L5

NOSQL Introduction

Agenda

- RDBMS(SQL) ….. NoSQL
- NoSQL
  - What is it?
  - Types of NoSQL Databases
  - Why NoSQL?
  - Advantages of NoSQL
  - CAP Theorem
  - NoSQL Vendors
  - SQL versus NoSQL
  - NewSQL
  - Comparison of SQL, NoSQL and NewSQL

RDBMS(SQL) ---- NOSQL?

Value of RDBMS

- Getting at Persistent Data
- Concurrency
- Shared DB Integration
- A (Mostly) standard Model

Trouble with RDBMS

- Impedance mismatch

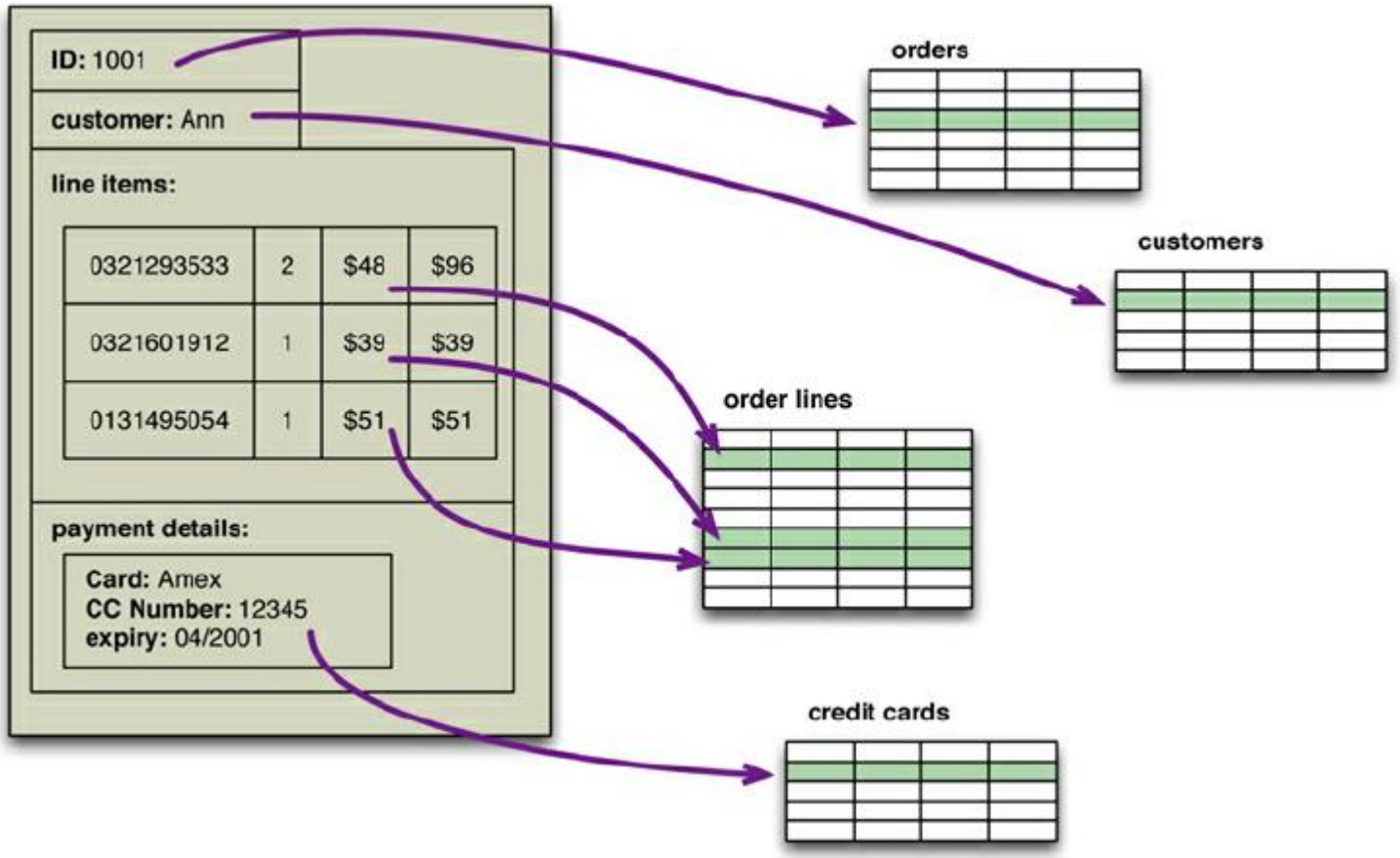


Figure 1.1. An order, which looks like a single aggregate structure in the UI, is split into many rows from many tables in a relational database

Trouble with RDBMS..

- Application and Integration Database
  - Shared DB Integration
    - Consistent Set of persistent data
    - Downside:
      - Complex DB Structure
      - Different structural and performance needs
      - DB Integrity -- within DB itself

Trouble with RDBMS..

- Application and Integration Database
  - Shared DB Integration
    - Consistent Set of persistent data
    - Downside:
      - Complex DB Structure
      - Different structural and performance needs
      - DB Integrity -- within DB itself
  - Application Database
    - DB Integrity
    - Interoperability Concerns

Trouble with RDBMS..

- Application and Integration Database
  - Shared DB Integration
    - Consistent Set of persistent data
    - Downside:
      - Complex DB Structure
      - Different structural and performance needs
      - DB Integrity -- within DB itself
  - Application Database
    - DB Integrity
    - Interoperability Concerns
- Web Services
  - Applications communicate over HTTP(Integration Mechanism)
  - a challenger to using the SQL with shared databases.
  - more flexibility for the structure of the data that was being exchanged.
    - XML , JSON
  - Freedom of choosing DB - nonrelational options!!

Trouble with RDBMS….

- Attack of the Clusters
  - The 2000s did see several large web properties dramatically increase in scale.
  - Two choices:

    Scale up or out(Bigger Machines vs Cluster of Small machines)
  - Cluster of commodity machines
    - Cheaper
    - High Reliability

Trouble with RDBMS….

- Attack of the Clusters
  - The 2000s did see several large web properties dramatically increase in scale.
  - Two choices:

    Scale up or out(Bigger Machines vs Cluster of Small machines)
  - Cluster of commodity machines
    - Cheaper
    - High Reliability
  - Problem: RDBMS not designed to run on clusters
    - Oracle RAC, Microsoft SQL Server
      - Highly available Disk Subsystem(Cluster aware file system)
      - Single point of failure
    - Separate Servers for different sets of Data
      - Sharding is controlled by Application

Trouble with RDBMS….

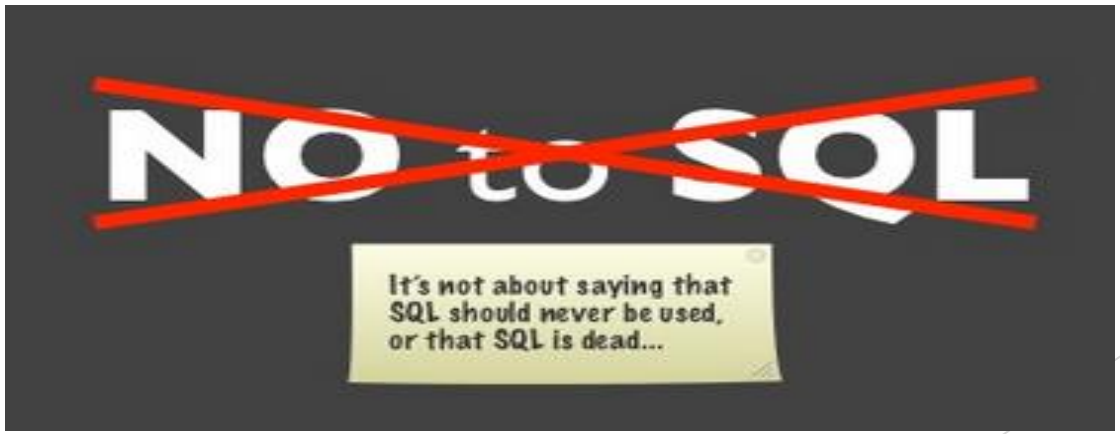
- Attack of the Clusters
  - The 2000s did see several large web properties dramatically increase in scale.
  - Two choices:

    Scale up or out(Bigger Machines vs Cluster of Small machines)
  - Cluster of commodity machines
    - Cheaper
    - High Reliability
  - Problem: RDBMS not designed to run on clusters
    - Oracle RAC, Microsoft SQL Server
      - Highly available Disk Subsystem(Cluster aware file system)
      - Single point of failure
    - Separate Servers for different sets of Data
      - Sharding is controlled by Application
- Mismatch b/w RDBMS and Clusters: Alternate data storage
  - Google: BigTable and Amazon: Dynamo
  - NoSQL

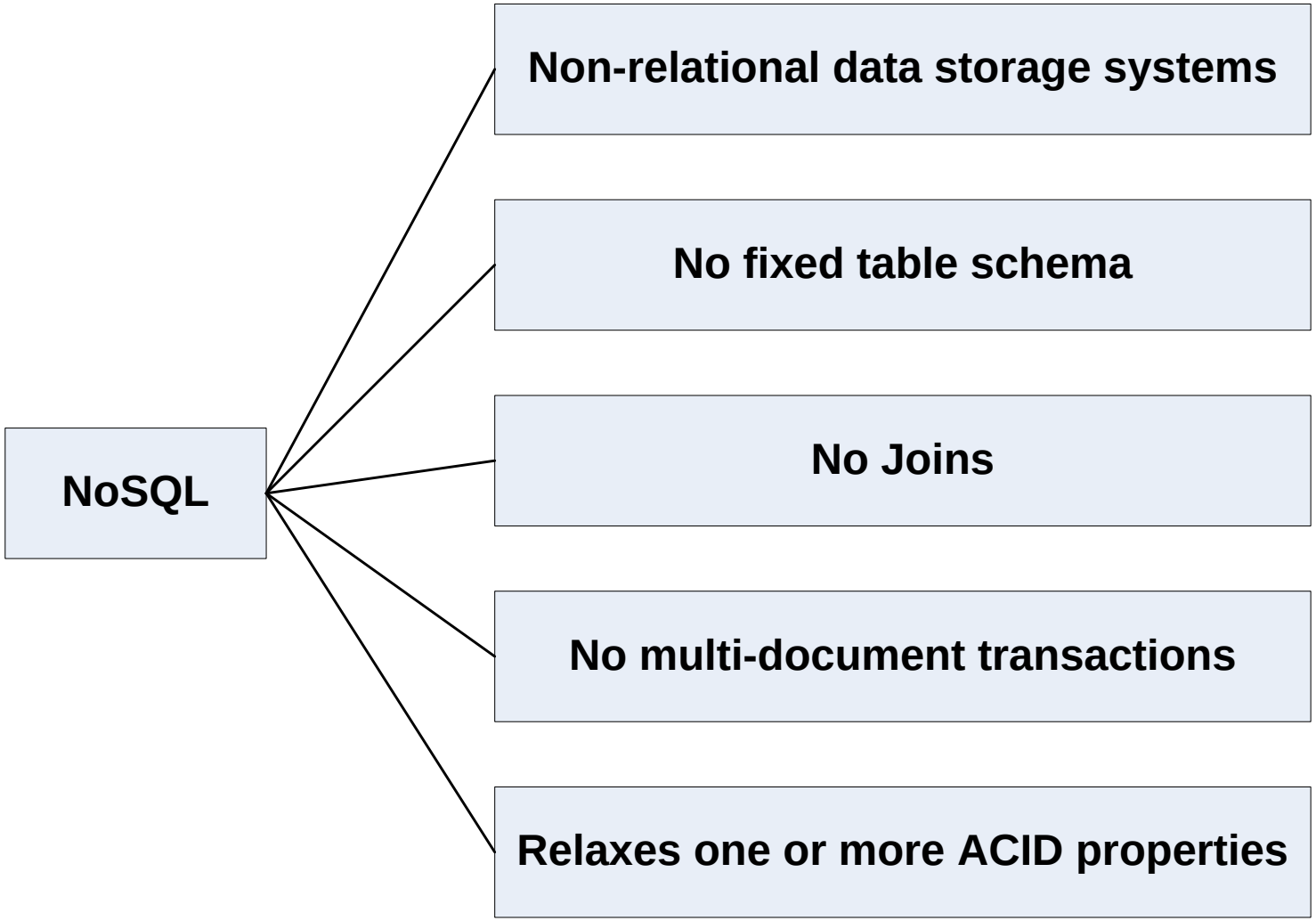
What is NoSQL?

What is NOSQL?

- The Name:
  - Stands for **N**ot **O**nly **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. …"*



What is NoSQL?



NOSQL: Common Characteristics

- Not using the relational model
- Running well on clusters
- Open-source

NOSQL: Common Characteristics

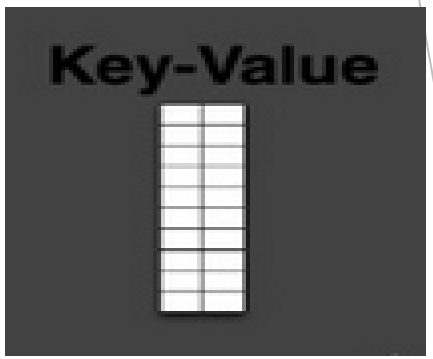
- Not using the relational model
- Running well on clusters
- Open-source
- **Built for the 21st century web estates**
- **Schemaless**
- **The most important result of the rise of NoSQL is Polyglot Persistence.**

Types of NoSQL

Types of NoSQL

Key value data store	Column-oriented data store	Document data store	Graph data store
• Riak • Redis • Membase	• Cassandra • HBase • HyperTable	• MongoDB • CouchDB • RavenDB	• InfiniteGraph • Neo4 • Allegro Graph

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 multi-valued like set.**
  - **items are combined into a *table***

Key-value


```
{
"lastVisit":1324669989288,
"user":{
 "customerId":"91cfdf5bcb7c",
 "name":"buyer",
 "countryCode":"US",
 "tzOffset":0
 }
}
```

Key-value

- Basic API access:
  - get(key): extract the value given a key
  - put(key, value): create or update the value given its key

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)**

Key-value

Pros:

- very fast
- very scalable (horizontally distributed to nodes based on key)
- simple data model
- eventual consistency
- fault-tolerance

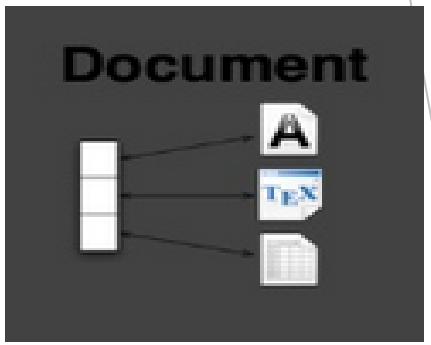
Cons:

- Can’t model more complex data structure such as objects

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	LinkeId	like SimpleDB	similar to Dynamo

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.**

Document-based

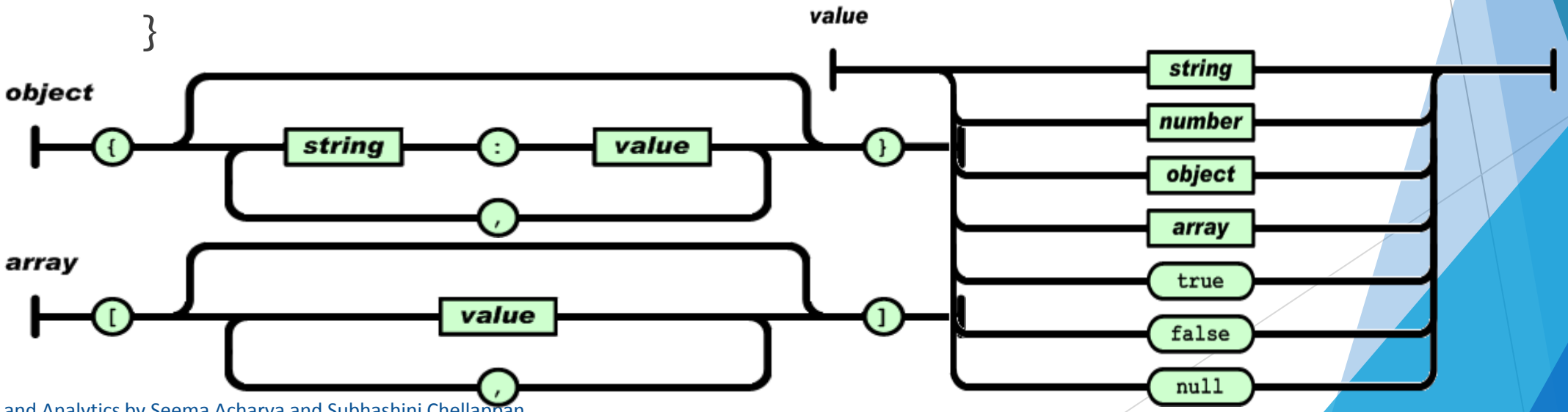
- Example: (MongoDB) document
  - {Name:"Jaroslav",

    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”]

    }

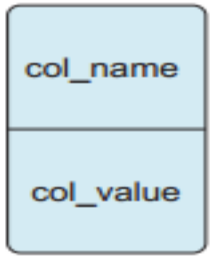


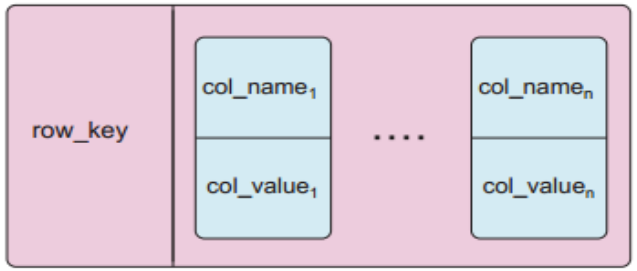
Document-based

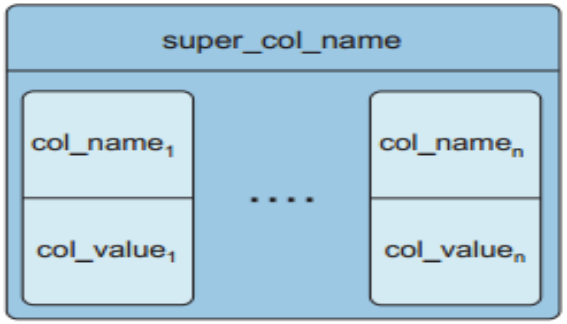
Name	Producer	Data model	Querying
MongoDB	10gen	object-structured documents stored in collections; each object has a primary key called ObjectId	manipulations with objects in collections (find object or objects via simple selections and logical expressions, delete, update,)
Couchbase	Couchbase[1]	document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

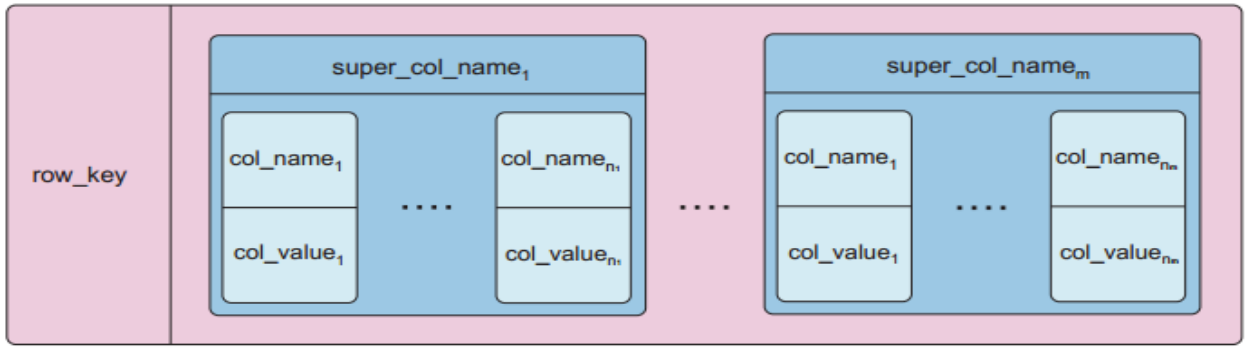
Column-based

- 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*

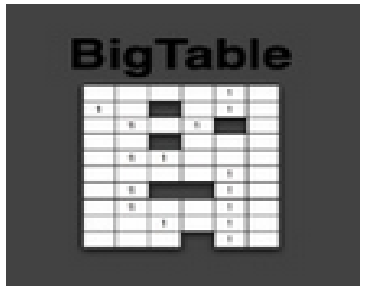








Column-based



- 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)

Column-based

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

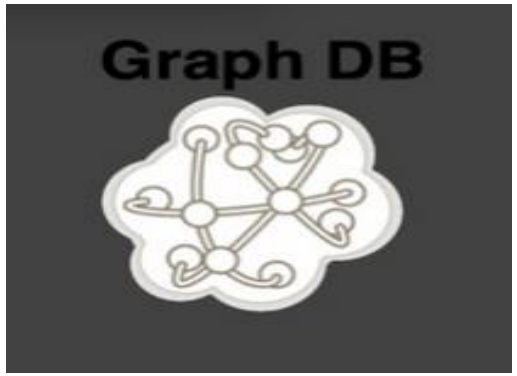
```
UserProfile = {
        Cassandra = { emailAddress:”casandra@apache.org” , age:”20”}
        TerryCho = { emailAddress:”terry.cho@apache.org” , gender:”male”}
        Cath = { emailAddress:”cath@apache.org” ,
                 age:”20”,gender:”female”,address:”Seoul”}
 }
```

Column-based

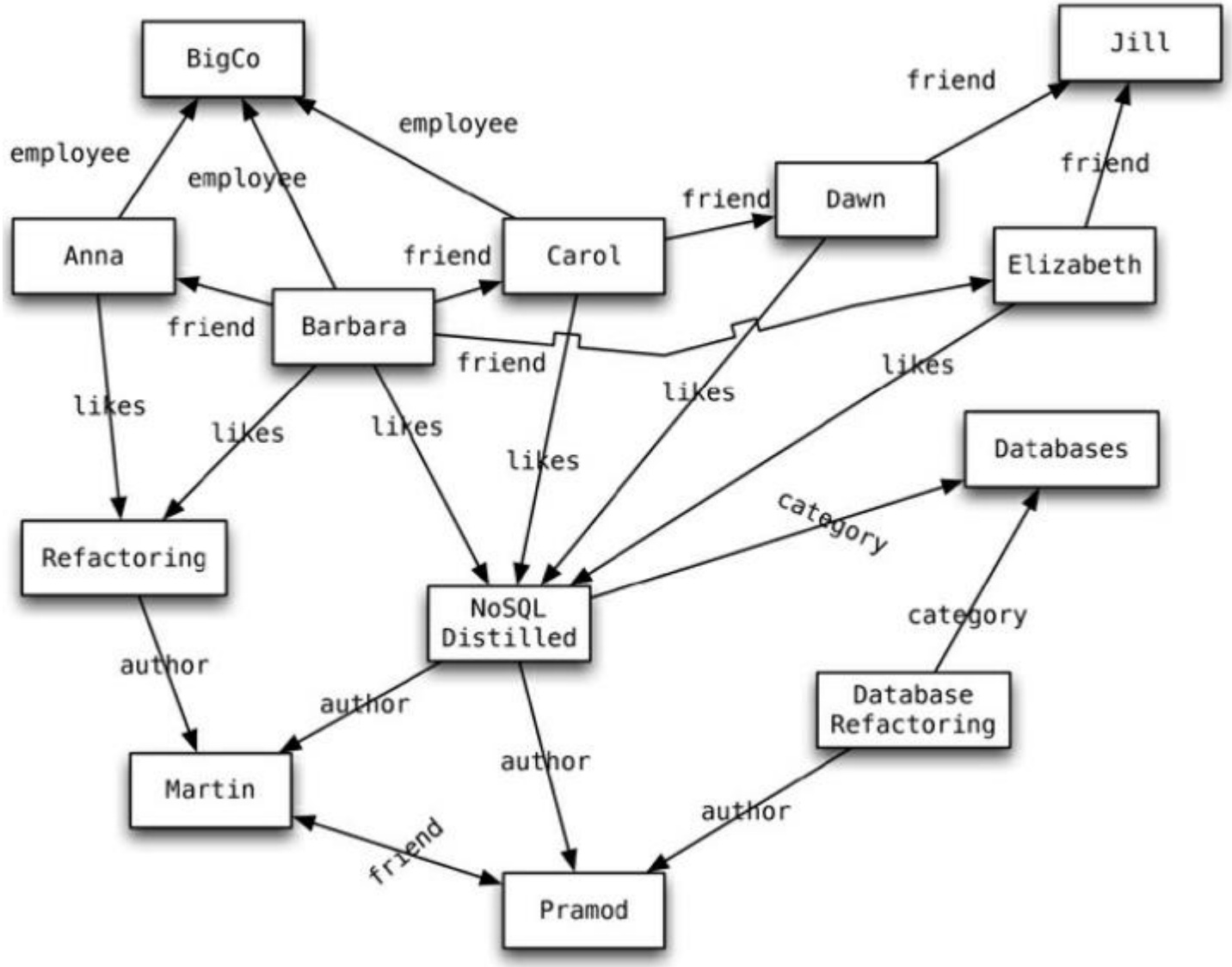
Name	Producer	Data model	Querying
BigTable	Google	set of couples (key, {value})	selection (by combination of row, column, and time stamp ranges)
HBase	Apache	groups of columns (a BigTable clone)	JRUBY IRB-based shell (similar to SQL)
Hypertable	Hypertable	like BigTable	HQL (Hypertext Query Language)
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)

Graph-based

- Focus on modeling the structure of data (*interconnectivity*)
- 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
- Interfaces and query languages vary
- *Single-step* vs *path expressions* vs *full recursion*
- Example:
  - Neo4j, FlockDB, Pregel, InfoGrid …



Graph Data Store



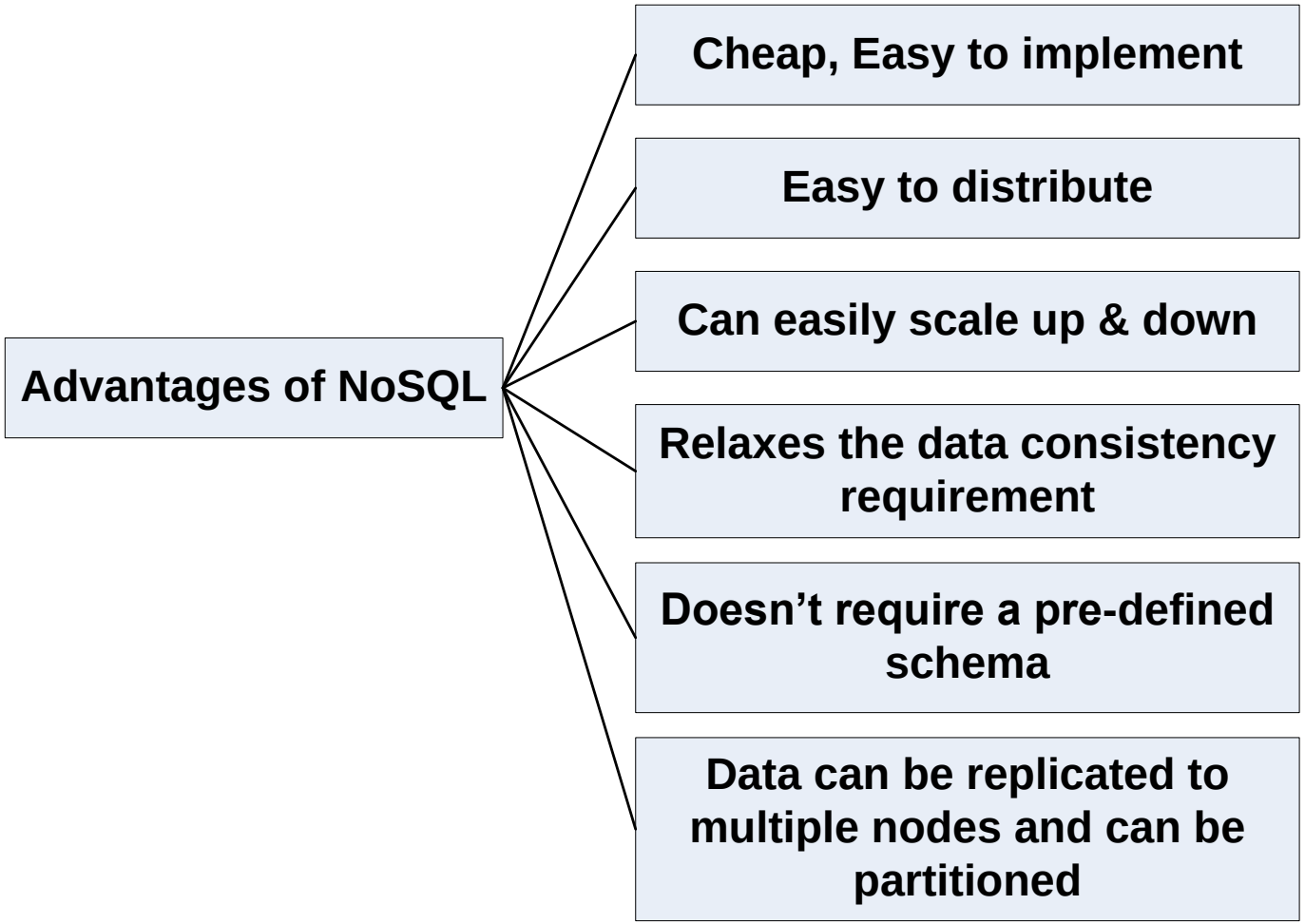
Why NoSQL?

Why NoSQL?

- Scale out architecture
- Large volumes of structured / semi structured / unstructured data
- Dynamic schema
- Auto-sharding
- Replication

Advantages of NoSQL

Advantages of NoSQL

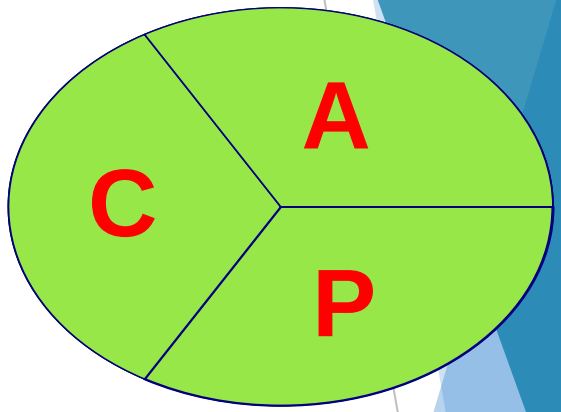


Disadvantages of NoSQL

- 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

CAP Theorem

CAP Theorem

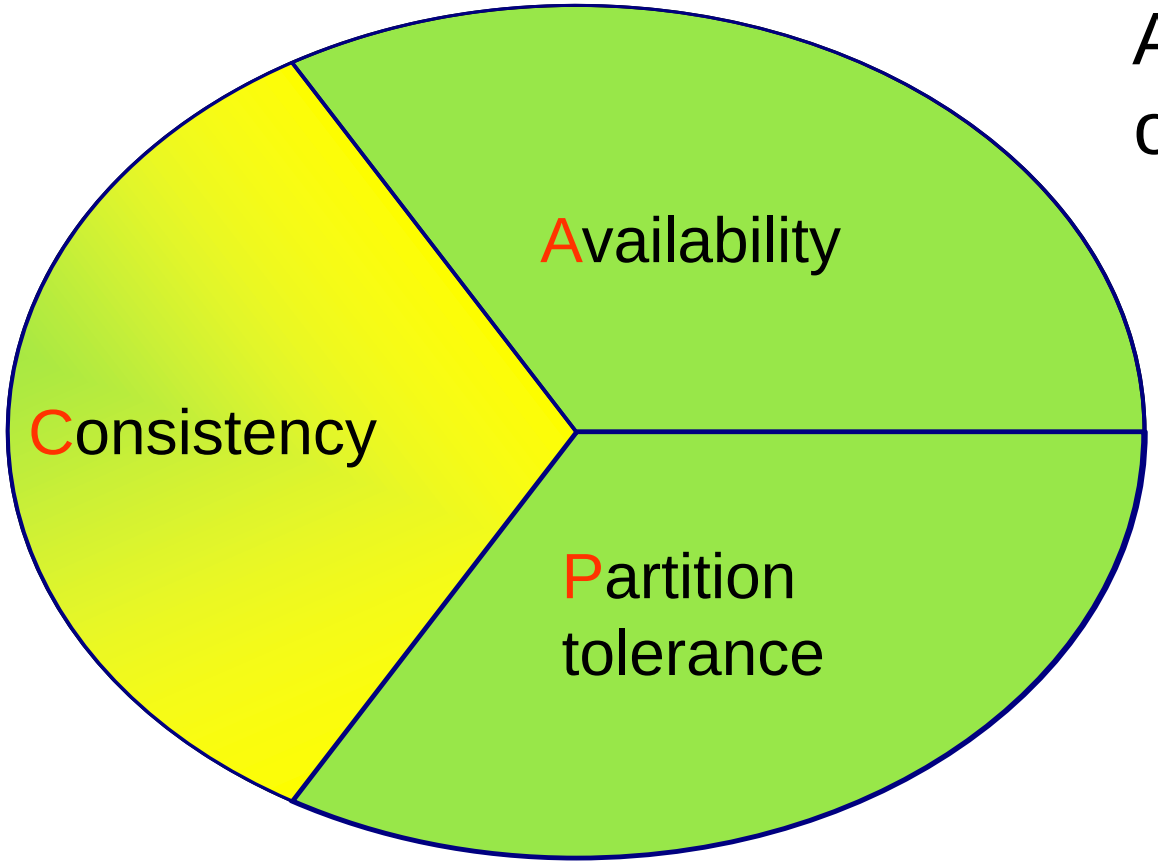


- Suppose three properties of a distributed system (sharing data)
  - **Consistency:**
    - all copies have same value
  - **Availability:**
    - reads and writes always succeed
  - **Partition-tolerance:**
    - system properties (consistency and/or availability) hold even when network failures prevent some machines from communicating with others

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)

CAP Theorem



All client always have the same view of the data

CAP Theorem

- **Consistency**
  - 2 types of consistency:
    1. Strong consistency – ACID (**A**tomicity, **C**onsistency, **I**solation, **D**urability)
    2. Weak consistency – BASE (**B**asically **A**vailable **S**oft-state **E**ventual consistency)
       - High Availability
       - Replica Convergence
       - Conflict Resolution
         - Read repair
         - Write repair
         - Asynchronous repair

CAP Theorem

- **ACID**
  - A DBMS is expected to support "ACID transactions," processes that are:
  - **A**tomicity: either the whole process is done or none is
  - **C**onsistency: only valid data are written
  - **I**solation: one operation at a time
  - **D**urability: once committed, it stays that way
- **CAP**
  - **C**onsistency: all data on cluster has the same copies
  - **A**vailability: cluster always accepts reads and writes
  - **P**artition tolerance: guaranteed properties are maintained even when network failures prevent some machines from communicating with others

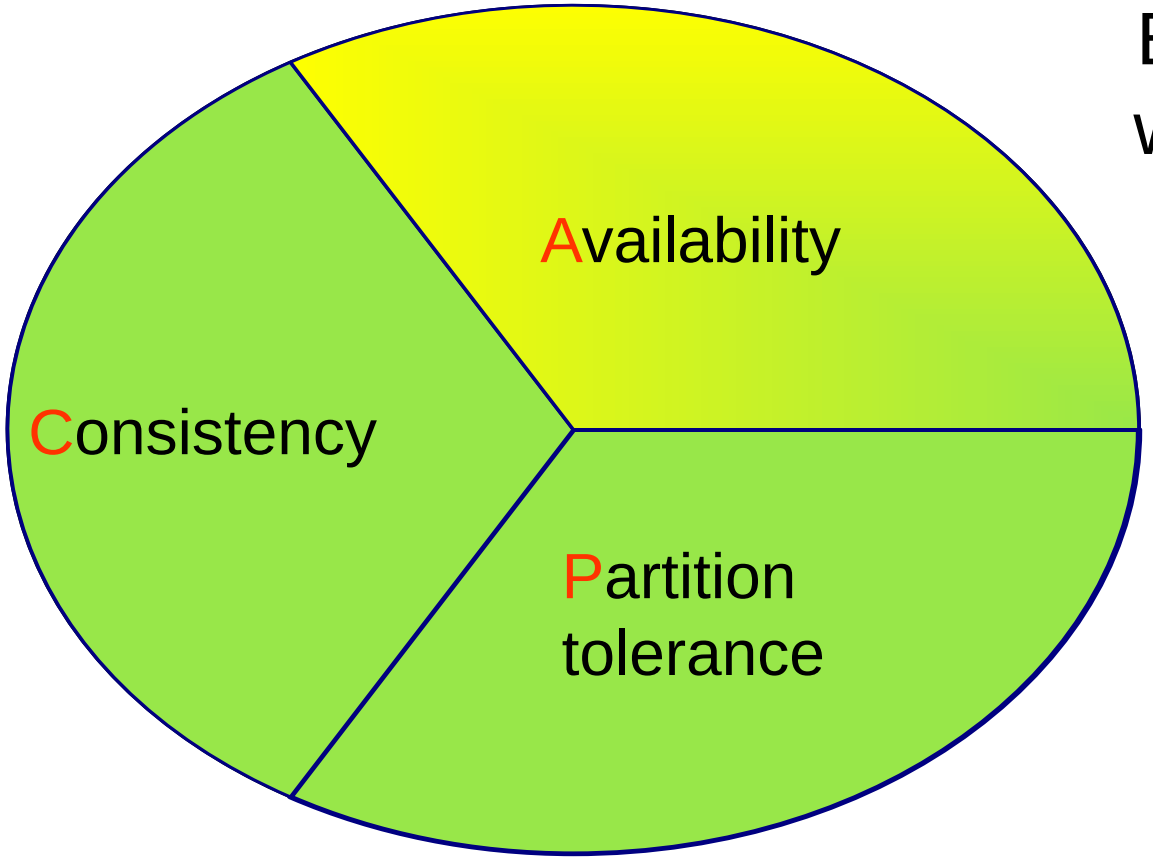
CAP Theorem

- 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"*

CAP Theorem

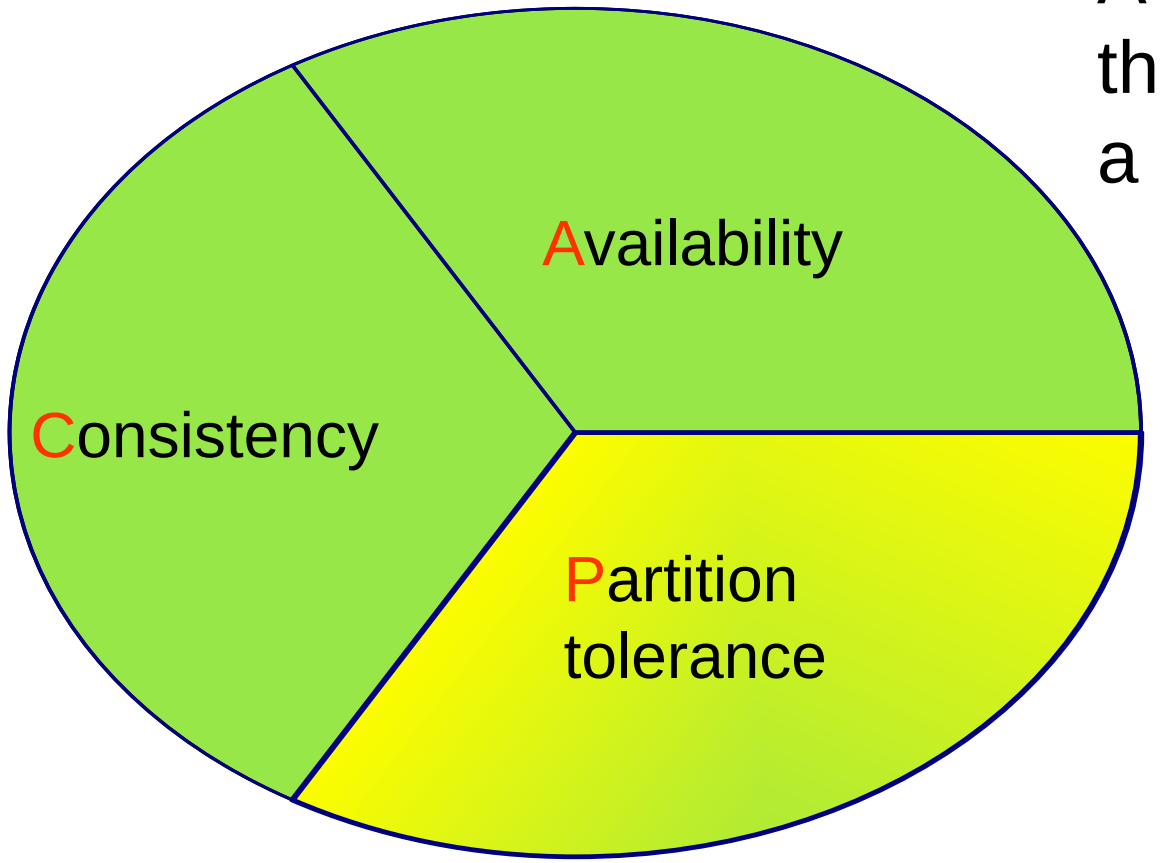
- Eventual consistency
  - When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent
- Cloud computing
  - ACID is hard to achieve, moreover, it is not always required, e.g. for blogs, status updates, product listings, etc.

CAP Theorem



Each client always can read and write.

CAP Theorem

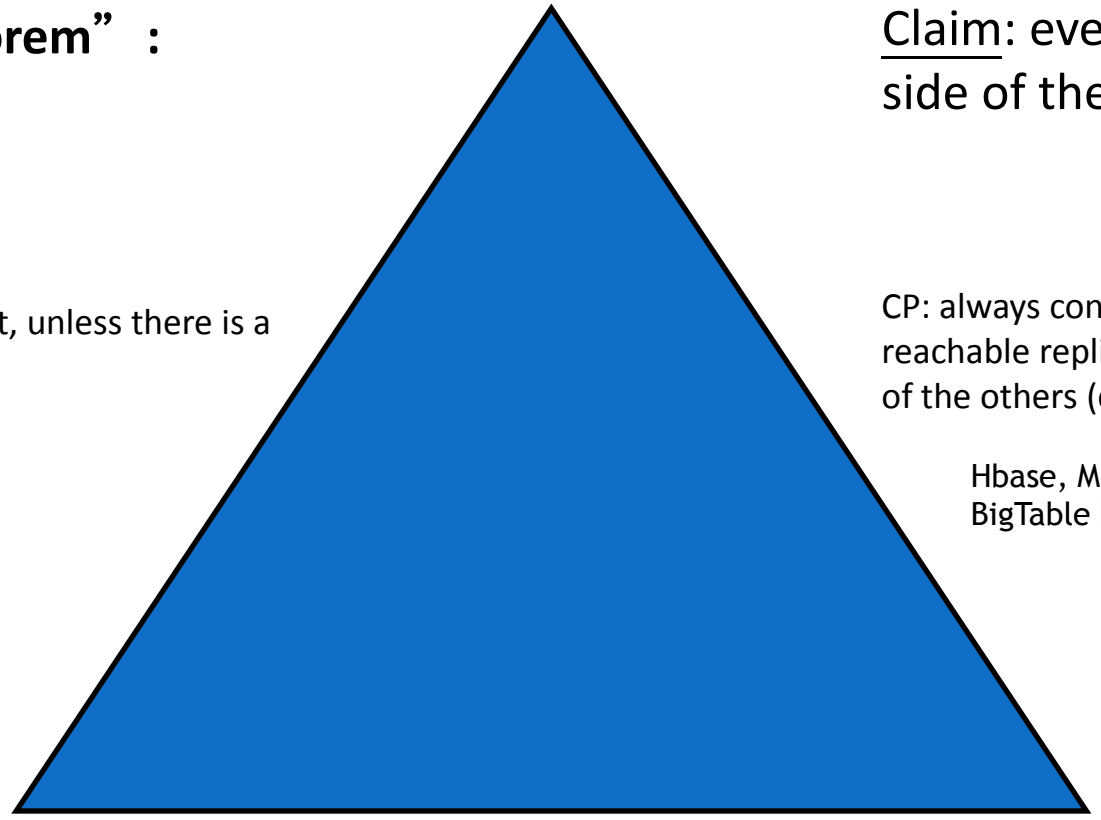


A system can continue to operate in the presence of a network partitions

Fox&Brewer "CAP Theorem": C-A-P: choose two.

consistency

C



A

Availability

P

Partition-resilience

Claim: every distributed system is on one side of the triangle.

CA: available, and consistent, unless there is a partition.

Traditional RDBMS
PostgreSQL, MYSQL
Etc.

CP: always consistent, even in a partition, but a reachable replica may deny service without agreement of the others (e.g., quorum).

Hbase, MongoDB, Redis,
BigTable like Systems

AP: a reachable replica provides service even in a partition, but may be inconsistent if there is a failure.

Riak, Cassandra, CouchDB,
Dynamo like systems

NoSQL in Industry

Use of NoSQL in Industry

- Key-Value
  - Shopping carts
  - Web user data analysis
  - Amazon, Linkedin
- Document based
  - Real-time Analysis
  - Logging
  - Document archive management
- Column-oriented
  - Analyze huge web user actions
  - Sensor feeds
  - Facebook, Twitter, eBay, Netfix
- Graph-based
  - Network modeling
  - Recommendation
  - Walmart-upsell, cross-sell

NoSQL Vendors

NoSQL Vendors

Company	Product	Most widely used by
Amazon	DynamoDB	LinkedIn, Mozilla
Facebook	Cassandra	Netflix, Twitter, eBay
Google	BigTable	Adobe Photoshop

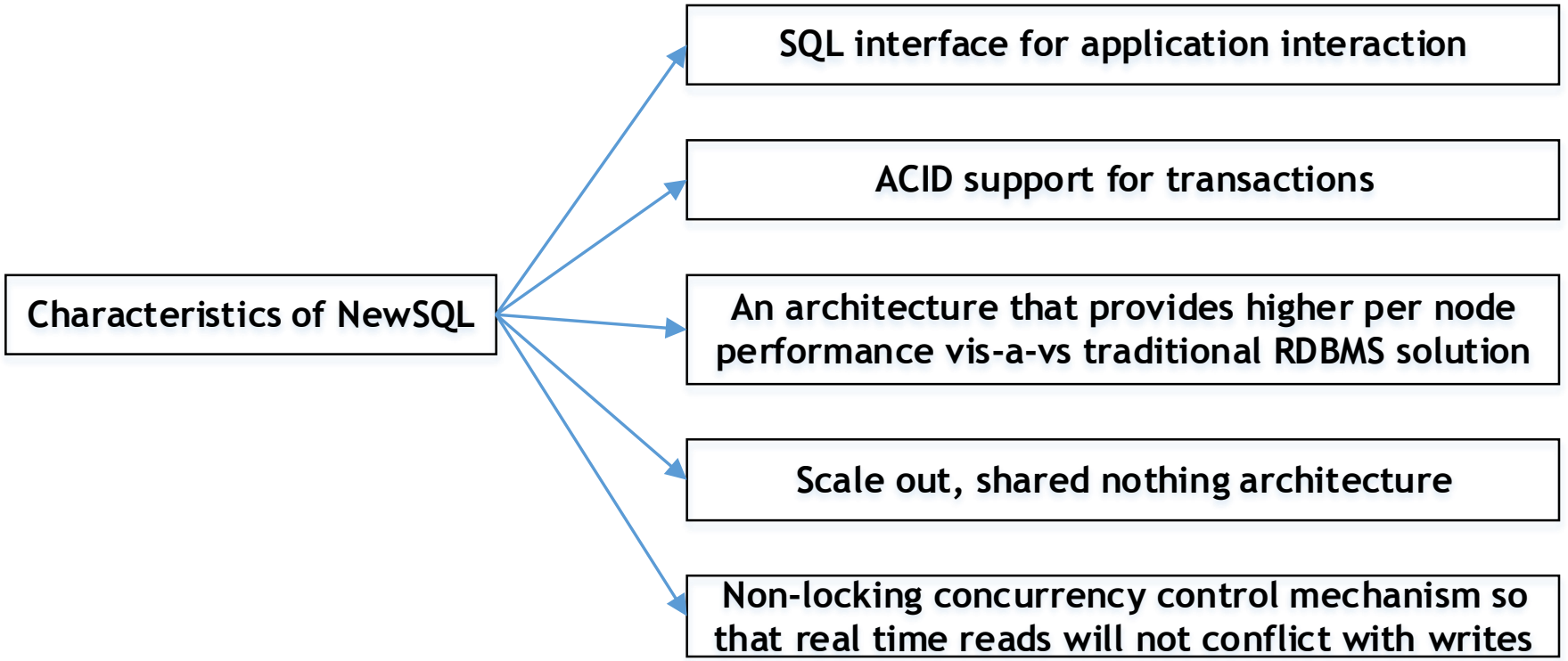
SQL Vs. NoSQL

SQL Vs. NoSQL

SQL	NoSQL
Relational database	Non-relational, distributed database
Relational model	Model-less approach
Pre-defined schema	Dynamic schema for unstructured data
Table based databases	Document-based or graph-based or wide column store or key-value pairs databases
Vertically scalable (by increasing system resources)	Horizontally scalable (by creating a cluster of commodity machines)
Uses SQL	Uses UnQL (Unstructured Query Language)
Not preferred for large datasets	Largely preferred for large datasets
Not a best fit for hierarchical data	Best fit for hierarchical storage as it follows the key-value pair of storing data similar to JSON (Java Script Object Notation)
Emphasis on ACID properties	Follows Brewer's CAP theorem
Excellent support from vendors	Relies heavily on community support
Supports complex querying and data keeping needs	Does not have good support for complex querying
Can be configured for strong consistency	Few support strong consistency (e.g., MongoDB), few others can be configured for eventual consistency (e.g., Cassandra)
Examples: Oracle, DB2, MySQL, MS SQL, PostgreSQL, etc.	MongoDB, HBase, Cassandra, Redis, Neo4j, CouchDB, Couchbase, Riak, etc.

NewSQL

NewSQL



SQL Vs. NoSQL Vs. NewSQL

SQL Vs. NoSQL Vs. NewSQL

	SQL	NoSQL	NewSQL
Adherence to ACID properties	Yes	No	Yes
OLTP/OLAP	Yes	No	Yes
Schema rigidity Adherence to data model	Yes Adherence to relational model	No	Maybe
Data Format Flexibility	No	Yes	Maybe
Scalability	Scale up Vertical Scaling	Scale out Horizontal Scaling	Scale out
Distributed Computing	Yes	Yes	Yes
Community Support	Huge	Growing	Slowly growing

Answer a few quick questions …

Fill in the blanks

1. The expansion for CAP is _____, _____ and _____.
2. The expansion of BASE is _____.
3. MongoDB is _____ and _____.
4. Cassandra is _____ and _____.
5. _____ has no support for ACID properties of transactions.
6. _____ is a robust database that supports ACID properties of transactions and has the scalability of NoSQL.

Answer Me

- Compare and contrast SQL, NoSQL and NewSQL.

References …

Further Readings

- http://www.mongodb.com/nosql-explained
- http://nosql-database.org/
- http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduce_Compatibility_Hadoop1_Hadoop2.html
- http://hadoop.apache.org/

Thank you