Big Data
(NoSQL Databases)

Instructor: Thanh Binh Nguyen

September 1st, 2019

S3Lab
Smart Software System Laboratory

“Big data is at the foundation of all the megatrends that are happening today, from social to mobile to cloud to gaming.”

– Chris Lynch, Vertica Systems

Background

- Relational databases mainstay of business
- Web-based applications caused spikes
- Explosion of social media sites (Facebook, Twitter) with large data needs
- rise of cloud-based solutions such as Amazon S3 (simple storage solution)
- Hooking RDBMS to web-based application becomes trouble

S3LAB.

Issues with Scaling up

- Best way to provide **ACID** and **Rich Query Model** is to have the dataset on a single machine
- Limits to scaling up (or vertical scaling: make a “single” machine more powerful) dataset is just too big!
- Scaling out (or horizontal scaling: adding more smaller / cheaper servers) is a better choice
- Different approaches for horizontal scaling (multi-node database):
  - Master / Slave
  - Sharding (partitioning)

S3LAB.

Scaling out RDBMS

Master / Slave

- All writes are written to the master
- All reads performed against the replicated slave databases
- Critical reads may be incorrect as writes may not have been propagated down
- Large datasets can pose problems as master needs to duplicate data to slaves

S3LAB.

Scaling out RDBMS

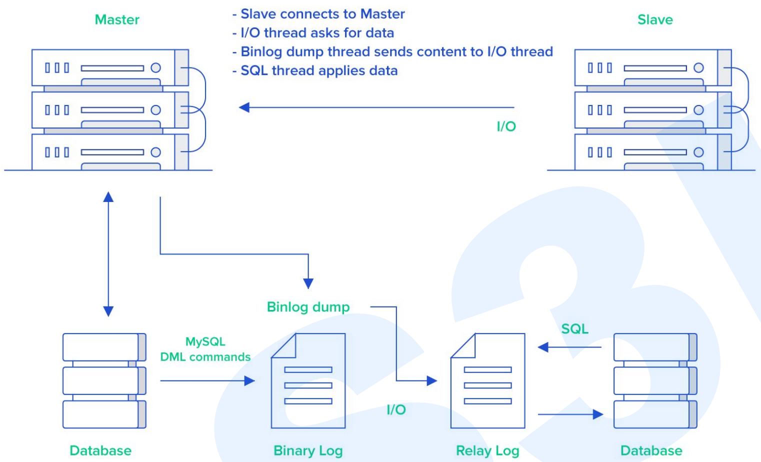
Sharding (Partitioning)

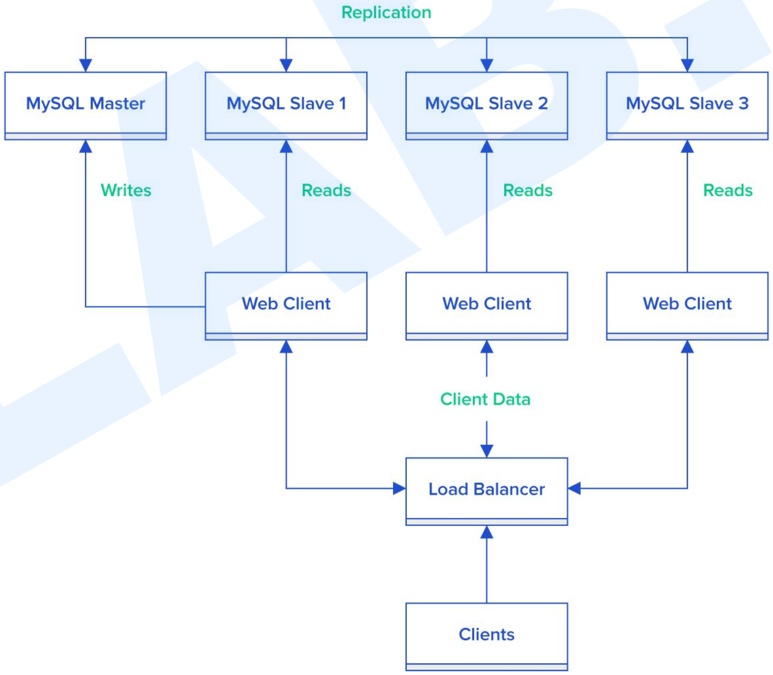
- Scales well for both reads and writes
- Not transparent, application needs to be partition-aware
- Can no longer have relationships/joins across partitions
- Loss of referential integrity across shards

S3LAB.

Scaling out RDBMS

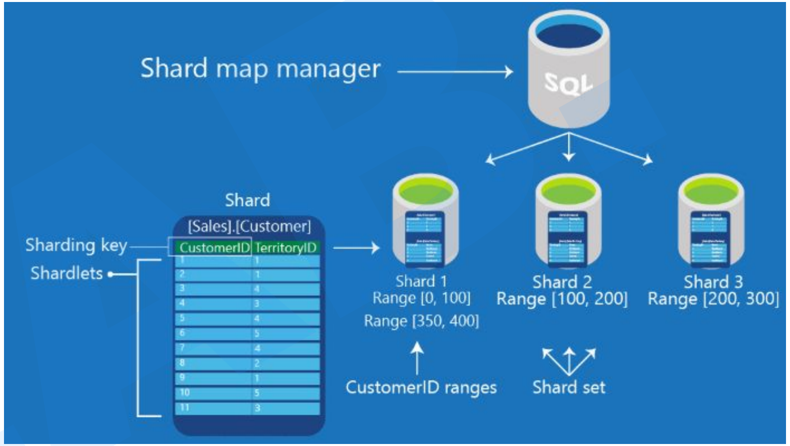
Master / Slave





Scaling out RDBMS

Sharding (Partitioning)



Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Vertical Partitions

VP1

CUSTOMER ID	FIRST NAME	LAST NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAĞCAN
4	JIM	PEPPER

VP2

CUSTOMER ID	FAVORITE COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

Horizontal Partitions

HP1

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

HP2

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Scaling out RDBMS

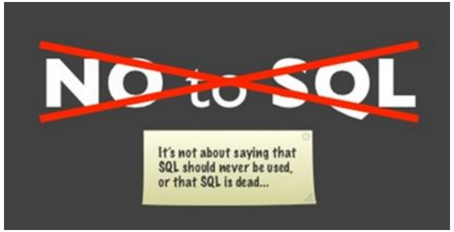
The other ways

- Multi-Master replication
- INSERT only, not UPDATES/DELETES
- No JOINs, thereby reducing query time
  - This involves de-normalizing data
- In-memory databases

S3LAB.

What is NoSQL?

- This name stands for **Not Only SQL**
- The term NOSQL was introduced by Carl Strozzi in 1998 to name his file-based database
- It was again re-introduced by Eric Evans when an event was organized to discuss open source distributed databases
    - Eric states that "... but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for. ..."



S3LAB.

What is NoSQL?

Key features (Advantages)

- non-relational
- don’t require schema
- data are replicated to multiple nodes (so, identical & fault-tolerant) and can be partitioned:
    - down nodes easily replaced
    - no single point of failure
- horizontal scalable

S3LAB.

What is NoSQL?

Key features (Advantages)

- cheap, easy to implement (open-source)
- massive write performance
- fast key-value access

S3LAB

What is NoSQL?

Disadvantages

- Don’t fully support relational features
  - no join, group by, order by operations (except within partitions)
  - no referential integrity constraints across partitions
- No declarative query language (e.g., SQL) more programming
- Relaxed **ACID** (see **CAP** theorem) fewer guarantees
- No easy integration with other applications that support SQL

S3LAB.

Who is using them?

3 major papers for NoSQL

- Three major papers were the "seeds" of the NOSQL movement:
  - BigTable (Google)
  - DynamoDB (Amazon)
    - Ring partition and replication
    - Gossip protocol (discovery and error detection)
    - Distributed key-value data stores
    - Eventual consistency
  - CAP Theorem

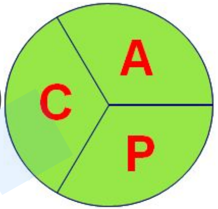
S3LAB.

The perfect storm

- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a “perfect storm”
- Not a backlash against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NOSQL offerings

S3LAB

CAP Theorem



- Suppose three properties of a **distributed system** (sharing data)
    - <u>C</u>**onsistency**:
        - Reads and writes are always executed atomically and are strictly consistent (linearizable). Put differently, all clients have the same view on the data at all times.
    - <u>A</u>**vailability**:
        - Every non-failing node in the system can always accept read and write requests by clients and will eventually return with a meaningful response, i.e. not with an error message.
    - <u>P</u>**artition-tolerance**:
        - system properties (consistency and/or availability) hold even when network failures prevent some machines from communicating with others. A system can continue to operate in the presence of a network partitions

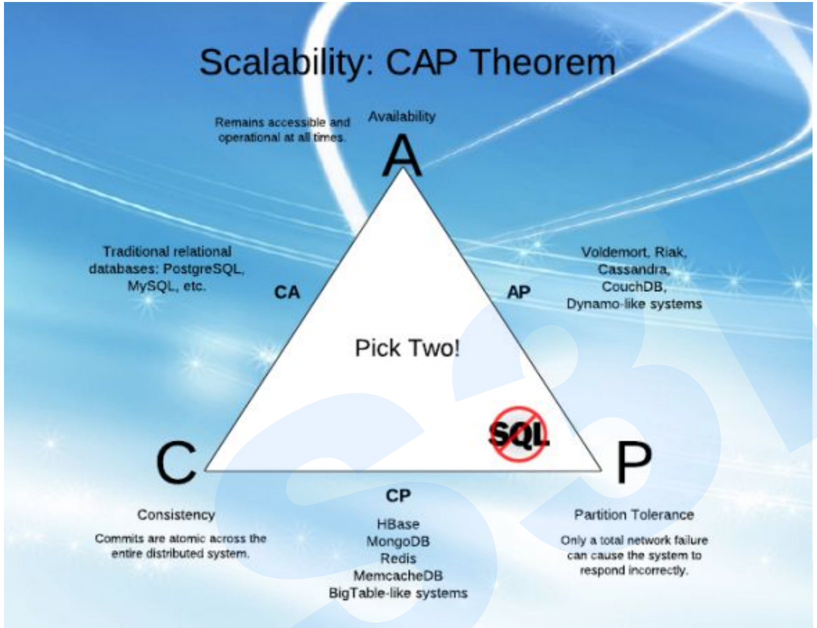
CAP Theorem

- Brewer's CAP Theorem:
    - For any system sharing data, it is "impossible" to guarantee simultaneously all of these three properties
    - You can have at most two of these three properties for any shared-data system
- Very large systems will "partition" at some point:
    - That leaves either **C** or **A** to choose from (traditional DBMS prefers **C** over **A** and **P**)
    - In almost all cases, you would choose **A** over **C** (except in specific applications such as order processing)

S3LAB.

CAP Theorem

Consistency



Data Model

- Key-Value
- Wide-Column
- Document
- Graph

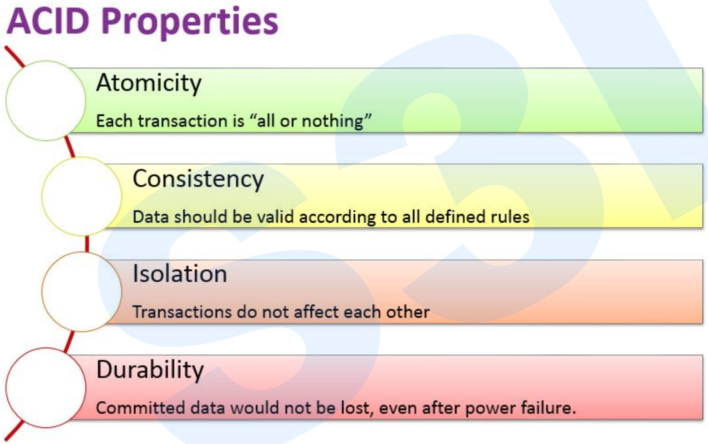
Consistency/Availability Trade-Off

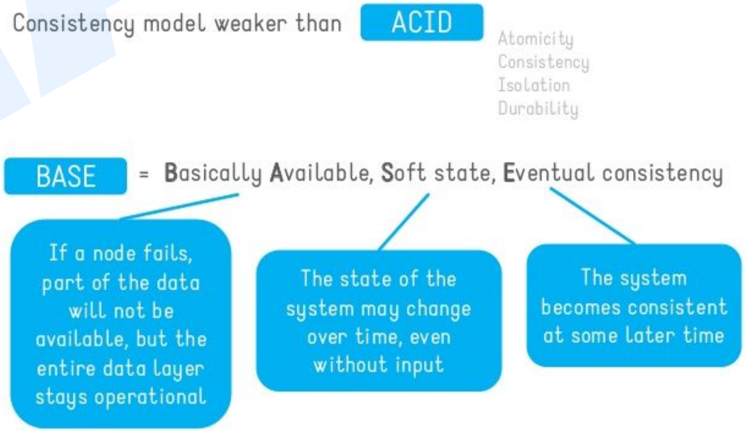
- **AP**: Available & Partition Tolerant
- **CP**: Consistent & Partition Tolerant
- **CA**: Not Partition Tolerant

CAP Theorem

Consistency

- Have 2 types of consistency:
  - Strong consistency – ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability)
  - Weak consistency – BASE (**B**asically **A**vailable **S**oft-state **E**ventual consistency)





CAP Theorem

Consistency

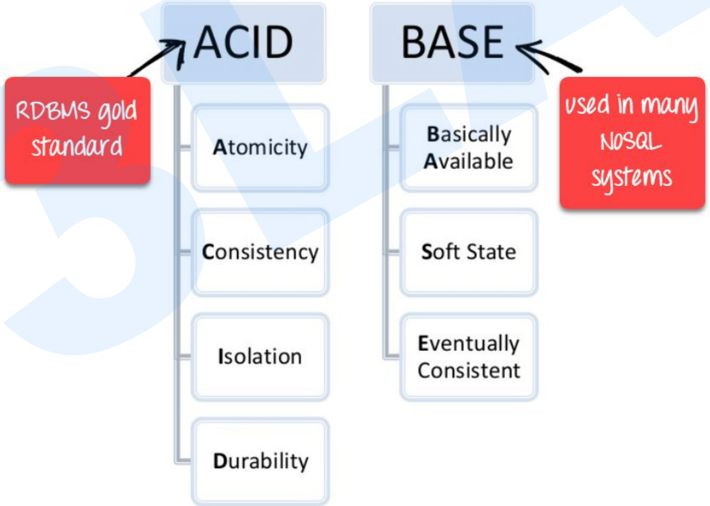
- A consistency model determines rules for visibility and apparent order of updates
- Example:
  - Row X is replicated on nodes M and N
  - Client A writes row X to node N
  - Some period of time t elapses
  - Client B reads row X from node M
  - **Does client B see the write from client A?**
  - Consistency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

S3LAB.

CAP Theorem

Consistency

- Cloud computing
  - ACID is hard to achieve, moreover, it is not always required, e.g. for blogs, status updates, product listings, etc.



NoSQL

- “No-schema” is a common characteristics of most NOSQL storage systems
- Provide “flexible” data types
- Other or additional query languages than SQL
- Distributed – horizontal scaling
- Less structured data
- Supports big data

S3LAB

NoSQL Categories

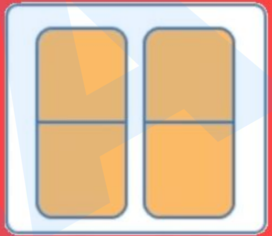
Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based

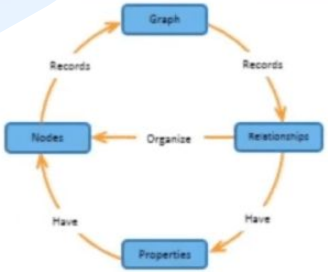
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



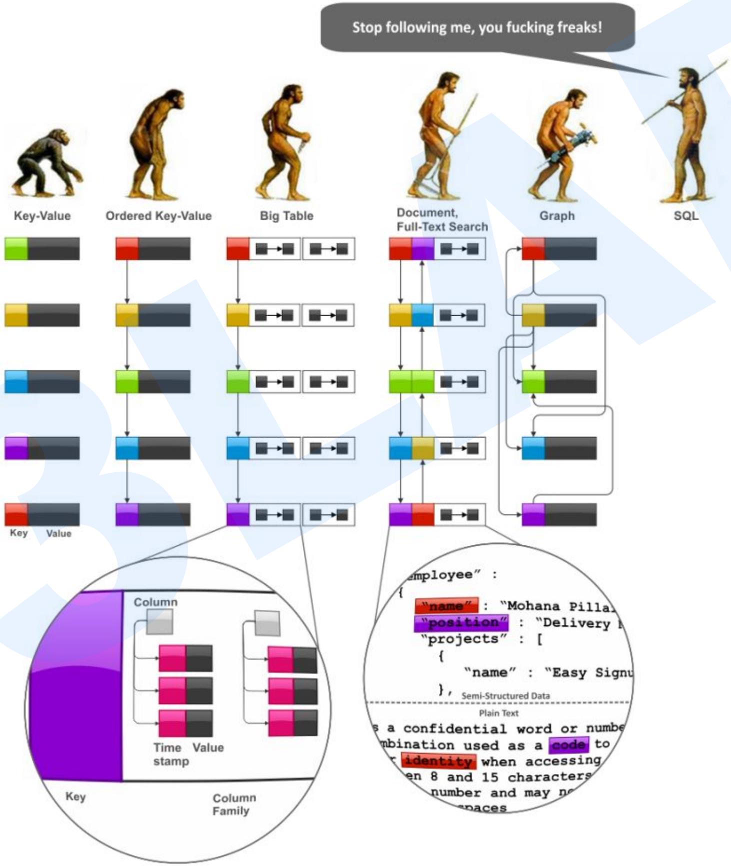
Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

NoSQL Categories



NoSQL Categories

Key-value

- Focus on scaling to huge amounts of data
- Designed to handle massive load
- Based on Amazon’s dynamo paper
- Data model: (global) collection of Key-value pairs
- Dynamo ring partitioning and replication
- Example: (DynamoDB)
    - items having one or more attributes (name, value)
    - An attribute can be single-valued or multivalued like set.
    - items are combined into a table

S3LAB

NoSQL Categories

Key-value

- Basic API access:
  - get(key): extract the value given a key
  - put(key, value): create or update the value given its key
  - delete(key): remove the key and its associated value
  - execute(key, operation, parameters): invoke an operation to the value (given its key) which is a special data structure (e.g. List, Set, Map .... etc)

S3LAB.

NoSQL Categories

Key-value

- Pros:
  - very fast
  - very scalable (horizontally distributed to nodes based on key)
  - simple data model
  - eventual consistency
  - fault-tolerance
- Cons:
  - Can’t model more complex data structure such as objects

S3LAB.

NoSQL Categories

Key-value

Name	Producer	Data model	Querying
SimpleDB	Amazon	set of couples (key, {attribute}), where attribute is a couple (name, value)	restricted SQL; select, delete, GetAttributes, and PutAttributes operations
Redis	Salvatore Sanfilippo	set of couples (key, value), where value is simple typed value, list, ordered (according to ranking) or unordered set, hash value	primitive operations for each value type
Dynamo	Amazon	like SimpleDB	simple get operation and put in a context
Voldemort	LinkedIn	like SimpleDB	similar to Dynamo

S3LAB.

NoSQL Categories

Key-value

employee_id	first_name	last_name	address
1	John	Doe	New York
2	Benjamin	Button	Chicago
3	Mycroft	Holmes	London

```
employee:$employee_id:$attribute_name = $value

employee:1:first_name = "John"
employee:1:last_name = "Doe"
employee:1:address = "New York"

employee:2:first_name = "Benjamin"
employee:2:last_name = "Button"
employee:2:address = "Chicago"

employee:3:first_name = "Mycroft"
employee:3:last_name = "Holmes"
employee:3:address = "London"
```

S3LAB.

NoSQL Categories

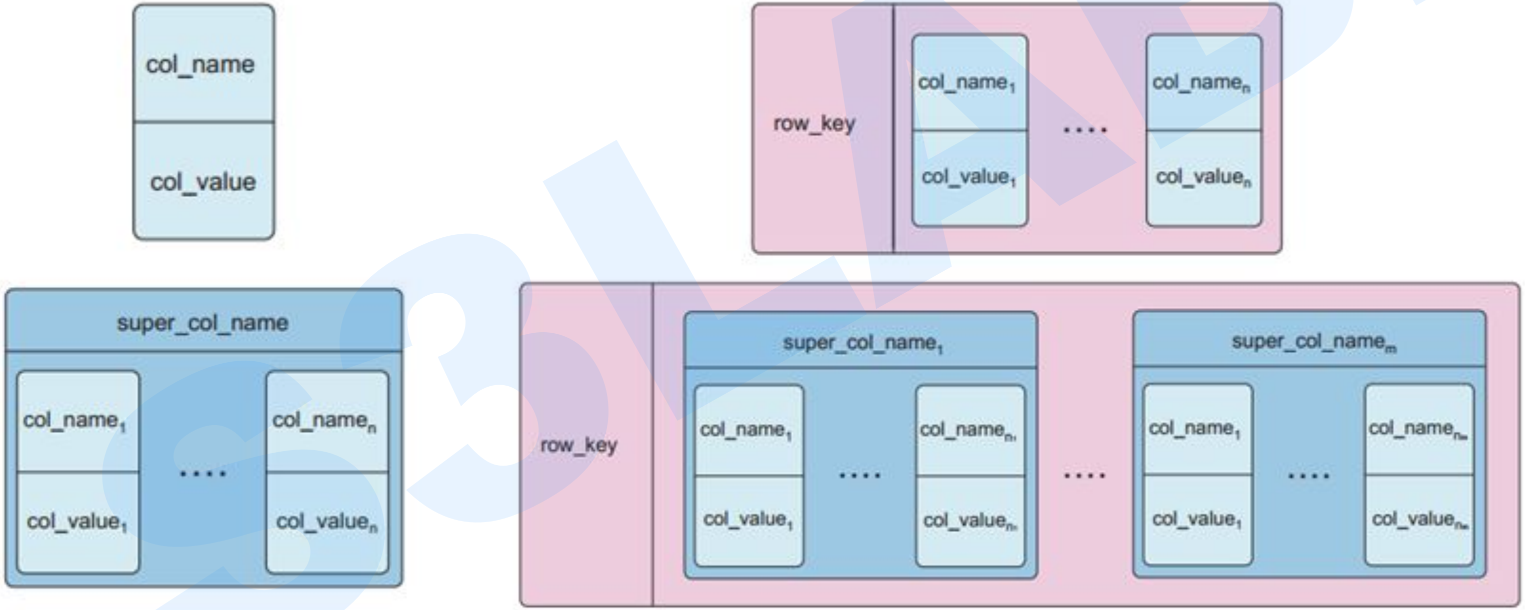
Column-based

- Based on Google's BigTable paper
- Like column oriented relational databases (store data in column order) but with a twist
- Tables similarly to RDBMS, but handle semi-structured
- Data model:
  - Collection of Column Families
  - Column family = (key, value) where value = set of related columns (standard, super)
  - indexed by row key, column key and timestamp

S3LAB.

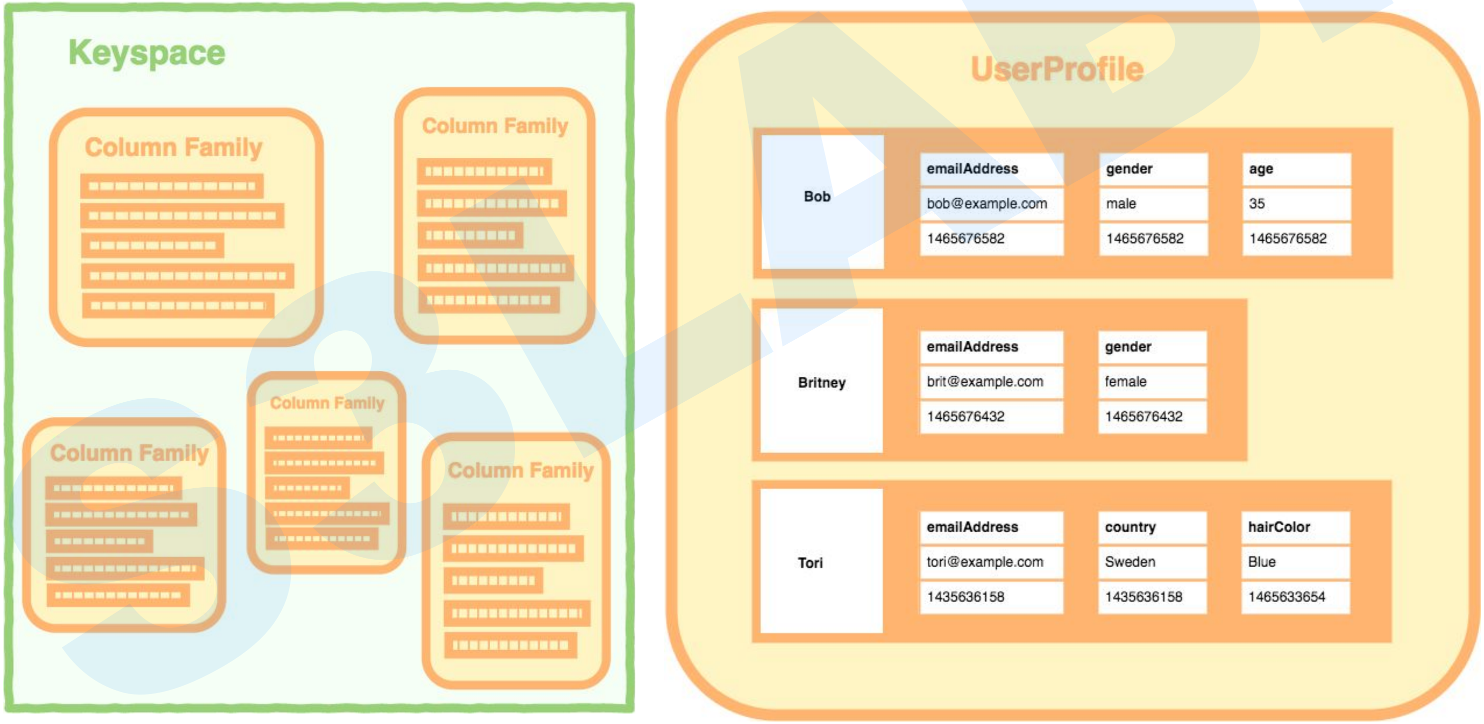
NoSQL Categories

Column-based



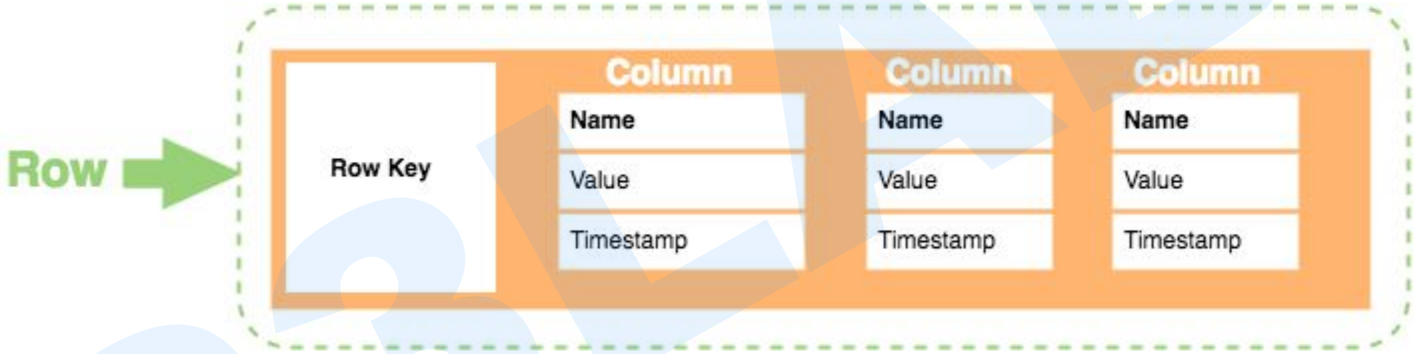
NoSQL Categories

Keyspace ~ Schema, Column Family ~ Table



NoSQL Categories

Row structure



NoSQL Categories

Column-based

- One column family can have variable numbers of columns
- Cells within a column family are sorted “physically”
- Very sparse, most cells have null values
- Comparison: RDBMS vs column-based NOSQL
  - Query on multiple tables
    - RDBMS: must fetch data from several places on disk and glue together
    - Column-based NOSQL: only fetch column families of those columns that are required by a query (all columns in a column family are stored together on the disk, so multiple rows can be retrieved in one read operation data locality)

S3LAB.

NoSQL Categories

Column-based

Operation	Column–Oriented Database	Row–Oriented Database
Aggregate Calulation of Single Column e.g. sum(price)	✔ fast	slow
Compression	✔ Higher. As stores similar data together	–
Retrieval of a few columns from a table with many columns	✔ Faster	has to skip over unnecessary data
Insertion/Updating of single new record	Slow	✔ Fast
Retrieval of a single record	Slow	✔ Fast

S3LAB.

NoSQL Categories

Column-based

- Example: (Cassandra column family--timestamps removed for simplicity)

```
UserProfile = {
        Cassandra    =  {
                        emailAddress:"casandra@apache.org" , age:"20"
                        }
        TerryCho     =  {
                        emailAddress:"terry.cho@apache.org" , gender:"male"
                        }
        Cath         =  {
                        emailAddress:"cath@apache.org",
                        age:"20",gender:"female",address:"Seoul"
                        }
    }
```

NoSQL Categories

Column-based

Name	Producer	Data model	Querying
BigTable	Google	set of couples (key, {value})	selection (by combination of row, column, and time stamp ranges)
HBase	Apache	groups of columns (a BigTable clone)	JRUBY IRB-based shell (similar to SQL)
Hypertable	Hypertable	like BigTable	HQL (Hypertext Query Language)
CASSANDRA	Apache (originally Facebook)	columns, groups of columns corresponding to a key (supercolumns)	simple selections on key, range queries, column or columns ranges
PNUTS	Yahoo	(hashed or ordered) tables, typed arrays, flexible schema	selection and projection from a single table (retrieve an arbitrary single record by primary key, range queries, complex predicates, ordering, top-k)

S3LAB.

NoSQL Categories

Document-based

- Can model more complex objects
- Inspired by Lotus Notes
- Data model: collection of documents
- Document: JSON (JavaScript Object Notation is a data model, key-value pairs, which supports objects, records, structs, lists, array, maps, dates, Boolean with nesting), XML, other semi-structured formats.

NoSQL Categories

Document-based

- **Example: (MongoDB) document**

{

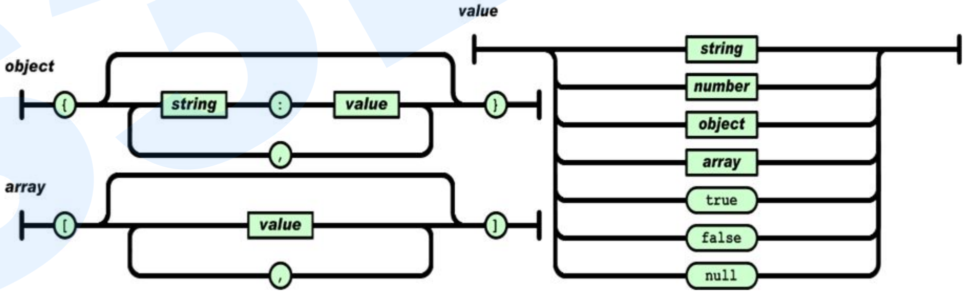
Name:"Jaroslav",

Address:"Malostranske nám. 25, 118 00 Praha 1”,

Grandchildren: {Claire: "7", Barbara: "6", "Magda: "3", "Kirsten: "1", "Otis: "3", Richard: "1“}

Phones: [“123-456-7890”, “234-567-8963”]

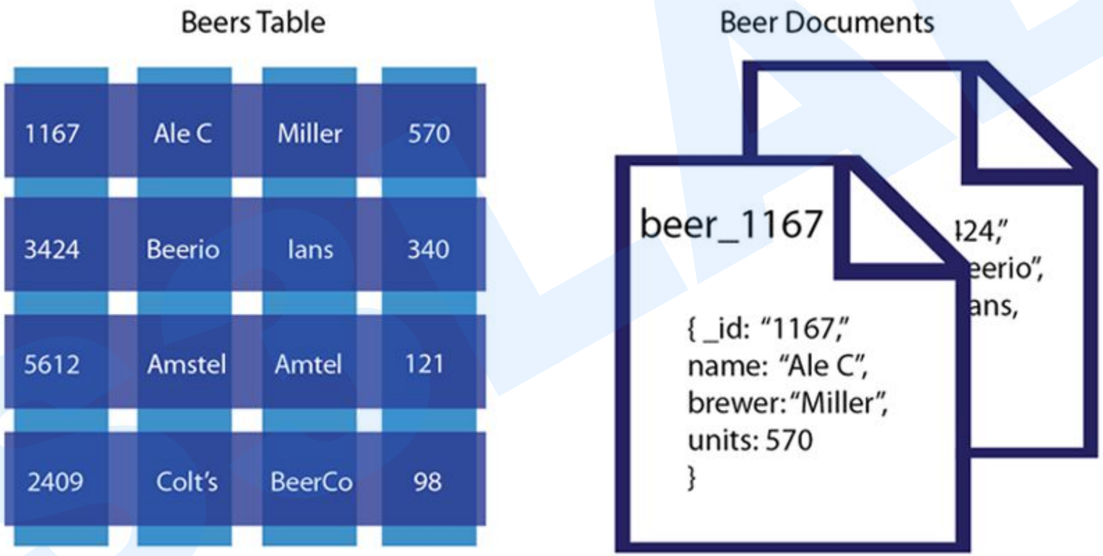
}



S3LAB.

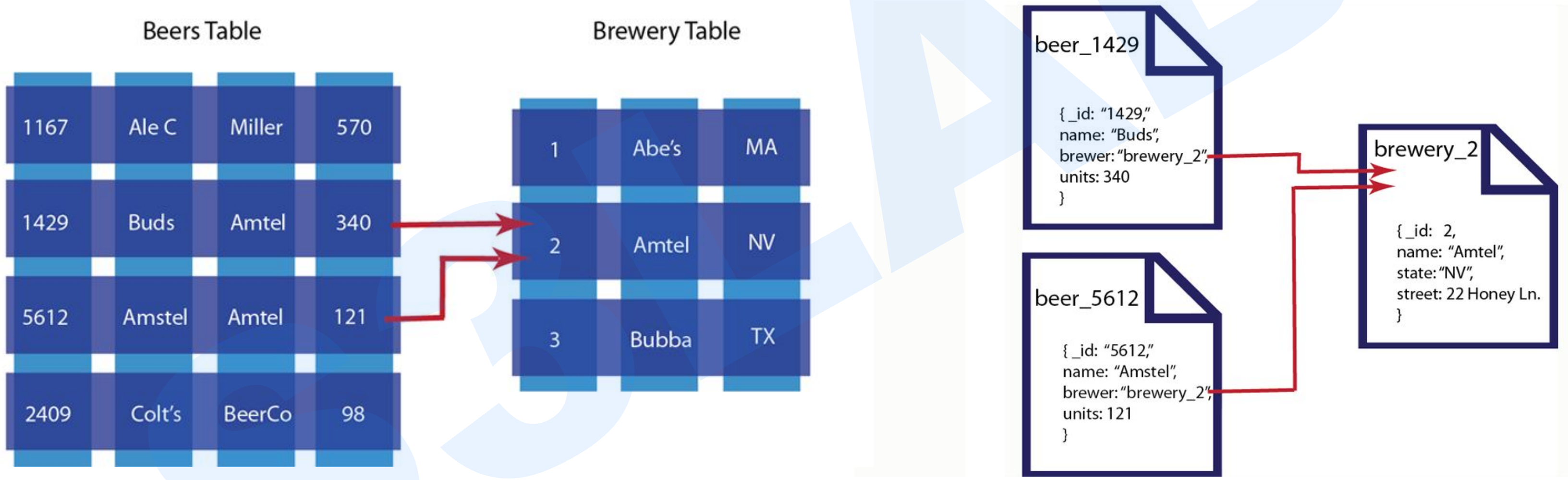
NoSQL Categories

Document-based



NoSQL Categories

Document-based



NoSQL Categories

Document-based

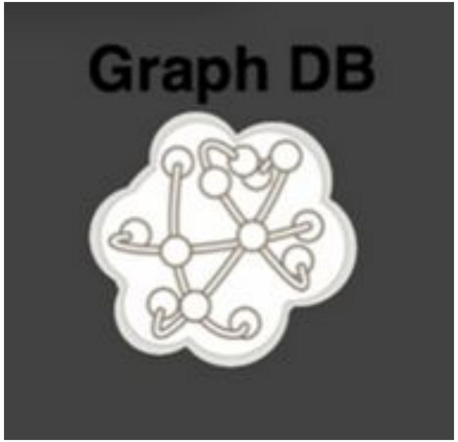
Name	Producer	Data model	Querying
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update,)
Couchbase	Couchbase[1]	document as a list of named (structured) items (JSON document)	by key and key range, views via Javascript and MapReduce

S3LAB.

NoSQL Categories

Graph-based

- Focus on modeling the structure of data (interconnectivity)
- A graph is composed of two elements: a node and a relationship.
- Scales to the complexity of data
- Inspired by mathematical Graph Theory (G=(E,V))
- Data model:
  - (Property Graph) nodes and edges
    - Nodes may have properties (including ID)
    - Edges may have labels or roles
  - Key-value pairs on both

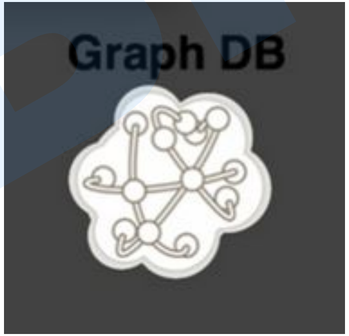


S3LAB.

NoSQL Categories

Graph-based

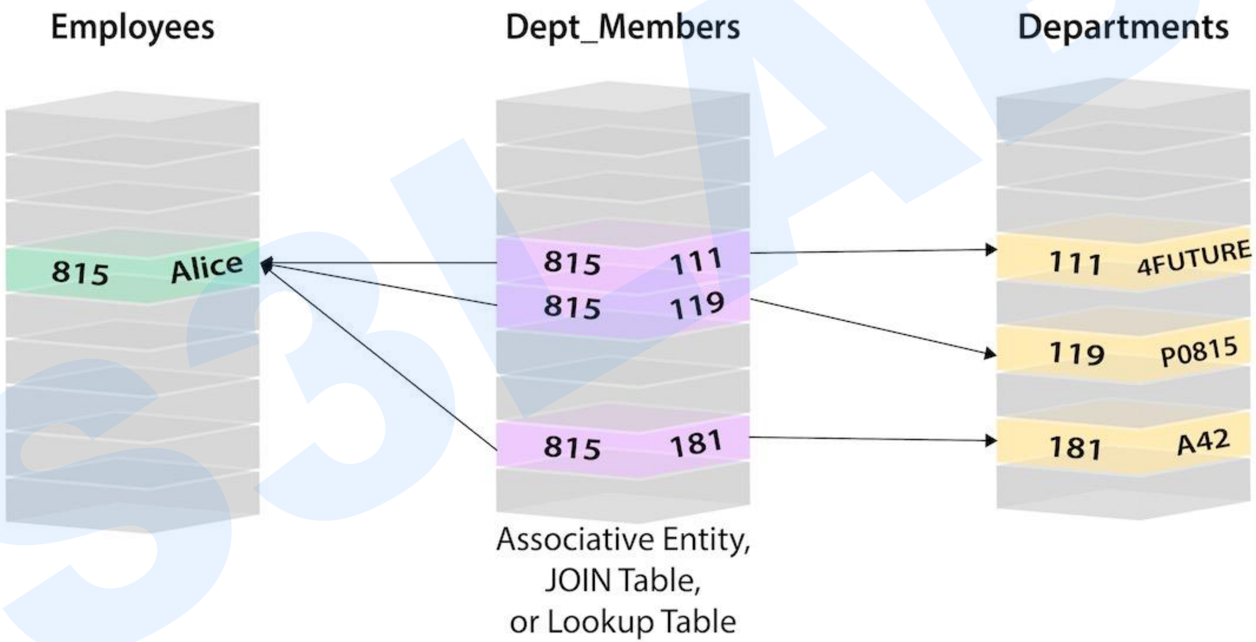
- Interfaces and query languages vary
- Single-step vs path expressions vs full recursion
- Example:
    - Neo4j, FlockDB, Pregel, InfoGrid ...



S3LAB

NoSQL Categories

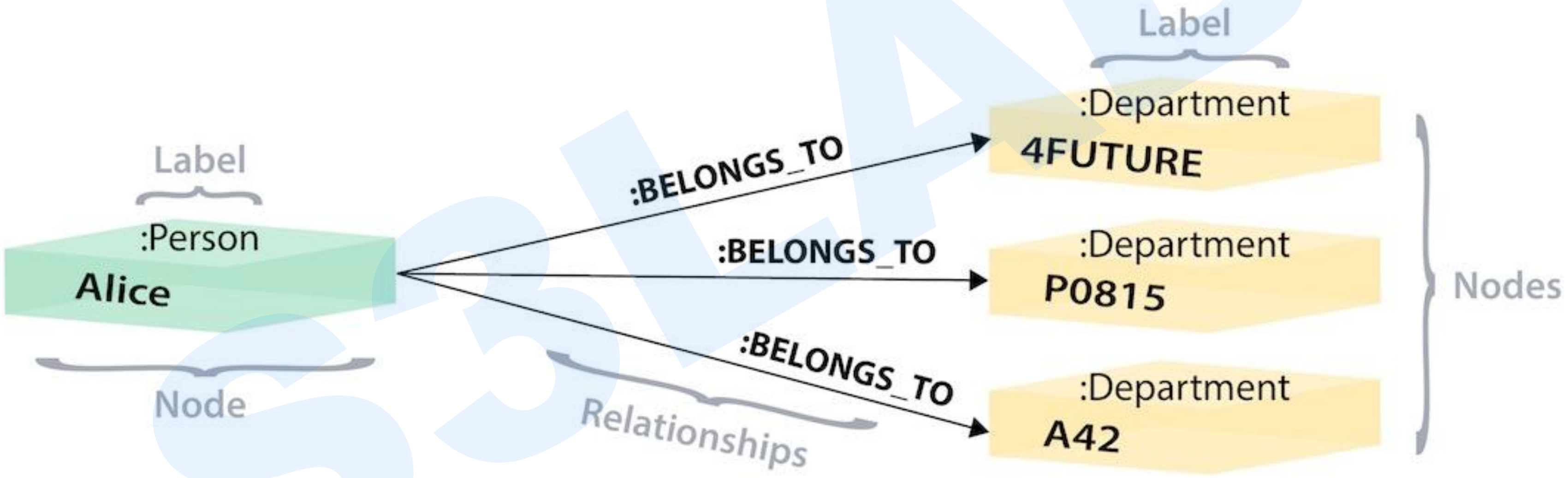
Graph-based



S3LAB.

NoSQL Categories

Graph-based



NoSQL Categories

Comparison

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored				Key-Value Stored		Graph-oriented
	Features	MongoDB	CouchDB	DynamoBD	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j
Design & Features	Data storage	Volatile memory File System	Volatile memory File System	SSD	HDFS		Hadoop	Volatile memory File System	Bitcask LevelDB Volatile memory	File System Volatile memory
	Query language	Volatile memory File System	JavaScript Memcached-protocol	API calls	API calls REST XML Thrift	API calls CQL Thrift		API calls	HTTP JavaScript REST Erlang	API calls REST SparQL Cypher Tinkerpop Gremlin
	Protocol	Custom, binary (BSON)	HTTP, REST	-	HTTP/REST Thrift	Thrift & custom binary CQL3	Thrift	Telnet-like	HTTP, REST	HTTP/REST Tembedding in Java
	Conditional entry updates	Yes	Yes	Yes	Yes	No	Yes	No	No	
	MapReduce	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
	Unicode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	TTL for Entries	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	
	Compression	Yes	Yes	-	Yes	Yes	Yes	Yes	Yes	
Integrity	Integrity model	BASE	MVCC	ASID	Log Replication	BASE	MVCC	-	BASE	ASID
	Atomicity	Conditional	Yes	Yes	Yes	Yes	Conditional	Yes	No	Yes
	Consistency	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
	Isolation	No	Yes	Yes	No	No	-	Yes	Yes	Yes
	Durability (data storage)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes
	Transactions	No	No	No	Yes	No	Yes	Yes	No	Yes
	Referential integrity	No	No	No	No	No	No	Yes	No	Yes
	Revision control	No	Yes	Yes	Yes	No	Yes	No	Yes	No
Indexing	Secondary Indexes	Yes	Yes	No	Yes	Yes	Yes	-	Yes	-
	Composite keys	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	-
	Full text search	No	No	No	No	No	Yes	No	Yes	Yes
	Geospatial Indexes	Yes	No	No	No	No	Yes	-	-	Yes
	Graph support	No	No	No	No	No	Yes	No	Yes	Yes
Distribution	Horizontal scalable	Yes	Yes	Yes	Yes	Yes	Yes		Yes	No
	Replication	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes
	Replication mode	Master-Slave-Replica Replication	Master-Slave Replication	-	Master-Slave Replication	Master-Slave Replication	-	Master-Slave Replication	Multi-master replication	-
	Sharding	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
	Shared nothing architecture	Yes	Yes	Yes	Yes	Yes	-	-	Yes	-
System	Value size max.	16MB	20MB	64KB	2TB	2GB	1EB	-	64MB	
	Operating system	Cross-platform	Ubuntu Red Hat Windows Mac OS X	Cross-platform	Cross-platform	Cross-platform	NIX 32 entries Operating system	Linux *NIX Mac OS X Windows	Cross-platform	Cross-platform
	Programming language	C++	Erlang C++ C Python	Java	Java	Java	Java	C C++	Erlang	Java

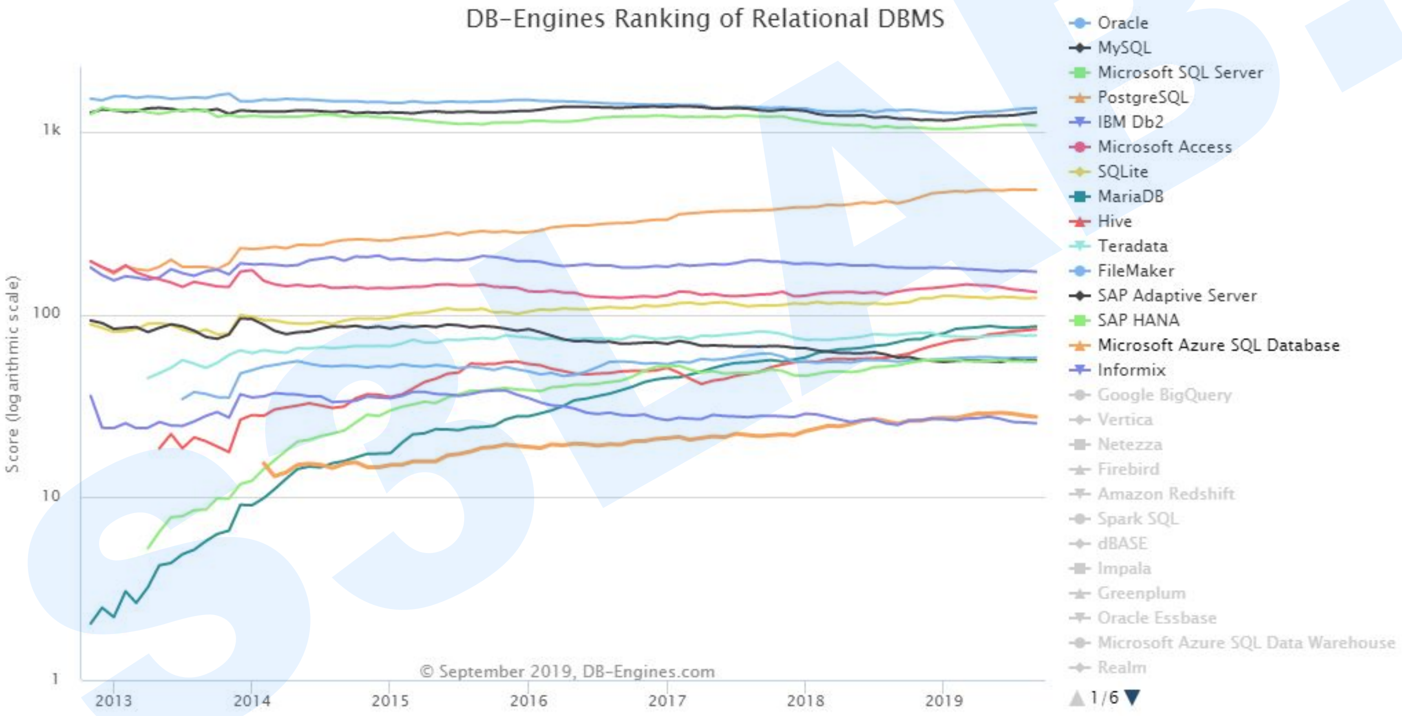
Conclusion

- NOSQL database cover only a part of data-intensive cloud applications (mainly Web applications)
- Problems with cloud computing:
  - SaaS (Software as a Service or on-demand software) applications require enterprise-level functionality, including ACID transactions, security, and other features associated with commercial RDBMS technology, i.e. NOSQL should not be the only option in the cloud
  - Hybrid solutions:
    - Voldemort with MySQL as one of storage backend
    - deal with NOSQL data as semi-structured data
      -> integrating RDBMS and NOSQL via SQL/XML

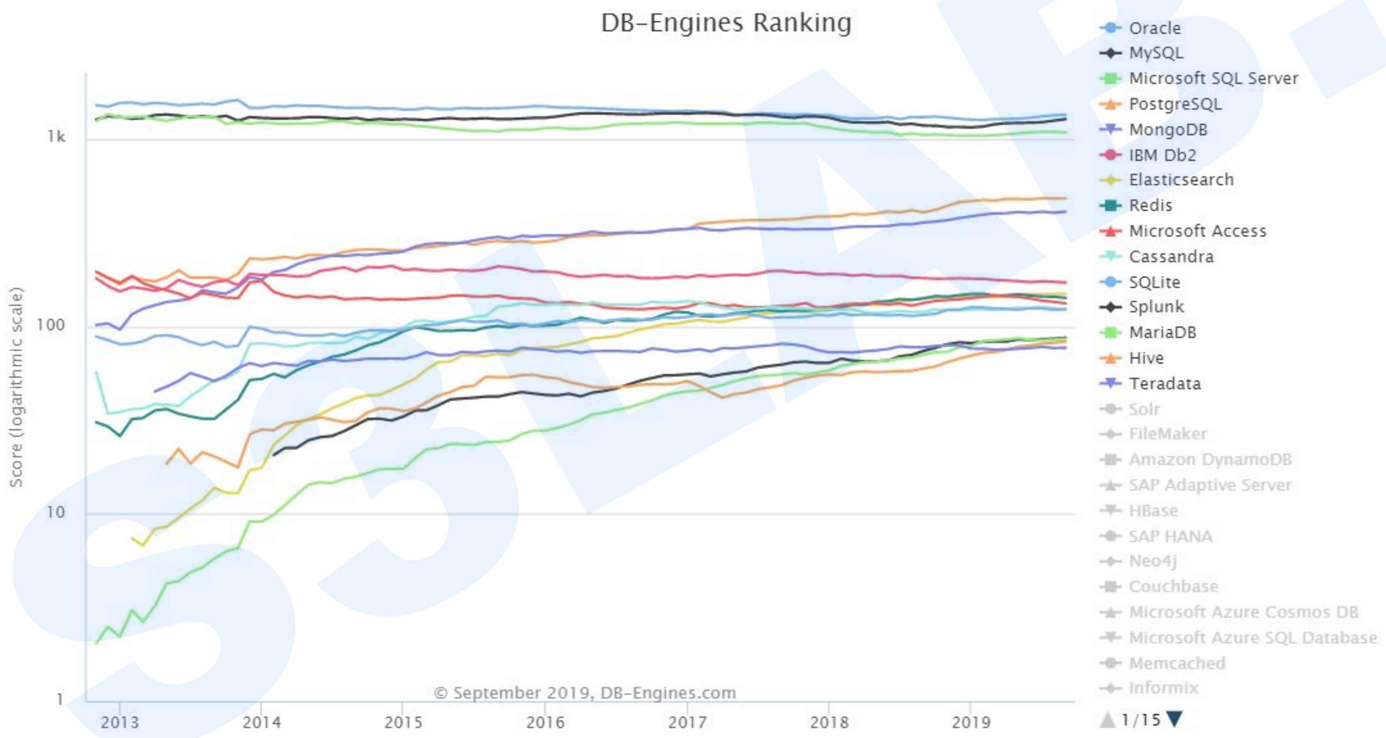
Conclusion

- next generation of highly scalable and elastic RDBMS: NewSQL databases (from April 2011)
  - they are designed to scale out horizontally on shared nothing machines,
  - still provide ACID guarantees,
  - applications interact with the database primarily using SQL,
  - the system employs a lock-free concurrency control scheme to avoid user shut down,
  - the system provides higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc, etc.

Conclusion


DB-Engines Ranking of Relational DBMS
Score (logarithmic scale)
1k
100
10
1
2013
2014
2015
2016
2017
2018
2019
© September 2019, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
IBM Db2
Microsoft Access
SQLite
MariaDB
Hive
Teradata
FileMaker
SAP Adaptive Server
SAP HANA
Microsoft Azure SQL Database
Informix
Google BigQuery
Vertica
Netezza
Firebird
Amazon Redshift
Spark SQL
dBASE
Impala
Greenplum
Oracle Essbase
Microsoft Azure SQL Data Warehouse
Realm
1/6

Conclusion



DB-Engines Ranking
Score (logarithmic scale)
1k
100
10
1
2013
2014
2015
2016
2017
2018
2019
© September 2019, DB-Engines.com
Oracle
MySQL
Microsoft SQL Server
PostgreSQL
MongoDB
IBM Db2
Elasticsearch
Redis
Microsoft Access
Cassandra
SQLite
Splunk
MariaDB
Hive
Teradata
Solr
FileMaker
Amazon DynamoDB
SAP Adaptive Server
HBase
SAP HANA
Neo4j
Couchbase
Microsoft Azure Cosmos DB
Microsoft Azure SQL Database
Memcached
Informix
1/15

Q & A

Cảm ơn đã theo dõi

Chúng tôi hy vọng cùng nhau đi đến thành công.

S3LAB.