University of CINCINNATI

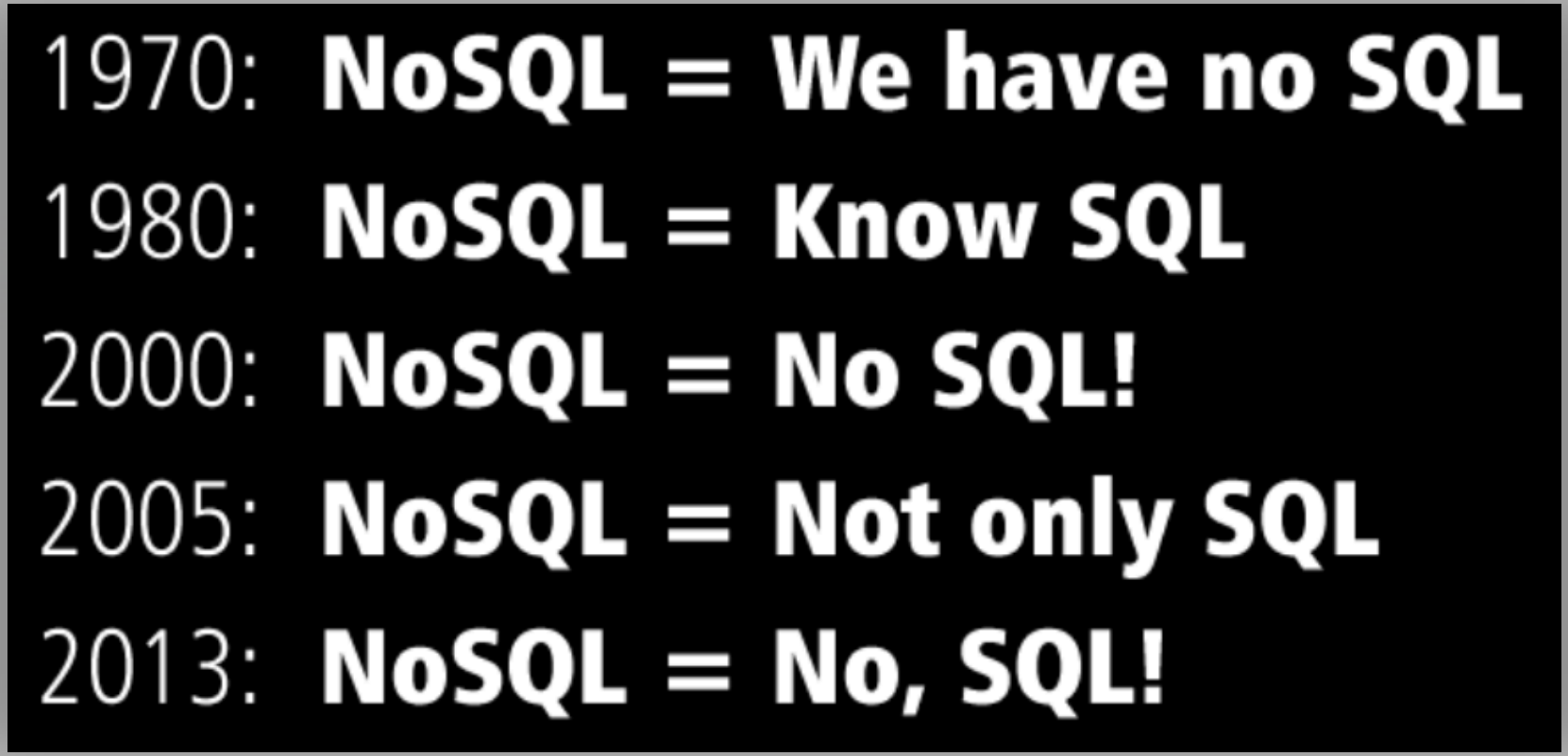
Cloud Computing

(SQL Vs NoSQL)

Rashmi Kansakar

History of database in **NO**-taition

Pass it on…
Knowledge is power

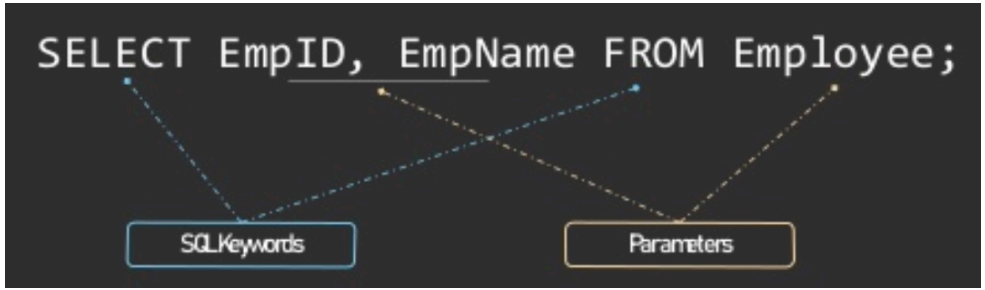


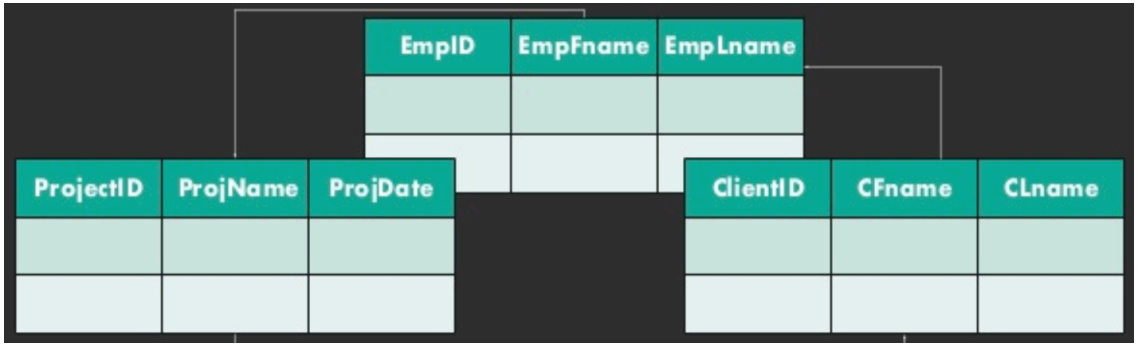
History

University of CINCINNATI

IBM introduced **SEQUEL** in 1970

- Structured English Query Language → SEQUEL
- Later SEQUEL → SQL





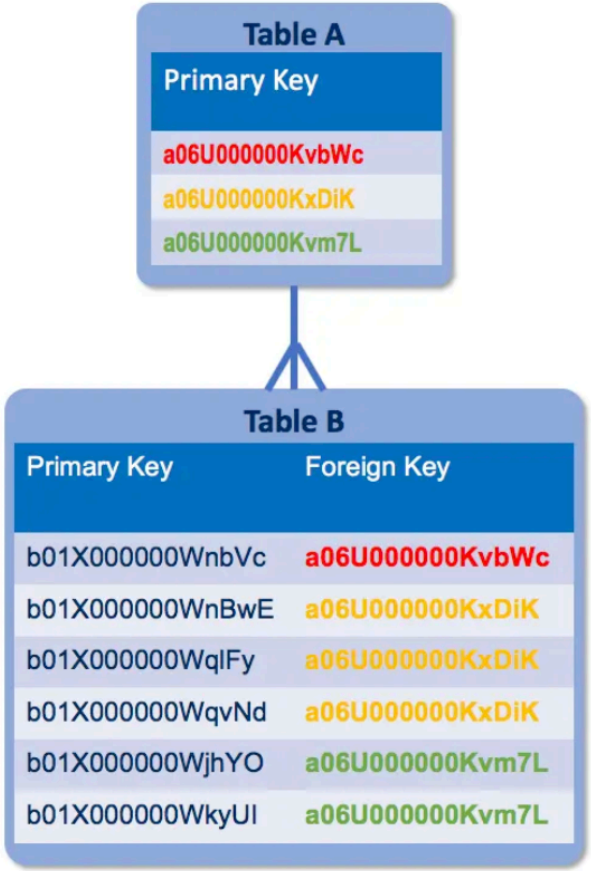
Relational Database Management System

- Main focus on ACID
  - **Atomicity**: Each transaction is atomic. If one part of it fails, the entire transaction fails (*and is rolled back*).
  - **Consistency**: every transaction is subject to a consistent set of rules (*constraints, triggers, cascades*).
  - **Isolation**: No transaction should interfere with another transaction
  - **Durability**: Once a transaction is committed, it remains committed.

Why ACID?

University of CINCINNATI

- ACID is important
  - Only when it's important.
  - Banking, Finance, System Safety etc.
- ACID adds overhead
  - Features like atomicity, isolation basically force database servers to use sequential evaluation.



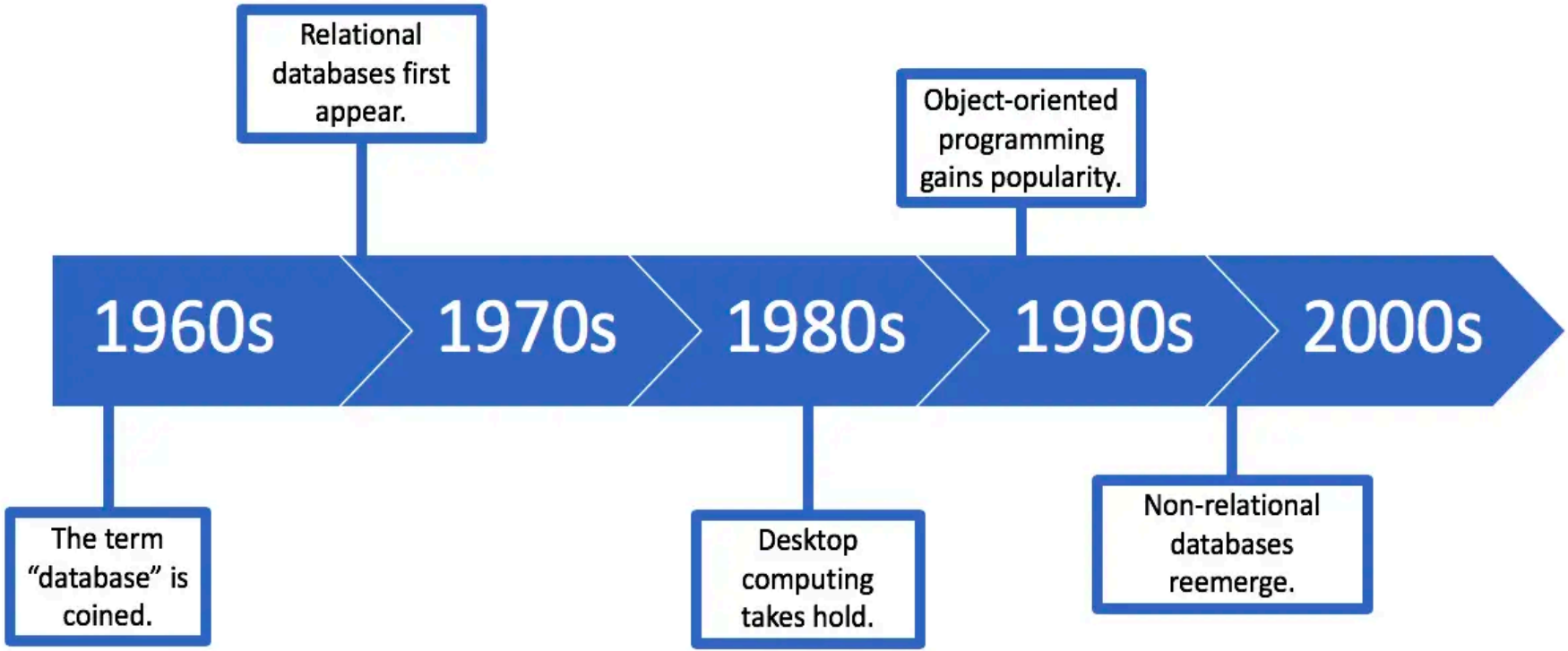
SQL Features

University of CINCINNATI

Other Features

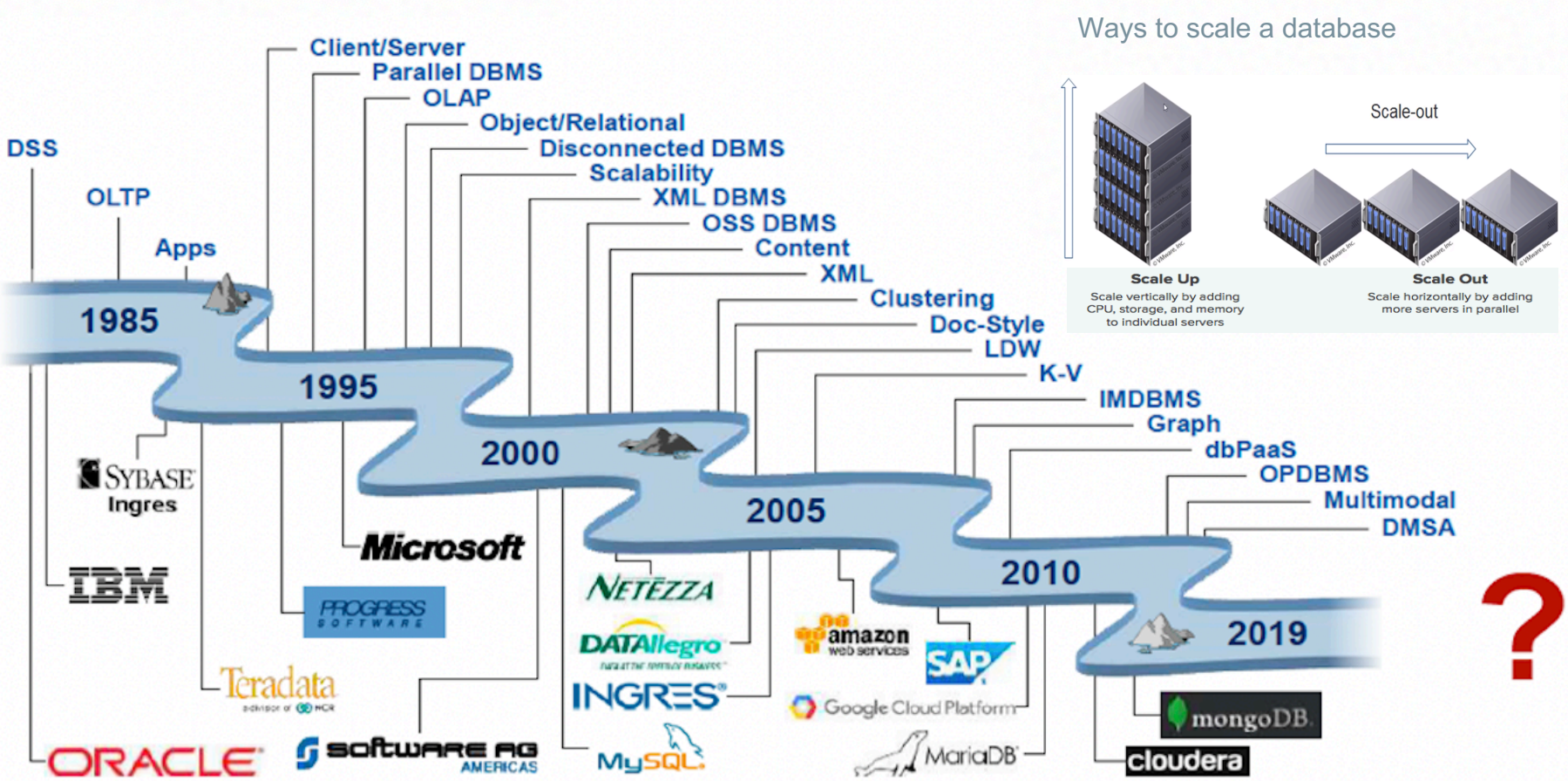
- Typed columns
- Normalized data
- Supports broad, unrestrained queries
- Has its own query optimizer
- Consistent way of accessing data (SQL)

History of SQL

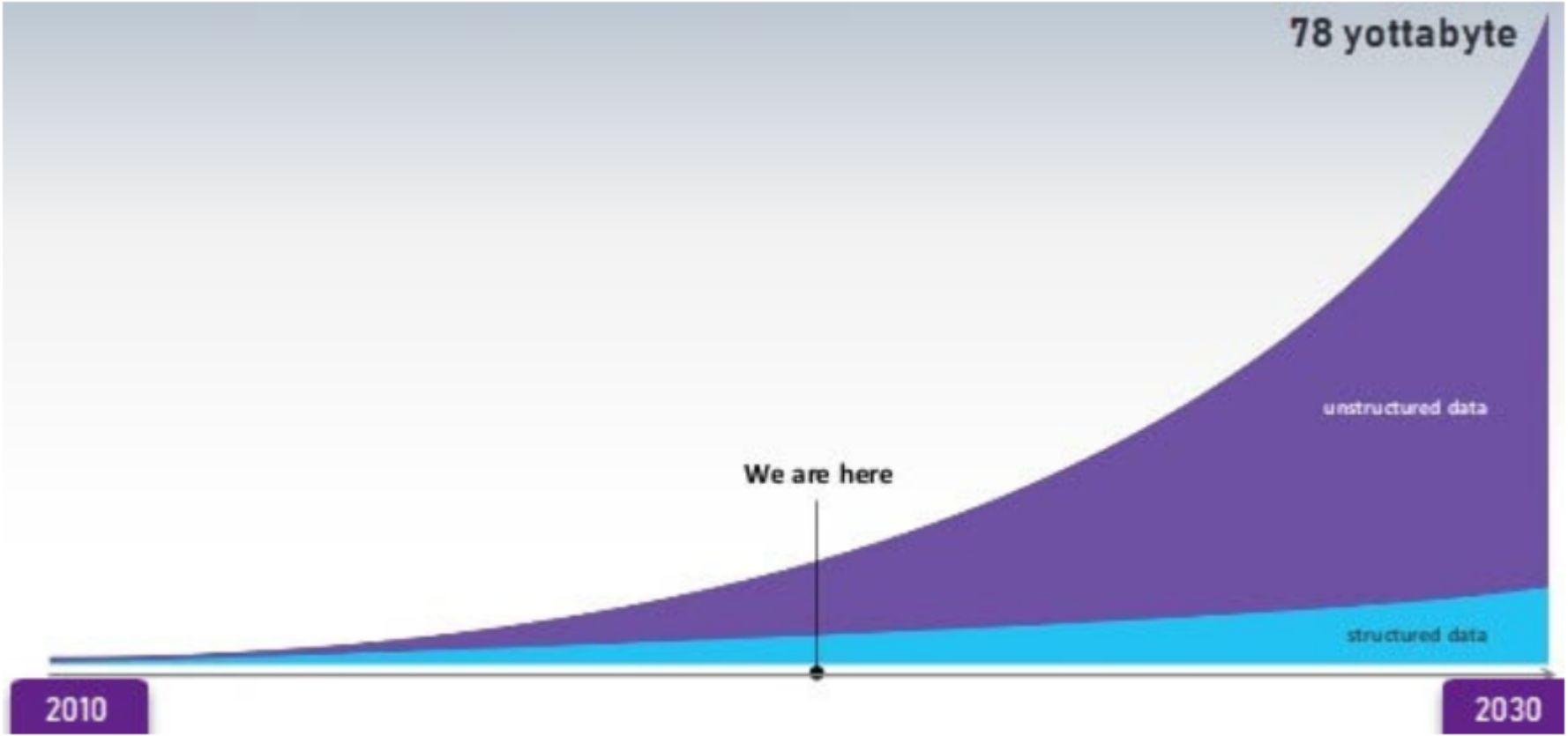


Evolving Databases for modern application

Pass it on… Knowledge is power

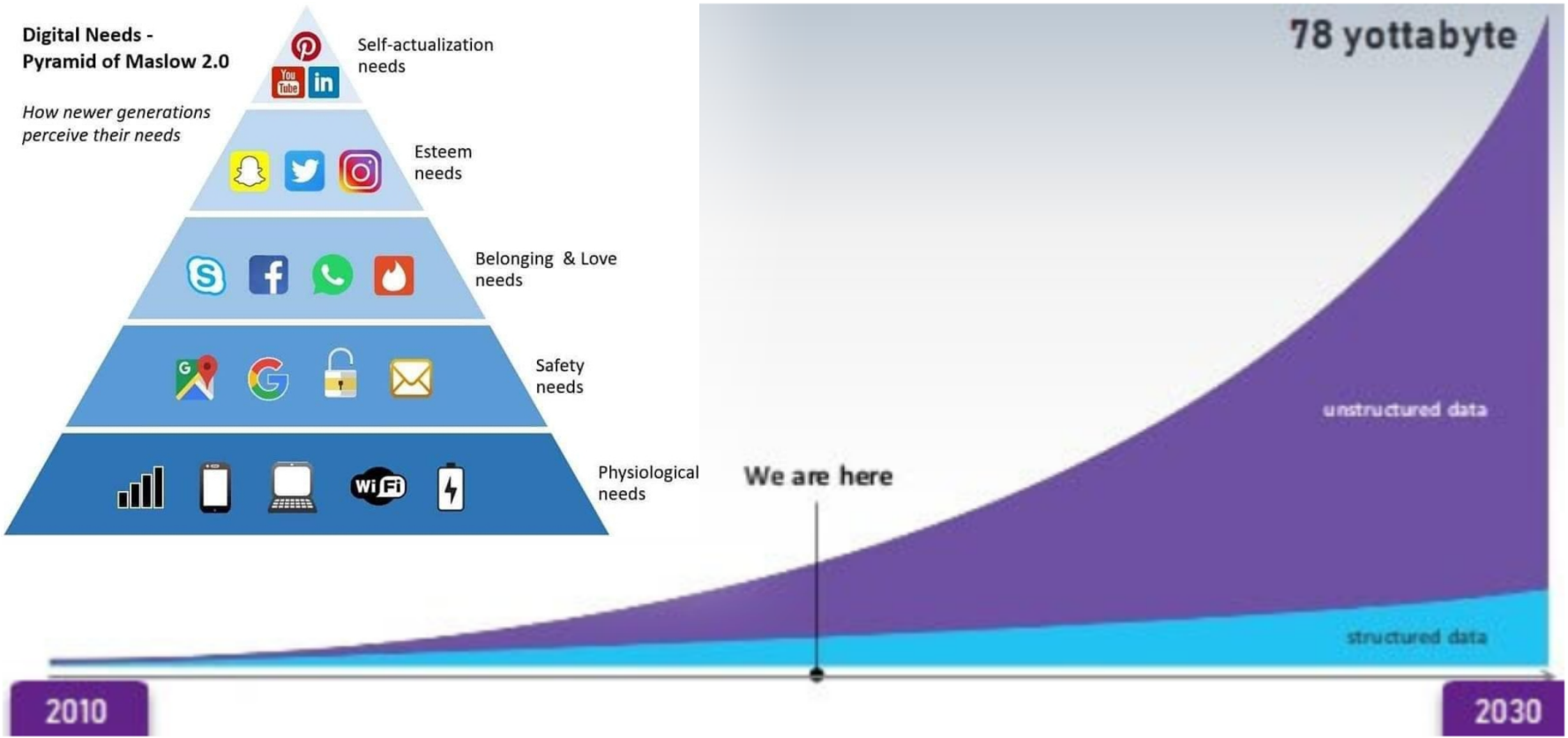


Data Growth over the years



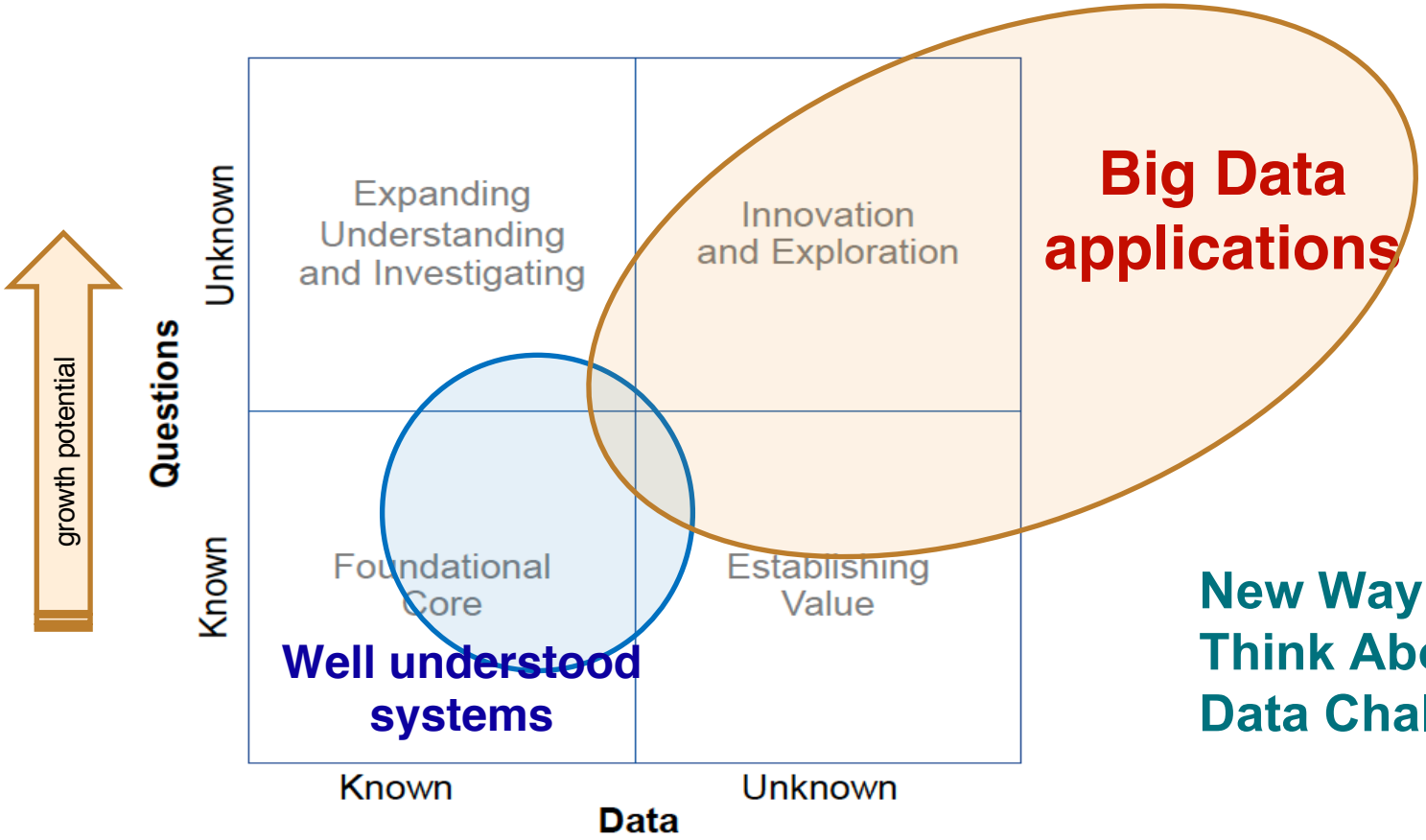
Data Growth over the years

Pass it on…
Knowledge is power



Known knowns, known unknowns, unknown unknowns.

Pass it on…
Knowledge is power

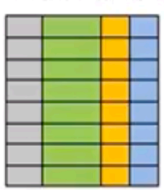


New Way to Think About Data Challenges

Types of Databases

SQL Databases

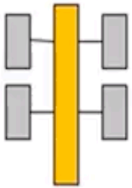
Relational



ORACLE

MySQL

Analytical (OLAP)

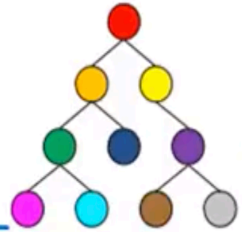


ORACLE

SAP

NoSQL Databases

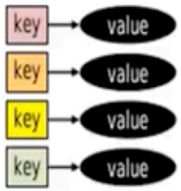
Document



mongoDB

Apache CouchDB relax

Key-Value

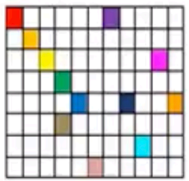


amazon DynamoDB

redis

riak

Column-Family



cassandra

APACHE HBASE

Graph

neo4j

When, Where & Why to Use NoSQL?

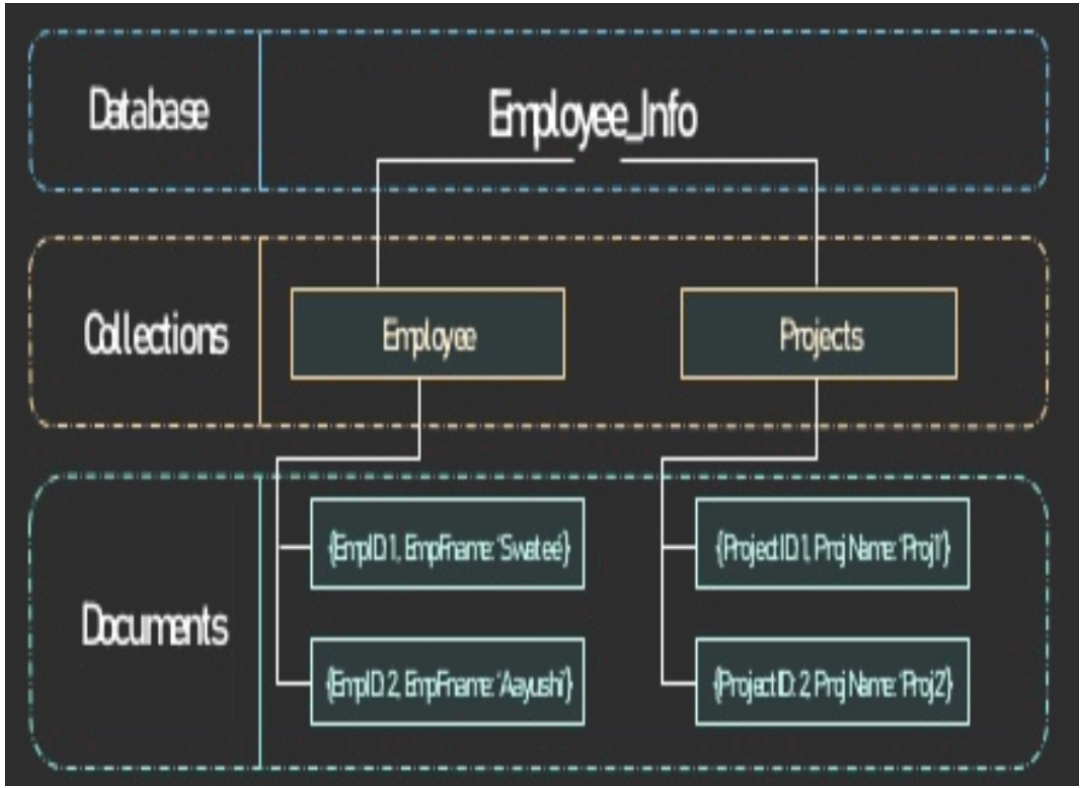
University of CINCINNATI

RDBMS doesn't quite fit for *some* requirements

- Azure Data Lake (ADLS gen2), Google (BigTable) and Amazon(S3) decided to make their own to meet their needs
  - Distributed
  - Scalability
  - Control over performance characteristics
  - High availability
  - Low latency
  - Cheap

Know NoSQL

- ✓ **NoSQL known as Not only SQL database.**
- ✓ **Provided mechanism for storage & retrieval of data.**
- ✓ **Next generation database.**
- ✓ **No specific schema & handle huge amount of data.**
- ✓ **Schema on Read vs Schema on Write**



What is NoSql?

University of CINCINNATI

- Basically a large serialized object store*
  - Retrieve objects by defined ID

- In general, doesn't support complicated queries*
- Doesn't have a structured schema*
  - Recommends denormalization

- Designed to be distributed (cloud-scale) out of the box
- Because of this, drops the ACID requirements
  - Any database can answer any query
  - Any write query can operate against any database and will "eventually" propagate to other distributed servers

***Dependent on vendor**

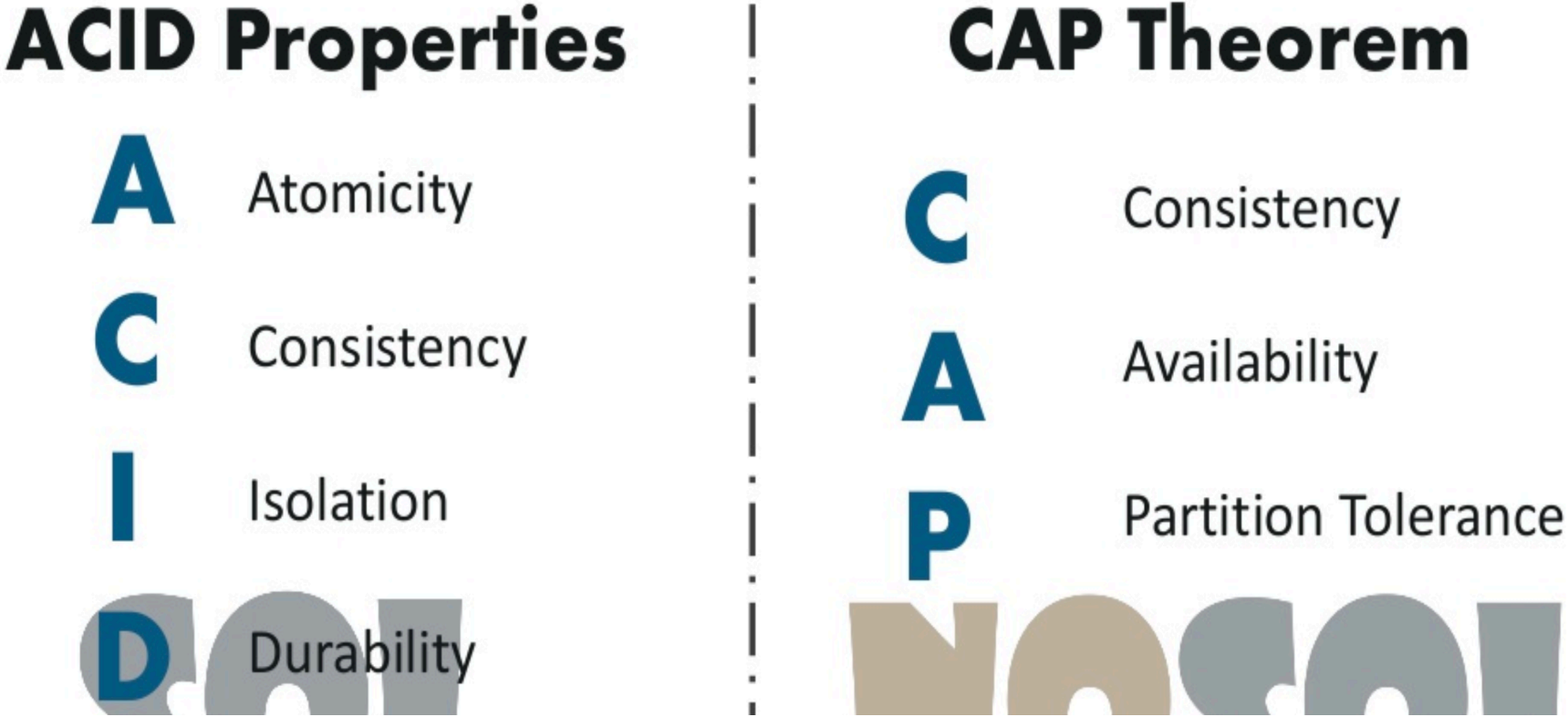
NoSQL / BASE

University of CINCINNATI

Opposite to ACID is BASE

- BASE
  - Basically Available → guaranteed availability
  - Soft state → state of system may change, even without a query (because of node updates)
  - Eventually Consistent → the system will become consistent over time

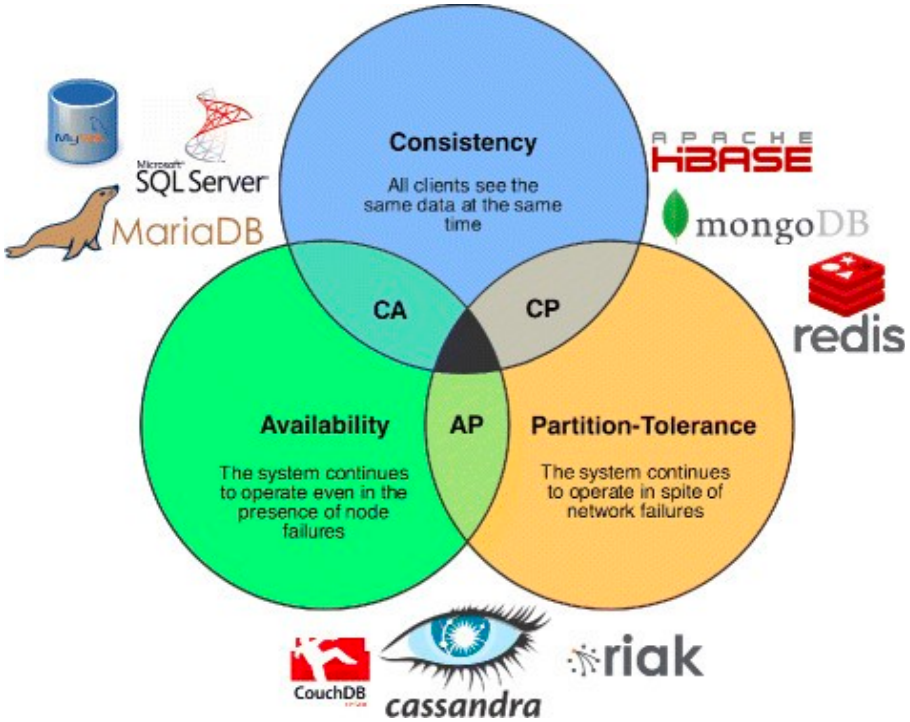
BASE Properties



CAP Theorem

University of CINCINNATI

Impossible for any shared data-system to guarantee simultaneously all of the following three properties

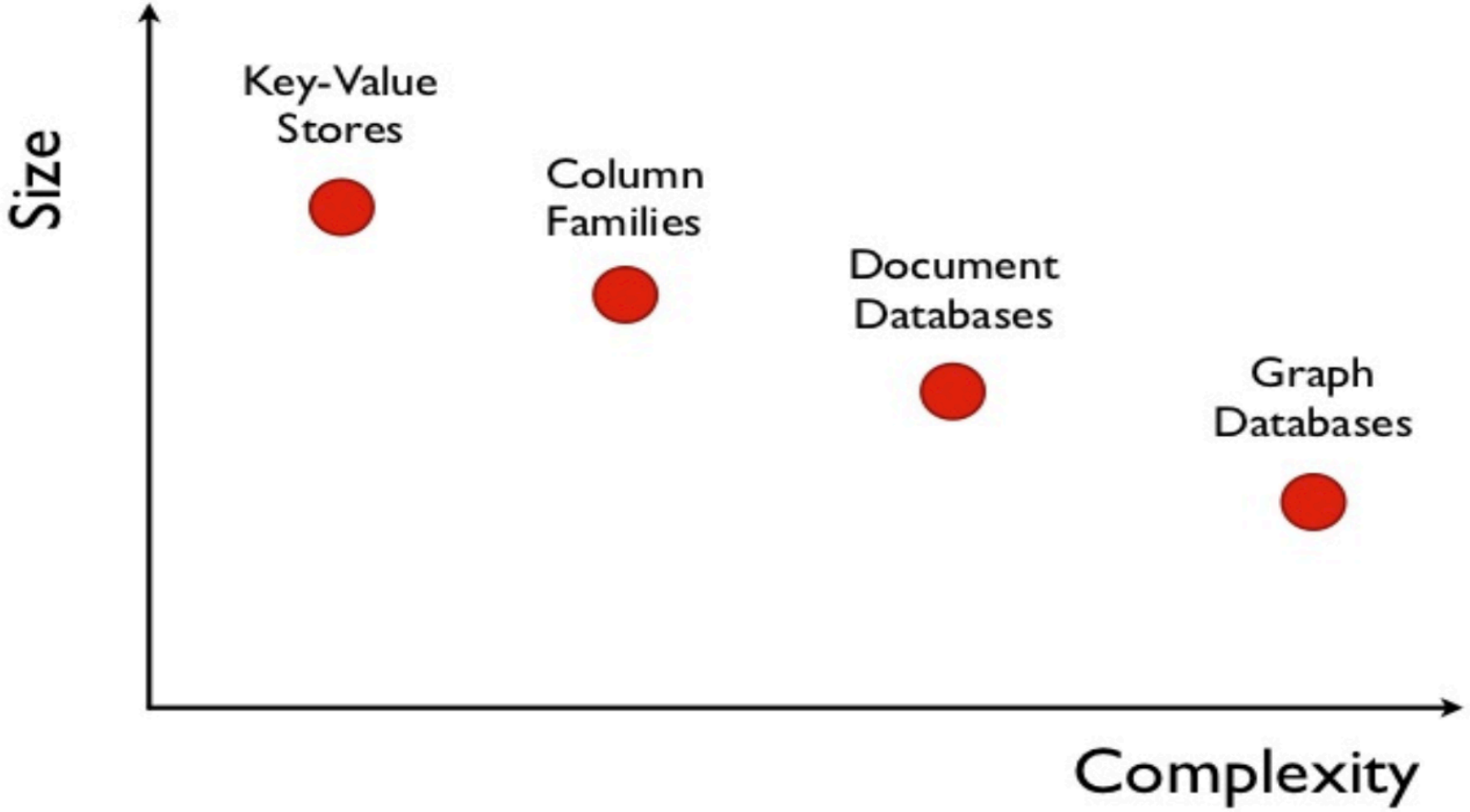


- Consistency: once data is written, all future read requests will contain that data
- Availability: the database is always available and responsive
- Partition Tolerance: if part of database is unavailable, other parts are unaffected.

NoSQL → Eventual Consistency

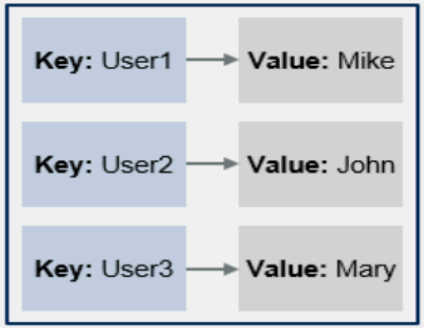
- Because of the distributed model, any server can answer any query
- Servers communicates amongst themselves at their own pace (*behind the scenes*)
- The server that answers your query might not have the latest data
- See a Facebook update from your friend delayed by some seconds may not make much difference

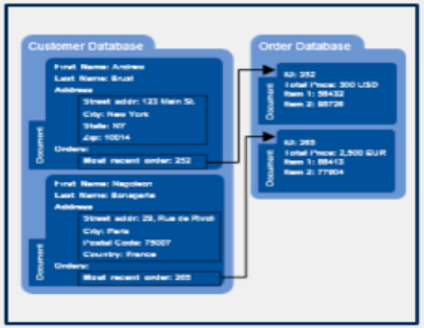
NoSQL Data models

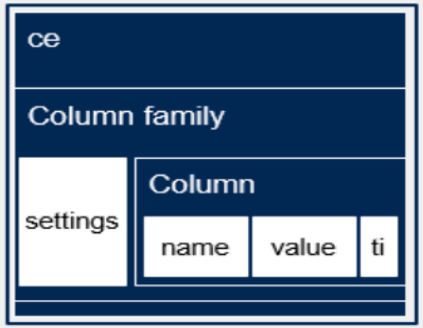


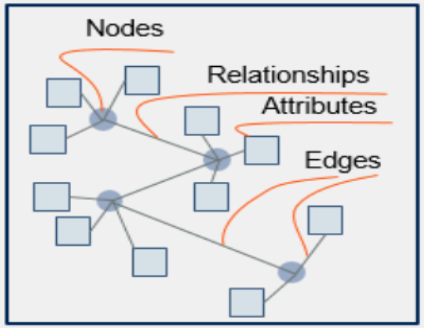
NoSQL Database types

University of CINCINNATI









Key-Value

- Records generally are stored in key-data format
- Examples:
  - Aerospike
  - Microsoft Azure Table storage
  - Amazon SimpleDB
  - Basho

Document

- Built to store document-centric data, such as JSON and/or XML documents
- Examples:
  - MarkLogic Server
  - Couchbase Server
  - Azure Cosmos DB
  - MongoDB
  - RavenDB

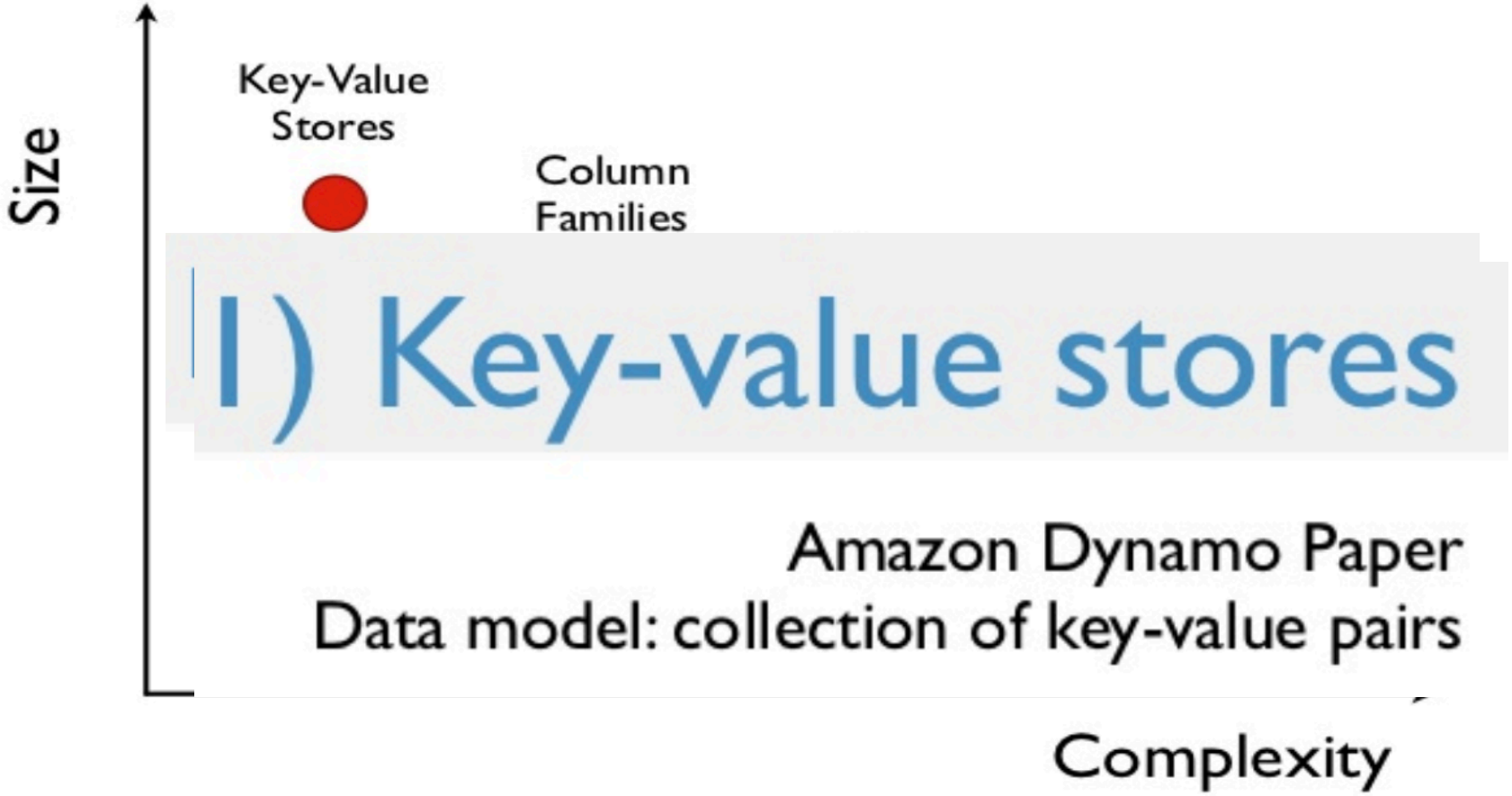
Table-Style

- Data model based on sparsely populated tables
- Examples:
  - Amazon DynamoDB
  - Apache HBase
  - DataStax Enterprise
  - Google Cloud Bigtable

Graph

- Accommodates graphs and relationship-centric data
- Examples:
  - Neo4j
  - Titan
  - Amazon Neptune

NoSQL Data models



Key-Value Data Store

- The simplest type of NoSQL database
- Uses key-value pairs in a hash table
- Unique key as pointer to a value
- Logical group of keys are called a "bucket"
- Large fallbacks with running queries. Most do not support update operations that modify only part of the data

Membase

DHT (K-V), no SPoF

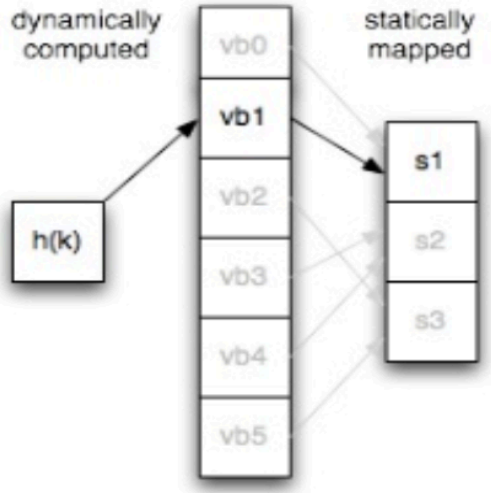
membase.org

LICENSE
Apache 2
LANGUAGE
C/C++
Erlang
API/PROTOCOL
REST/JSON
memcached

"VBuckets"

membase	memcached
persistence replication (fail-over HA) rebalancing	distributed in-memory

Unit of consistency and replication
Owner of a subset of the cluster key space



http://dustin.github.com/2010/06/29/memcached-vbuckets.html

h(k) → vb1 → s1

hash function + table lookup

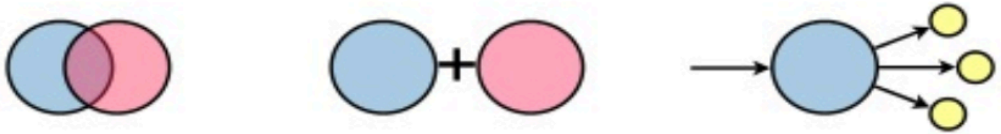
All metadata kept in memory (high throughput / low latency).
Manual/Programmatic failover via the Management REST API.

Redis

~~K-V store~~ "Data Structures Server"

redis

Map, Set, Sorted Set, Linked List
Set/Queue operations, Counters, Pub-Sub, Volatile keys



http://redis.io/presentation/Redis_Cluster.pdf

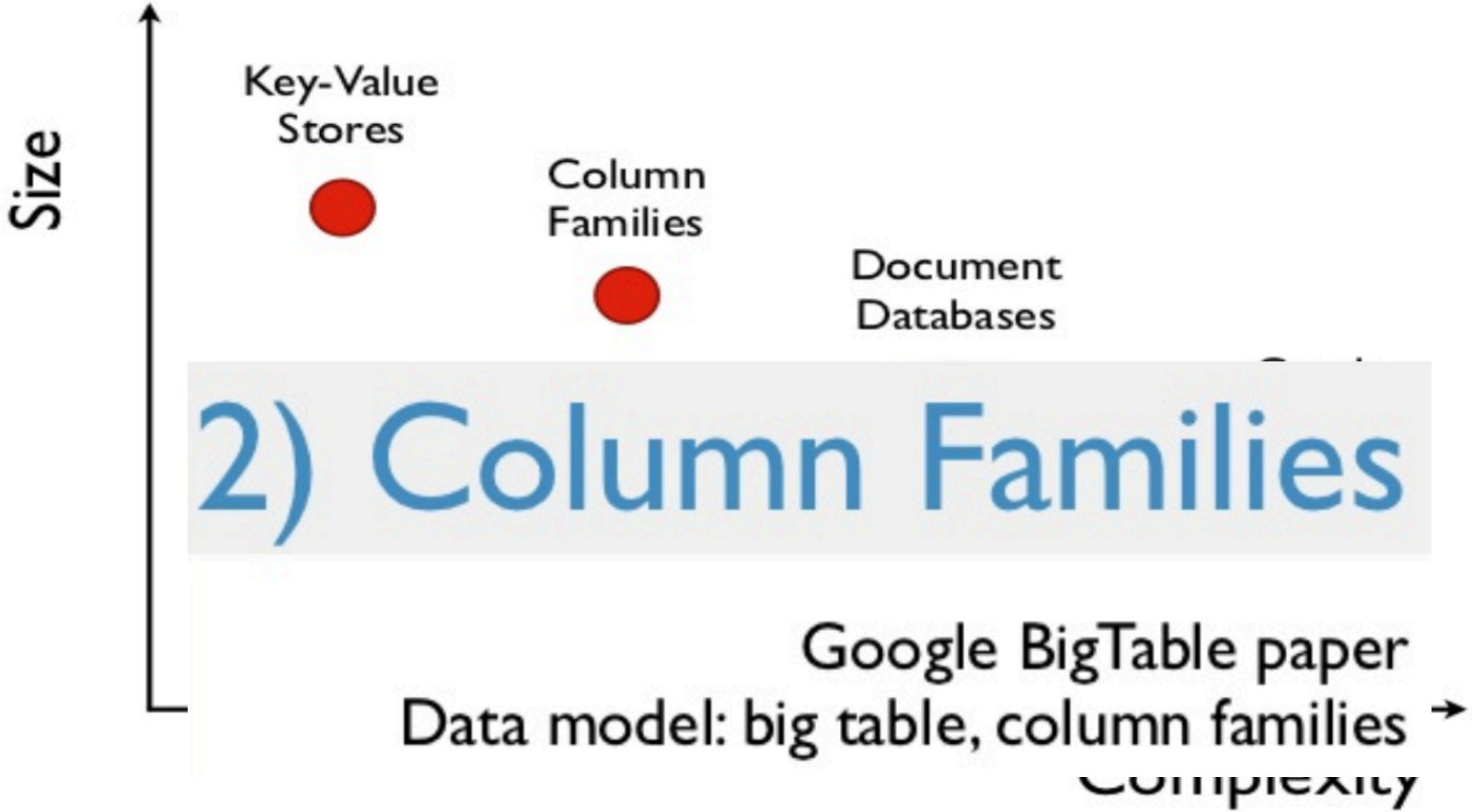
10-100K op/s (whole dataset in RAM + VM)

Persistence via snapshotting (tunable fsync freq.)

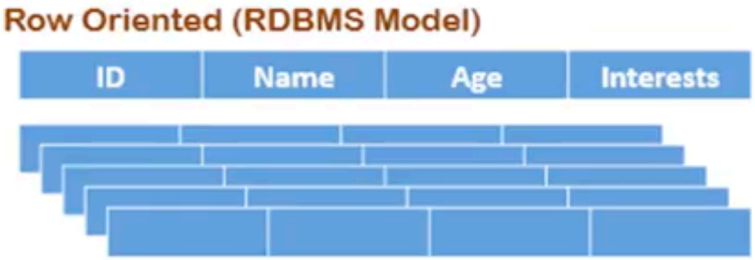
Distributed if client supports consistent hashing

LICENSE
BSD
LANGUAGE
ANSI C
API
*
PROTOCOL
Telnet-like
PERSISTENCE
in memory bg snapshots
REPLICATION
master-slave

NoSQL Data models



Column-Store Family Database



cassandra

APACHE HBASE

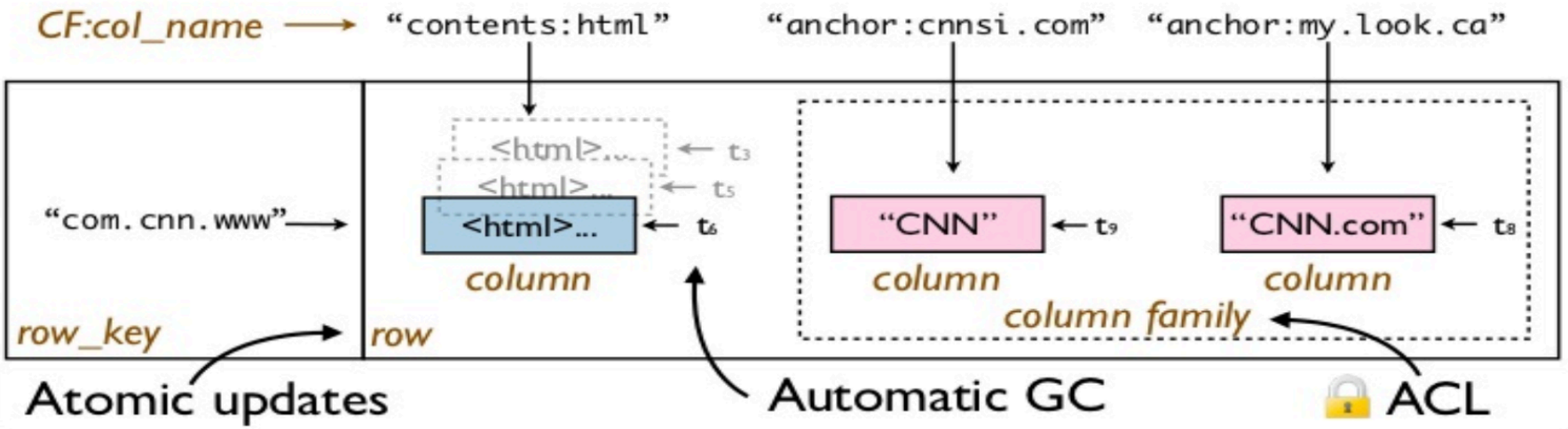
- Data is stored in cells that are grouped in columns of data rather than as rows of data
- Columns are grouped logically into **column families**. Column families can contain a virtually **unlimited** number of columns that can be created at **runtime** or the **definition** of the schema.
- These are generally for storing and processing very large amounts of data that is distributed across **many** machines.
- Faster access, search and data aggregation

Google BigTable Paper

Sparse, distributed, persistent multi-dimensional sorted map indexed by (*row_key, column_key, timestamp*)

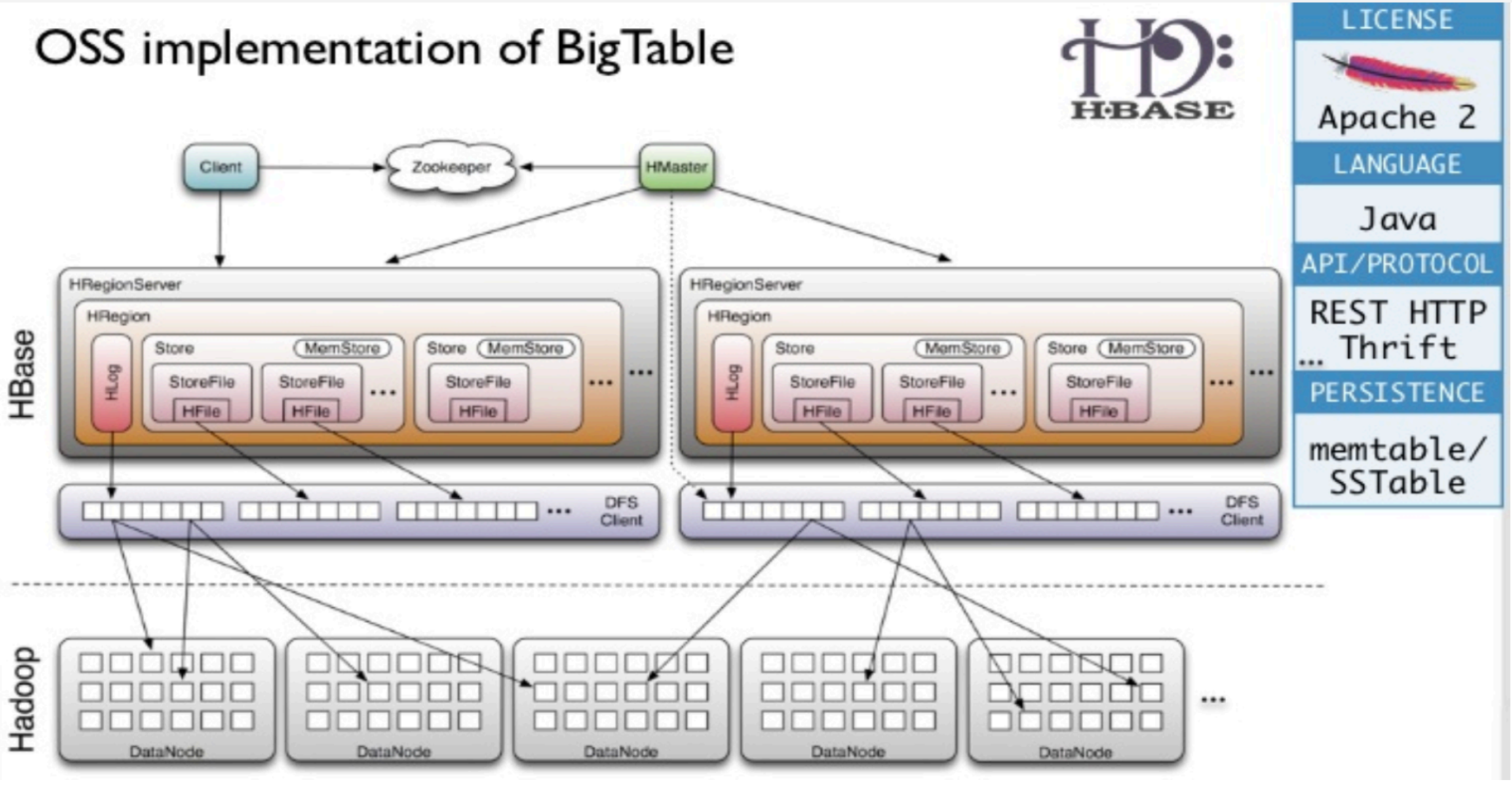
Google

http://labs.google.com/papers/bigtable-osdi06.pdf



HBase

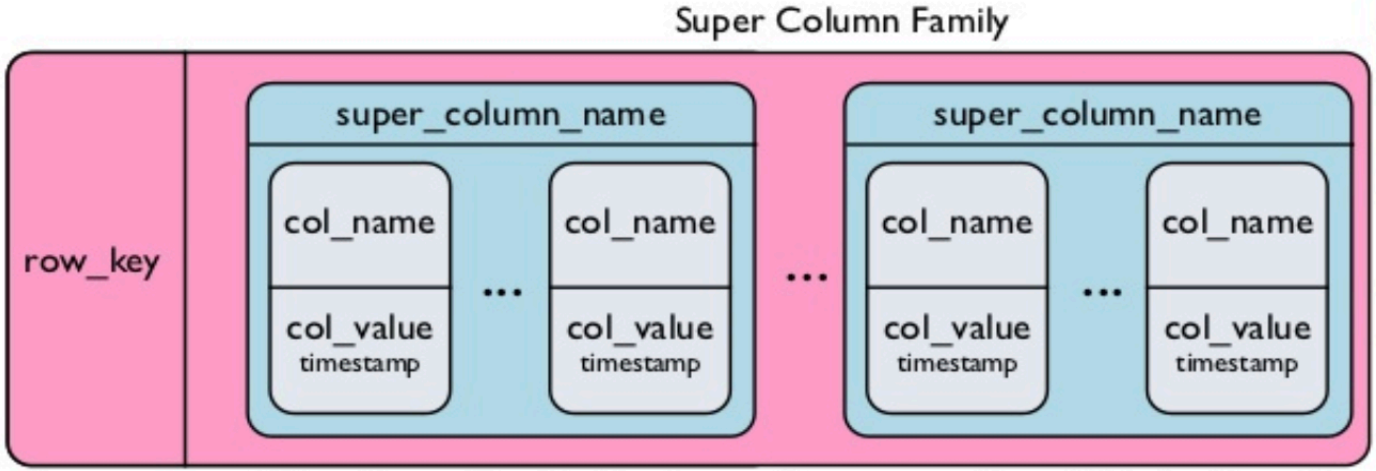
OSS implementation of BigTable



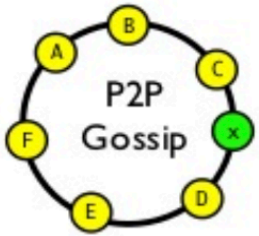
Cassandra

facebook

Data model of BigTable, infrastructure of Dynamo




```
keyspace.get("column_family", key, ["super_column",] "column")
```



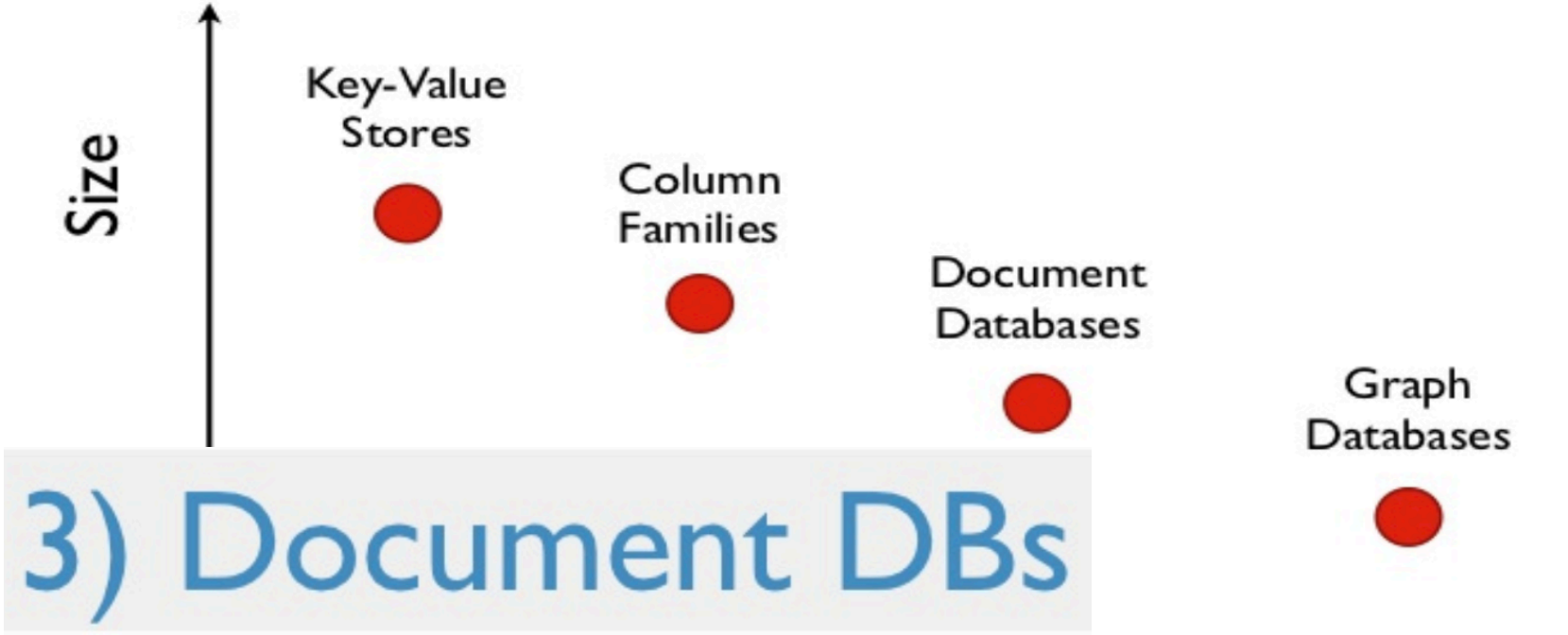
ALL
ONE
QUORUM

RandomPartitioner (MD5)
OrderPreservingPartitioner

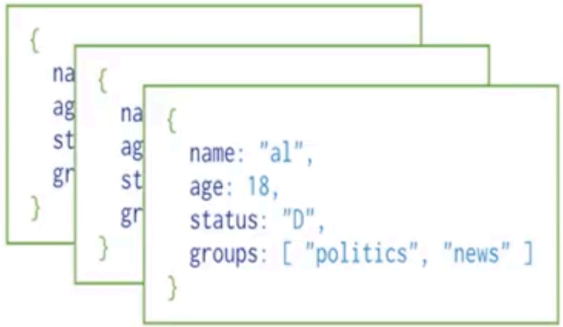
Range Scans, Fulltext Index (Solandra)

LICENSE
Apache 2
LANGUAGE
Java
PROTOCOL
Thrift Avro
PERSISTENCE
memtable/ SSTable
CONSISTENCY
Tunable R/W/N

NoSQL Data models



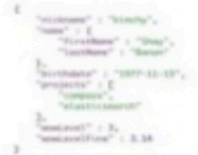
Document Database



- Uses key value pairs compressed as a document store similar to a key-value store, but values stored ("documents") provide some structure and encoding of the managed data.
- Can use multiple formats with document databases – **JSON, XML, BSON**
- Can perform nested queries and can do much more with your data when compared to a key-value store.
- **MongoDB** and **Couchbase** are the most popular document databases at this time

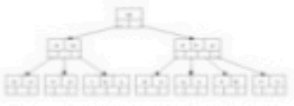
CouchDB

http://horicky.blogspot.com/2008/10/couchdb-implementation.html

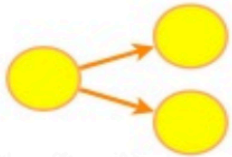


JSON docs

map-reduce "views"
(materialised resultset)

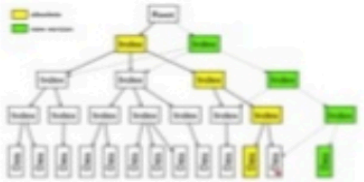


Storage + View Indexes (B+Tree)
[by_id_index, by_seqnum_index]



Replication used
as a way to scale
transactions volume

Conflict Resolution
at *application* level



MVCC (copy-on-modify)
Volatile Versioning

Online Compaction
(very primitive VACUUM)

Update validation /
Auth triggers

Delayed commits
(write performance)

LICENSE
Apache 2
LANGUAGE
Erlang
API/PROTOCOL
REST/JSON
PERSISTENCE
Append-only B+Tree
CONCURRENCY
MVCC
CONSISTENCY
crash-only design
REPLICATION
multi master

MongoDB

mongoDB

bson_encode()
bson_decode()

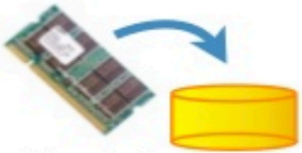
BSON serialisation
(storage & transfer)

Auto-Sharding,
Master-Slave,
Auto-Failover

B-Tree Indexes
(on different cols too)

Update in place
(no versioning, no append-only log)

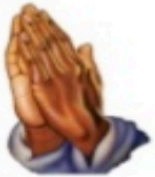
Geo-Spatial Indexes



Persistence via
Replication +
Snapshotting

GROUP BY

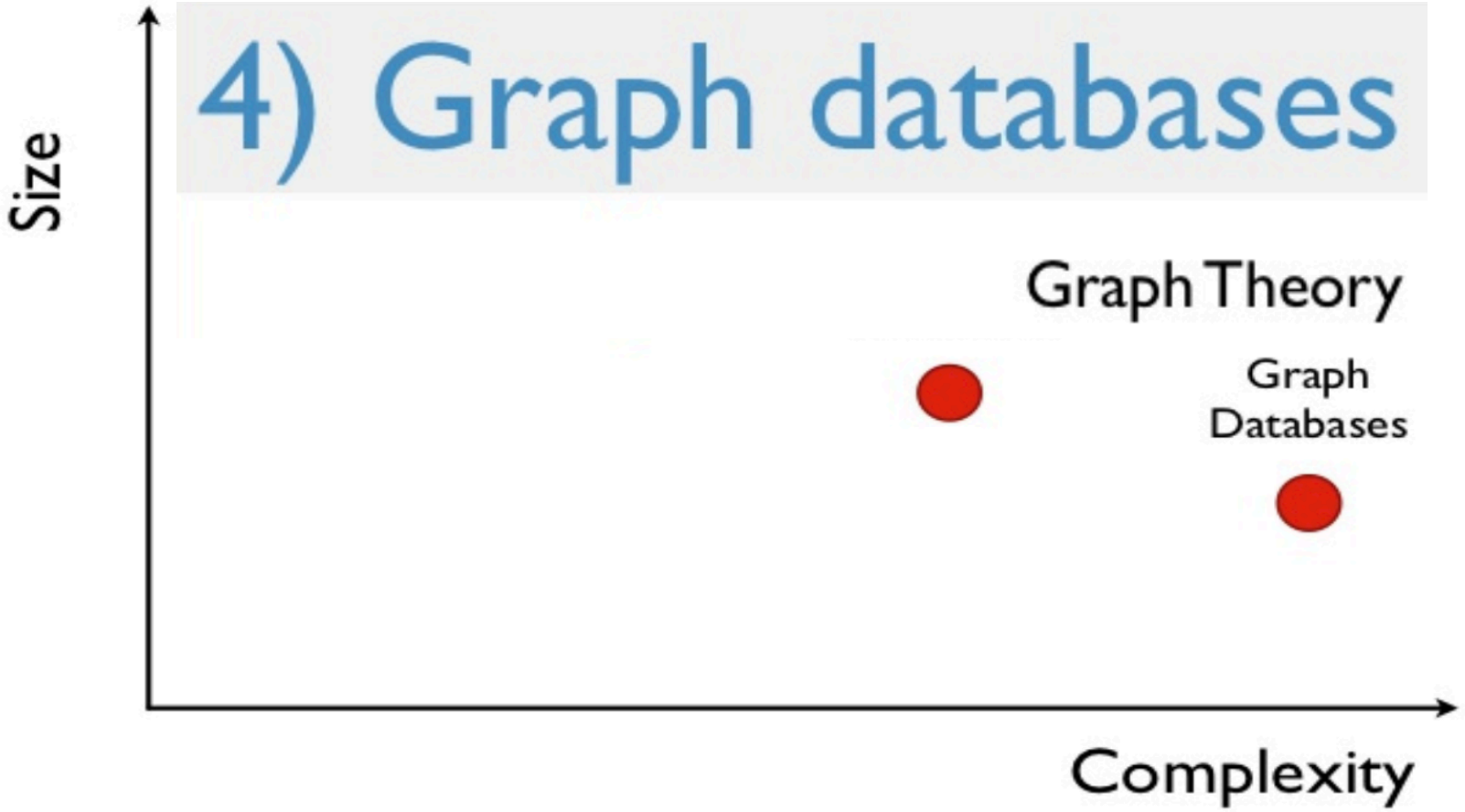
Map/Reduce
(well, aggregation)



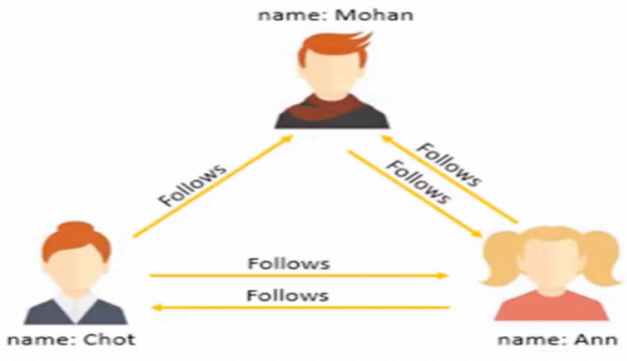
No ACK on Updates
(or ensure N replicas)

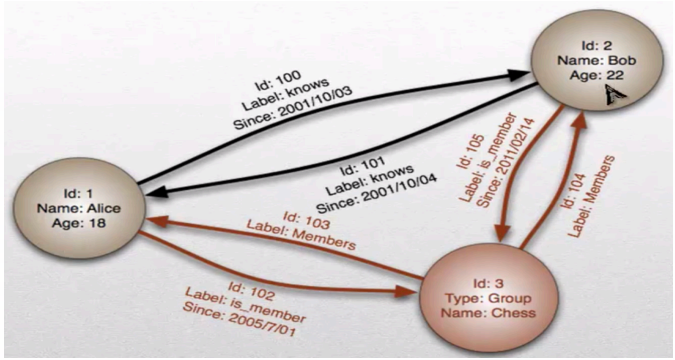
LICENSE
AGPLv3
LANGUAGE
C++
API/PROTOCOL
REST/BSON *
PERSISTENCE
B+Tree, Snapshots
CONCURRENCY
In-place updates
REPLICATION
master-slave replica sets

NoSQL Data models



Graph Database





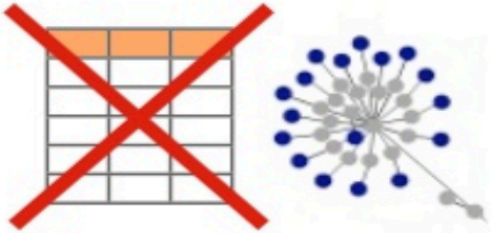
neo4j

- Based on "**graph theory**". Uses a flexible graphical representation as oppose to rigid tables and columns
- Uses "**edges**" and "**nodes**" instead of key-value pairs. This creates index-free adjacency
- Often used to store information about networks, such as social connections
- Compared to **RDBMS**, graph databases are usually faster for associative data sets and map more directly to the structure of object-oriented applications

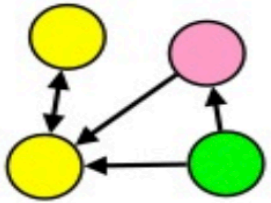
Neo4j

Neo4j
the graph database

http://docs.neo4j.org/chunked/stable/ha-how.html



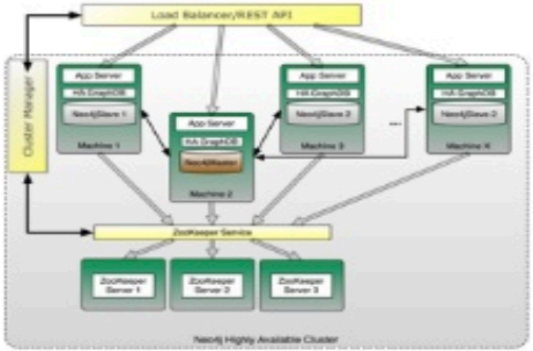
Graph Data Structure



Nodes,
Relationships,
Properties on both

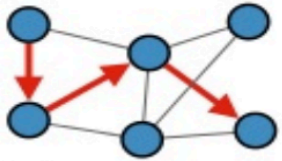
Vertical Scalability
(1000s times faster, but not distributed)

LICENSE
AGPLv3 / Commercial
LANGUAGE
Java
API/PROTOCOL
REST Java SPARQL
PERSISTENCE
On-disk linked-list



HA cluster with ZooKeeper
(nodes = exact replicas)

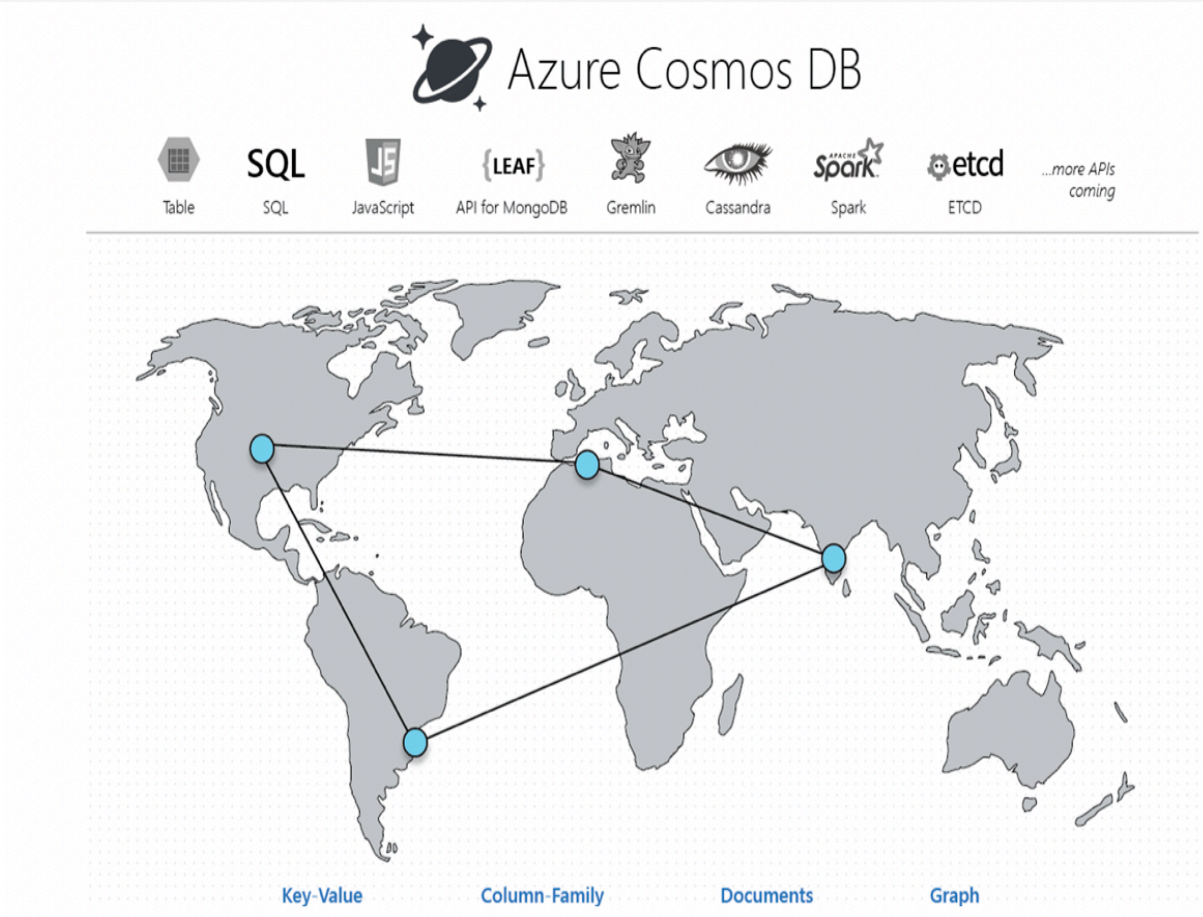
Physical structure:
LinkedList stored on disk



High-performance
node traversal

SPARQL

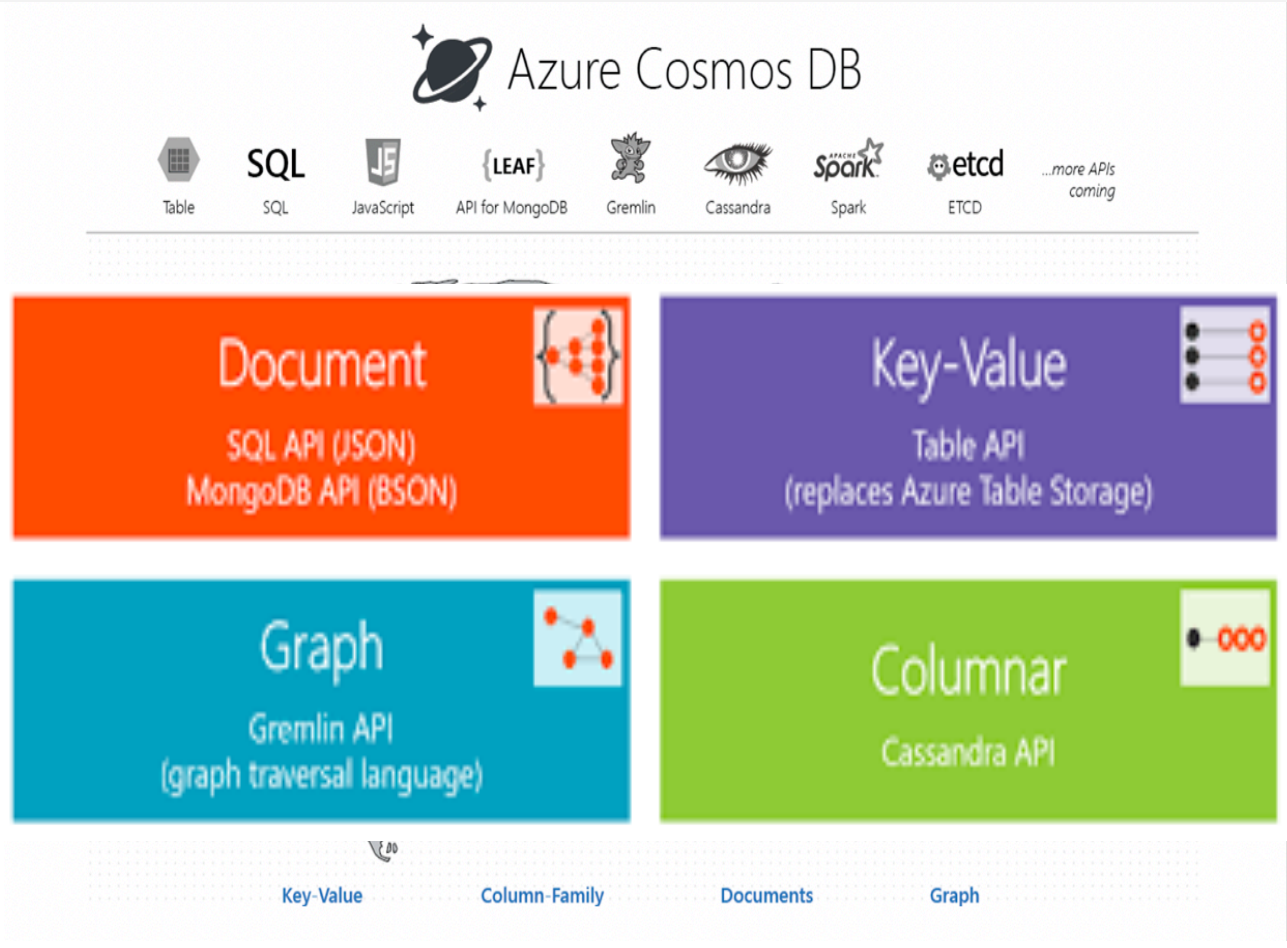
Azure Cosmos DB: multi-model DB



Microsoft's globally distributed multi-model database service on cloud

globally distributed multi-model DB

Pass it on… Knowledge is power



Microsoft's globally distributed multi-model database service on cloud

Azure Cosmos DB

... *more coming soon*

Global distribution | Elastic scale out | Guaranteed low latency | Five consistency models | Comprehensive SLAs

A globally-distributed, multi-model database service

Cosmos DB: Key Features

- **Global distribution** - transparently replicates your data, geo-redundancy, multi-master writes
- **Always On** – provides 99.999 high availability SLA
- **Elastic scalability –** transparent horizontal partitioning and multi-master replication, can scale storage and throughput separately
- **Predictable low latency:** latch-free and wire optimized engine, <10ms
- **5 consistency models:** strong, bounded staleness, session, consistent prefix, and eventual
- **Multi-model:** key-value pairs, Column family, Document and Graph
- **Automatic indexing** - not need any schema or any secondary indexes
- **Secure and enterprise ready –** data encrypted at rest and in motion, certified for a wide range of compliance standards

Comparison SQL vs. NoSQL

RDBMS	NoSQL
Structured Data	Semi-structured and Unstructured Data
Scale Up (Vertical Scalability)	Scale Out (Horizontal Scalability)
Strict Schema	Schema-free or Dynamic Schema
ACID Compliance	Eventual Consistency
Custom High-Availability	Auto High-Availability

SQL & NoSQL

Relational World

STORE

ID	Branch	Type	Count	SalesAssistant	Product
2313	Boston	Shoes	36	324	756
2654	Worcester	Shoes	39	354	567
6546	Weymouth	ShoeCareProducts	127	654	445

CUSTOMER

ID	GivenName	FamilyName
324	Jack	Bauer
354	Tony	Stark
654	Charles	Xavier

PRODUCT

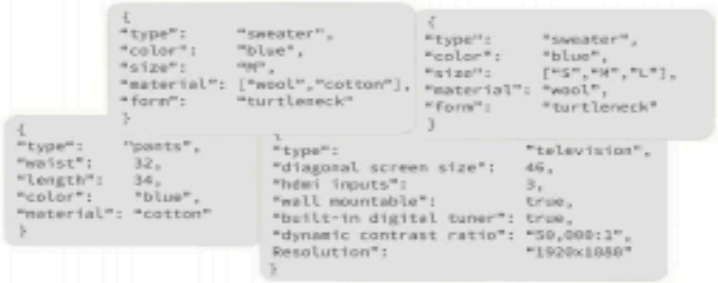
ID	Color	Size
756	red	20-28
567	blue	30-41
787	brown	36-48

TRANSACTION

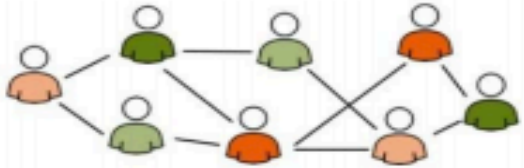
ID	Type	LeatherType
445	Cream	Velour
676	Brush	Smooth leather
987	Spray	all

NoSQL World

Document - JSON



GRAPHS



KEY VALUES

Not all DB type Are equal. Right DB for Workload Type

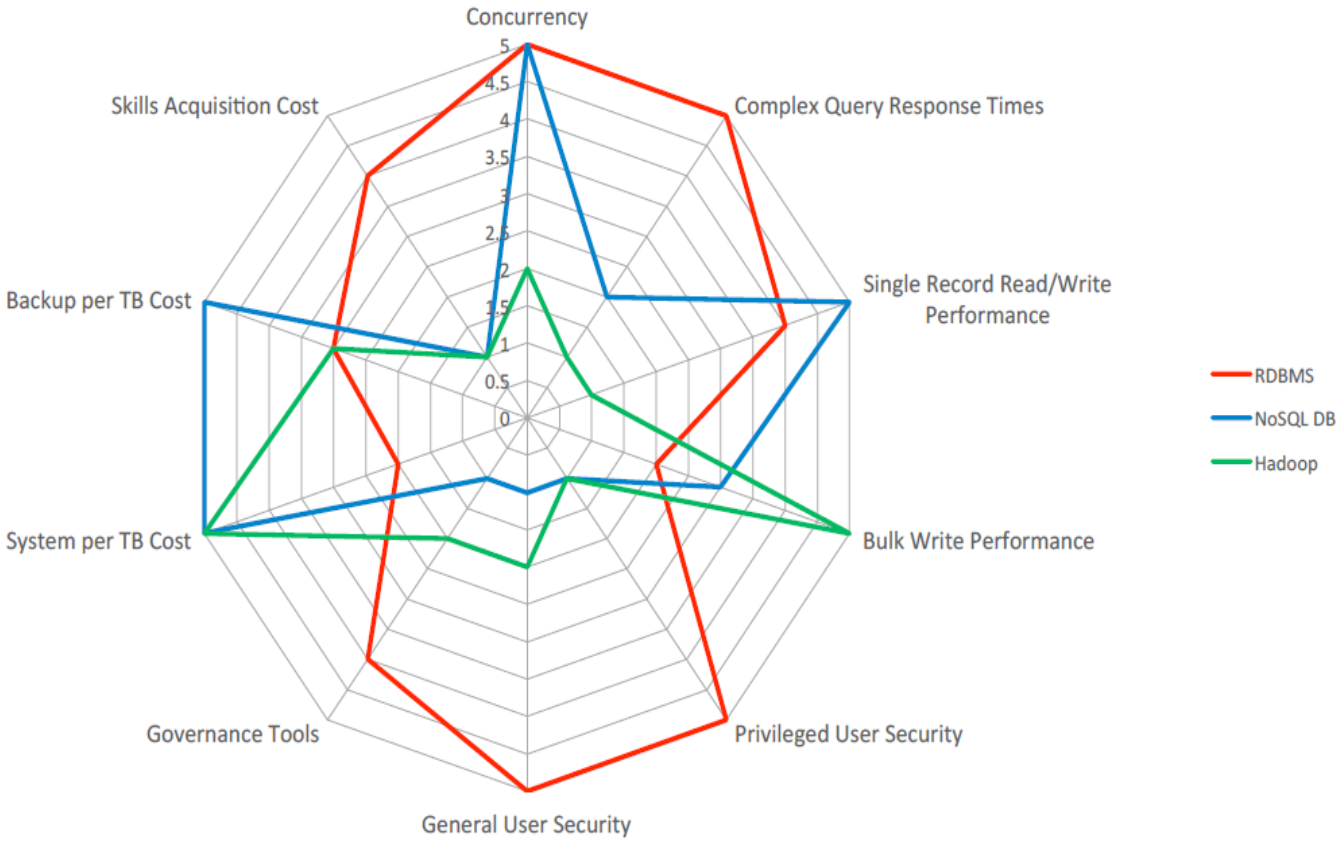
Comparison SQL, NoSQL, Hadoop

Pass it on…
Knowledge is power

Performance

Security

Cost



No one-size-fits-all

Pass it on…
Knowledge is power

- Polyglot Persistence

- No one-size-fits-all

Server Databases

MySQL
PostgreSQL
MarkLogic
Netezza
Hadoop
CouchDB
Greenplum
Vertica
Neo4J
SimpleDB
Drizzle
Cassandra
Riak
Voldemort
Hypertable
MariaDB
Redis
MongoDB
RethinkDB
OrientDB
~~Xeround~~
FlockDB
RavenDB
Clustrix
Membase
Translattice
NimbusDB/NuoDB
Citrusleaf/Aerospike
DynamoDB
Datomic
MemSQL
HyperDex
TokuDB
GenieDB
FoundationDB
Apollo
Cayley

1994 1995 2000 2003 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

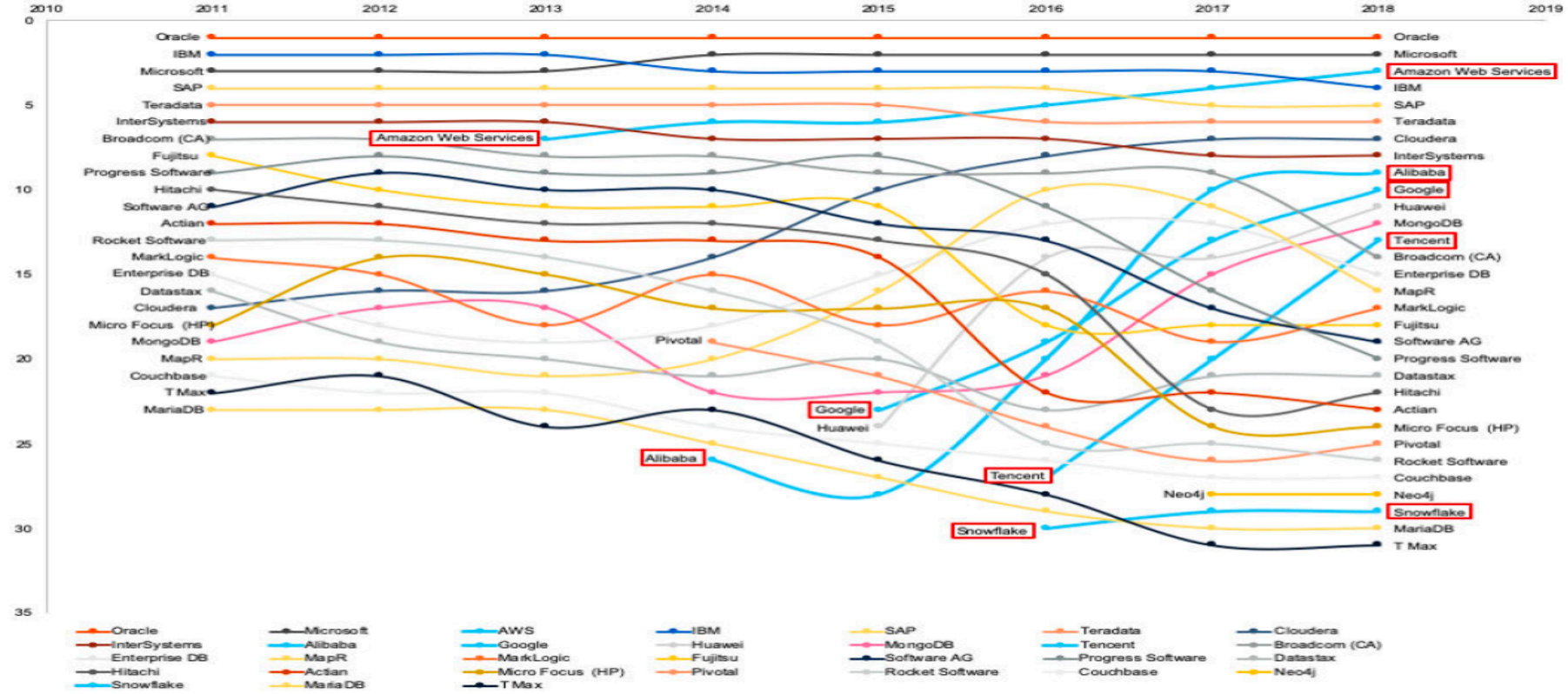
Arranged by date of first public release (source: Wikipedia)

Mobile Databases
SQLite
Realm

closed-source • **open-source**

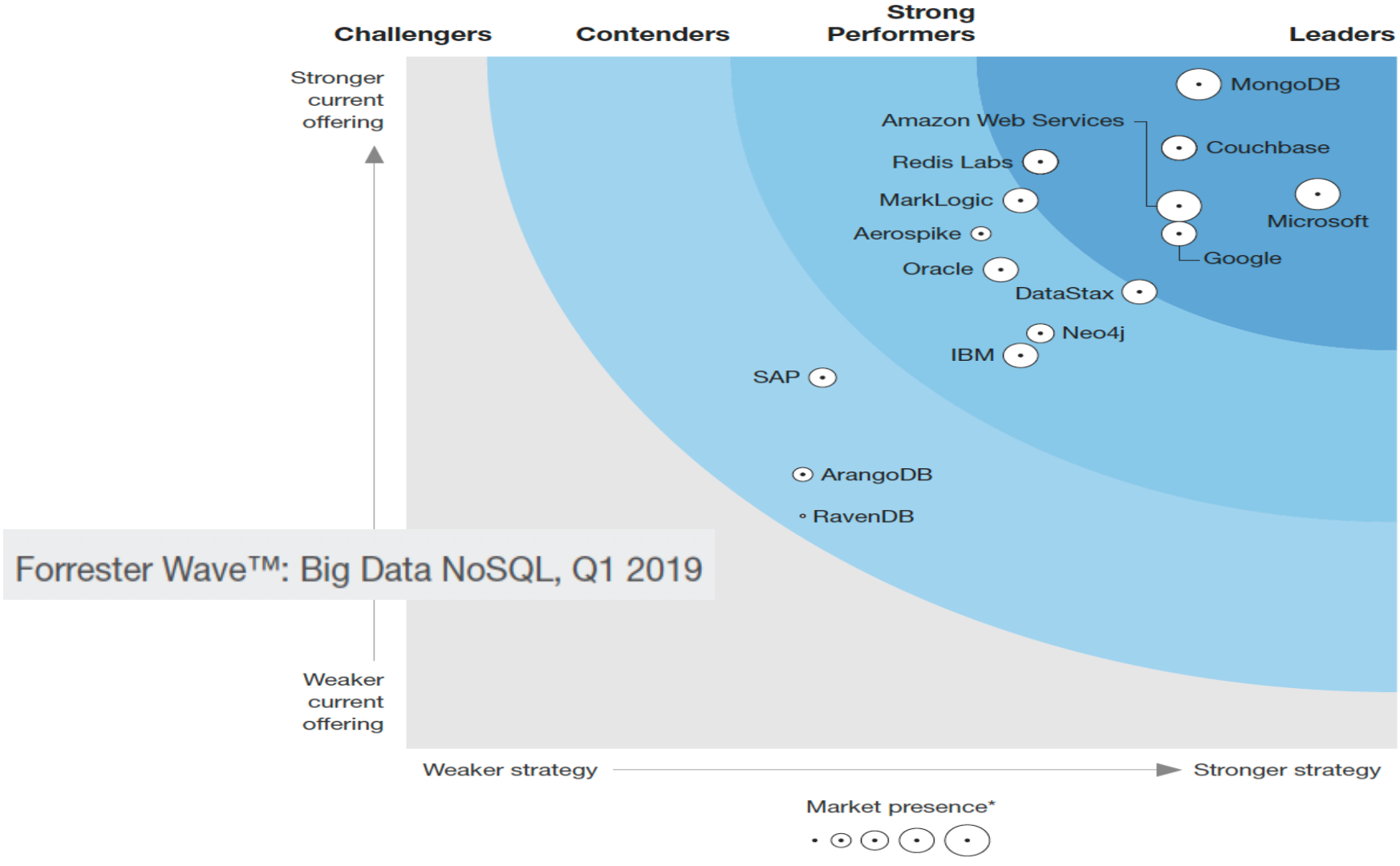
Gartner Market Share Ranking, 2011-2018

Rank



Source: Gartner (June 2019)

Note: The following historical vendor revenues were combined to reflect the state of the market in 2019: Cloudera, reflecting the merger with Hortonworks; Micro Focus, reflecting the acquisition of HPE Vertica; Broadcom, reflecting the acquisition of CA.


Challengers
Contenders
Strong Performers
Leaders
Stronger current offering
Weaker current offering
MongoDB
Amazon Web Services
Couchbase
Redis Labs
MarkLogic
Microsoft
Aerospike
Google
Oracle
DataStax
Neo4j
IBM
SAP
ArangoDB
RavenDB
Forrester Wave™: Big Data NoSQL, Q1 2019
Weaker strategy
Stronger strategy
Market presence*

SUMMARY SQL or NoSQL?

Projects where SQL is ideal:

- ✓ Logical related discrete data requirements which can be identified up-front.
- ✓ Data integrity is essential.
- ✓ Standards-based proven technology with good developer experience and support.

Projects where NoSQL is ideal:

- ✓ Unrelated, indeterminate or evolving data requirements.
- ✓ Simpler or looser project objectives, able to start coding immediately.
- ✓ Speed and scalability is imperative.