Chapter5

Using Big Data for Analytics

NoSQL Database:

New Era of Databases for Big Data Analytics – Classification, Characteristics and Comparison

Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

Program Coordinator, MSc in Information and Communication Engineering

Member, Laboratory for ICT Research and Development (LICT)

Member, Research Management Cell (RMC)

Institute of Engineering

basanta@ioe.edu.np

http://www.basantajoshi.com.np

https://scholar.google.com/citations?user=iocLiGcAAAAJ

https://www.researchgate.net/profile/Basanta_Joshi2

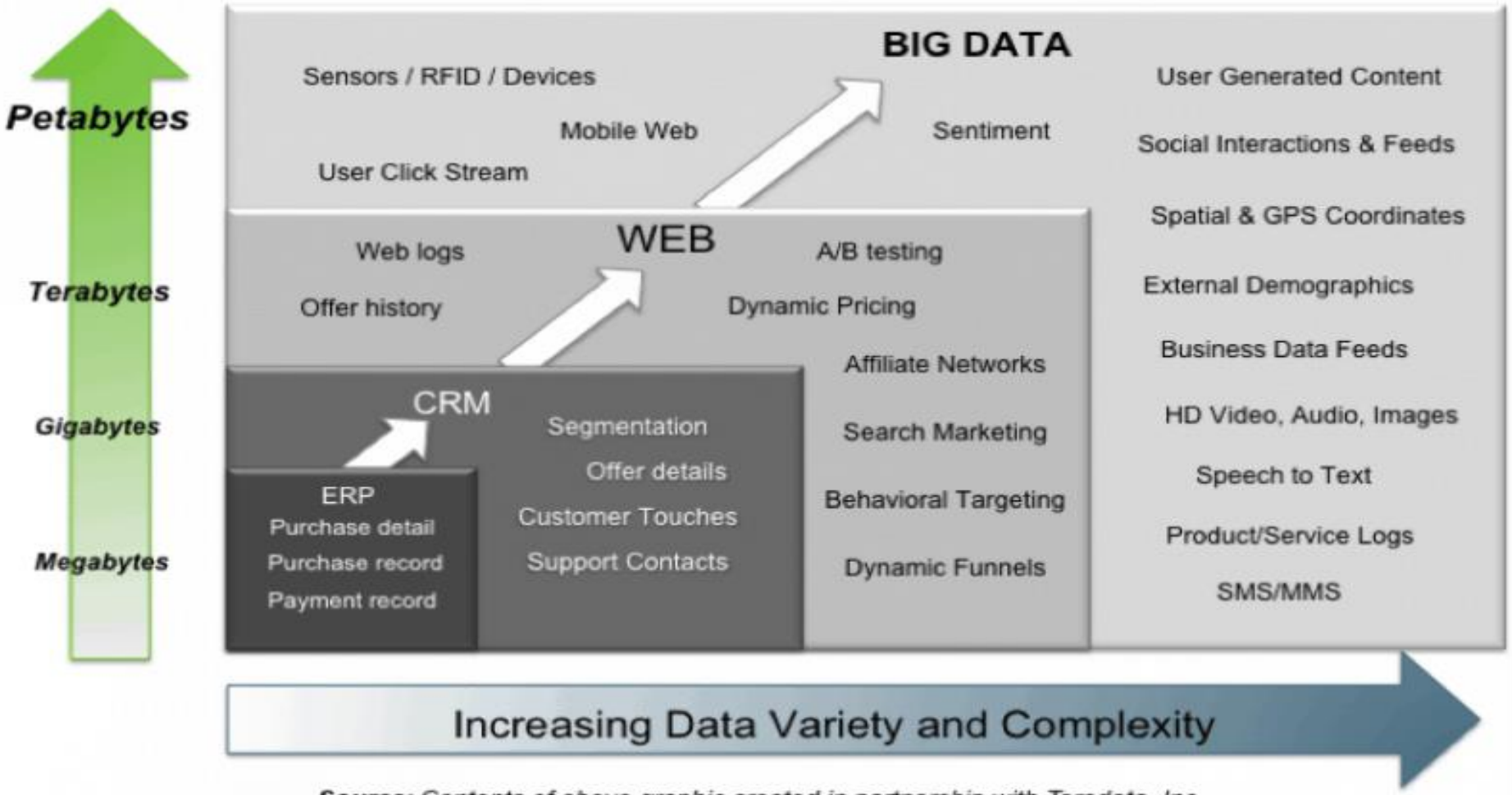
Keywords

- NoSQL Database
- Big Data
- NewSQL Database
  - ✓Provide the same scalable performanc ill maintaining the ACID
- Big Data Analytics
  - ✓Examining big data to uncover information
  - ✓Make informed business decision

Some History

- RDBMS
  - ✓Relational Database Management Sys
  - ✓Relational model of data
  - ✓Centralized database
- Trends
  - ✓Exponential growth of volume of data
  - ✓Increasing interdependency and complexity

Big Data = Transactions + Interactions + Observations



***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

Issues with RDBMS

- Scalability
  - ✓To scale relational database it has to be distributed on to multiple servers.
  - ✓Handling tables across different servers is difficult .
- Not all data is relational
- Limits to scaling up (vertical scaling)
  - ✓Costly

NoSQL System

- Distributed architecture
- Large scale data storage using non-relational databases
- Massively-parallel data processing across a large number of commodity servers
- Use non-SQL language (can use api's that translate SQL to non-SQL)
- Support exploratory and predictive analytics, ETL-style data transformation and non mission-critical OLTP

6

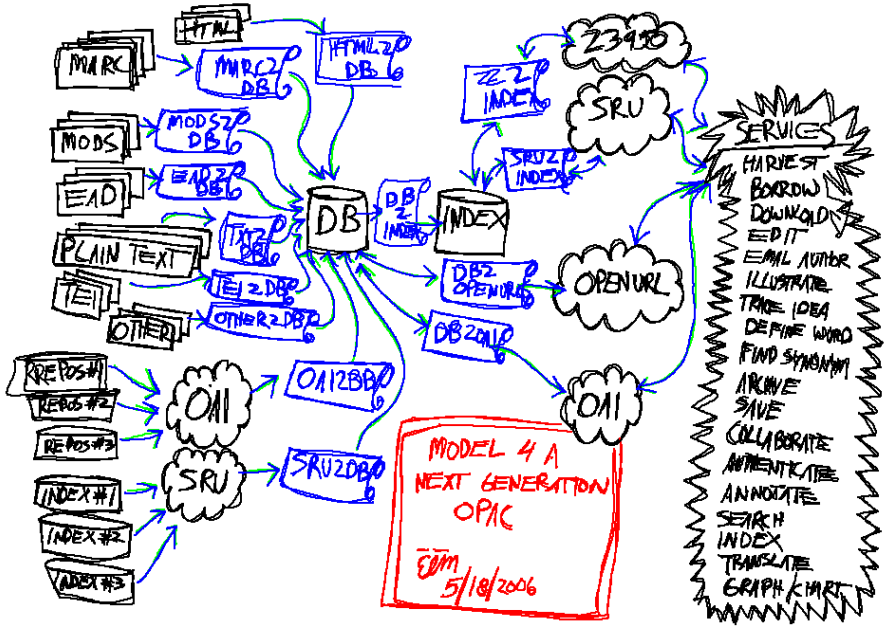
NoSQL

- Stand for “Not Only SQL”
- Non-relational data management systems
- Horizontally scalable

“NoSQL systems are distributed, non-relational databases designed for large-scale data storage and for massively-parallel data processing across a large number of commodity servers.”

Background

- *Relational Model dominated since 80s*
  - *MySQL, Oracle, MS-SQL Server*
- *Problems include deficits and modeling of data/constrains of horizontal scalability over several servers and big data*
- *MAJOR Trends*
  - *Exponential growth of volume of data generated by users, system,, sensors, Big distributed systems like Amazon and Google*
  - *Increasing independency and complexity of data accelerated by Internet, Web2.0, social networks*



The Era of Databases

newSQL Systems

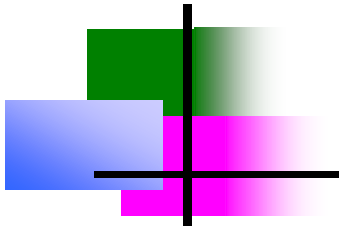
- Provide (Atomicity, Consistency, Isolation, Durability) ACID – compliant real-time OLTP
  - Eg. Manage long duration or inter-organization transactions
- Support conventional SQL-based OLAP (online analytical processing) in Big Data environments

No-SQL Systems

- Break into conventional RDBMS
  - Column –oriented data storage
  - Distributed architectures
  - In-memory processing
  - Symmetric Multi-processing(SMP)
  - Massively parallel processing(MPP)

Characteristics of NoSQL Databases

- Relaxed ACID (CAP Theorem) → fewer gurantees
- CAP Theorem
  - ✓Consistency
    - ❑All copies have same value
  - ✓Availability
    - ❑Reads and writes always succeed
  - ✓Partition-tolerance
    - ❑System properties (consistency and/or availability hold even when network failures prevent some machines from communicating with others

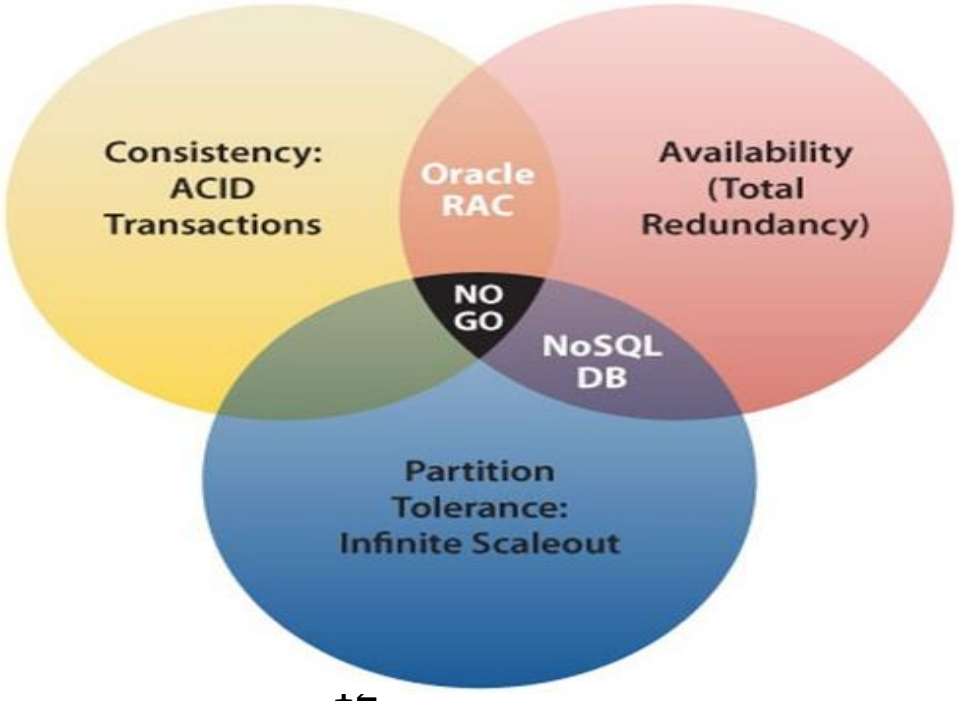


Why Non-Relational Databases?

- They focus on analytical processing of large scale datasets
- Offer Increased Scalability over commodity hardware
- Business Intelligence, Big Data Analytics and social networking have computational and storage requirements over **peta-Bytes**
- Scalable with Data-Warehousing, Grid, Web2.0 and CloudApplications
- Exhibit the ability to store and index arbitrarily big data sets --- while enabling a large no. of concurrent user requests

CAP Theorem

- Drop A or C of ACID
  - relaxing C makes replication easy, facilitates fault tolerance, speed up transactions
  - relaxing A reduces (or eliminates) need for distributed concurrency control.



CAP Theorem

- Supposes three properties of a system
  - Consistency (all copies have same value)
  - Availability (system can run even if parts have failed)
  - Partitions (network can break into two or more parts, each with active systems that can not influence other parts)

- Eric Brewer's CAP "Theorem" (1999) :
  For any system sharing data it is impossible
  to guarantee simultaneously all of these three properties

CAP Theorem

- Pick 2!

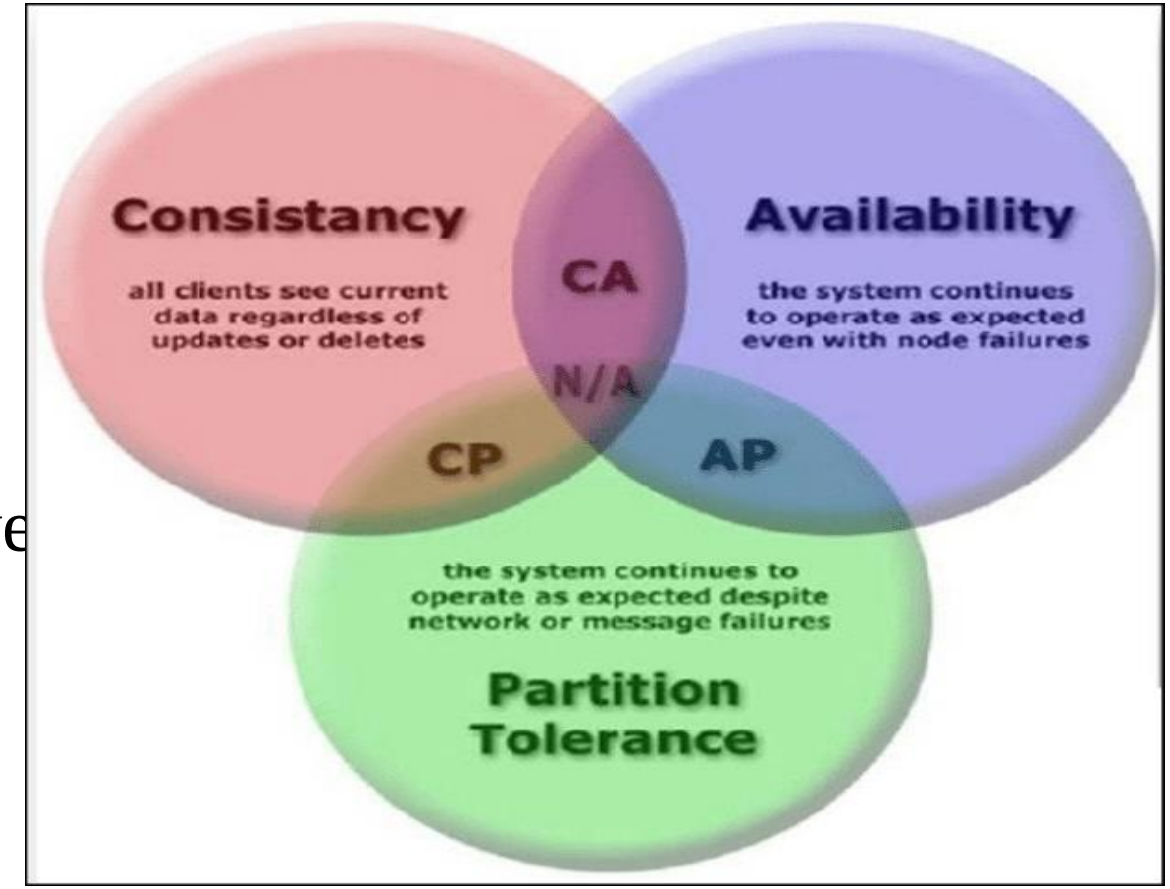
❖To scale out, partition is needed

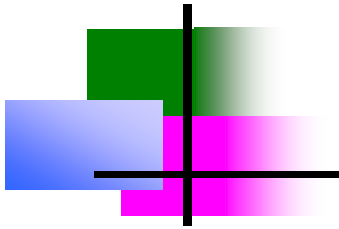
✓In almost all cases, availability and partition over consistency

Resulted in BASE

"Basically Available, Soft-state, Eve

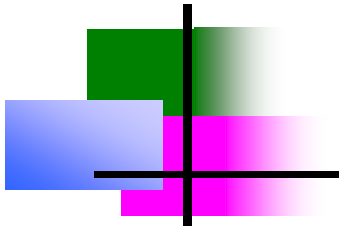
Consistent"





CAP Theorem

- Traditional RDBMS implement C and A
- Very large systems will partition at some point
  - Node failure
    (Google reports MTTF=9 hours in its
    1800’ node cluster)
  - Network failure
  - Network delay



CAP Theorem

- To deal with Big Data it is necessary to decide between C and A
- Most Web applications choose A
  (except in specific applications such as order processing)
- P depends on timeout settings
- C and A are more than just binary

BASE

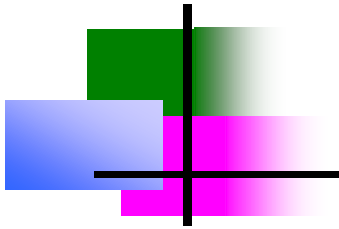
BASE

- **B**asically **A**vailable
- **S**oft state
- **E**ventual consistency

- BASE as opposed to ACID
  - Soft state: copies of a data item may be inconsistent
  - Eventually Consistent – copies becomes consistent at some later time if there are no more updates to that data item
  - Basically Available – possibilities of failures but not a failure of the whole system

BASE: Relaxing ACID properties

- BigData
  - ACID is hard to achieve, moreover, it is not always required, *e.g. for blogs, status updates, product listings, etc.*
- Availability
  - Traditionally, thought of as the server/process available 99.999% of time
  - For a large-scale node system, there is a high probability that a node is either down or that there is a network partitioning
- Partition tolerance
  - ensures that write and read operations are redirected to available replicas when segments of the network become disconnected

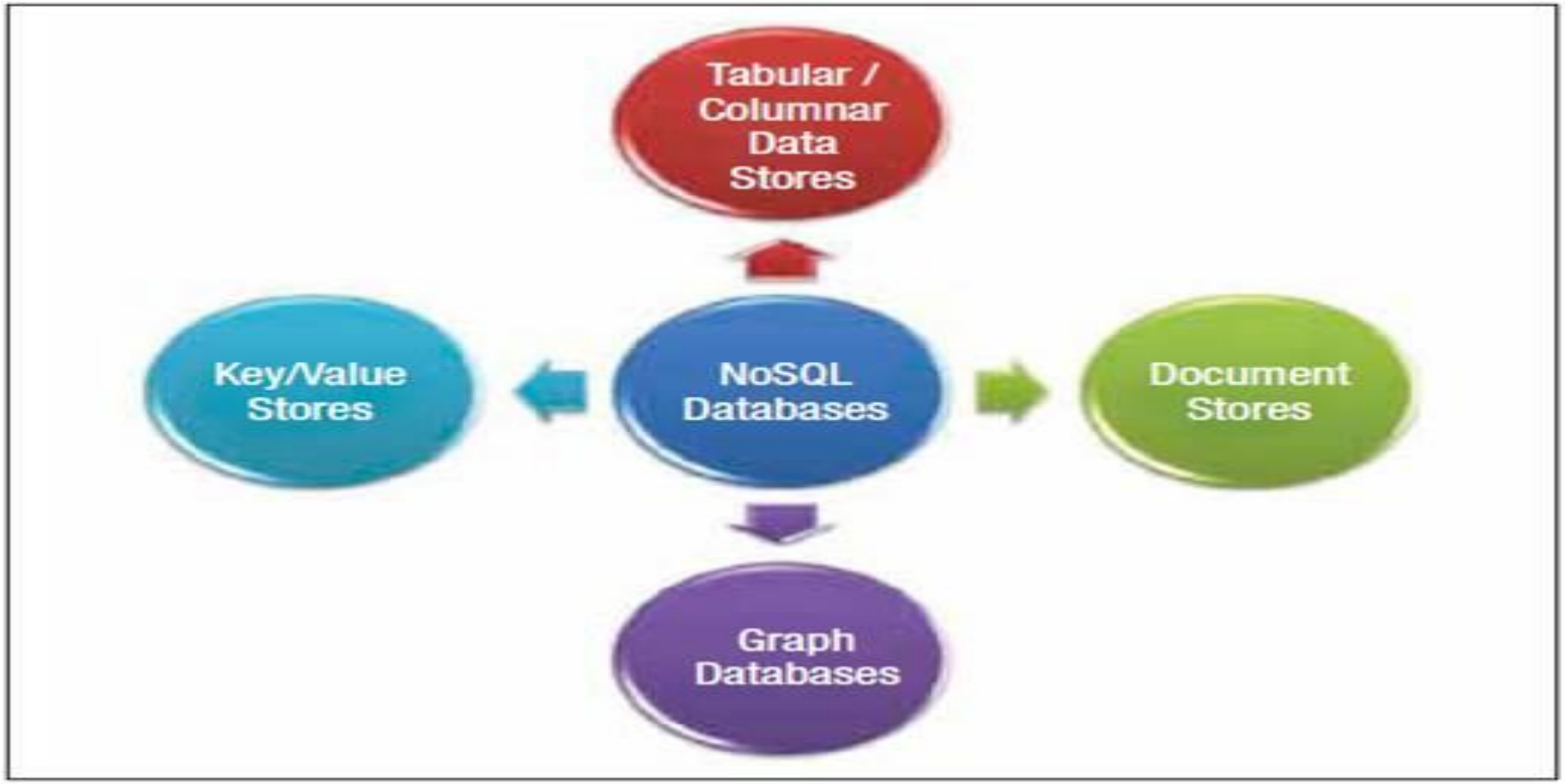


BASE: Eventual Consistency

- 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

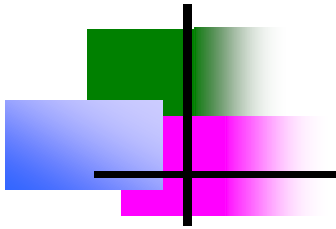
- For a given accepted update and a given node, eventually either the update reaches the node or the node is removed from service

Classification of NoSQL Databases




Redis
Neo4j
the graph database
Cassandra
CouchDB
relax
mongoDB
membase
riak


SQL



NoSQL databases

- The name stands for **N**ot **O**nly **SQL**
  - 2009, Eric Evans (Rackspace)

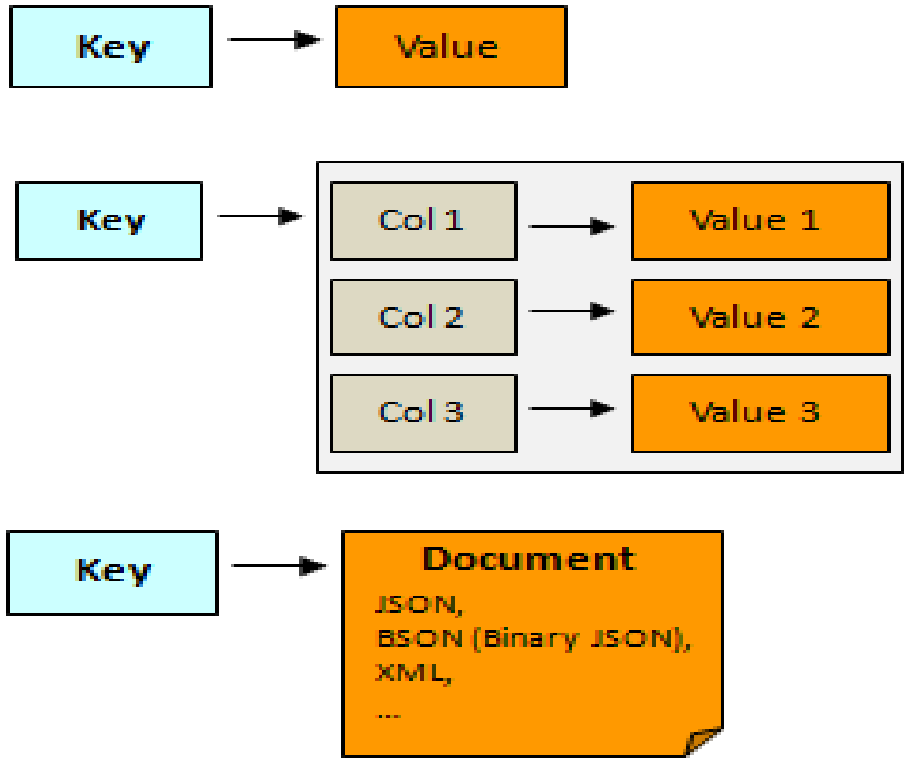
- Common features:
  - non-relational
  - do not require a fixed table schema
  - horizontal scalable
  - very fast data access (reads and writes)
  - mostly open source

NoSQL databases

- More characteristics
  - the data structure (e.g. key-value, graph, or document) differs from the RDBMS
  - relax one or more of the ACID properties
  - choose A-P or C-P (see CAP theorem)
  - replication support
  - easy API (if SQL, then only its very restricted variant)
- Do not fully support relational features
  - no join operations (except within partitions),
  - no referential integrity constraints across partitions.

Categories of NoSQL databases

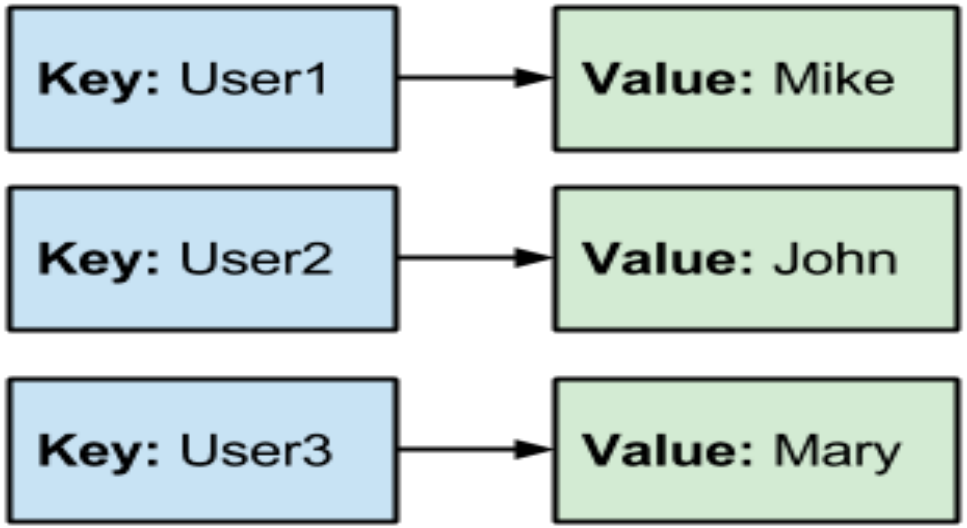
- **Key/Value**
- **Column-oriented**
- **Document-oriented**



- Graph database (neo4j, InfoGrid)
- XML databases (myXMLDB, Tamino, Sedna)

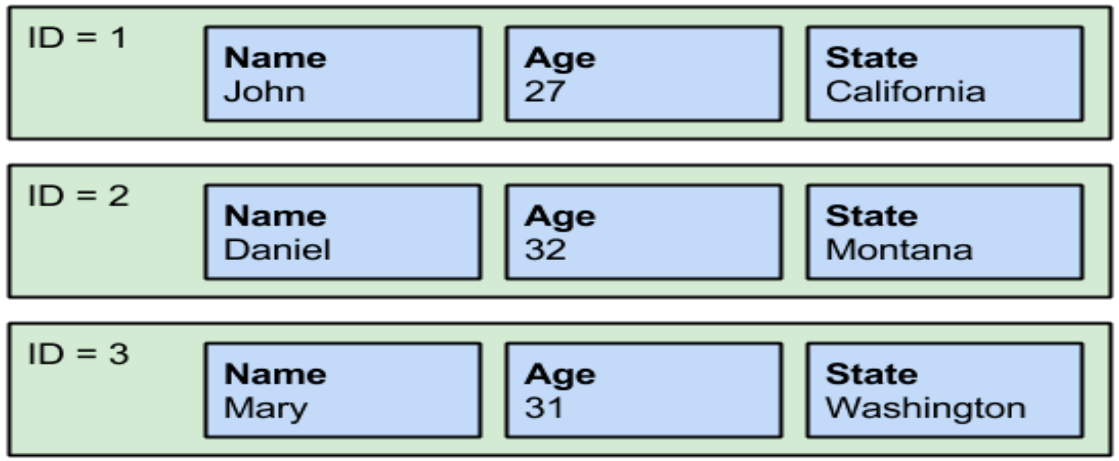
NoSQL databases: Key/Value

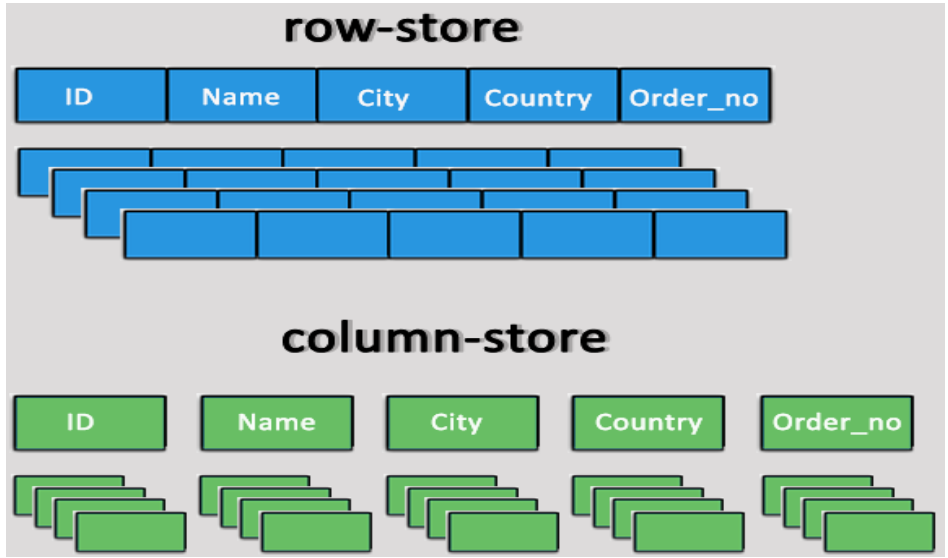
- Works as a big cache stored on a remote server.
- Each value is associated with a key.
- The value is “opaque” (no specific structure, not indexed), though can be string, JSON, BLOB, etc.



NoSQL databases: Column-oriented

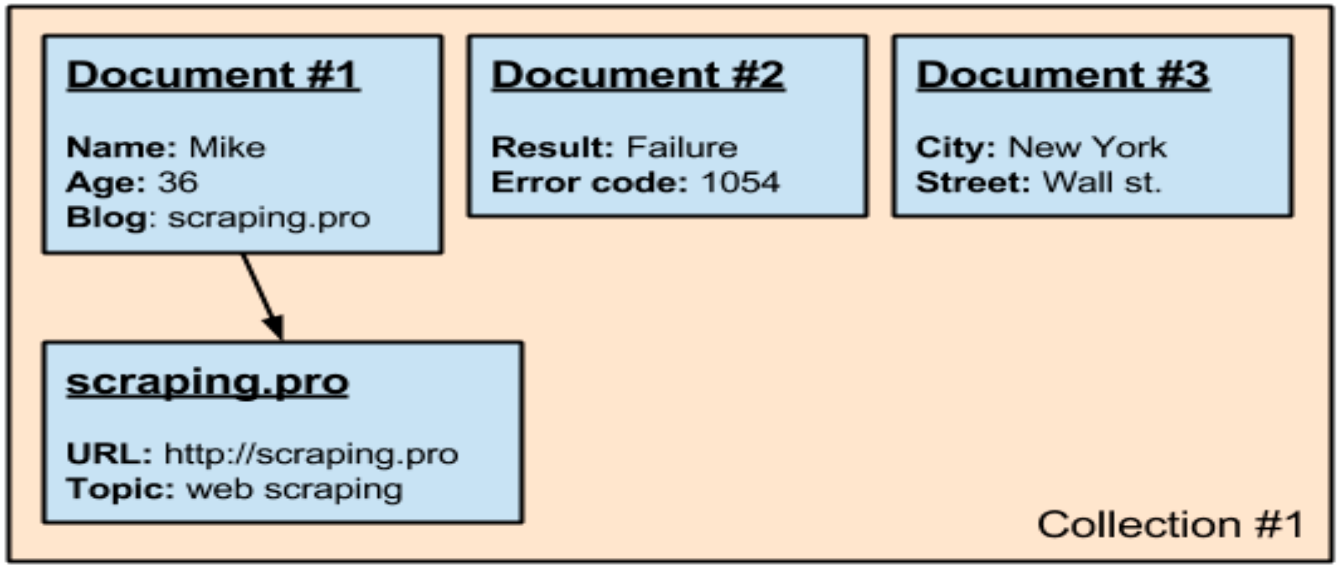
- the data is also associated with a “key” but it is organized by “columns”
- the columns can be grouped by “family”.
- There is no fixed column definition (no schema)
- the values for each column are physically stored sequentially into disk blocks





NoSQL databases: Document

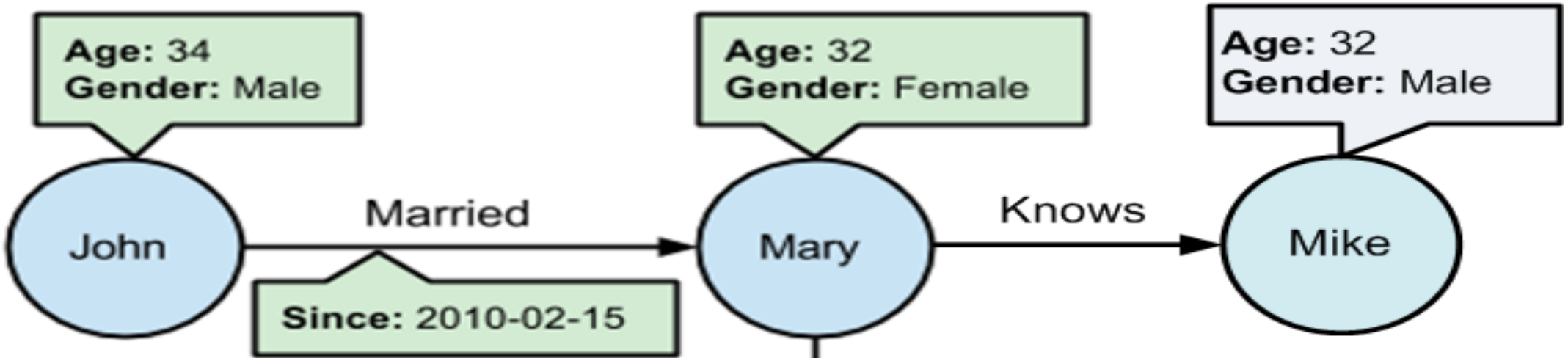
- each key is associated with a “document”
- Document is usually formatted in JSON or XML
- Each document has an arbitrary set of properties



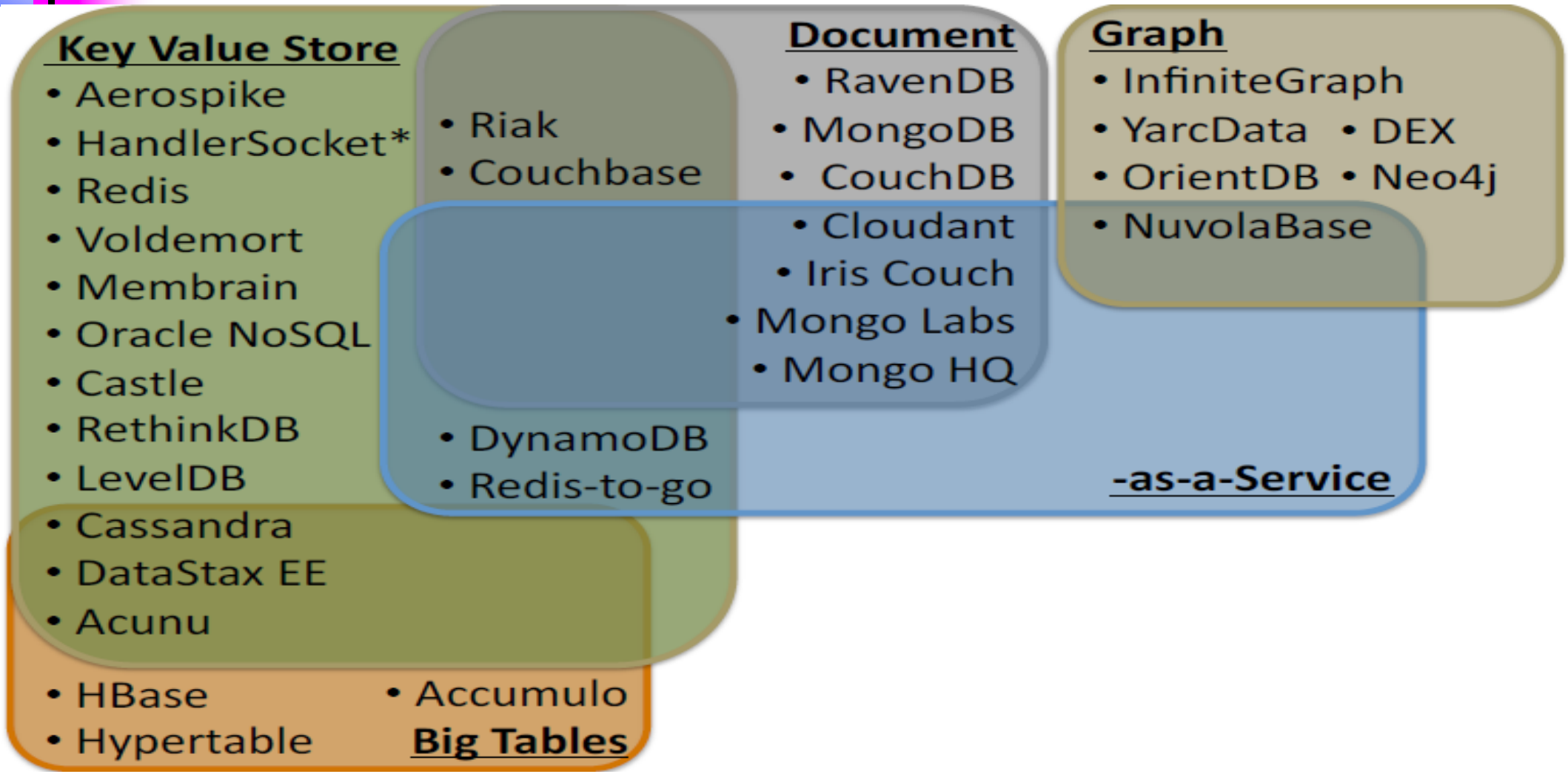
{Name: "Jack",
Address: “Graham str. 25, NE1 7RU, Newcastle, UK“
Grandchildren: [Claire: "7", Barbara: "6", Magda: "3", Kirsten: "1", Otis: "3", Richard: "1"]
}

NoSQL databases: Graph-oriented

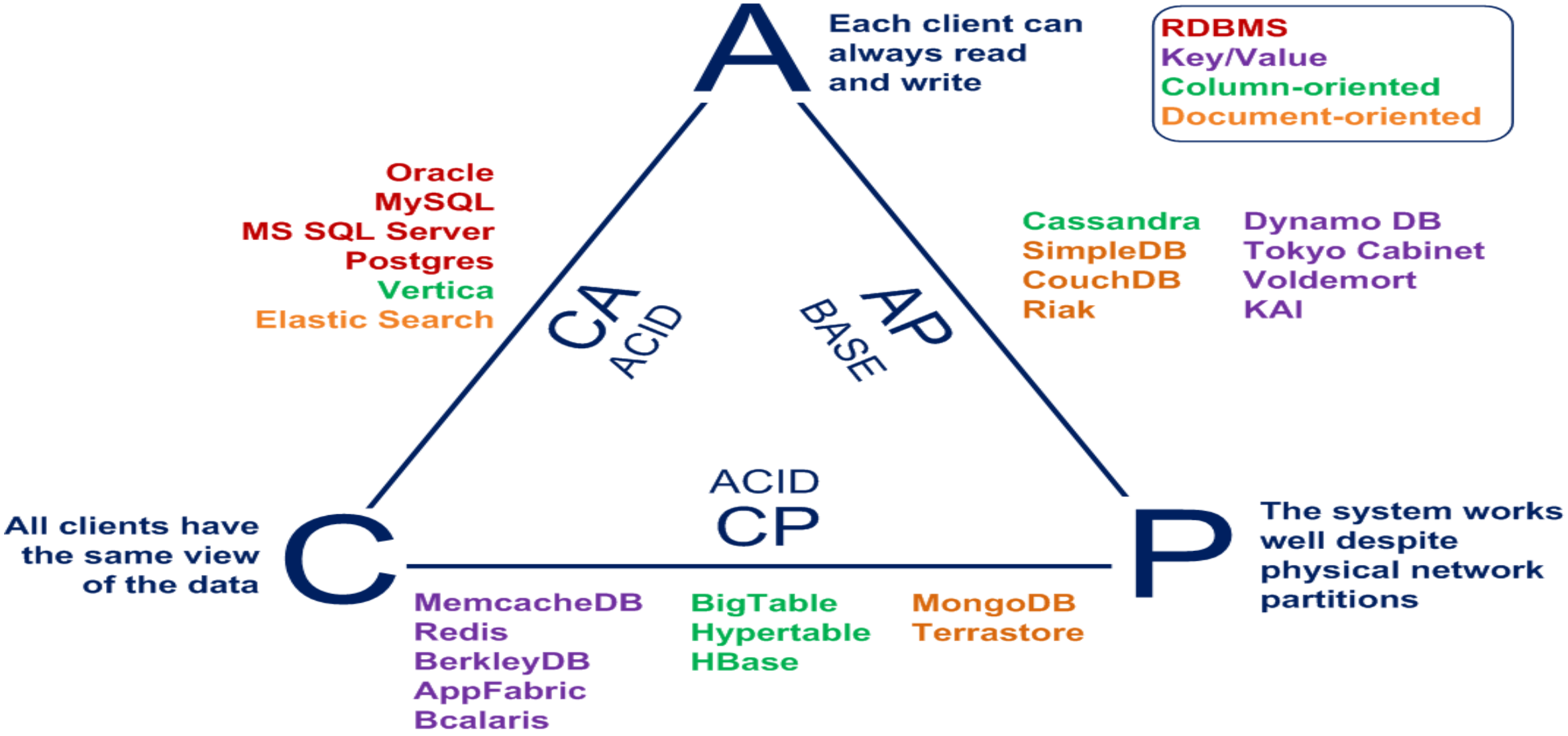
- Keep data in the forms of nodes, properties and edges
- Nodes stand for objects whose data we want to store
- Properties represent the features of those objects
- Edges show the relationships between those objects



NoSQL family



NoSQL databases: CAP implication



SQL vs NoSQL. Data definition

SQL (RDBMS)	NoSQL
1. DB contains tables, table contains columns and rows, rows are made of column values. Rows within a table all have the same schema 2. Data model is well defined in advance. A schema is strongly typed, has constraints and relationships enforcing data integrity 3. The data model is normalized to remove data duplication. Normalization establishes table relationships that associate data between tables	1. DB contains domains, domain contains rows (items), but rows contain variable set of attributes and can have different schema 2. Rows (items) are identified by keys. New attributes can be added into the row. 3. Attributes usually are textual or of a simple type 4. No relationships are explicitly defined between domains

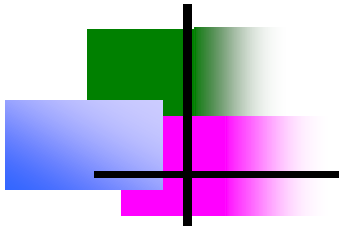
SQL vs NoSQL. Data access

SQL (RDBMS)

1. Data is created, updated, deleted and retrieved using SQL
2. SQL queries can access data from multiply tables (table joins)
3. SQL queries include functions for aggregation and complex filtering
4. DB contains means of supporting data integrity and embedding logic close to data (triggers, stored procedures)
5. Object-Relational Mapping is needed

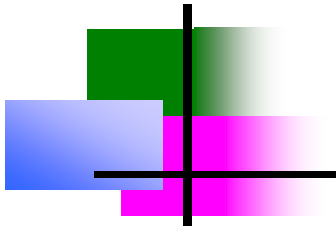
NoSQL

1. Data is created, updated, deleted and retrieved using API calls usually only by key
2. Tables joins are hardly supported
3. Only basic filter predicates (=, !=, >, <) can often be applied
4. All application and data integrity logic is contained in the application code



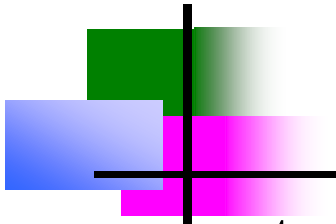
Typical NoSQL API

- 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).



NewSQL

- **NewSQL** is a class of modern RDBMS (from April 2011) that seek to:
  - provide the same scalable performance of NoSQL systems
  - still maintain the ACID guarantees of a traditional RDBMS.



NewSQL

- next generation of highly scalable and elastic RDBMS: NewSQL databases
  - still provide ACID guarantees,
  - still use SQL
  - designed to scale out horizontally on shared nothing machines,
  - employ a lock-free concurrency control scheme to avoid user shut down,
  - provide higher performance than available from the traditional systems.
- Examples: MySQL Cluster (most mature solution), VoltDB, Clustrix, ScalArc, …

NoSQL vs NewSQL

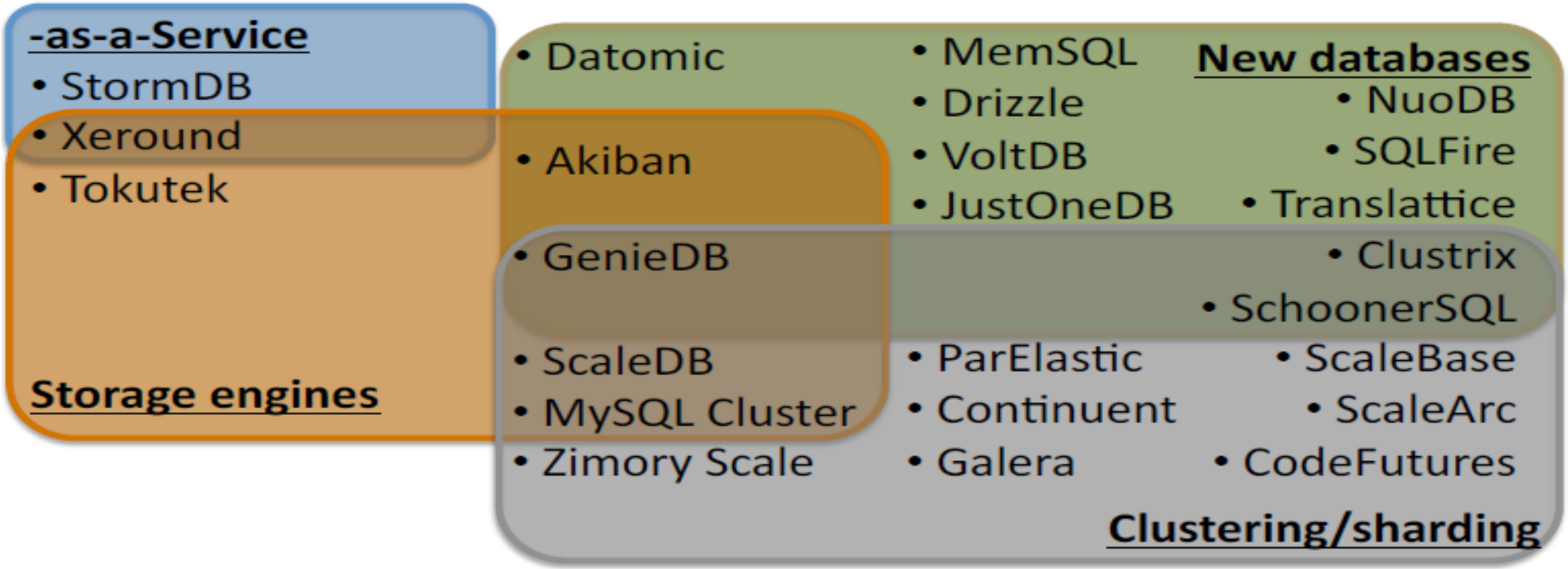
NoSQL

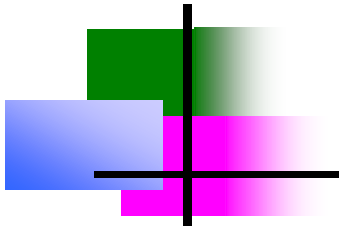
- New breed of non-relational database products
- Rejection of fixed table schema and join operations
- Designed to meet scalability requirements of distributed architectures
- And/or schema-less data management requirements

NewSQL

- New breed of relational database products
- Retain SQL and ACID
- Designed to meet scalability requirements of distributed architectures
- Or improve performance so horizontal scalability is no longer a necessity

NewSQL family





NoSQL Database Uses

- Large Scale data processing (Parallel Processing over Distributed Processing)
- Embedded IR (basic machine to machine information lookup and retrieval)
- Exploratory analytics on semi-structured data (expert Level)
- Large volume data storage (un-, semi-, small-packet structured)

10

NoSQL Databases -- Classification

- Key-Value Stores
- Column-oriented databases
- Wide-column stores
- Graph Databases

41

Key Value Stores

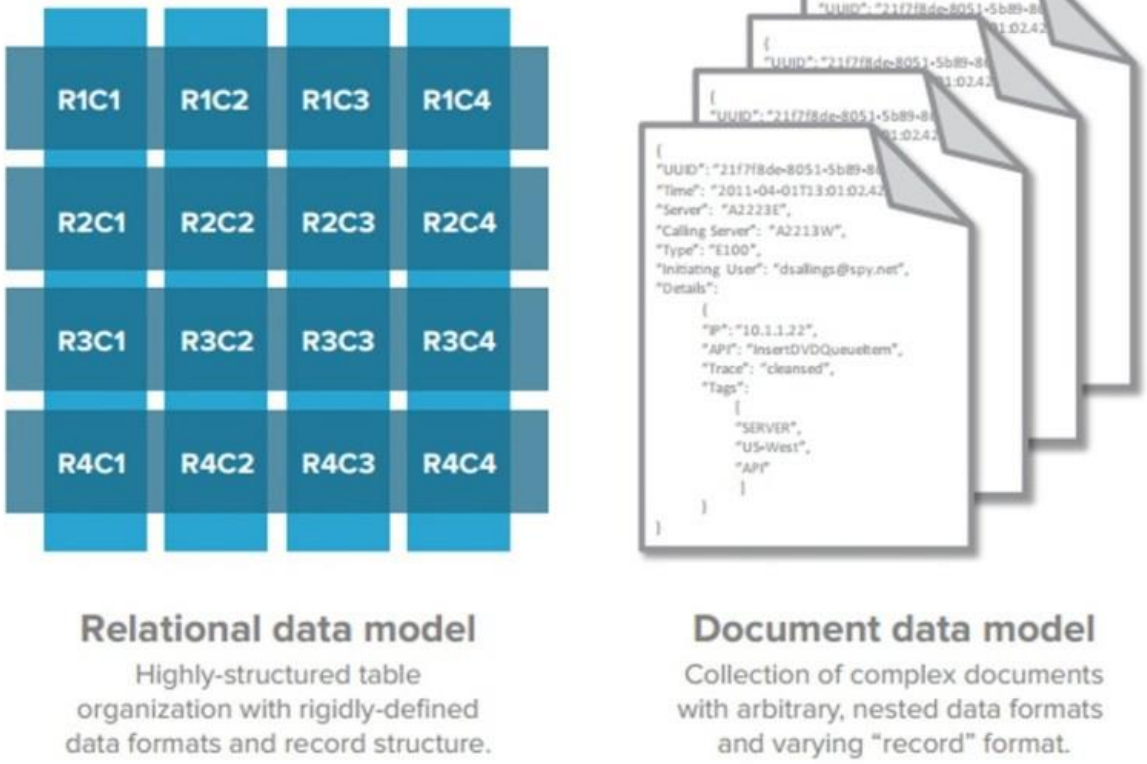
- Stores alpha-numeric identifiers (keys)
- Associated values in hash tables
  - Simple text strings
  - Complex lists
  - Sets
- Data Search – Keys, not values, limited to exact matches
  - Primary USE:
    - Manage user profiles/sessions/retrieve product names
  - Eg. Dynamo of Amazon, Voldemort(LinkedIn), Redis, BerkeleyDB, Riak

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

Document Databases

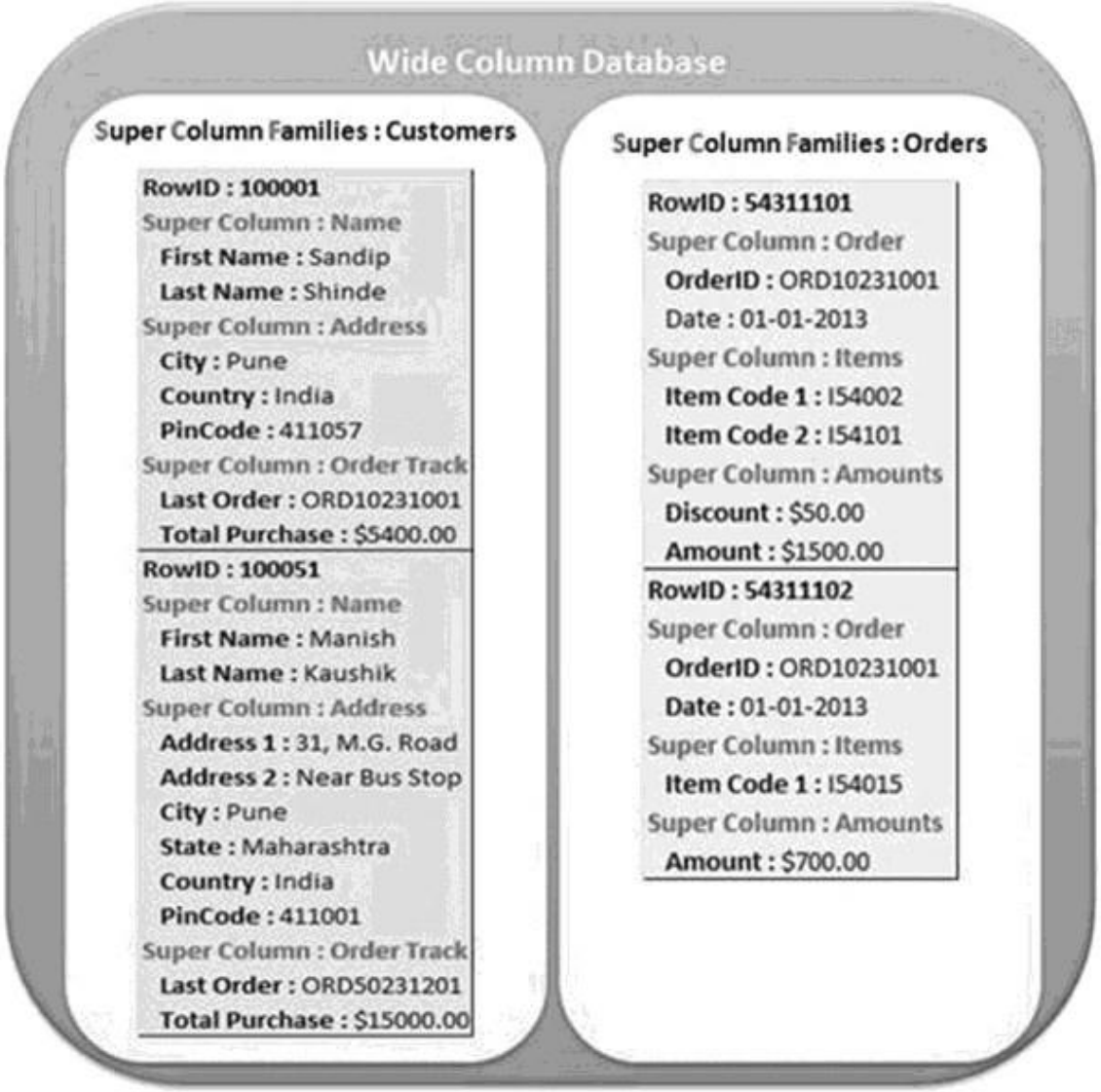
- Value column contains semi-structued data
  - Esp. attribute name/value pairs
- Keys and values are fully searchable in document databases

  - Primary USE:
    - Store and manage Big Data-size collections of literal documents – email, xml, email, de-normalized database entity such as product/customer
    - Storing parse data
    - Eg. CouchDB (JSON), MongoDB(BSON)



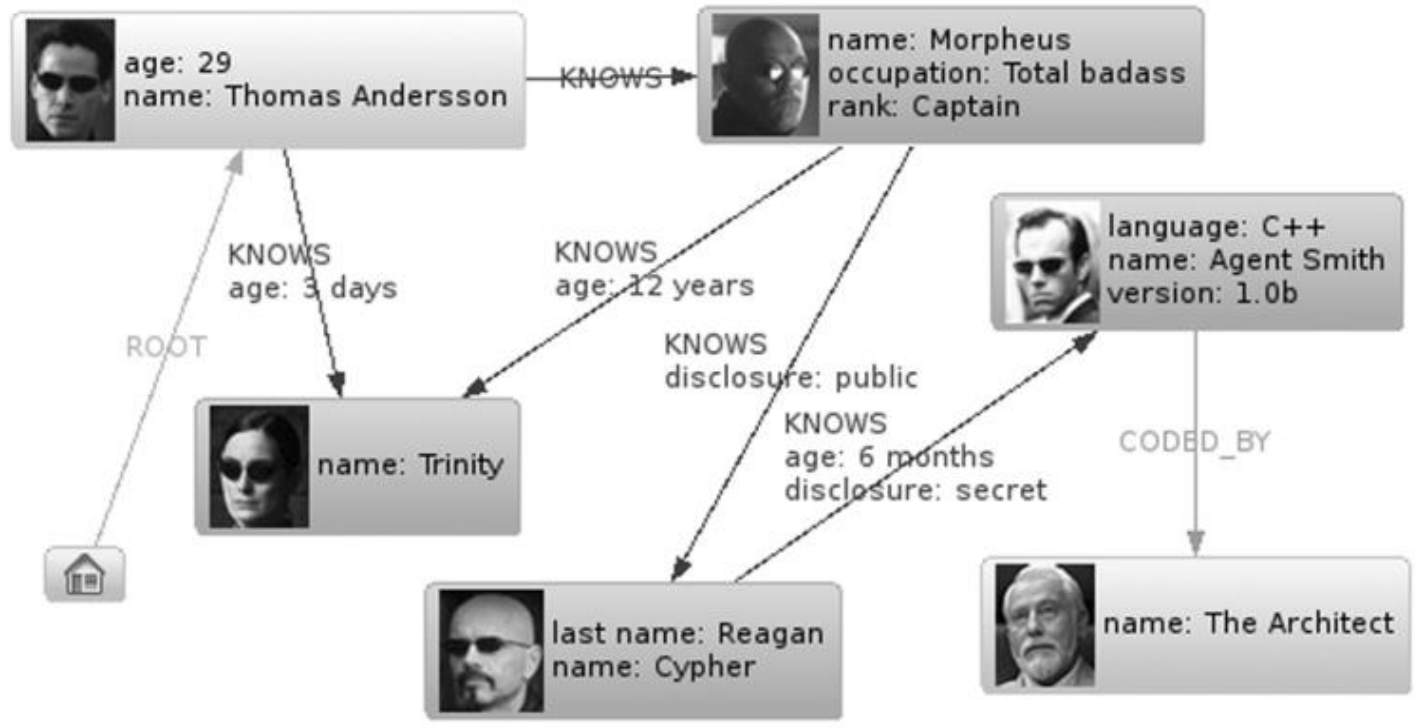
Wide-Column Stores

- Similar to Doc DB in that one key can accommodate multiple attributes
- Patterned after --- Google's Big Table Data Storage (Google Search Engine)
- GFS filesystem, MapReduce parallel processing framework, Hadoop File System, Hbase
- Primary Uses:
  - Distributed Data Storage
  - Large-scale, batch-oriented data processing
  - Exploratory and Predictive Analytics
- It uses MapReduce – Batch Processing Method
  - Recently upgraded process is **Caffeine --** search

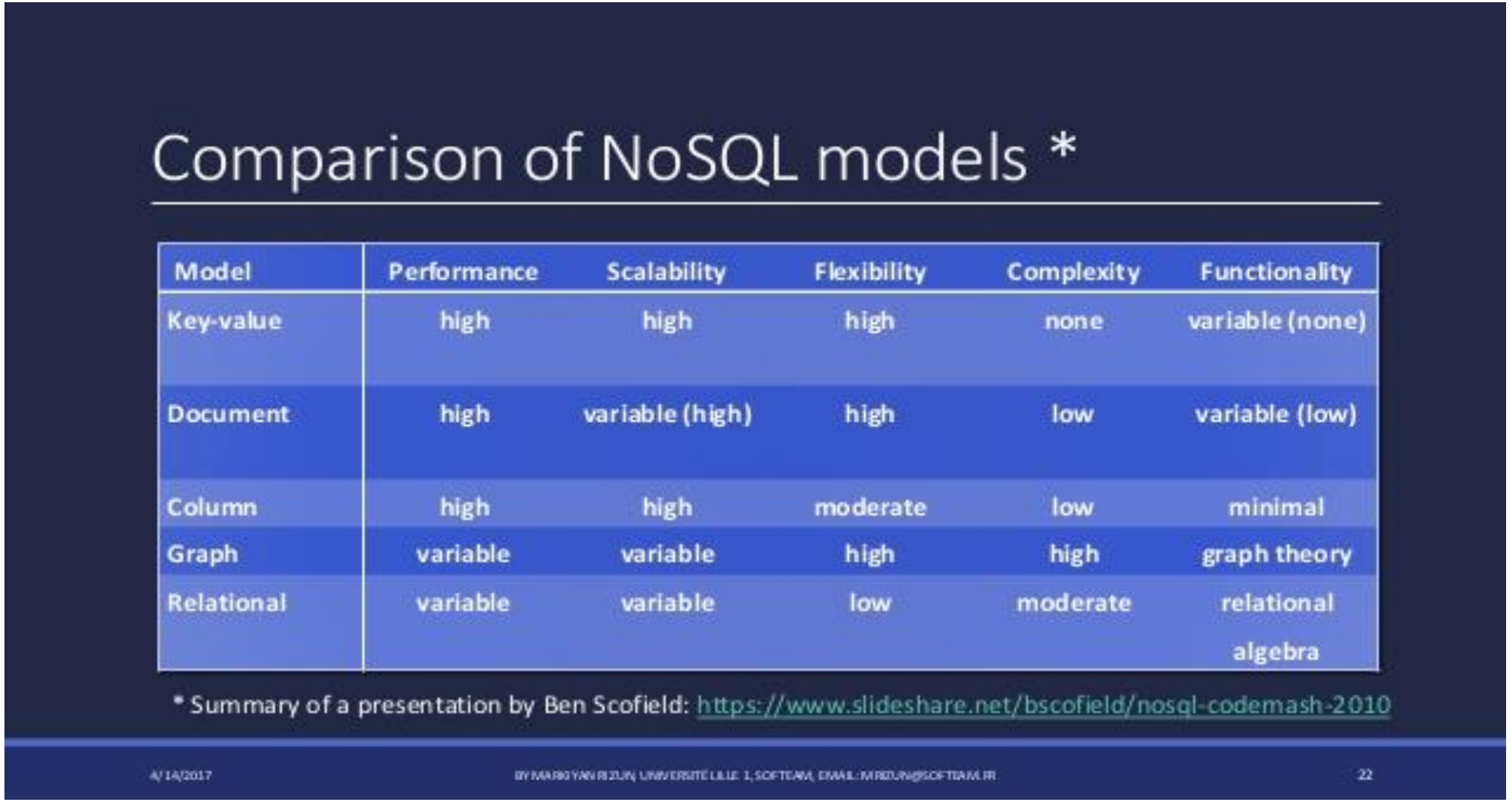


Graph Databases

- Replace Relational tables with
  - Structured Relational graphs of interconnected key-value pairings
- Resemble Object Oriented DBs.
  - Nodes – Conceptual Objects
  - Edges (Node Relationships)
  - Properties -- Object attributes (K-V pairs)
- Prime Uses:
  - Human-Friendly DB
  - Interesting Relationship Representation
    - Social Networks
    - Recommendation System
    - Forensic Investigations (Pattern Recognitio
  - Traversing data || Not querying!!
- Eg. InfoGrid, Neo4j, Sones GraphDB, AllegroGraph, Infinite Graph



Comparison of NoSQL Database



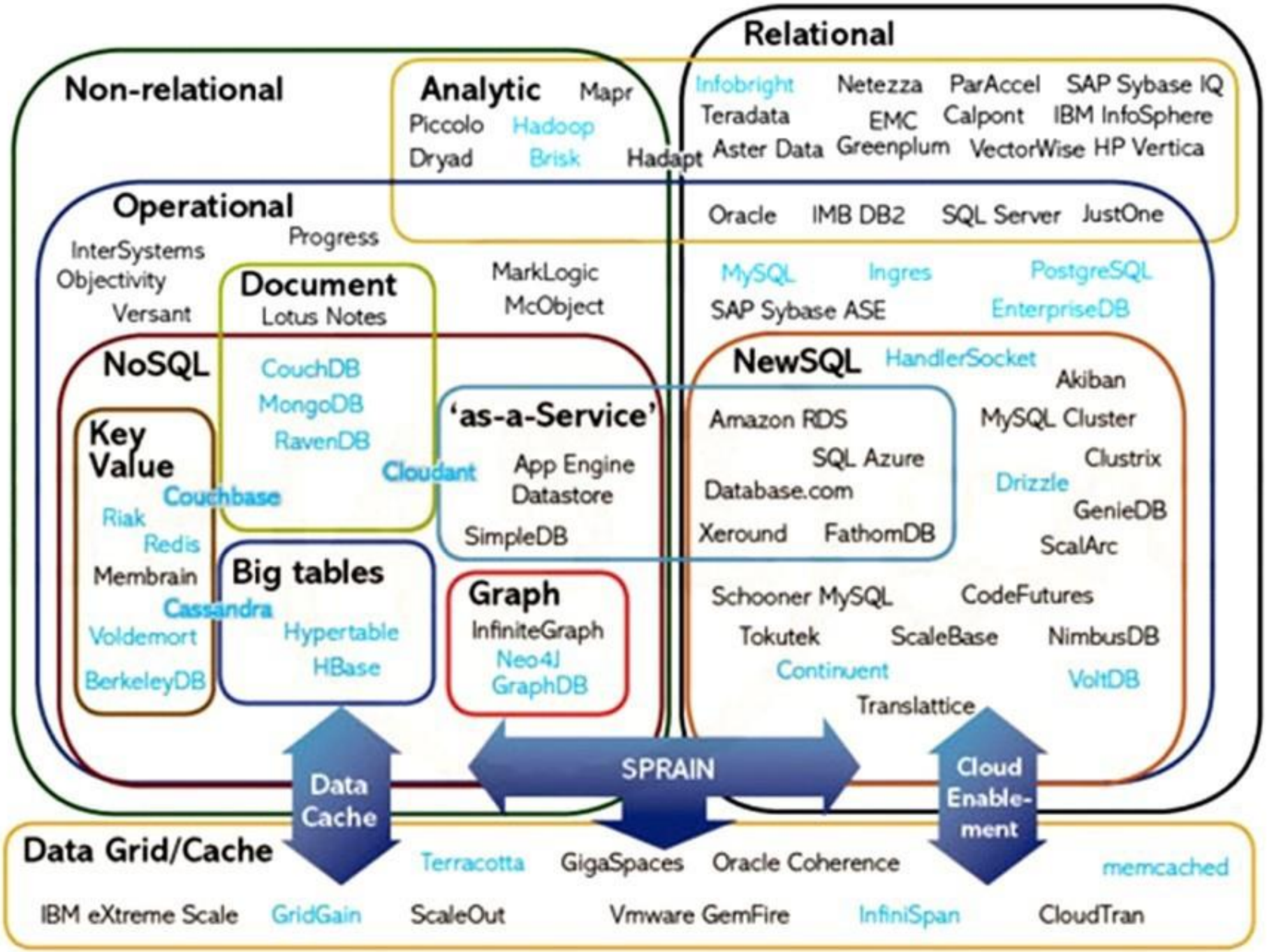
Comparison of NoSQL models *

Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value	high	high	high	none	variable (none)
Document	high	variable (high)	high	low	variable (low)
Column	high	high	moderate	low	minimal
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

* Summary of a presentation by Ben Scofield: https://www.slideshare.net/bscofield/nosql-codemash-2010

4/14/2017 BY MARJAN RIZUN, UNIVERSITÉ LILLE 1, SOFTEAM, EMAIL: M.RIZUN@SOFTEAM.FR 22

Comparision – Design, Integrity, Indexing, Distribution, System



Comparison of NoSQL Database

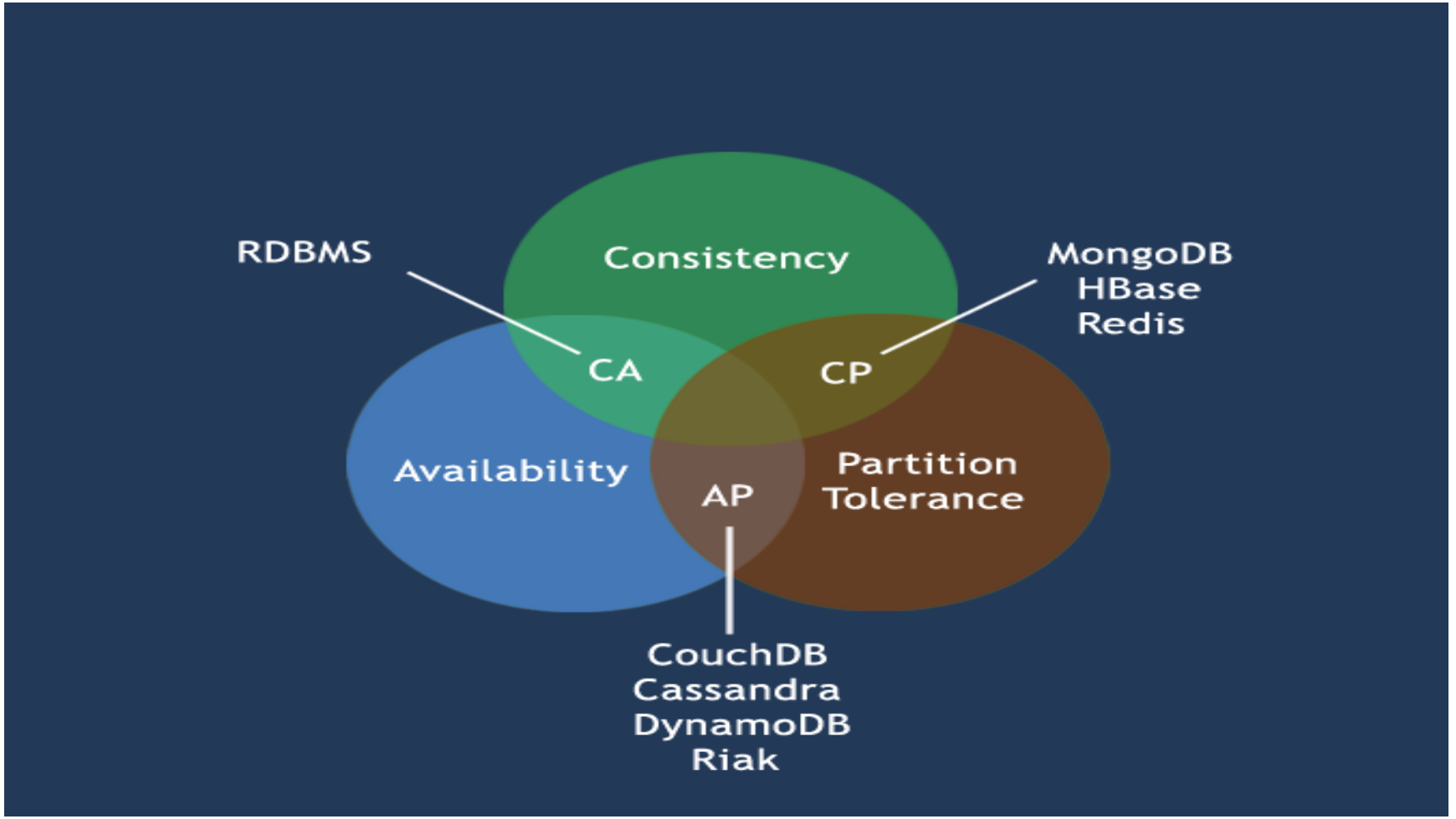
Comparison of NoSQL models *

Model	Performance	Scalability	Flexibility	Complexity	Functionality
Key-value	high	high	high	none	variable (none)
Document	high	variable (high)	high	low	variable (low)
Column	high	high	moderate	low	minimal
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

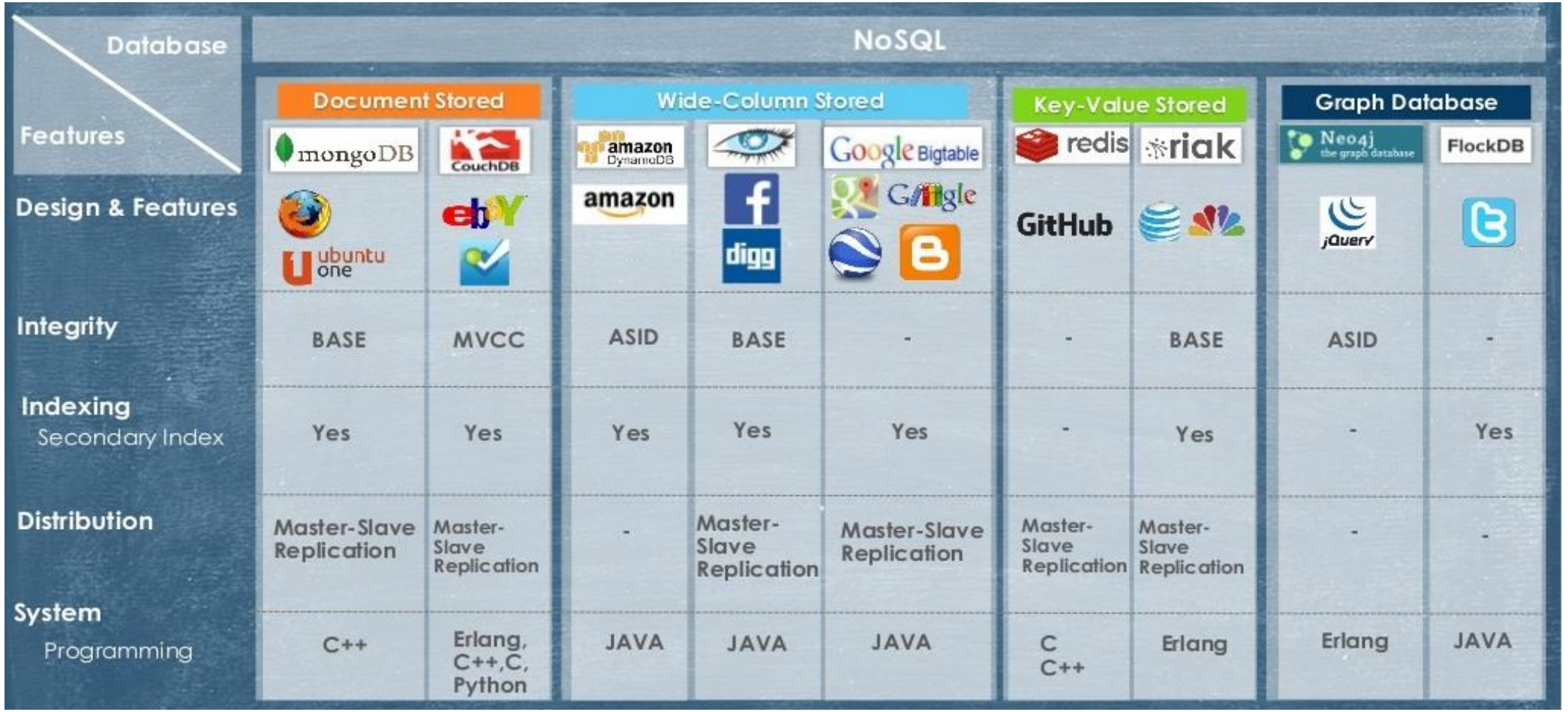
* Summary of a presentation by Ben Scofield: https://www.slideshare.net/bscofield/nosql-codemash-2010

4/14/2017 BY MARIO YAN RIZUN, UNIVERSITÉ LILLE 1, SOFTEAM, EMAIL: M.RIZUN@SOFTEAM.FR 22

Comparison of NoSQL Database



Comparison of NoSQL Database



Database / Features	NoSQL								
	Document Stored		Wide-Column Stored			Key-Value Stored		Graph Database	
Design & Features	mongoDB, CouchDB, ubuntu one, ebay		amazon DynamoDB, amazon	digg	Google Bigtable, Google, Blogger	redis, GitHub	riak	Neo4j the graph database, jQuery	FlockDB
Integrity	BASE	MVCC	ASID	BASE	-	-	BASE	ASID	-
Indexing Secondary Index	Yes	Yes	Yes	Yes	Yes	-	Yes	-	Yes
Distribution	Master-Slave Replication	Master-Slave Replication	-	Master-Slave Replication	Master-Slave Replication	Master-Slave Replication	Master-Slave Replication	-	-
System Programming	C++	Erlang, C++,C, Python	JAVA	JAVA	JAVA	C C++	Erlang	Erlang	JAVA

Comparison of NoSQL Database

Attributes		NoSQL Databases								
Database model		Document-Stored		Wide-Column Stored				Key-Value Stored		Graph-oriented
Design & Features	Features	MongoDB	CouchDB	DynamoBD	HBase	Cassandra	Accumulo	Redis	Riak	Neo4j
	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/RES 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	

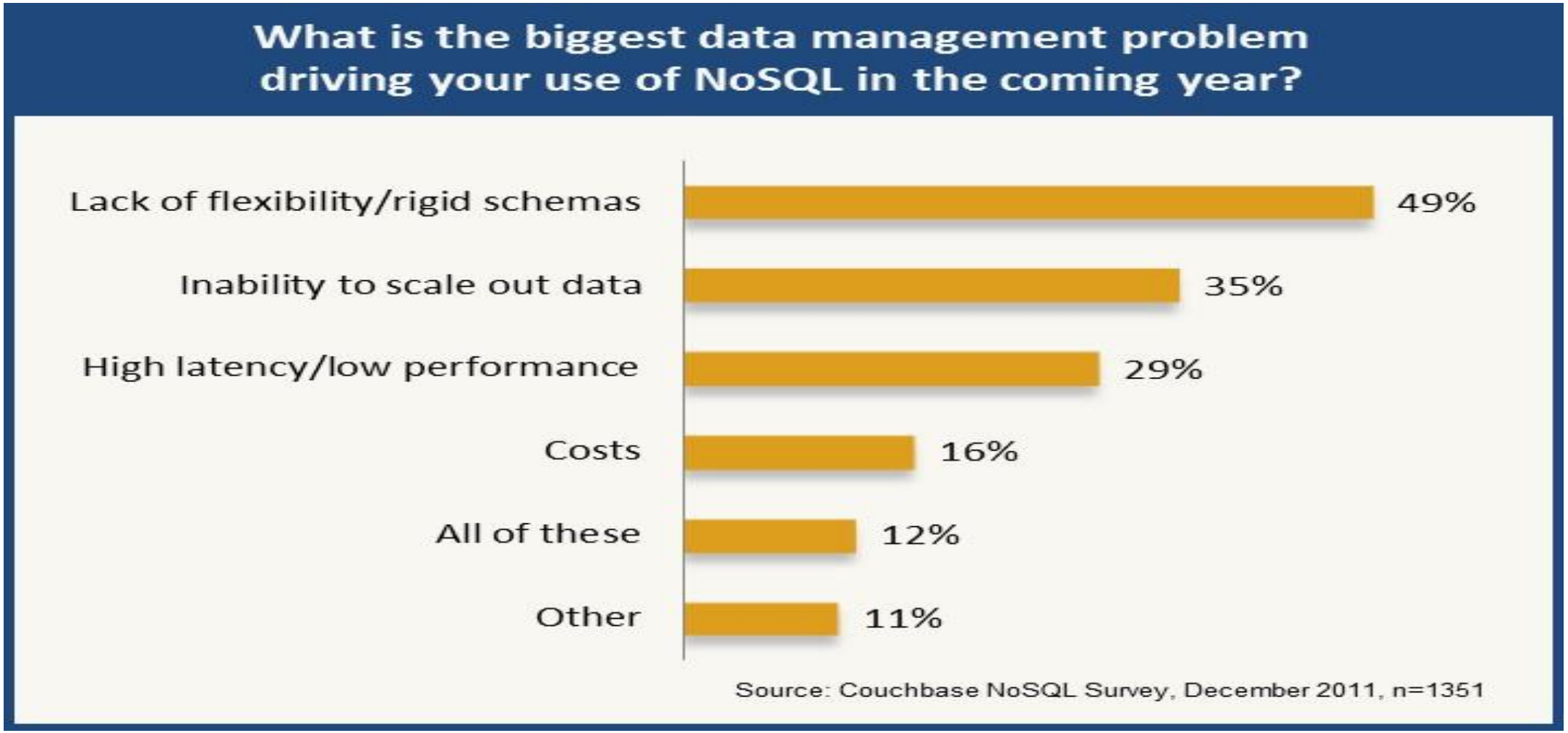
Comparison of NoSQL Database

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
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

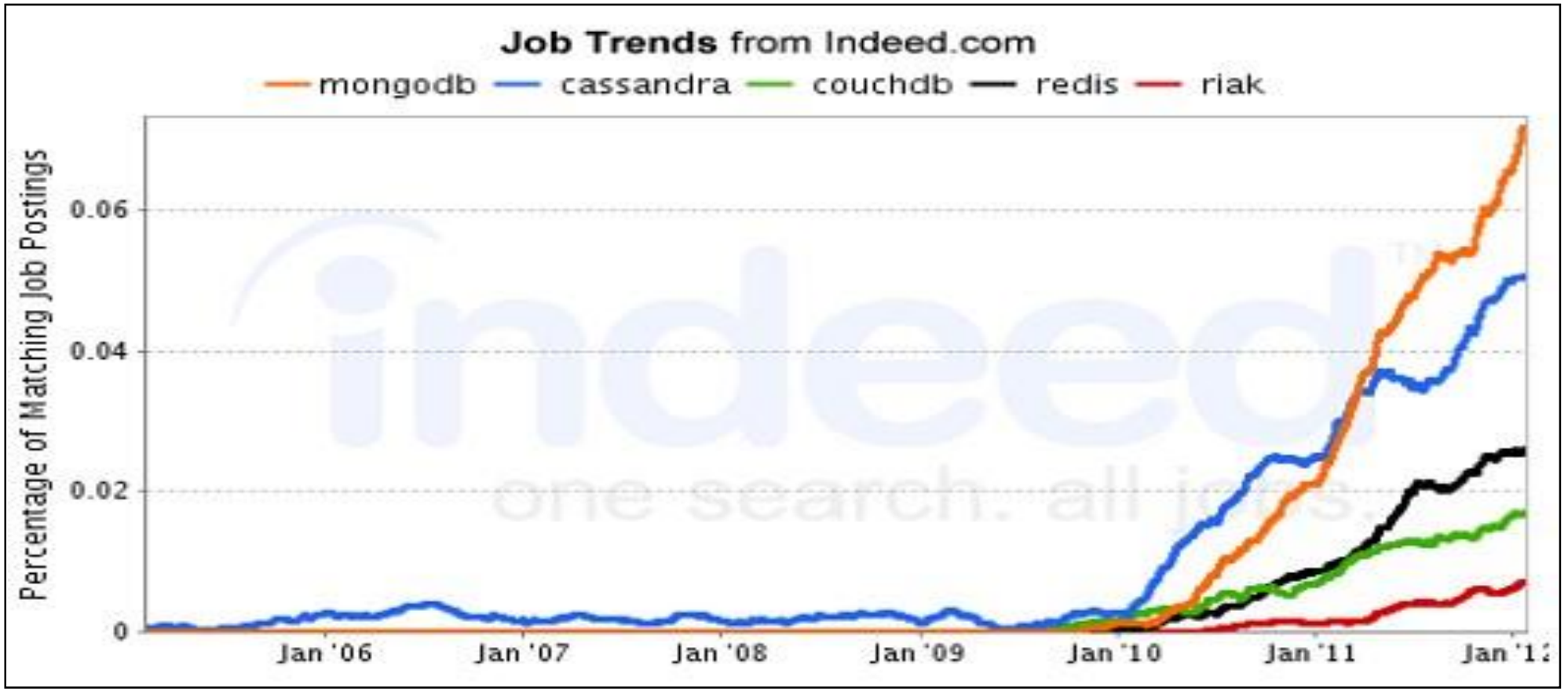
Comparison of NoSQL Database

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
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

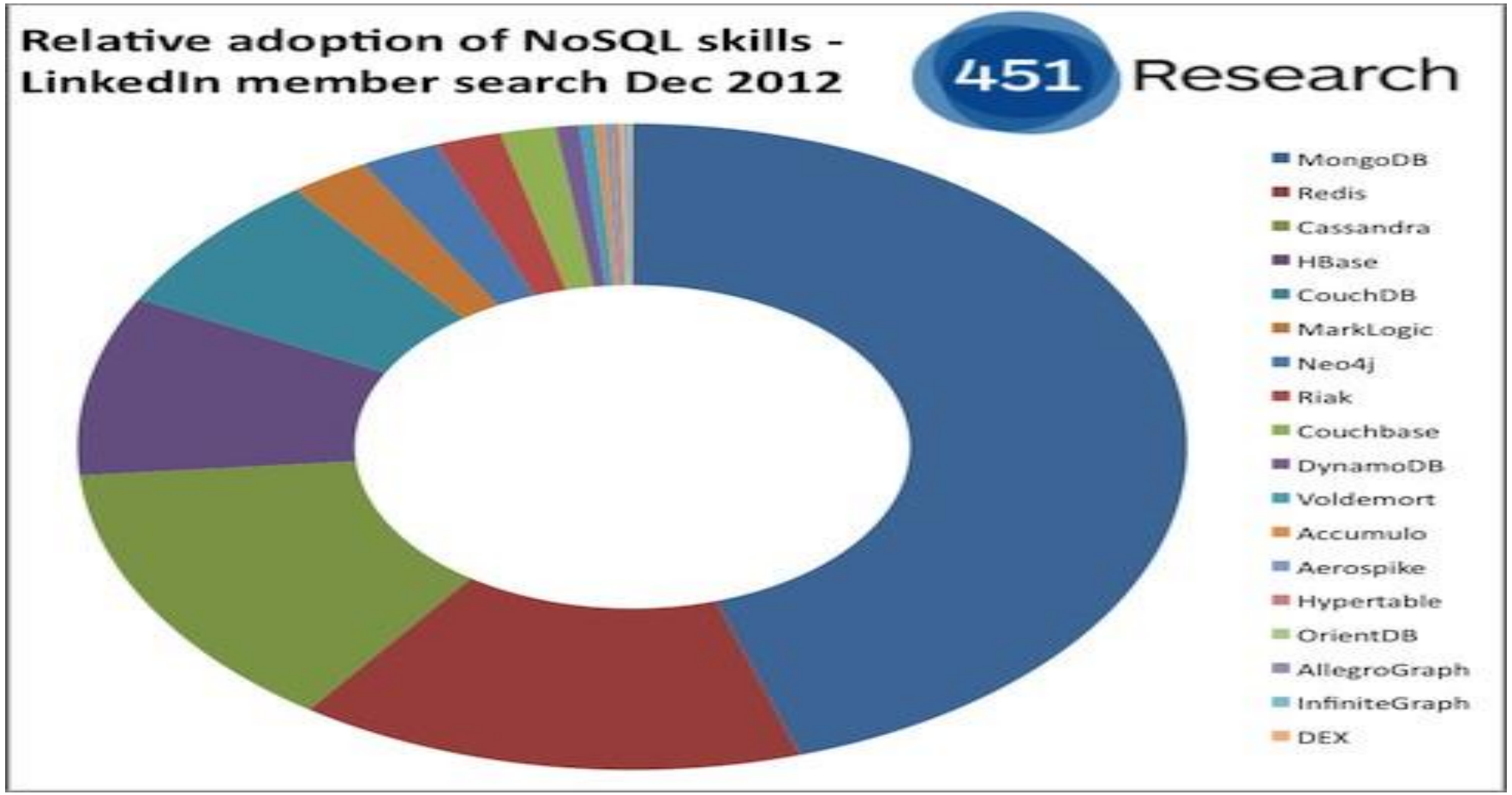
Adoption of NoSQL Database



Job trends of five NoSQL Databases (source: Indeed.com)



NoSQL LinkedIn Skills Index – December 2012



source: http://blogs.the451group.com

Summary

- Computational and storage requirements of applications led to the development of horizontally scalable, distributed non-relational No-SQL databases.
- Primary Uses
  - ✓ Large-scale data processing
  - ✓ Large volume data storage
  - ✓ Embedded IR (basic machine-to-machine information look-up & retrieval
- CAP Theorem
- Flexibility, scalability

References

- https://www.researchgate.net/publication/243963821_NoSQL_Database_New_Era_of_Databases_for_Big_data_Analytics_-_Classification_Characteristics_and_Comparison
- https://www.slideshare.net/mayureesrikulwong/nosql-database-classification-characteristics-and
- https://www.linkedin.com/pulse/20141021201313-156372715-vertical-scaling-vs-horizontal-scaling-big-data/

Conclusion

- Computational and Storage Requirement unhandled by sql-like centralized DBs
  - Big Data Analytics, Business Intelligence, Social-Networking (peta-byte datasets)
- NoSQL – horizontally scalable, distributed non-relational DBs
- Motivational understanding of various types of NoSQL DBs.

59