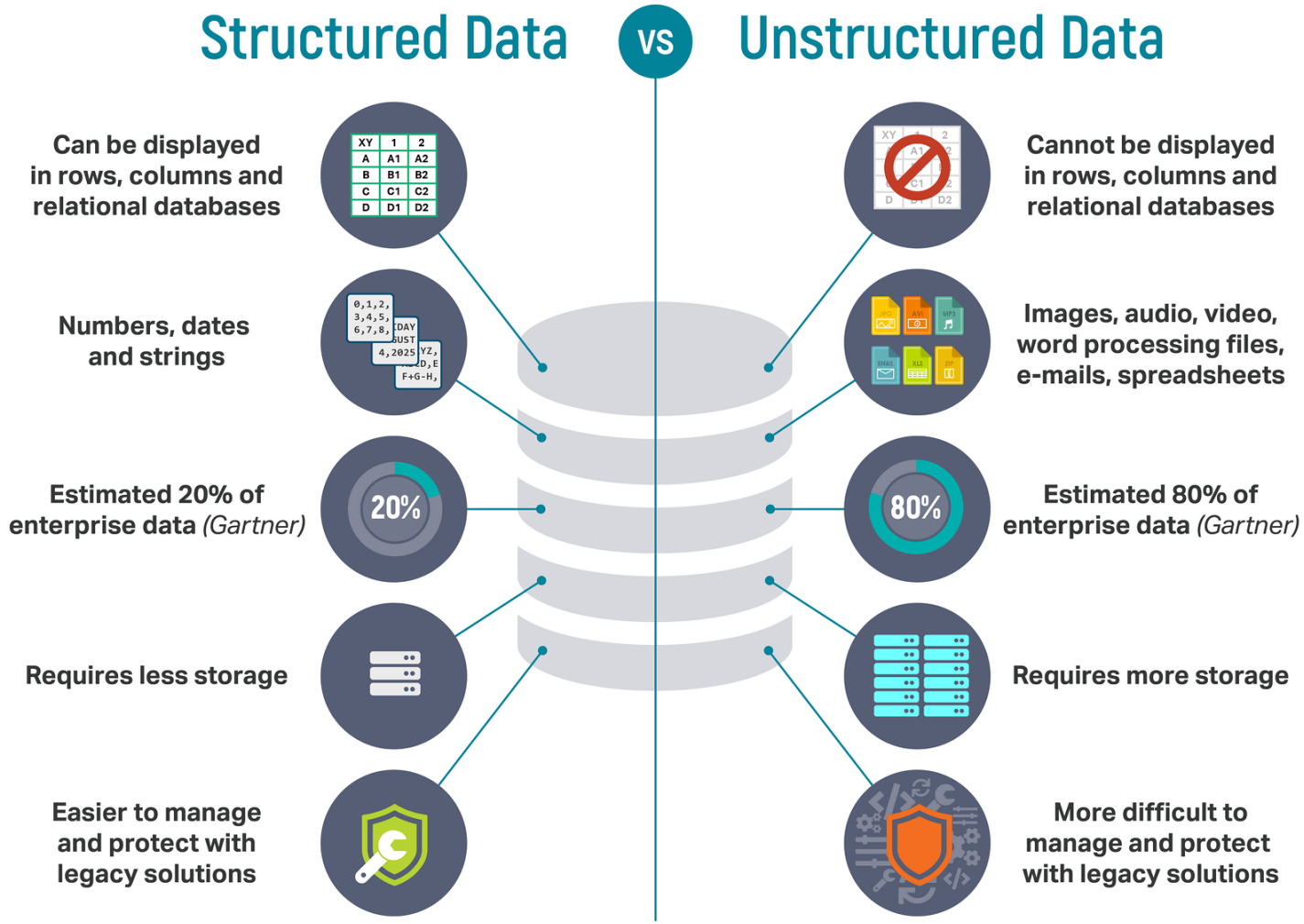
- **Big Data**
- **NoSQL**
- **CAP Theorem**
- **ACID vs BASE**
- **MongoDB**

Unit	Value	Size
bit (b)	0 or 1	1/8 of a byte
byte (B)	8 bits	1 byte
kilobyte (KB)	1000^1 bytes	1,000 bytes
megabyte (MB)	1000^2 bytes	1,000,000 bytes
gigabyte (GB)	1000^3 bytes	1,000,000.000 bytes
terabyte (TB)	1000^4 bytes	1,000,000,000,000 bytes
petabyte (PB)	1000^5 bytes	1,000,000,000,000,000 bytes
exabyte (EB)	1000^6 bytes	1,000,000,000,000,000,000 bytes
zettabyte (ZB)	1000^7 bytes	1,000,000,000,000,000,000,000 bytes
yottabyte (YB)	1000^8 bytes	1,000,000,000,000,000,000,000,000 bytes

2021 This Is What Happens In An Internet Minute

Created By:
@LoriLewis
@OfficiallyChadd

Structured Data vs Unstructured Data



JSON Example

```
{
    "custkey": "450002",
    "useragent": {
        "devicetype": "pc",
        "experience": "browser",
        "platform": "windows"
    },
    "pagetype": "home",
    "productline": "television",
    "customerprofile": {
        "age": 20,
        "gender": "male",
        "customerinterests": [
            "movies",
            "fashion",
            "music"
        ]
    }
}
```

snowflake

Characteristics of Big Data

- **Grows at a fast pace**
- **Diverse**
- **Not formally modeled**
- **Unstructured**
- **Heterogeneous**
- **Data is valuable**
- **Standard DBs and cannot capture diversity and heterogeneity**
- **Cannot achieve satisfactory performance**

3 V’s

- **Volume** = because of the large amount of data, storing data on a single machine is impossible.
- **Variety** = How can we deal with data coming from varied sources which have been formatted using different schemas?
- **Velocity** = How can we quickly store and process new data?

Volume: scale of data

- 90% of today's data has been created in just the last 2 years
- Every day we create around 900 petabytes of data or enough to fill 35 million Blu-ray discs
- According to **Statista**, by 2025, the amount of data created globally is projected to reach **181 zettabytes**
- Most companies in the US have over 100 terabytes (100,000 gigabytes) of data stored

Variety: different forms of data

Velocity - Analysis of data

- **Big Data Velocity deals with the pace at which data flows in from sources like business processes, machines, networks and human interaction with things like social media sites, mobile devices, etc.**

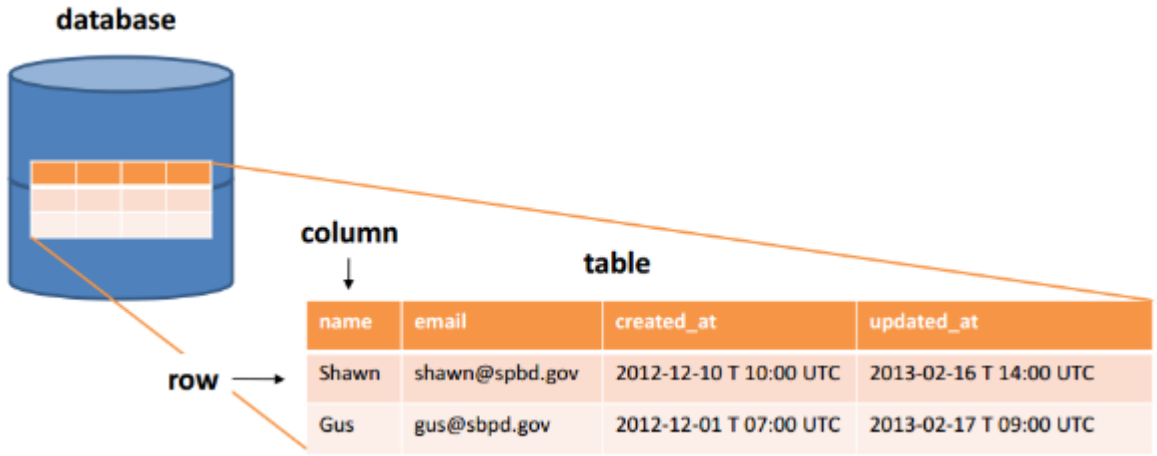
- **The flow of data is massive and continuous.**

- **This real-time data can help researchers and businesses make valuable decisions that provide strategic competitive advantages**

Velocity: analysis of streaming data

Relational Databases

- Tables
- SQL (Structured Query Language)
- MySQL, Sybase, Oracle



MySQL

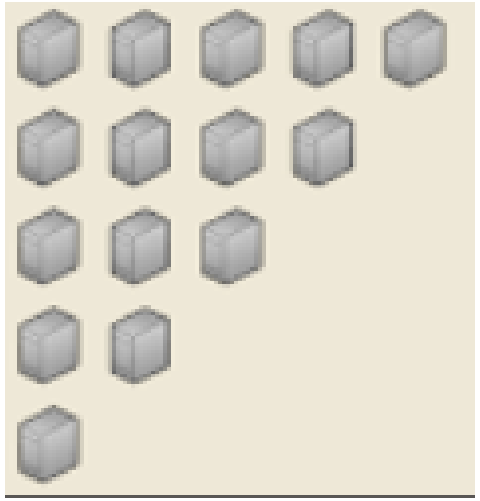
ORACLE

SYBASE
An SAP Company

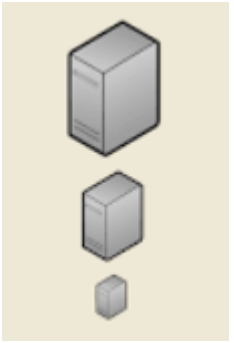
Newer databases for large data sets

- **NoSQL databases are increasingly used in big data and real-time web applications.**
- **Data is modelled in means other than the tabular relations used in relational databases**
- **Horizontal rather than vertical scaling (see next slide)**

Horizontal Scaling



Vertical Scaling



Horizontal scaling means that you scale by adding more machines into your pool of resources where Vertical scaling means that you scale by adding more power (CPU, RAM) to your existing machine.

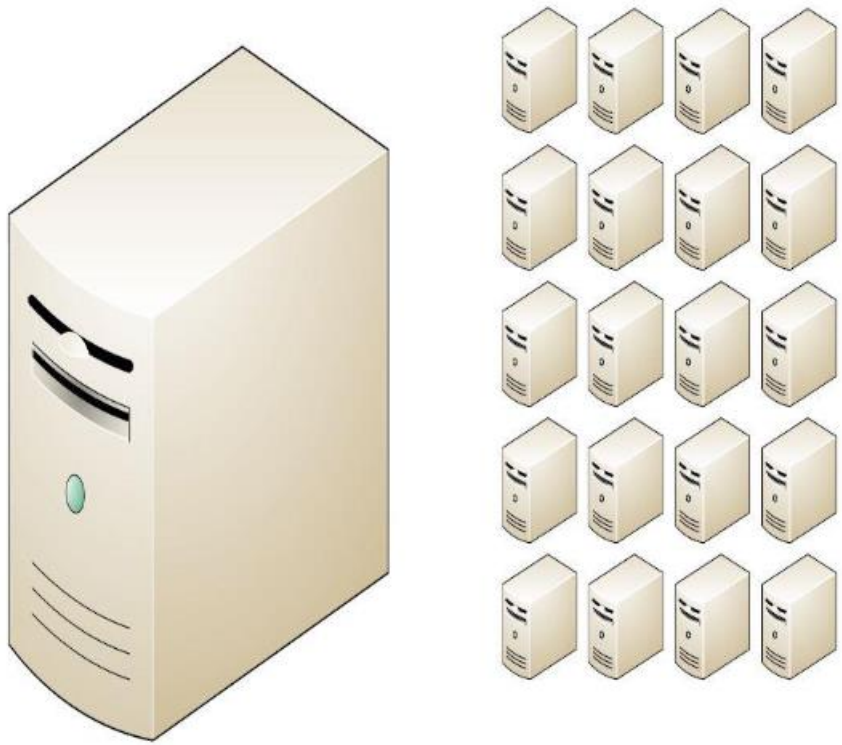
But...

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

Because...

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

Era of Distributed Computing



But...

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

Because...

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

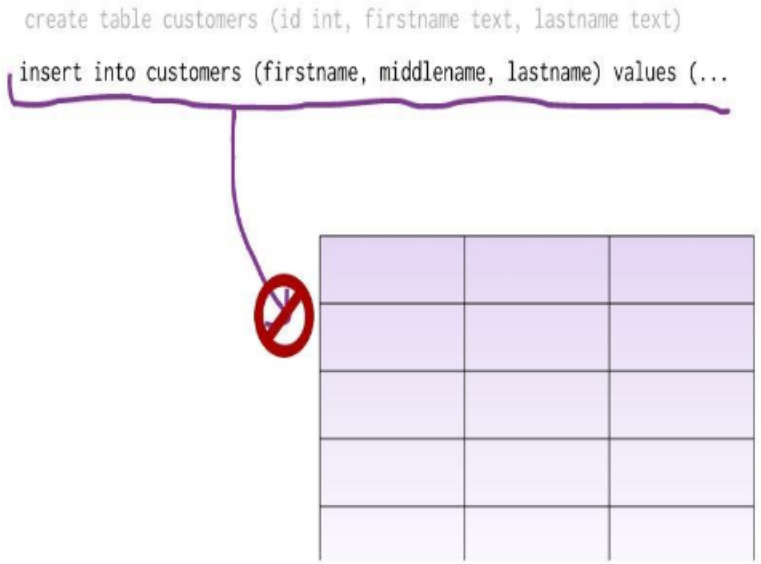
And....

It's weak in:

- Speed (performance)
- High availability
- Partition tolerance

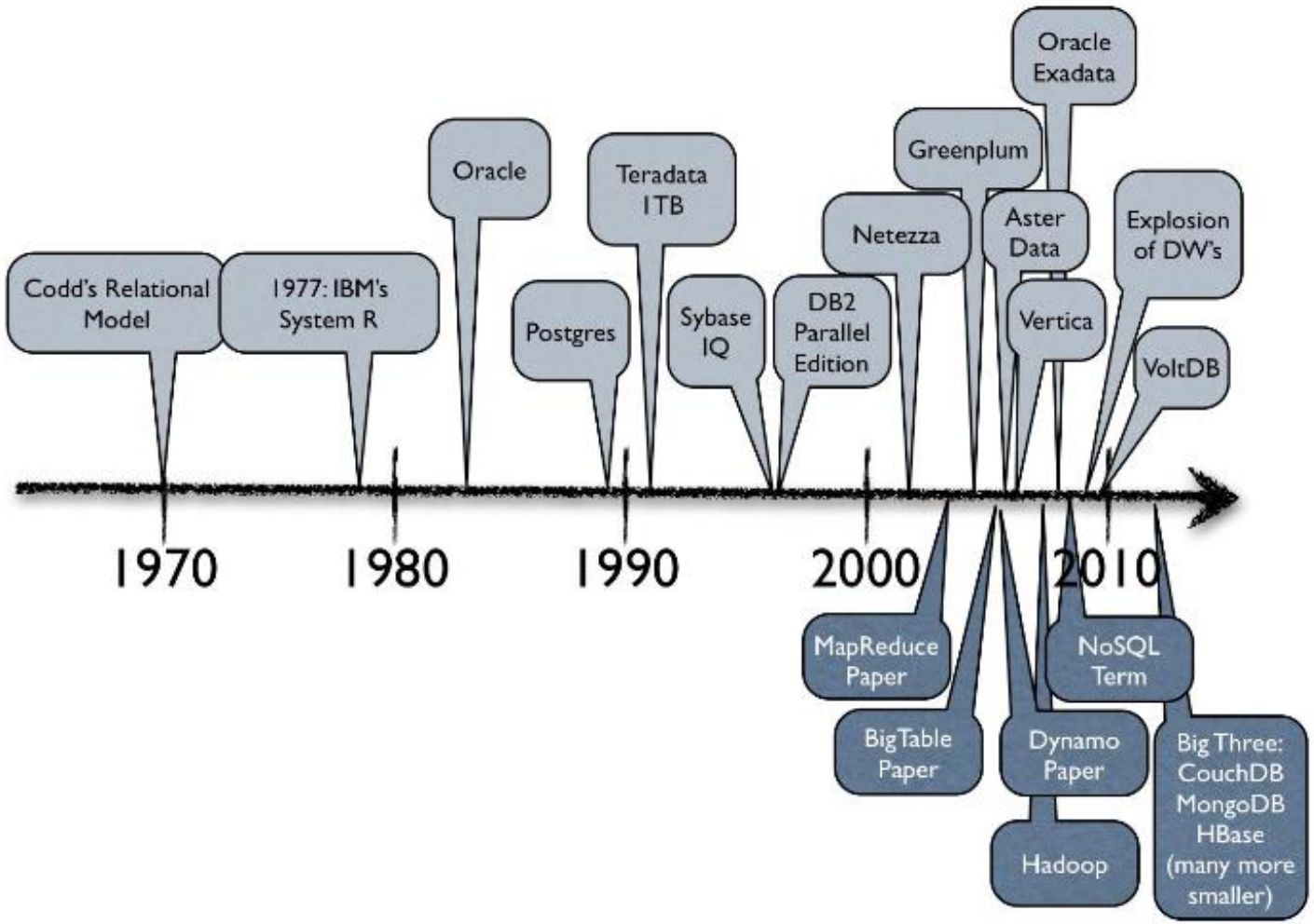
In relational Databases:

- 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 stirng to an interger field
- You can’t add multiple items in a field (You should create another table: primary-key, foreign key, joins, normalization, ... !!!)



What is NoSQL?

- **Stands for Not Only SQL**
- **Class of non-relational data storage systems**
- **Usually do not require a fixed table schema nor do they use the concept of joins**
- **All NoSQL offerings relax one or more of the ACID properties (will talk about the CAP theorem)**


Codd's Relational Model
1977: IBM's System R
Oracle
Postgres
Teradata 1TB
Sybase IQ
DB2 Parallel Edition
Netezza
Greenplum
Oracle Exadata
Aster Data
Vertica
Explosion of DW's
VoltDB
1970
1980
1990
2000
2010
MapReduce Paper
BigTable Paper
Dynamo Paper
Hadoop
NoSQL Term
Big Three: CouchDB MongoDB HBase (many more smaller)

Why NoSQL?

- **For data storage, an RDBMS cannot be the be-all/end-all**

- **Just as there are different programming languages, need to have other data storage tools in the toolbox**

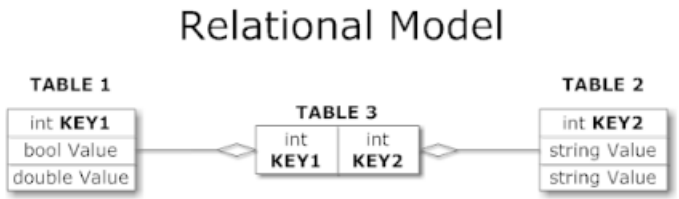
NoSQL avoids:

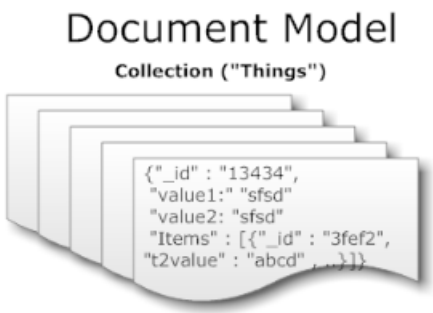
- Overhead of ACID transactions
- Complexity of SQL query
- Burden of up-front schema design
- DBA presence
- Transactions (It should be handled application layer)

Provides:

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

In NoSQL Databases:

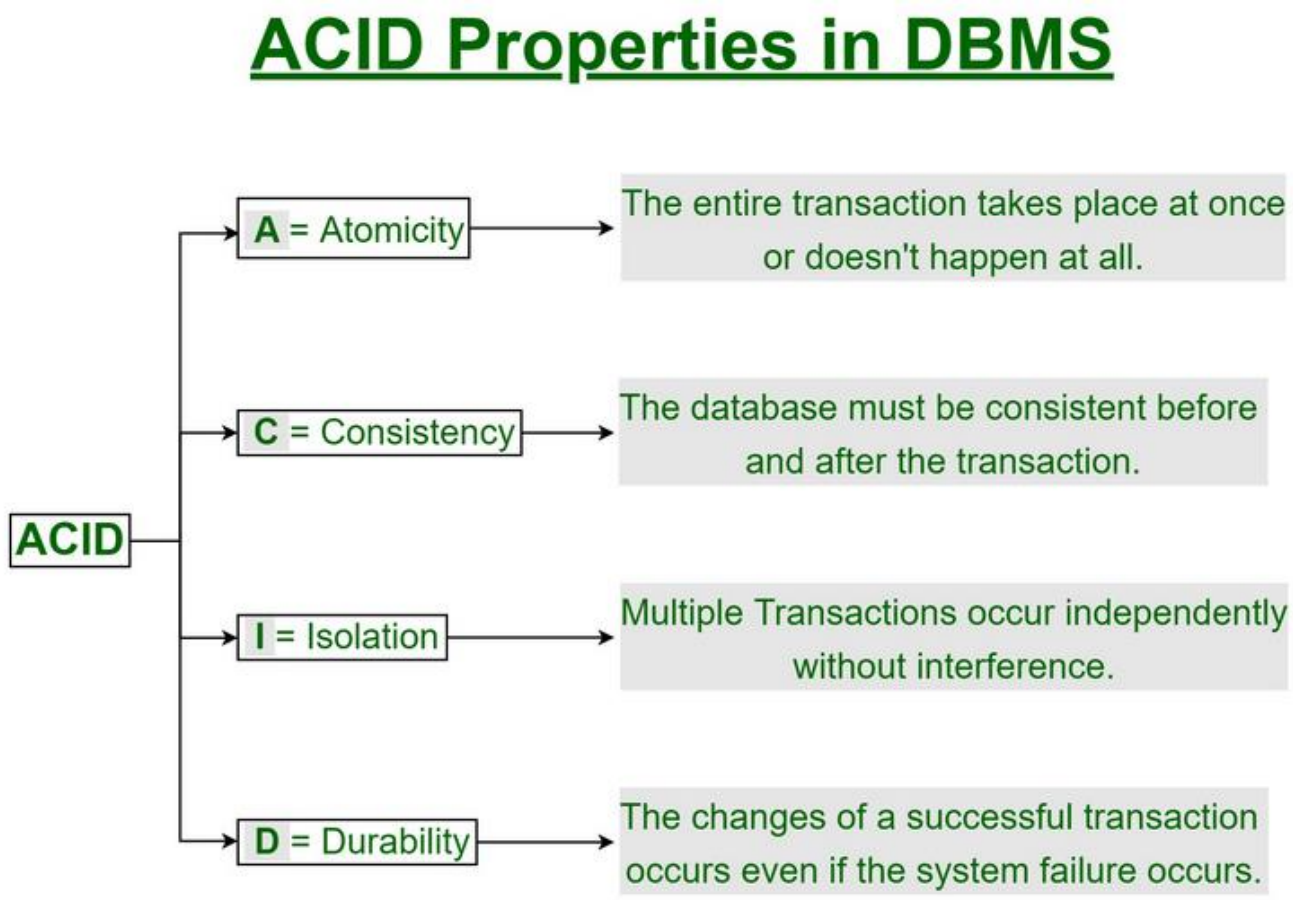




- There is no schema to consider
- There is no unused cell
- There is no datatype (implicit)
- Most of considerations are done in application layer
- We gather all items in an aggregate (document)

ACID

A transaction is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.
In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called ACID properties.



CAP Theorem

- Database systems traditionally support ACID requirements:
  - Atomicity, Consistency, Isolation, Durability

- In distributed web applications the focus shifts to:
  - Consistency, Availability, Partition tolerance

- CAP theorem - This term was coined by Eric Brewer.
- At most two of the above can be enforced at any given time.

- To scale out, you have to partition. That leaves either consistency or availability to choose from
  - In almost all cases, you would choose availability over consistency

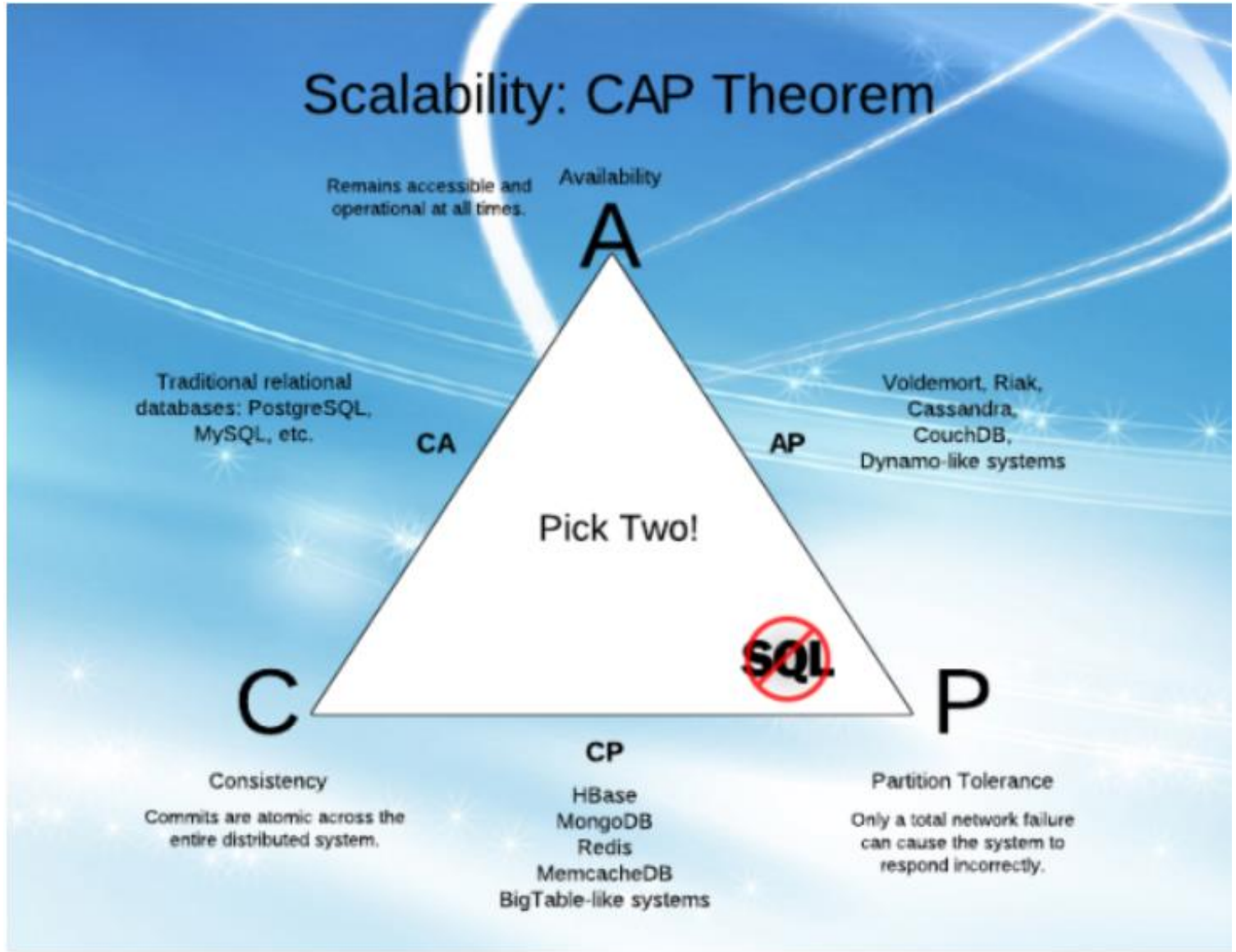
The CAP Theorem

- The limitations of distributed databases can be described in the so called the CAP theorem
  - **C**onsistency: every node always sees the same data at any given instance (i.e., strict consistency)
  - **A**vailability: the system continues to operate, even if nodes in a cluster crash, or some hardware or software parts are down due to upgrades
  - **P**artition Tolerance: To scale out, you have to partition.
  - The system continues to operate in the presence of network partitions

CAP theorem: any distributed database with shared data, can have *at most two* of the three desirable properties, C, A or P

Consistency Model

- A consistency model determines rules for visibility and apparent order of updates.
- For example:
  - Row X is replicated on nodes A and B
  - Client 1 writes row X to node A
  - Some period of time t elapses.
  - Client 2 reads row X from node M
  - Does client 2 see the write from client 1?
  - Consistency is a continuum with tradeoffs
  - For NoSQL, the answer would be: maybe
  - CAP Theorem states: Strict Consistency can't be achieved at the same time as availability and partition-tolerance.


Scalability: CAP Theorem
Availability
A
Remains accessible and operational at all times.
Traditional relational databases: PostgreSQL, MySQL, etc.
CA
AP
Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems
Pick Two!
SQL
C
P
CP
Consistency
Commits are atomic across the entire distributed system.
HBase
MongoDB
Redis
MemcacheDB
BigTable-like systems
Partition Tolerance
Only a total network failure can cause the system to respond incorrectly.

- **CA** - data is consistent between all nodes - as long as all nodes are online - and you can read/write from any node and be sure that the data is the same, but if you ever develop a partition between nodes, the data will be out of sync (and won't re-sync once the partition is resolved).

- **CP** - data is consistent between all nodes, and maintains partition tolerance

- **AP** - nodes remain online even if they can't communicate with each other and will resync data once the partition is resolved, but you aren't guaranteed that all nodes will have the same data (either during or after the partition)

Large-Scale Databases

- When companies such as Google and Amazon were designing large-scale databases, 24/7 Availability was a key
  - A few minutes of downtime means lost revenue
- When *horizontally* scaling databases to 1000s of machines, the likelihood of a node or a network failure increases tremendously
- Therefore, in order to have strong guarantees on Availability and Partition Tolerance, they had to sacrifice “strict” Consistency (*implied by the CAP theorem*)

The BASE Properties

- The CAP theorem proves that it is impossible to guarantee strict Consistency and Availability while being able to tolerate network partitions

- This resulted in databases with relaxed ACID guarantees

- In particular, such databases apply the BASE properties:
  - **B**asically **A**vailable: the system guarantees Availability
  - **S**oft-State: the state of the system may change over time
  - **E**ventual Consistency: the system will *eventually* become consistent

Eventual Consistency

- A database is termed as *Eventually Consistent* if:
  - All replicas will *gradually* become consistent in the absence of updates
  - When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent

ACID vs. BASE

- **Distributed NoSQL systems are typically said to support some form of BASE:**
  - **Basic Availability**
  - **Soft state**
  - **Eventual consistency***

- ***"We'd really like everything to be structured, consistent and harmonious,…, but what we are faced with is a little bit of punk-style anarchy. And actually, whilst it might scare our grandmothers, it's OK..."***
- ***-Julian Browne***

- Everyone who builds big applications builds them on CAP and BASE: Google, Yahoo, Facebook, Amazon, eBay, etc

NoSQL databases are classified in four major datamodels:

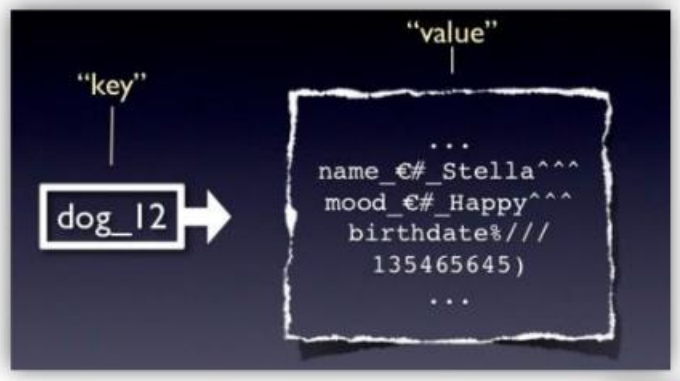
- Key-value
- Document
- Column family
- Graph

Each DB has its own query language

Key Value Model

- 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) pairs
- Basic Operations:
  Insert(key,value),
  Fetch(key),
  Update(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



Key-Value Stores

id	hair_color	age	height
1923	Red	18	6’0”
3371	Blue	34	NA
…	…	…	…

Table in relational db

```
user1923_color     Red
user1923_age       18
user3371_color     Blue
user4344_color     Brackish
user1923_height    6' 0"
user3371_age       34
```

Store/Domain in Key-Value db

Key/Value

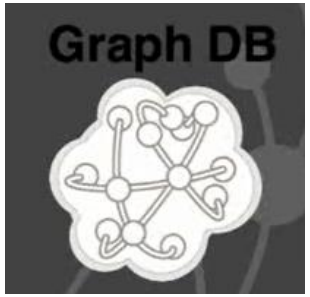
- **The basic data model:**
  - **Database is a collection of key/value pairs**
  - **The key for each pair is unique**

- **Primary operations:**
  - **insert(key,value)**
  - **delete(key)**
  - **update(key,value)**
  - **lookup(key)**

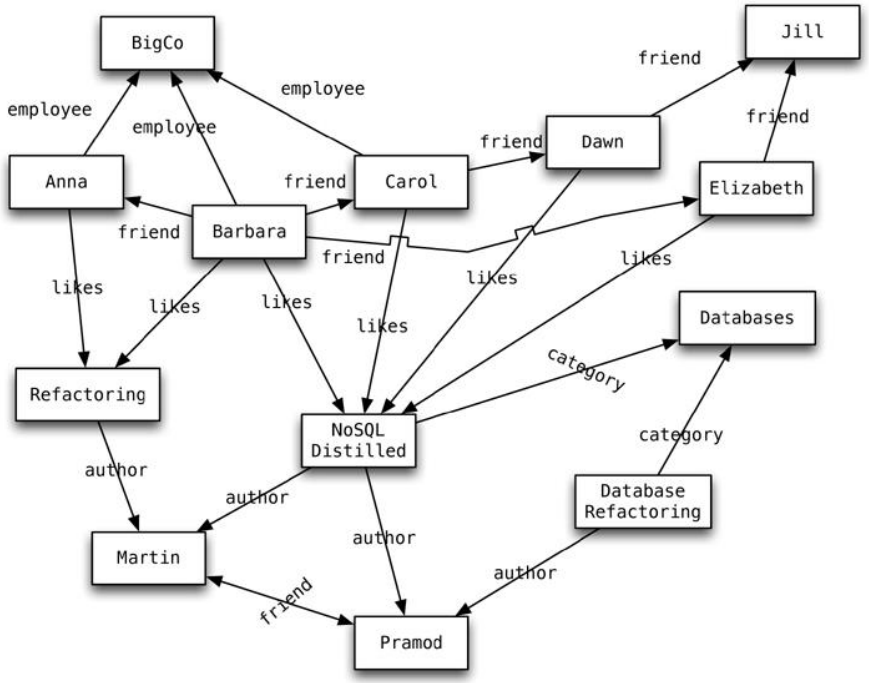
- **Additional operations:**
  - **variations on the above, e.g., reverse lookup**
  - **iterators**

DynamoDB
Azure Table Storage
Riak
Rdis
Aerospike
FoundationDB
LevelDB
Berkeley DB
Oracle NoSQL Database
GenieDb
BangDB
Chordless
Scalaris
Tokyo Cabinet/Tyrant
Scalien
Voldemort
Dynomite
KAI
MemcacheDB
Faircom C-Tree
LSM
KitaroDB
HamsterDB
STSdb
TarantoolBox
Maxtable
Quasardb
Pincaster
RaptorDB
TIBCO Active Spaces
Allegro-C
nessDB
HyperDex
SharedHashFile
Symas LMDB
Sophia
PickleDB
Mnesia
LightCloud
Hibari
OpenLDAP
Genomu
BinaryRage
Elliptics
Dbreeze
RocksDB
TreodeDB
(www.nosql-database.org
www.db-engines.com
www.wikipedia.com)

Graph Database



- **Inspired by Euler & graph theory**
  - **Key value pairs – nodes (equivalent to rows in rdbms)**
  - Types of relationships between nodes
  - Properties of the relationships (e.g. how do two people know each other , work, college, social etc.)
  - allow to find interesting patterns



BigTable clones / ColumnFamily

- **Based on Google's BigTable**
  - Distributed system for managing structured data that is designed to scale to a very large size.
  - Data is indexed by row key, column key, and a timestamp.

```
(row_key,column_key,time) → string
```

- Each cell in a Bigtable can contain multiple versions of the same data; These versions are indexed by a time stamp.

Some statistics about Facebook Search (using **Cassandra**)

- MySQL > 50 GB Data
  - Writes Average : ~300 ms
  - Reads Average : ~350 ms

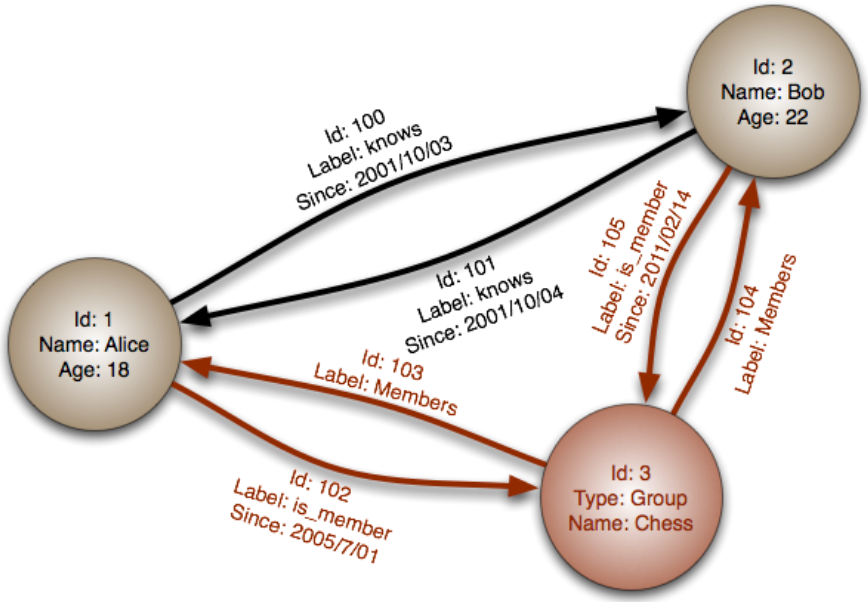
- Rewritten with Cassandra > 50 GB Data
  - Writes Average : 0.12 ms
  - Reads Average : 15 ms

Graph Store

•Neo4j - “The Neo Database – A Technology Introduction,” 2006.

•The basic data model:

- Directed graphs
- Nodes & edges, with properties, i.e., “labels”



AllegroGraph
ArangoDB
Bigdata
Bitsy
BrightstarDB
DEX/Sparksee
Execom IOG
Fallen *
Filament
FlockDB
GraphBase
Graphd
Horton
HyperGraphDB
IBM System G Native Store
InfiniteGraph
InfoGrid
jCoreDB Graph
MapGraph
Meronymy
Neo4j
Orly
OpenLink virtuoso
Oracle Spatial and Graph
Oracle NoSQL Datbase
OrientDB
OQGraph
Ontotext OWLIM
R2DF
ROIS
Sones GraphDB
SPARQLCity
Sqrrl Enterprise
Stardog
Teradata Aster
Titan
Trinity
TripleBit
VelocityGraph
VertexDB
WhiteDB
(www.nosql-database.org
www.db-engines.com
www.wikipedia.com)

Document Store

- MongoDB - "How a Database Can Make Your Organization Faster, Better, Leaner," February 2015.

The basic data model:

- The general notion of a document – words, phrases, sentences, paragraphs, sections, subsections, footnotes, etc.
- Flexible schema – subcomponent structure may be nested, and vary from document-to-document.
- Metadata – title, author, date, embedded tags, etc.
- Key/identifier.

One implementation detail:

- Formats vary greatly – PDF, XML, JSON, BSON, plain text, various binary, scanned image.

AmisaDB
ArangoDB
BaseX
Cassandra
Cloudant
Clusterpoint
Couchbase
CouchDB
Densodb
Djondb
EJDB
Elasticsearch
eXist
FleetDB
iBoxDB
Inquire
JasDB
MarkLogic
MongoDB
MUMPS
NeDB
NoSQL embedded db
OrientDB
RaptorDB
RavenDB
RethinkDB
SDB
SisoDB
Terrastore
ThruDB
(www.nosql-database.org
www.db-engines.com
www.wikipedia.com)

What am I giving up?

- ACID transactions
- SQL as a sometimes frustrating but still powerful query language
  - JOINS
  - Group By
  - Order By
- Easy integration with other applications that support SQL

NoSQL Databases – the End of Relational Databases?

- Relational databases are not going away
- Compelling arguments for most projects
  - familiarity, stability, feature set, and available support
- We should see relational databases as one option for data storage
  - **polyglot persistence** - using different data stores in different circumstances
- The problem they solve:
  - huge amounts of data are now handled in **real-time**
  - both data and use cases are getting more and more **dynamic**
  - social networks (relying on **graph data**) have gained impressive momentum

NoSQL - **Characteristics**

- NoSQL databases have a number of distinctive characteristics
- NoSQL does **not use SQL** as its query language.
- NoSQL database systems arose alongside major Internet companies, e.g. Google, Amazon, and Facebook, which had challenges in dealing with huge quantities of data which conventional RDBMS solutions could not cope with.
- NoSQL database systems are developed to manage large volumes of data that do not necessarily follow a fixed schema.
- Data is partitioned among different machines (for performance reasons and size limitations) so that traditional JOIN operations cannot be used.

NoSQL - **Characteristics**

- NoSQL cannot necessarily give full ACID guarantees.
  - Usually only eventual consistency is guaranteed.

  - This means that given a sufficiently long length of time over which no changes are sent, **all updates can be expected to propagate eventually through the system**.

- NoSQL has a distributed, fault-tolerant architecture.
  - Several NoSQL systems employ a distributed architecture, with the data held in a redundant manner on several servers.
  - In this way, the system can easily scale out by adding more servers, and failure of a server can be tolerated.

NoSQL Advantages

- Elastic scaling
- Big data
- Flexible Data Models
- Reduced Personnel Costs
- Reduced Equipment Costs

Elastic Scaling

- For years, database administrators have relied on **scale up** — buying bigger servers as database load increases — **rather than scale out** — distributing the database across multiple hosts as load increases.

- However, as transaction rates and availability requirements increase, and as databases move into the cloud or onto virtualized environments, the economic advantages of scaling out on commodity hardware become irresistible.

- RDBMS might not scale out easily on commodity clusters, but the new breed of NoSQL databases are designed to expand transparently to take advantage of new nodes, and they're usually designed with low-cost commodity hardware in mind.

Big Data

NoSQL Advantages

- Elastic scaling
- Big data
- Flexible Data Models
- Reduced Personnel Costs
- Reduced Equipment Costs

- Just as transaction rates have grown out of recognition over the last decade, the volumes of data that are being stored also have increased massively.
- O'Reilly has cleverly called this the "industrial revolution of data."
- RDBMS capacity has been growing to match these increases, but as with transaction rates, the constraints of data volumes that can be practically managed by a single RDBMS are becoming intolerable for some enterprises.
- Today, the volumes of "big data" that can be handled by NoSQL systems, such as Hadoop, outstrip what can be handled by the biggest RDBMS.

Flexible Data Models

NoSQL Advantages

- Elastic scaling
- Big data
- Flexible Data Models
- Reduced Personnel Costs
- Reduced Equipment Costs

- Change management is a big headache for large production RDBMS. Even minor changes to the data model of an RDBMS have to be carefully managed and may necessitate downtime or reduced service levels.

- NoSQL databases have far more relaxed — or even nonexistent — data model restrictions. NoSQL Key Value stores and document databases allow the application to store virtually any structure it wants in a data element. Even the more rigidly defined BigTable-based NoSQL databases (Cassandra, HBase) typically allow new columns to be created without too much fuss.

Reduced Personnel Costs

- Despite the many manageability improvements claimed by RDBMS vendors over the years, high-end RDBMS systems can be maintained only with the assistance of expensive, highly trained DBAs.
- DBAs are involved in the design, installation, and on-going tuning of high-end RDBMS systems.
- NoSQL databases are generally designed from the ground up to require less management: automatic repair, data distribution, and simpler data models lead to lower administration and tuning requirements — in theory.

Reduced Equipment Costs

NoSQL Advantages

- Elastic scaling
- Big data
- Flexible Data Models
- Reduced Personnel Costs
- Reduced Equipment Costs

- NoSQL databases typically use clusters of cheap commodity servers to manage the exploding data and transaction volumes, while RDBMS tends to rely on expensive proprietary servers and storage systems.

- The result is that the cost per gigabyte or transaction/second for NoSQL can be many times less than the cost for RDBMS, allowing you to store and process more data at a much lower price point.

NoSQL - Recap

- In computing, NoSQL is a broad class of database management systems identified by non-adherence to the widely used relational database management system model.

- NoSQL databases are **not** built primarily on tables, and **do not use SQL** for data manipulation.

- NoSQL database systems are often highly optimized for retrieval and appending operations and often offer little functionality beyond record storage (e.g. key–value stores).

NoSQL - Recap

- The reduced run-time flexibility compared to full SQL systems is compensated by gains in **scalability and performance** for certain data models.

- NoSQL database management systems are useful when working with a huge quantity of data when the data's nature does not require a relational model.

- The data can be structured, but NoSQL is used when what really matters is **the ability to store and retrieve great quantities of data, not the relationships between the elements.**