MATES ED2MIT

Education and Training for Data Driven Maritime Industry

Tutorial A03

Big Data Types, SQL and NoSQL Databases

Yuri Demchenko MATES Project
University of Amsterdam

Maritime Alliance for fostering the European Blue economy through a Marine Technology Skilling Strategy

Co-funded by the Erasmus+ Programme of the European Union

Outline

- Data types and data models
- SQL databases
- Distributed systems: CAP theorem, ACID and BASE properties
- NoSQL databases overview
- Modern cloud databases

- Additional Information: Data structures and Normalisation

Co-funded by the
Erasmus+ Programme
of the European Union

Data structures and data models

Cloud based IaaS/PaaS/SaaS platforms need to provide storage and processing environment for different types of data generated and used by enterprise and user applications.

- Different stages of the data transformation will use or produce data of different structures, models and formats.

Data types can be defined:

- Data described via a formal data model, which are the majority of structured data, data stored in databases, archives, etc.
- Data described via a formalized grammar (e.g. machine generated textual data or forms)
- Data described via a standard format (e.g. digital images, audio or video files)
- Arbitrary textual or binary data

Data models

- Structured data (e.g. relational)
- Unstructured data (e.g. text or HTML pages)
- Semi-structured Data (e.g. tables)
- Key-value pairs
- XML: Hierarchical data (e.g. document)
- RDF: Semantic data (e.g. RDF, triple store)

Structured Data and SQL – Example Data Structure Normalisation 1NF, 2NF, 3NF *(Detailed Example Provided)*

- Structured data is the most widely used in data management applications
- Essentially structured relational data are tables where rows are records and columns are properties
- Structured data can be stored in SQL/relational databases
- Structured data simplify many operations on data analysis and reports generation – the major uses of SQL databases

Table 1 contains 3 records for laptops and has 5 columns/properties

- Notice, first 2 columns contain related data Manufacturer and Model; Lenovo model contains also screen size
  - Replace first 2 columns with references/keys to a separate table with Manufacturer-Model data
  - Split ThinkPad, 14 inch cell on two
- In database world such operation is called normalization

Table 1. Laptop models and characteristics

Manufacturer	Model	Screen size	Color	OS
Apple	MacBook Pro	14	white	OS X
Dell	Inspiron	15	Black	Windows 8
Lenovo	ThinkPad, 14 inch		Black	Windows 7

Laptop Type	Screen size	Color	OS
M1	14	white	OS X
M2	15	Black	Windows 8
M3	14	Black	Windows 7

Key	Manufacturer	Model
M1	Apple	MacBook Pro
M2	Dell	Inspiron
M3	Lenovo	ThinkPad

Semi-structured data - Example

Semi-structured data non-formally can be defined as data having some structure but cannot be used with the structural databases.

- The reasons for this can be many, for example, because the data are presented as a string containing specific information that is not consistent from record to record.

Using the same example with laptop models, a couple of records can be taken from the webshop catalogue:

APPLE MacBook Air MD761N/A 13 inch
13 inch LED • Intel Core i5 (1,3 GHz) • 4 GB • 256 GB SSD • Intel HD Graphics 5000

Dell XPS 13 Ultrabook (9333-0987NL)
Silver, 256GB SSD, WLAN, Touch, Win 8.1Pro; Processor: Intel® Core i7-4500U; Memory: 8 GB; Display: 33.8 cm (13.3 inch)

A simple data processing can create a table that extracts the laptop model and display size.

- Although the remaining information is quite understandable for humans, it cannot be easily parsed by machine and can be just stored as a string.

Model	Display size	Other
APPLE MacBook Air MD761	13 inch	Intel Core i5 (1,3 GHz) • 4 GB • 256 GB SSD • Intel HD Graphics 5000
Dell XPS 13 Ultrabook (9333-0987NL)	13,3 inch	Silver, 256GB SSD, WLAN, Touch, Win 8.1Pro; Processor: Intel® Core i7-4500U; Memory: 8 GB

Key-Value Pairs - Example

In the key-value dataset the data are stored as pairs of key and value, where the KEY is structured and the VALUE is not structured.

- This allows flexibility in defining the overall data structure.
- Key-value data can be converted into structured form, semi-structured form or text.
  - Table below illustrates example of the key-value data set.
- Key-value data structures are specifically adopted for using with MapReduce and Hadoop. The reason for this is that key-value data can be easily split and processed in parallel.
- Note: Value can be quite diverse type data, e.g. set of parameters, values

KEY	VALUE
Apple MacBook Pro	14 inch | white | OS X
Dell Inspiron	15 inch | black | Windows 8
Lenovo ThinkPad	14 inch | black | Windows 7

XML: Hierarchical Data

- XML data can be considered as hierarchical data where XML structure defines an XML document that has a root element and a tree of parent and child elements.
- Information query and access to data in XML documents requires parsing the whole document which is quite time consuming operation.
- The example below illustrates a simplified structure of the XML records having only one level of hierarchy.
  - XML file below shows content of the <Laptop> XML document.

Parent	Child Element	Value
Laptop	ID	UID
	Model	Dell XPS13 Ultrabook
	Color	Silver
	Display	13 inch
	Memory	8 GB
	HDD	256 GB SSD
	OS	Windows 8.1

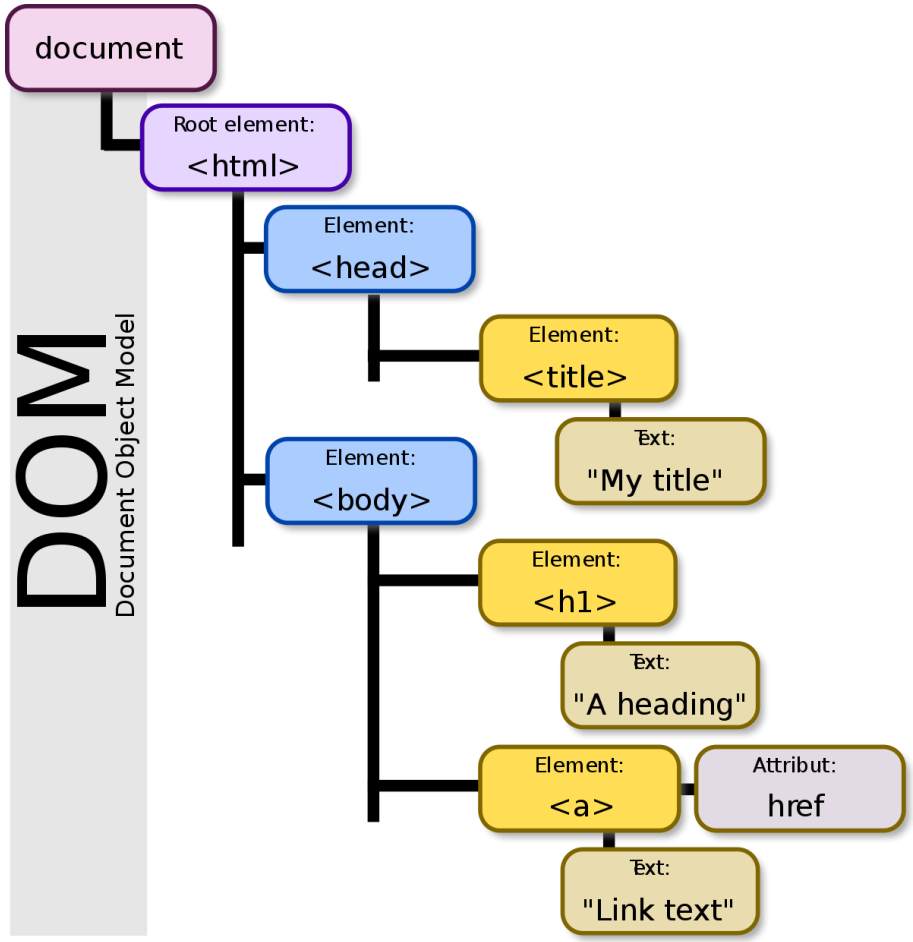
XML document (note 3 levels of hierarchy)

```
<ProductType ID=3736358>
   <Laptop ID=2346883625390092>
     <Model>DELL XPS13 Ultrabook</Model>
     <Color>Silver</Color>
     <Display>13 inch</Display>
     <Memory>8 GB</Memory>
     <HDD type=SSD>256 GB</HDD>
     <OS>Windows 8.1 Pro</OS>
  </Laptop>
</ProductType>
```

Document Object Model (DOM)

- DOM is a concept behind XML, HTML, JSON
- Documents are parsed against DOM and available in RAM memory providing quick access to elements attributes and content
- Methods available in Java, JavaScipt
- Python BeautifulSoup Parser and lxml/libxml2
  - However based on regular expressions than on DOM

Example: DOM based Structure of HTML document



"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The W3C DOM standard is separated into 3 different parts:

- Core DOM - standard model for all document types
- XML DOM - standard model for XML documents
- HTML DOM - standard model for HTML documents

What is the HTML DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements
- In other words:
  The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

DOM: Processing approaches (Java/JavaScript)

- Use of document parsing tree
  - document.forms['*formid*'].elements[*'inputid'*]
  - document.forms['*formid*'].*inputid*
- Select by id, tag or name
  - document.getElementById('*elementId*')
  - document.getElementsByTagName('div')
  - document.getElementsByName('*elementName*')
- Combined
  - document.getElementById('*formId*').*inputid*
- Note: "camelCaseType" in Javascript semantics

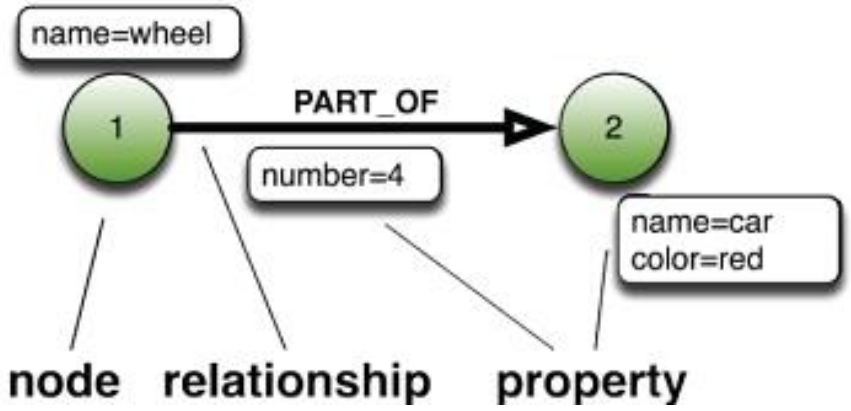
- Properties expressed as sub-elements vs as attributes

```
<td color=green opacity=30%>Name:</td>
  <ul>
    <li>First name:</li>
    <li>Last name:</li>
  </ul>
<td>Age:</td>
```

- Select by attributes
  - `document.getElementsByTagName("color")`

RDF: Semantic Data and Graph Data

- Resource Description Framework (RDF) is a format for expressing a **relation between subject and object**.
  - Initially designed as a metadata data model and currently used as the main format for describing relations in the Semantic web and for knowledge description.
  - The core of RDF is a statement in a form of triple “subject-predicate-object” which allows describing complex and conceptual relations between elements.
  - The collection of RDF statements represents a directed graph.
  - A Social graph can also be described with the RDF triples like this **“person1-isFriendOf-person2”**.
- RDF data can be stored in the SQL database but for performance reasons it is better to use specialized Triplestores or Quad stores if RDF represents a named graph.
  - In the latter case the structure of quads contains context element or graph name **“graphname-subject-predicate-object”**.



Property Graph Model defines the following three basic building blocks:

- node (or vertex)
- relationship (or edge) - with direction and Type (labeled and directed)
- property (or attribute) on nodes and relationships

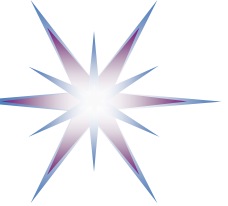
[ref] http://www.infoq.com/articles/graph-nosql-neo4j

Mentimeter Question 7.1

- Texts are considered not structured data but we easy understand textual information. What properties of text make it easy understandable?
  - Language
  - Grammar
  - Character set
  - Visual appearance
  - Dictionary
  - Domain knowledge

Mentimeter Question 7.2

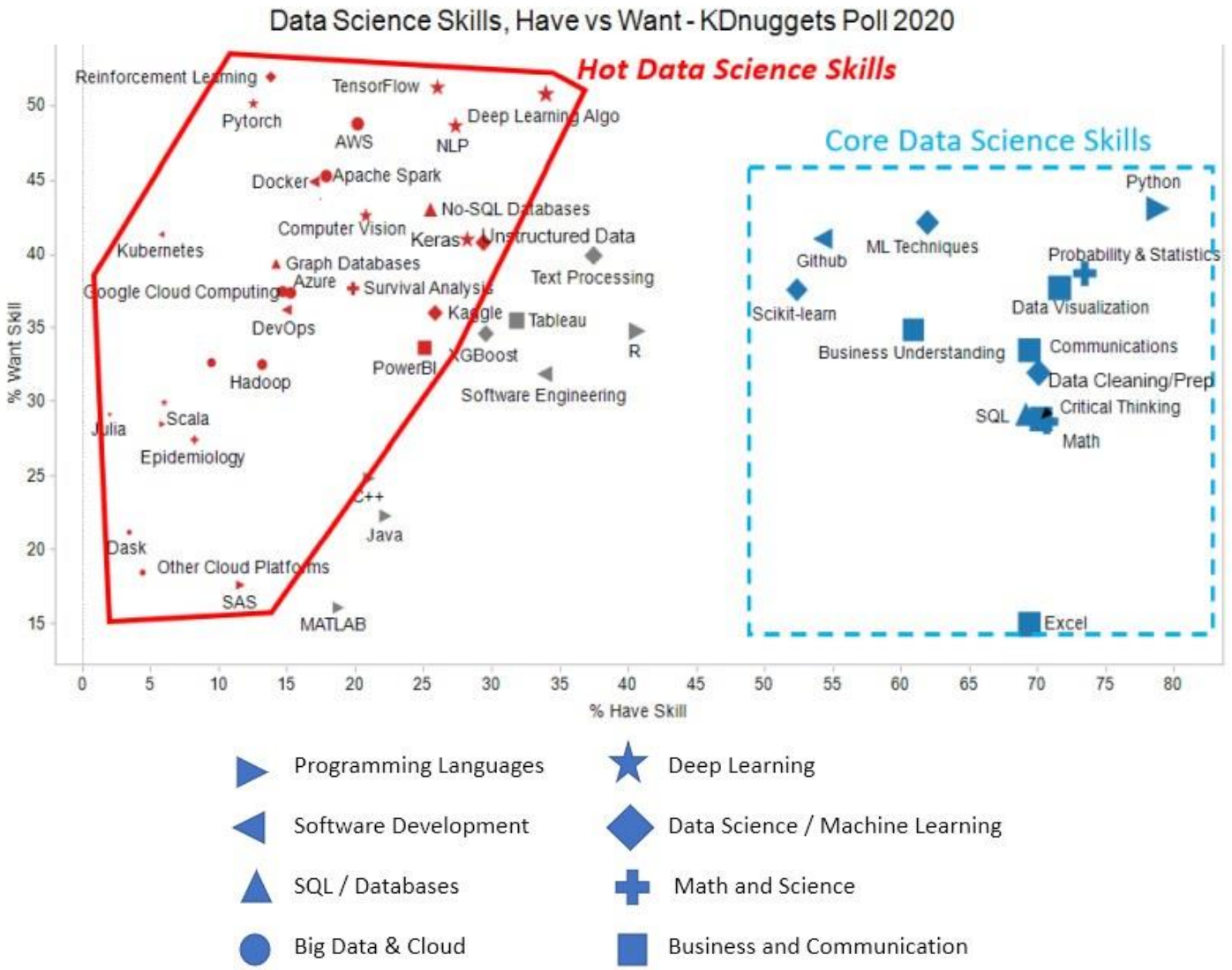
- XML, HTML, JSON are widely used in many application. What makes them so popular?



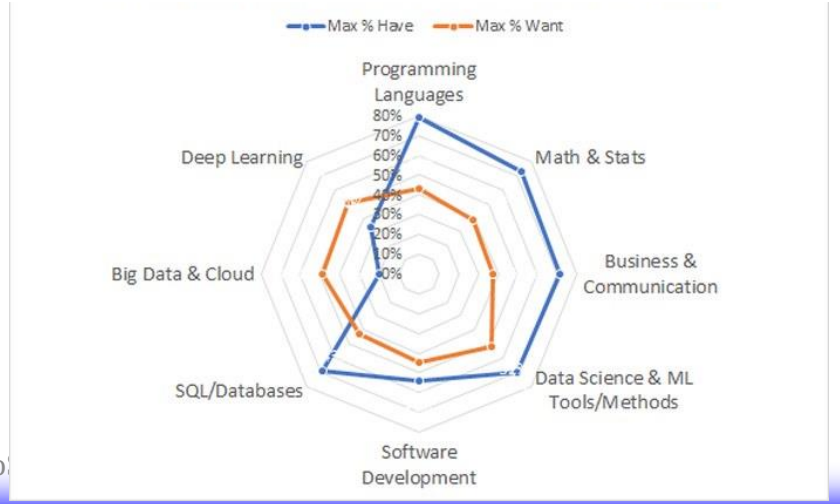
RDBMS and SQL

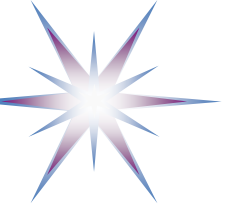
- SQL
- ACID properties
- SQL databases
- SQL scalability issues

Modern Data Science Skills (2020)
https://www.kdnuggets.com/2020/09/modern-data-science-skills.html



Category	Max %Have	Max %Want
Programming Languages	79%	43%
Math & Stats	73%	39%
Business & Communication	72%	38%
Data Science & ML Tools/Methods	70%	52%
Software Development	54%	45%
SQL/Databases	69%	43%
Big Data & Cloud	**20%**	**49%**
Deep Learning	34%	51%





Standard SQL

SQL is widely used for enterprise and business data management and reporting.

SQL has a long history and is specified by a number of standards – ISO/IEC 9075 group

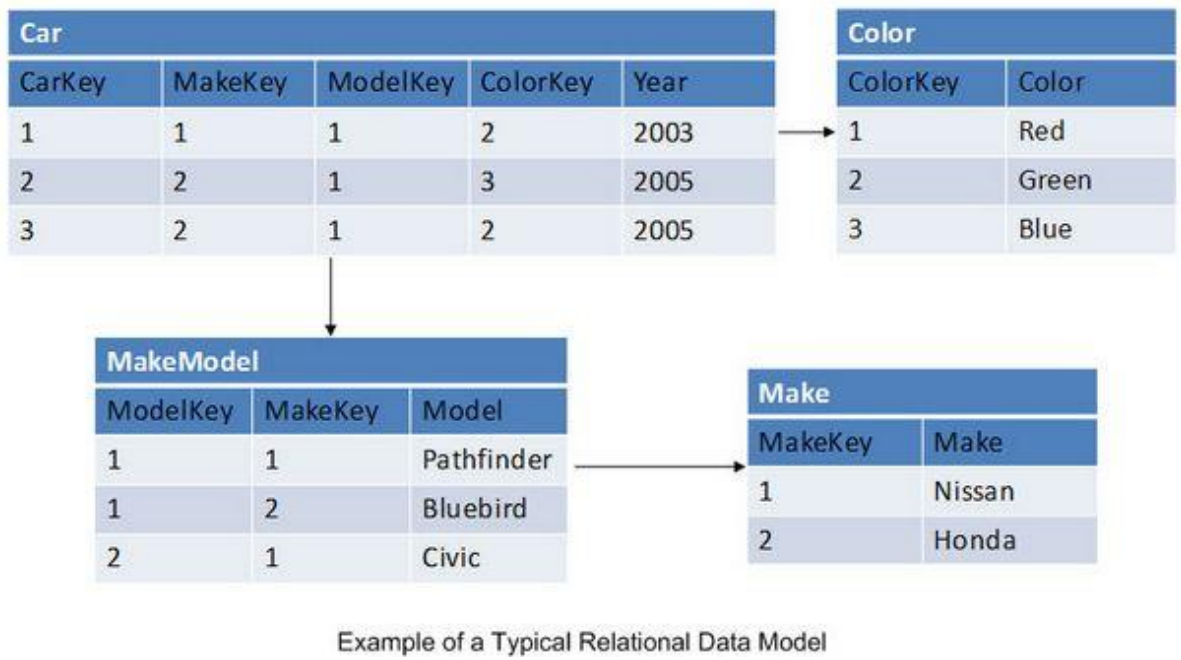
- 1987 – Initial ISO/IEC Standard
- 1989 – Referential Integrity
- 1992 – SQL2
  - 1995 SQL/CLI (ODBC)
  - 1996 SQL/PSM – Procedural Language extensions
- 1999 – User Defined Types
- 2003 – SQL/XML
- 2008 – Expansions and corrections
- 2011 (or 2012) System Versioned and Application Time Period Tables

SQL Characteristics

- Data stored in columns and tables
- Relationships represented by data
- Data Definition Language (DDL) - commands for creating, modifying and deleting SQL objects
- Data Manipulation Language (DML) - commands for adding, updating and removing data from SQL storage objects
- Data Control Language (DCL) – commands for managing access to Hive storage objects
- Transactions
- Abstraction from physical layer

Relational Database (1)

- Essentially a group of tables (entities)
  - Tables are made up of columns and rows (tuples)
  - Tables have constraints
  - Relationships are defined between tables
- Facilitated through Relational Database Management Systems (RDBMS)



Car

CarKey	MakeKey	ModelKey	ColorKey	Year
1	1	1	2	2003
2	2	1	3	2005
3	2	1	2	2005

Color

ColorKey	Color
1	Red
2	Green
3	Blue

MakeModel

ModelKey	MakeKey	Model
1	1	Pathfinder
1	2	Bluebird
2	1	Civic

Make

MakeKey	Make
1	Nissan
2	Honda

Example of a Typical Relational Data Model

- Multiple tables being accessed in a single query are "joined" together
- Normalization is a data-structuring model used with relational databases
  - Ensures data consistency
  - Removes data duplication

Relational Database Advantages

- Advantages
  - Simplicity, easy, well defined programming
  - Robustness
  - Flexibility
  - Performance
  - Easy scalable up on one server, but problems with scalability down and horizontally
  - Compatibility in managing generic data
- Under condition
  - To offer all of these, relational databases have to be incredibly complex internally

SQL Database Examples

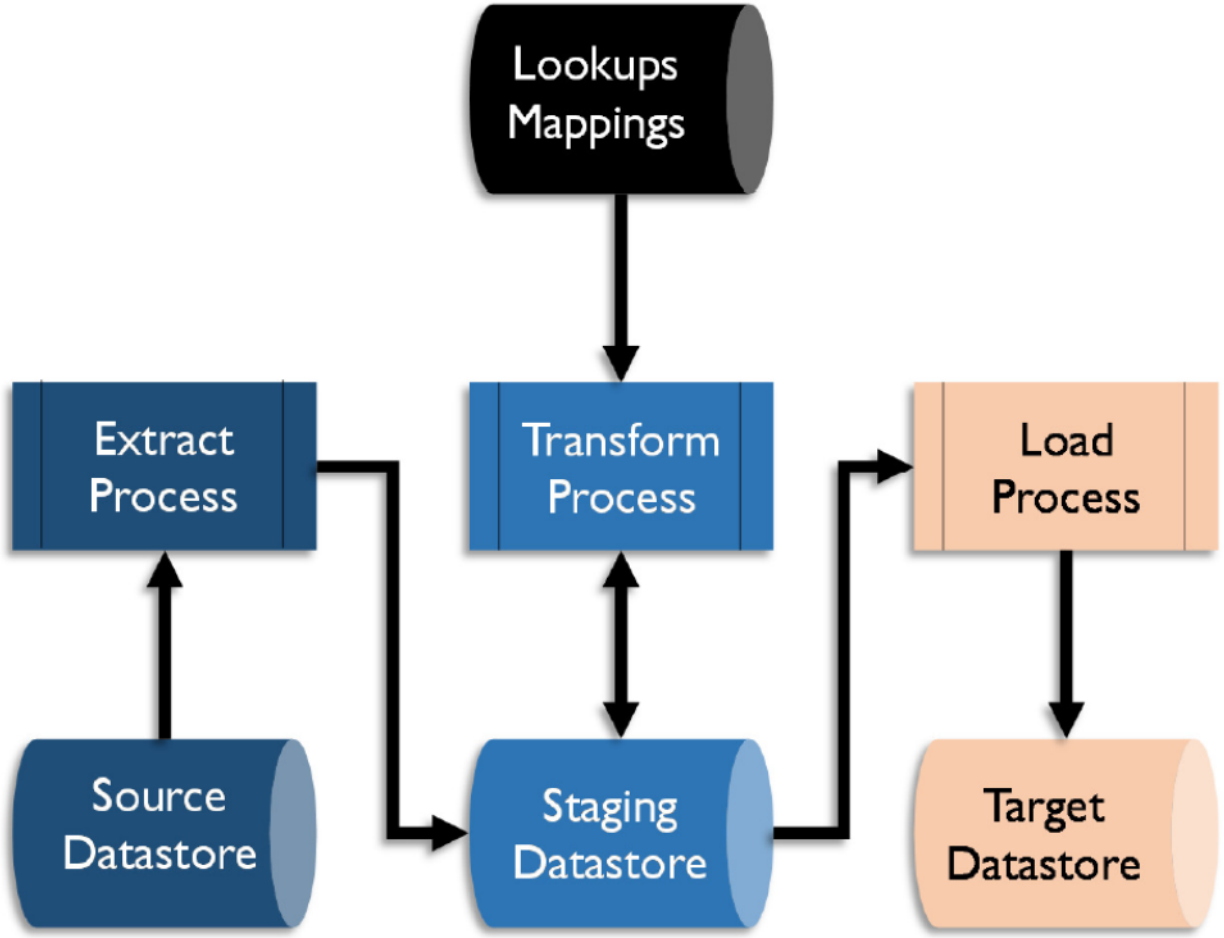
- Commercial
  - IBM DB2
  - Oracle RDMS
  - Microsoft SQL Server
  - Sybase SQL Anywhere
- Open Source (with commercial options)
  - MySQL, PostgreSQL, Ingres

Majority of enterprises and businesses in the world run SQL databases!

Many NoSQL Databases use the SQL-like commands and semantics

Modern Cloud based SQL databases approach performance and scalability of specialised NoSQL databases

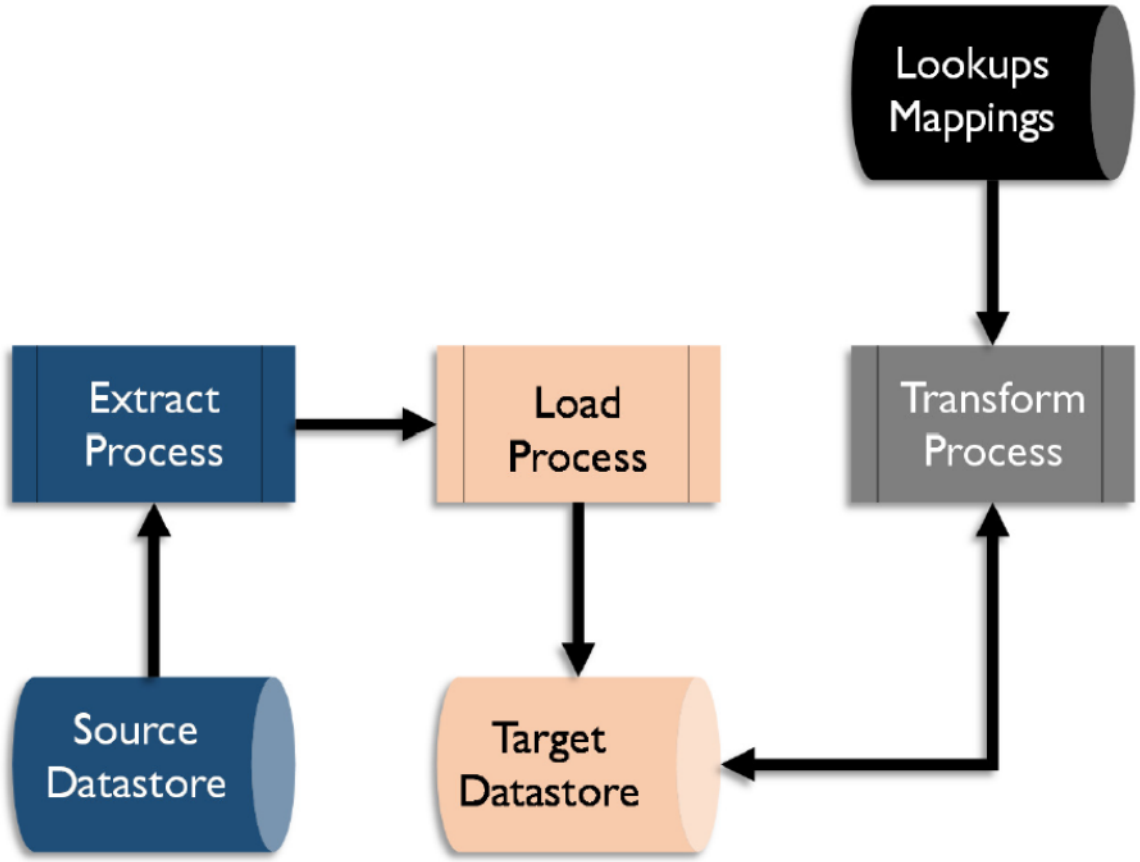
ETL Processes



Extract - Transfer – Load

- Primary model for RDBMS and SQL
- Enforces schema
  - Schema on write

ELT Processes



Extract – Load - Transfer

- Modern Big Data infrastructure and NoSQL databases
- Load in native data format and schema apply on read
- CEP Complex Events Processing
- Data Exchange Standards
- ESB – Enterprise Service Bus

SQL Scalability Issues in Cloud and at Web Scale

Scaling Up

- Issues with scaling up when the dataset is just too big
- RDBMS were not designed to be distributed
- Multi-node database solutions
  - Known as ‘scaling out’ or ‘horizontal scaling’
- Different approaches include
  - Master-slave
  - Sharding

Partitioning or sharding

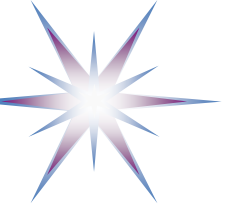
- Scales well for both reads and writes
- Not transparent, application needs to be partition-aware
- Can no longer have relationships/joins across partitions
- Loss of referential integrity across shards

Scaling 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 data sets can pose problems as master needs to duplicate data to slaves

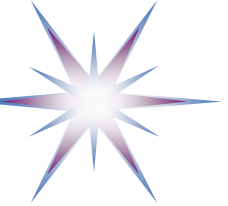
Other ways to scale RDBMS

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



Large scale SQL databases on cloud

- AWS Aurora
  - Terabyte scale SQL database
- Google Cloud Spanner
  - Globally-distributed, and strongly consistent database service
  - Combines SQL query model and strong geographical consistency
- IBM Big SQL
  - IBM MPP SQL database engine optimised to work with HDFS and HBase
- Azure SQL database
  - Traditional SQL database of Terabyte scale

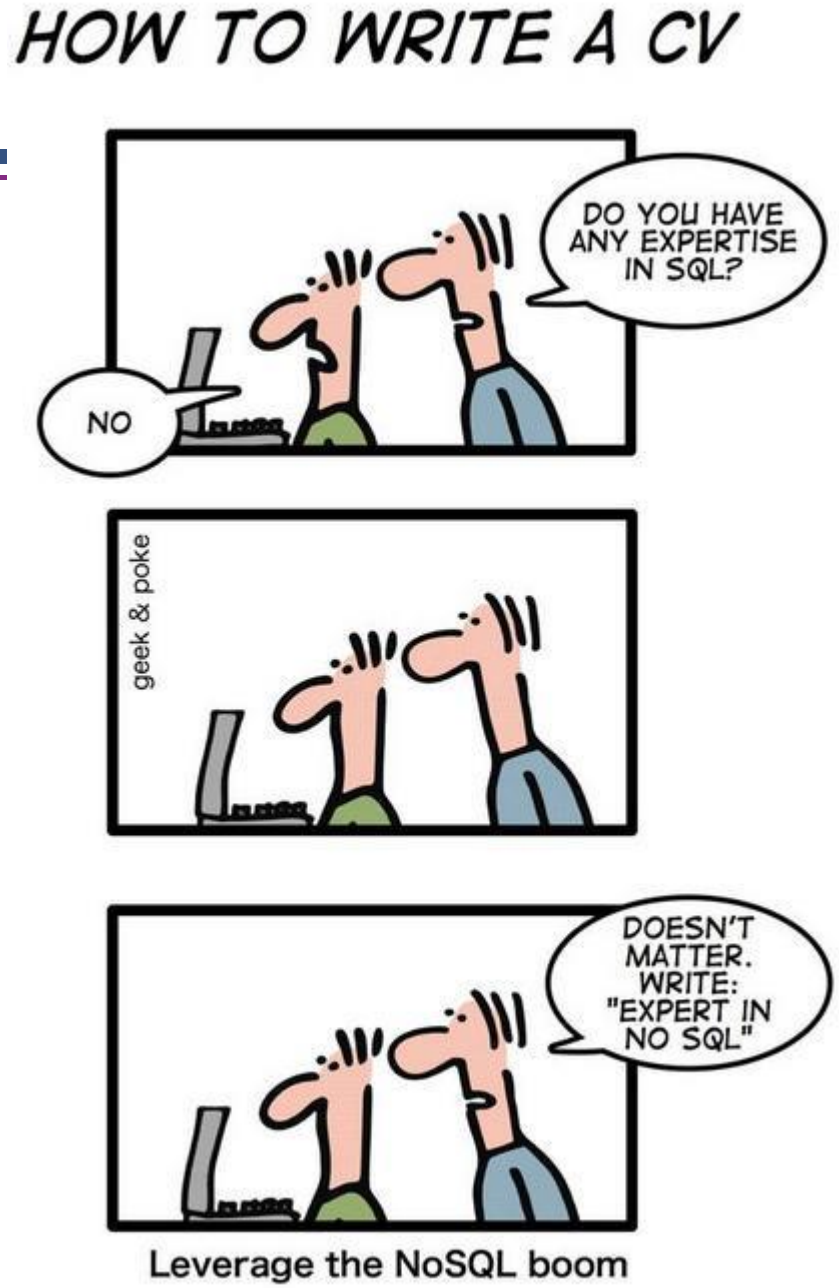


NoSQL Databases

- NoSQL definition
- BASE vs ACID
- CAP Theorem
- NoSQL databases overview
  - HBase
  - Cassandra
  - MongoDB
  - Apache Accumulo

NoSQL Databases

- NoSQL definition
- BASE vs ACID
- CAP Theorem
- NoSQL databases overview
  - HBase
  - Cassandra
  - MongoDB
  - Apache Accumulo



NoSQL Definition: Not only SQL

From www.nosql-database.org:

> Next Generation Databases mostly addressing some of the points:
>
> being **non-relational, distributed, open-source** and **horizontal scalable**. The original intention has been **modern web-scale databases**.
>
> The movement began early 2009 and is grown rapidly. Often more characteristics apply as: **schema-free, easy replication support, simple API, eventually consistent** / **BASE** (not ACID), a **huge data amount**, and more.

NoSQL Distinguishing Characteristics

- Large data volumes
  - Web scale “Big Data” (not only Google)
- Scalable replication and distribution – *However data skew*
  - Potentially thousands of machines
  - Potentially distributed around the world
- Queries need to return answers quickly
  - Not necessary precisely
  - Employing ***probabilistic*** search/decision
- Mostly query, few updates
- Asynchronous Inserts and Updates
- Schemaless – on write (*schema flexible on read*)
- Paradigm shift from ACID transaction properties to BASE
- CAP Theorem optimization and challenges
- Open source development

SQL Transactions – ACID Properties

RDBMS databases design principles

- **A**tomic – All of the work in a transaction completes (commit) or none of it completes
- **C**onsistent – A transaction transforms the database from one consistent state to another consistent state. Consistency is defined in terms of constraints/conditions.
- **I**solated – The results of any changes made during a transaction are not visible until the transaction has committed.
- **D**urable – The results of a committed transaction survive failures

- Relaxing ACID properties is a common architectural compromise where transaction capability is traded out for achieving higher performance and scalability.

BASE and ACID: Paradigm Shift

BASE Semantics

- **B**asically **A**vailable: Nodes in a distributed environment can go down, but the whole system shouldn't be affected
- **S**oft State: The state of the system and data changes over time
- **E**ventual Consistency: Given enough time, data will be consistent across the distributed system

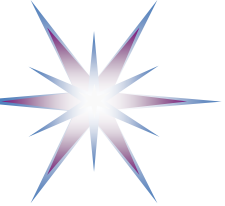
BASE and ACID Compared

ACID Properties

- Strong consistency
- Less availability
- Pessimistic concurrency
- Complex

BASE Properties

- Availability is the most important thing. Willing to sacrifice for this (CAP)
- Weaker consistency (Eventual)
- Best effort
- Simple and fast
- Optimistic



Consistency Model

- A consistency model determines rules for visibility and apparent order of updates.
- For example (sequence):
  - 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: Strict Consistency can't be achieved at the same time as availability and partition-tolerance.

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
- Part of BASE (**B**asically **A**vailable, **S**oft state, **E**ventual consistency) property, as opposed to ACID

Brewer's CAP Theorem

Brewer's (CAP) Theorem (original formulation) [ref]

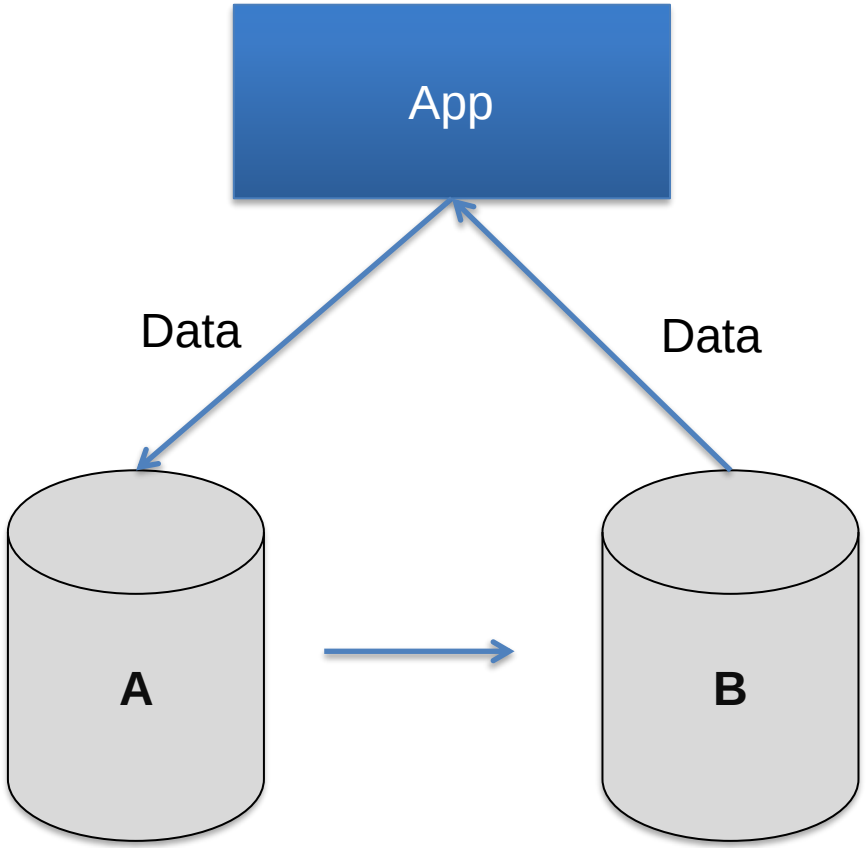
> *"There are three core systemic requirements that exist in a special relationship when it comes to designing and deploying applications in a distributed environment."*

A distributed system can support only two of the following characteristics:

- Consistency
  - All nodes see the same data at the same time
- Availability
  - Node failures do not prevent survivors from continuing to operate
- Partition tolerance
  - The system continues to operate despite arbitrary message loss

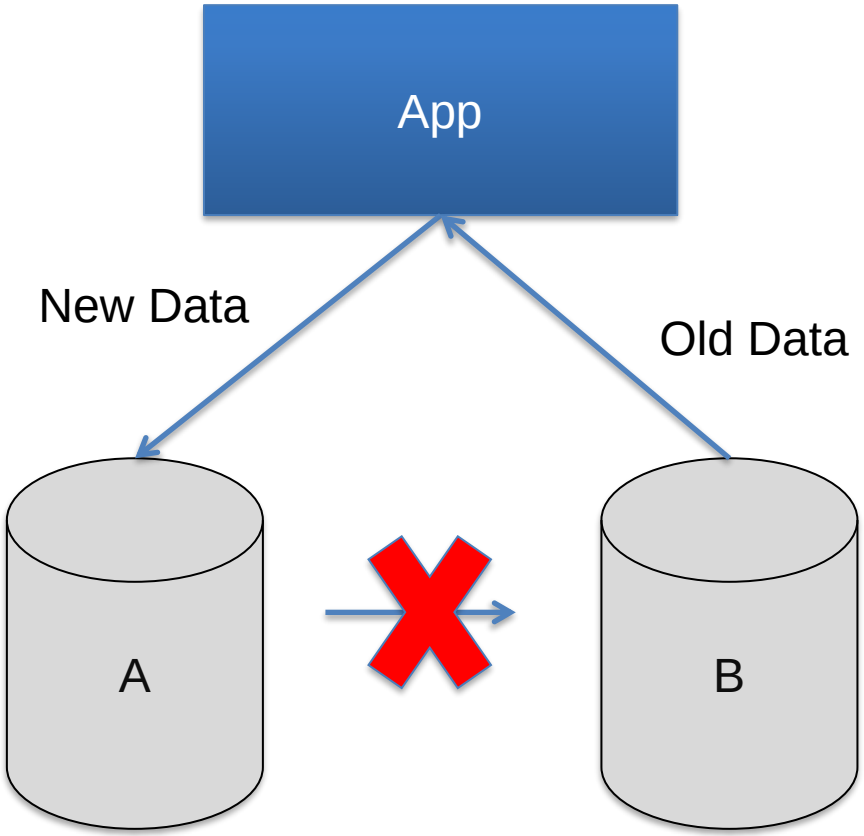
[ref] http://www.julianbrowne.com/article/viewer/brewers-cap-theorem

CAP: A Simple Proof



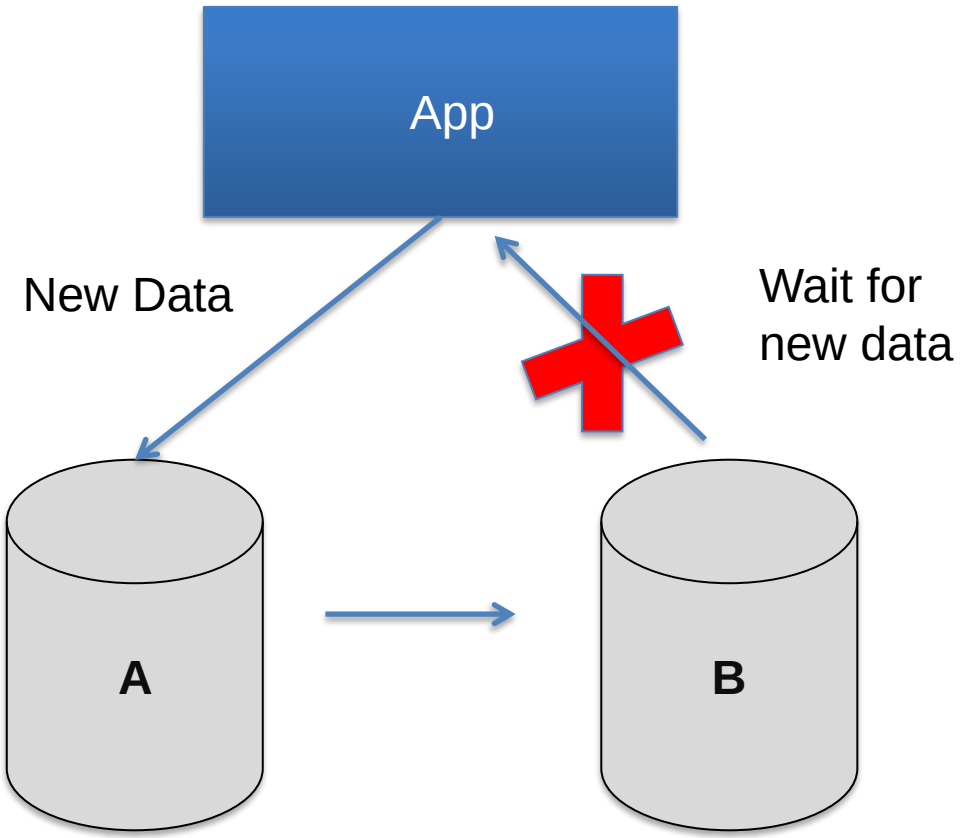
Consistent and available
No partition.

CAP: A Simple Proof



Available and partitioned
Not consistent, we get back old data.

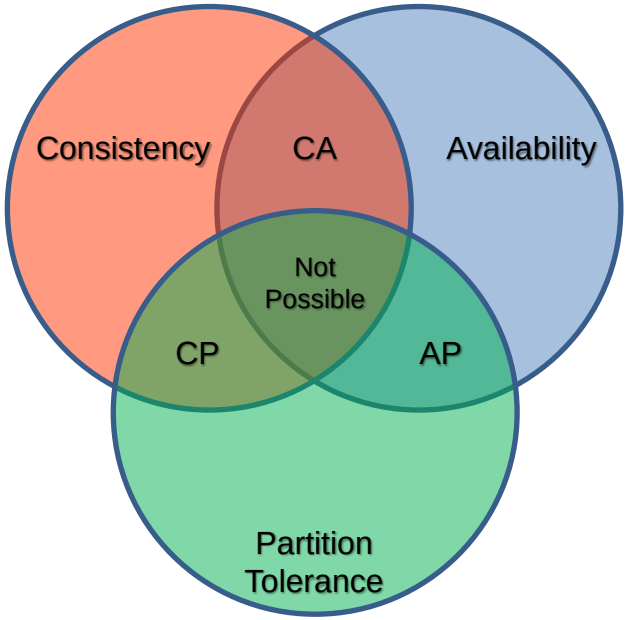
CAP: A Simple Proof



Consistent and partitioned
Not available, waiting…

CAP Theorem for Cloud (distributed) Storage and Databases

- In Cloud Computing, everything has to work at large scale, web or planet size, and in distributed form
- The different types of Cloud Storage often realise different tuples of CAP properties
  - For example, typical object storage sacrifices absolute consistency
  - Writes succeed before data is replicated
  - Copies eventually become globally consistent
- A consistency model determines rules for visibility and apparent order of updates
  - Strict Consistency – RDBMS
  - Eventual Consistency – Amazon Dynamo
  - Tunable Consistency – Azure CosmosDB, Cassandra, Google Spanner



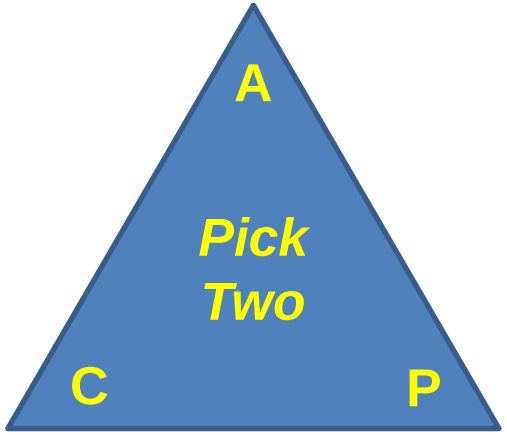
The CAP Theorem Dilemma in Cloud Database and Storage

Visualizing the CAP Theorem and it's members

Configurable consistency:
Azure CosmosDB (K,C,D,G), AWS Aurora (R), Google Spanner (R)

Data Models:
(R) Relational (Comparison)
(K) Key Value
(C) Column-Oriented/Tabular
(D) Document-Oriented
(G) Graph

Availability
Each client can always read and write



CA

(R) RDBMs such as MySQL, Oracle, DB2, PostgreSQL, SQL Server, etc.
(R) Aster Data
(R) Greenplum
(D) CouchDB
(C) Vertica

AP

(K) Dynamo
(K) Voldemort
(K) Tokyo Cabinet
(K) KAI
(C) Cassandra
(D) SimpleDB
(D) CouchDB
(D) Riak

Consistency
All clients always have the same view of data

Partition Tolerance
The system works well despite physical network partitions

CP

(C) BigTable
(C) Hypertable
(C) Hbase
(D) MongoDB
(D) Terastore
(K) Scalaris
(K) BerkeleyDB
(K) MemcacheDB
(K) Redis
(R) VoltDB

NoSQL Database - Main Types

Discussing NoSQL databases is complicated because there are a variety of types:

- **Column Store** – Each storage block contains data from only one column (family)
- **Document Store** – Stores documents made up of tagged elements
- **Key-Value Store** – Hash table of keys
- **Graph Databases** – Graph

NoSQL Databases Overview

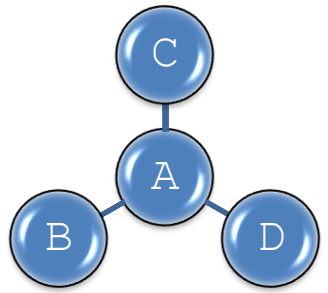
Key Value Store

- Data is stored in key/value pairs
- Designed to handled large data quantities and heavy load
- Based on Amazon's Dynamo Paper
- Example: Voldermort, developed by LinkedIn

Key	Value
Name	Fred Foobar
Age	21
Member Number	453339
Member Since	2011

Graph

- Focus on modeling data and associated connections
- Based on mathematical principles of Graph Theory
- Example: FlockDB developed by Twitter, Neo4J



BigTable / Column

- Data is grouped in Columns, not Rows
- Based on the BigTable paper from Google
- Example: Apache HBase, Apache Cassandra, originally developed by Facebook

Column Family			
Row Key	Column Name		
	Key	Key	Key
	Value	Value	Value
	Column Name		
	Key	Key	Key
	Value	Value	Value

Document

- Data is stored as whole documents
- JSON and XML are popular formats
- Maps well to Object Oriented programming model
- Example: CouchDB Apache project

```
{
"id": "123",
"name":      Fred Foobar",
"dob" :      {
             "year":      1985,
             "month":     5,
             "day":       12
             }
}
```

Data Models and Database Complexity

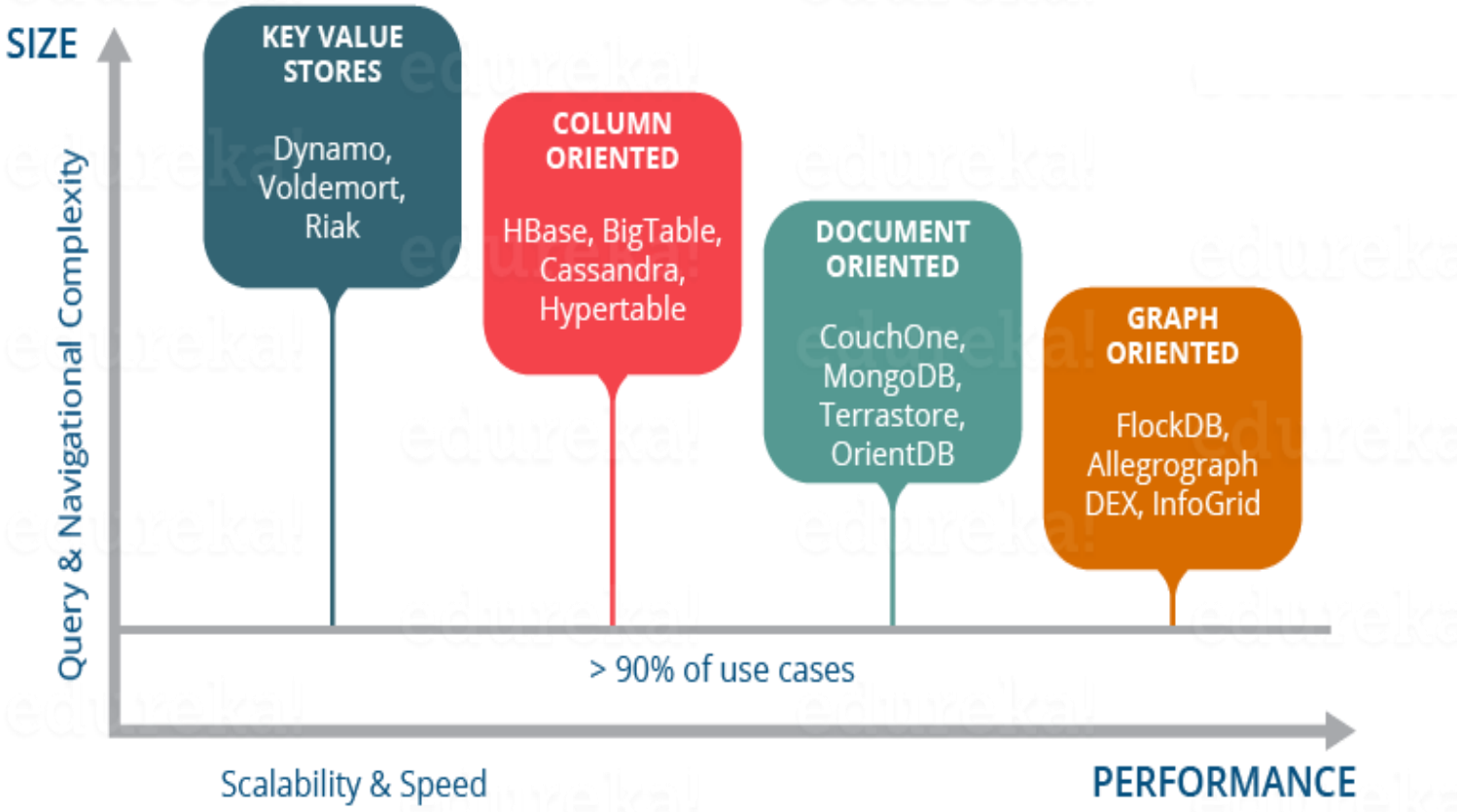


Figure: Comparison of different types of NoSQL databases

Source: Edureka

Data models and corresponding data stores

- Key/Value Pairs
- Documents
- Columns
- Objects
- Graphs

Row oriented vs Column oriented

Row-oriented data stores – Data is stored and retrieved one row at a time and hence could read unnecessary data if only some of the data in a row is required.

- Easy to read and write records
- ***Well suited for OLTP systems (Online Transaction Processing)***
- Not efficient in performing operations applicable to the entire dataset (because of typically many tables) and hence aggregation is an expensive operation
- Typical compression mechanisms provide less effective results than those on column-oriented data stores

Column-oriented data stores – Data is stored and retrieved in columns and hence can read only relevant data if only some data is required

- Read and Write are typically slower operations
- ***Well suited for OLAP systems (Online Analytical Processing)***
- Can efficiently perform operations applicable to the entire dataset and hence enables aggregation over many rows and columns
- Permits high compression rates due to few distinct values in columns

Apache HBase

- Open-source clone of the Google BigTable
- *Implemented on top of Hadoop Distributed File System (HDFS)*

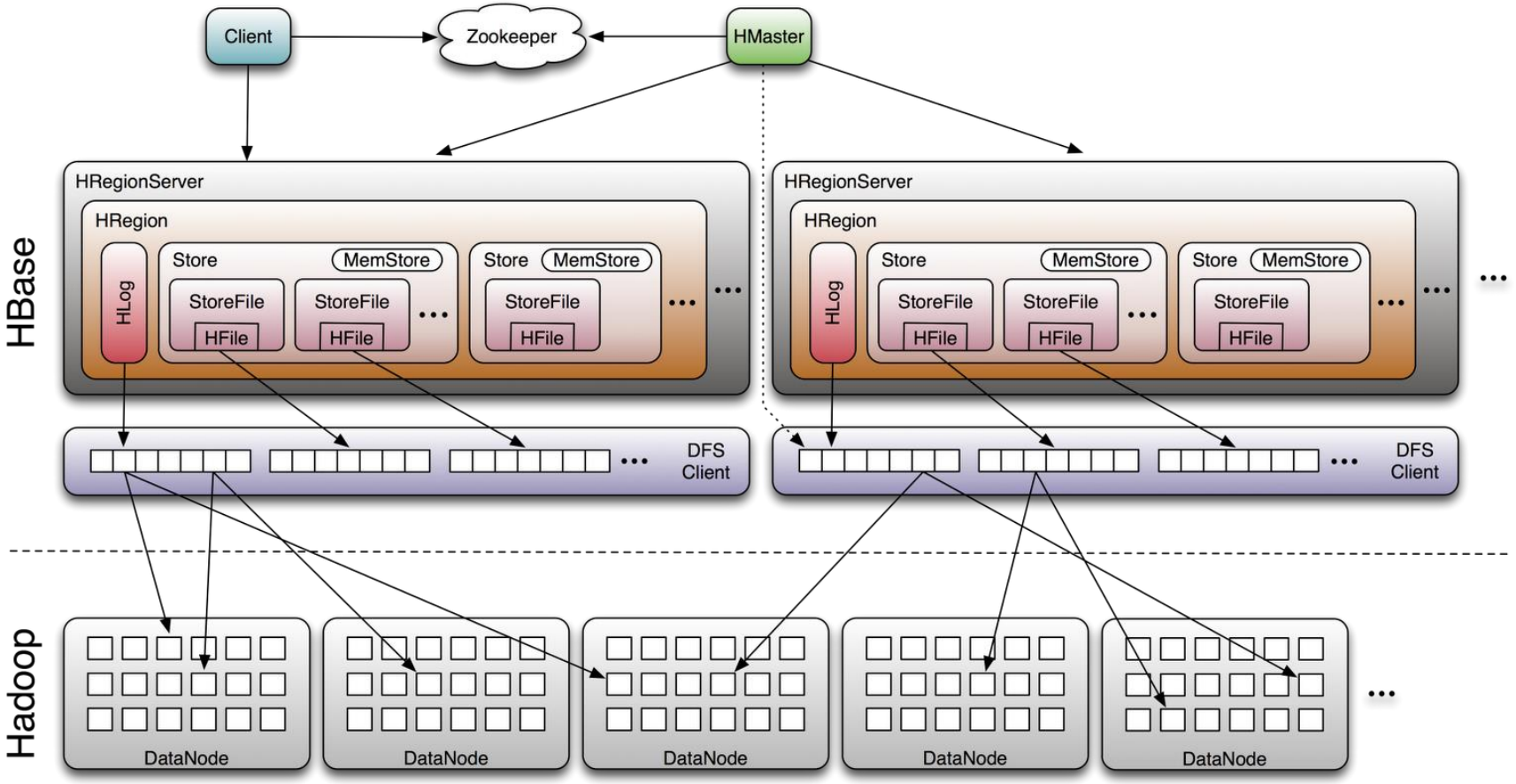


Image Source: http://www.larsgeorge.com/2009/10/hbase-architecture-101-storage.html

HBase/BigTable Columnar Database

- Apache HBase and BigTable
  - Cheap, can use commodity equipment
- Data are replicated to multiple nodes (therefore identical and fault-tolerant) and can be partitioned
  - Down nodes easily replaced
  - No single point of failure
- Easy to distribute
- Doesn't require a schema
- Can scale up and down
- Relies on Relaxation of the data consistency requirement in CAP

HBase Data Model

- A table in BigTable is a sparse, distributed, persistent multidimensional sorted map
- Map indexed by a row key, column key, and a timestamp
  - (row, column, timestamp) → cell contents
- Supports lookups, inserts, deletes
  - Single row transactions only
- Good match for most of Google's applications

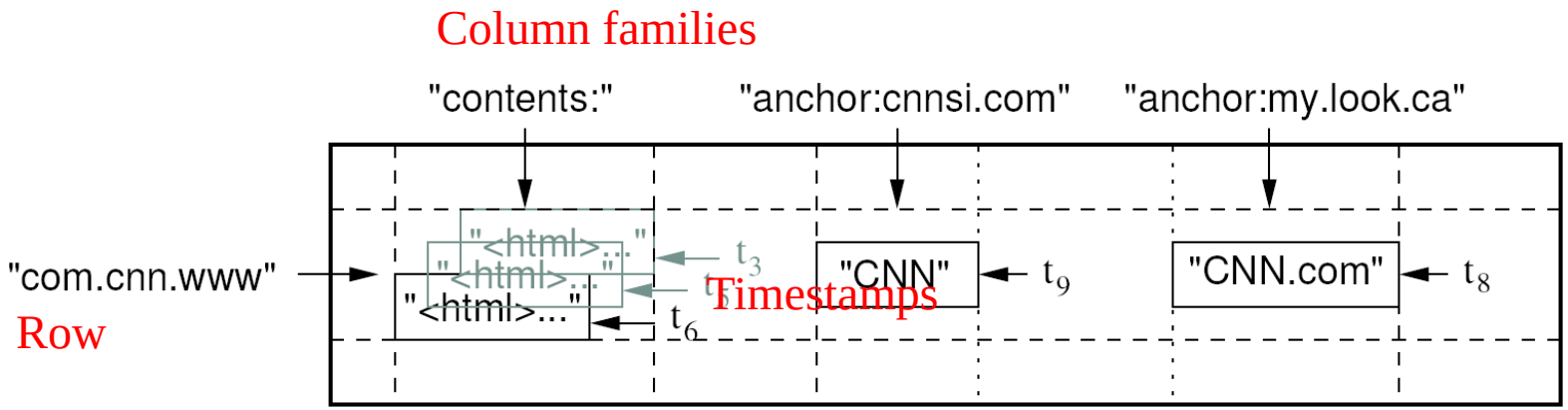


Image Source: Chang et al., OSDI 2006

Rows, Columns, Column Family

- Rows maintained in a sorted lexicographic order
  - Everything is a String
  - Every row has a single key
  - Row ranges dynamically partitioned into tablets
  - Rows close together lexicographically usually on one or a small number of machines
- Columns grouped into column families
  - Column key = *family:qualifier*
  - Column names are arbitrary strings
  - Data in the same locality group are stored together
  - Unbounded number of columns
- Column Family
  - Must be created before any column in the family can be written
  - Basic unit of access control and usage accounting
    - different applications need access to different column families
    - careful with sensitive data

Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

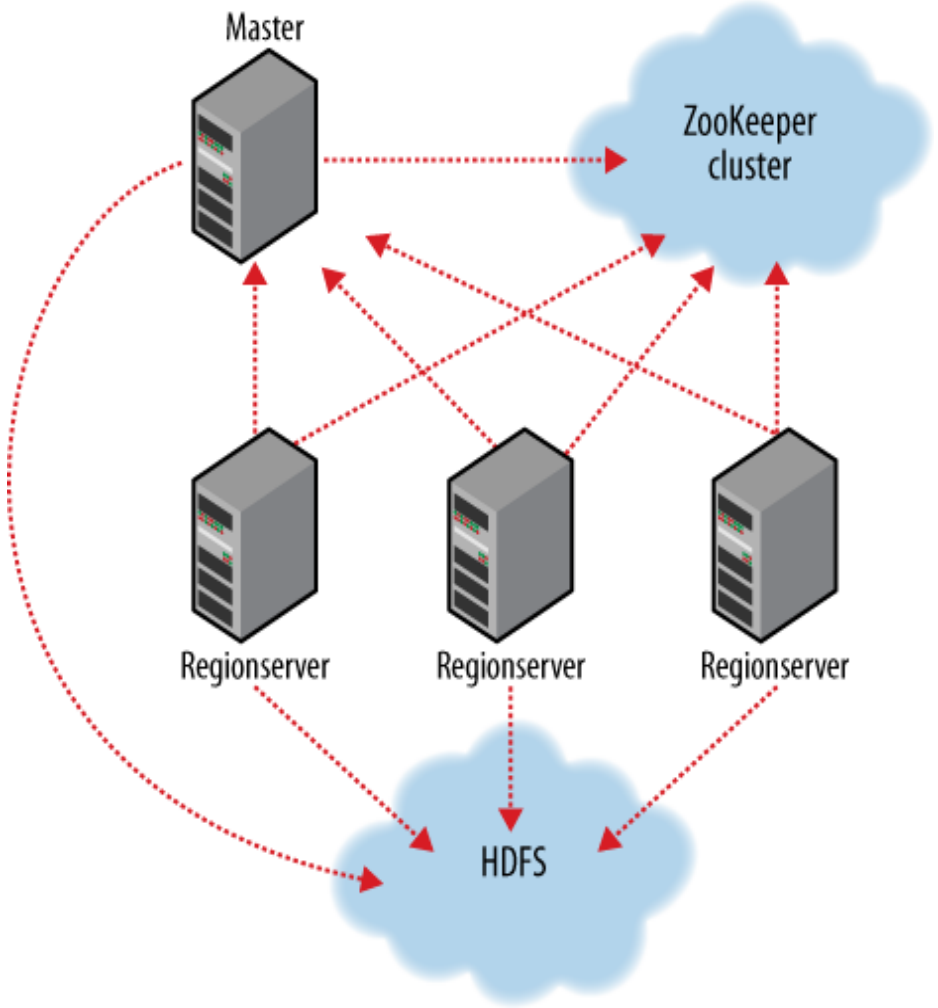
Column Families

ZooKeeper and HBase

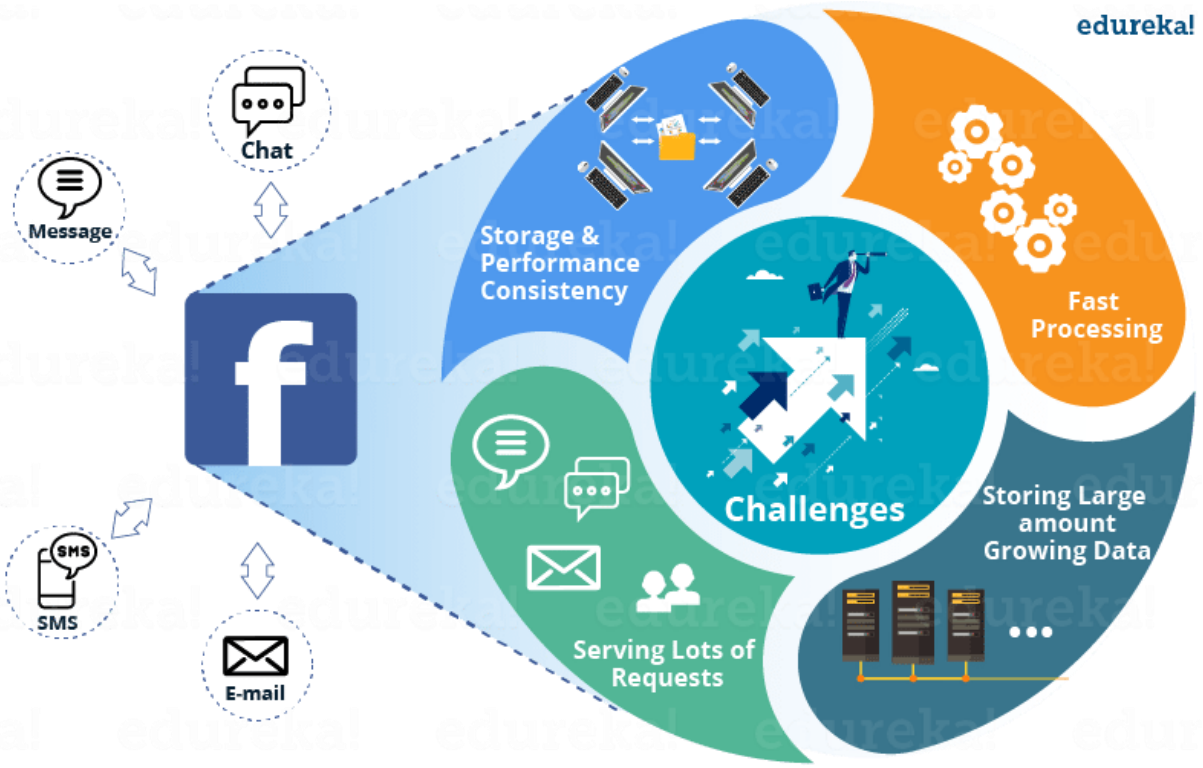
- HBase uses ZooKeeper to manage the authority on cluster state

Master Failover

- Region Servers and Master discovery via ZooKeeper
  - HBase clients connect to ZooKeeper to find configuration data
  - Region Servers and Master failure detection



Challenges faced by Facebook Messenger



- Storing the large sets of continuously growing data from various Facebook services.
- Requires Database which can leverage high processing on it.
- High performance needed to serve millions of requests.
- Maintaining consistency in storage and performance.
- HBase since 2010
- Recently, moved to MyRocks, Facebook's open source database project that integrates RocksDB as a MySQL storage engine

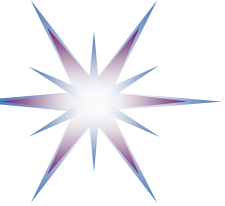
Source: Edureka

Cassandra: Hybrid HBase (C) and DynamoDB (K)

- Originally developed at Facebook
  - Now an Apache Open Source project
- Follows the HBase data model: column-oriented
- Uses the DynamoDB Eventual Consistency model
- Written in Java
- Open-sourced and exists within the Apache family
- Uses Apache Thrift as it's API
- Good integration with Hadoop stack: Pig Latin script and Hive query

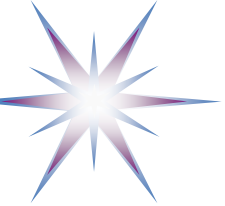
HBase
- Strong consistency
- Sparse map data model
- HDFS, Chubby, et al

DynamoDB
- O(1) distributed hash table (DHT) - scales w/o limit to data size
- BASE (i.e. eventually consistent)
- Client tunable consistency/availability



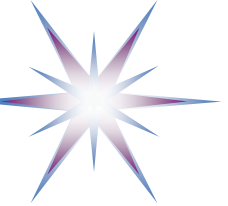
Cassandra Properties

- Big Table data model, runs on Dynamo
- High availability (eventual consistency)
- Eventually consistent, tunable consistency
- Partition tolerant
- Linearly scalable
- Uses consistent hashing (logical partitioning) when clustered
- Writes directly to the FS
- No single point of failure
- Ease of administration
- Flexible partitioning, replica placement
- Multi data center support



Cassandra Data Model

- Column is a basic unit of storage
- **ColumnFamily**: There's a single structure used to group both the Columns and SuperColumns. Called a ColumnFamily (think table), it has two types, Standard & Super
  - Column families must be defined at startup
  - Record-level Atomicity
  - Indexed
- **Key**: the permanent name of the record
- **Keyspace**: the outer-most level of organization. This is usually the name of the application. For example, 'Acme' (think database name)



Cassandra and Consistency

- Implements BASE and eventual consistency
- Cassandra has programmable read/writable consistency
  - **One:** Return from the first node that responds
  - **Quorom:** Query from all nodes and respond with the one that has latest timestamp once a majority of nodes responded
  - **All:** Query from all nodes and respond with the one that has latest timestamp once all nodes responded. An unresponsive node will fail the node

Document Stores

Document oriented databases is an example of the schema free database which means that the records don't have uniform structure and each record can have different column value types, and records may have a nested structure.

- Each record or document can have its own schema or structure that can be defined by the XML schema or JSON script
- Query Model: JavaScript or custom.
- Integrates with Map/Reduce
- Indexes are done via B-Trees
- Products/Implementations
  - MongoDB
  - CouchDB
  - CosmosDB (former DocumentDB)

MongoDB: an example of document oriented DB

- Data types: bool, int, double, string, object(bson), oid, array, null, date.
- Integrates with multiple languages
  - Written in C++
  - Native Python bindings
  - Supports aggregation with MapReduce with JavaScript
- Connection pooling
- Supports indexes, B-Trees. IDs are always indexed.
- Updates are atomic. Low contention locks.
- Querying mongo done with a document:
  - Lazy, returns a cursor, can iterate on cursor to obtain more details
  - Reducible to SQL, select, insert, update limit, sort etc.
  - Several operators:
    - $ne, $and, $or, $lt, $gt, $incr, $decr, and so on.
- Repository Pattern makes development very easy.

Key-Value Stores

Key-value databases represent a wide range of databases that are simply described as item-based. All necessary information is stored in the item. This simplifies data processing as there is no need for JOIN operations always present in SQL databases.

- Memcached – Key value stores
- Membase – Memcached with persistence and improved consistent hashing
- AppFabric Cache – Multi region Cache
- Redis – Data structure server
- Riak – Based on Amazon's Dynamo
- Project Voldemort – eventual consistent key value stores, auto scaling
- **Apache Accumulo**

Apache Accumulo (K)

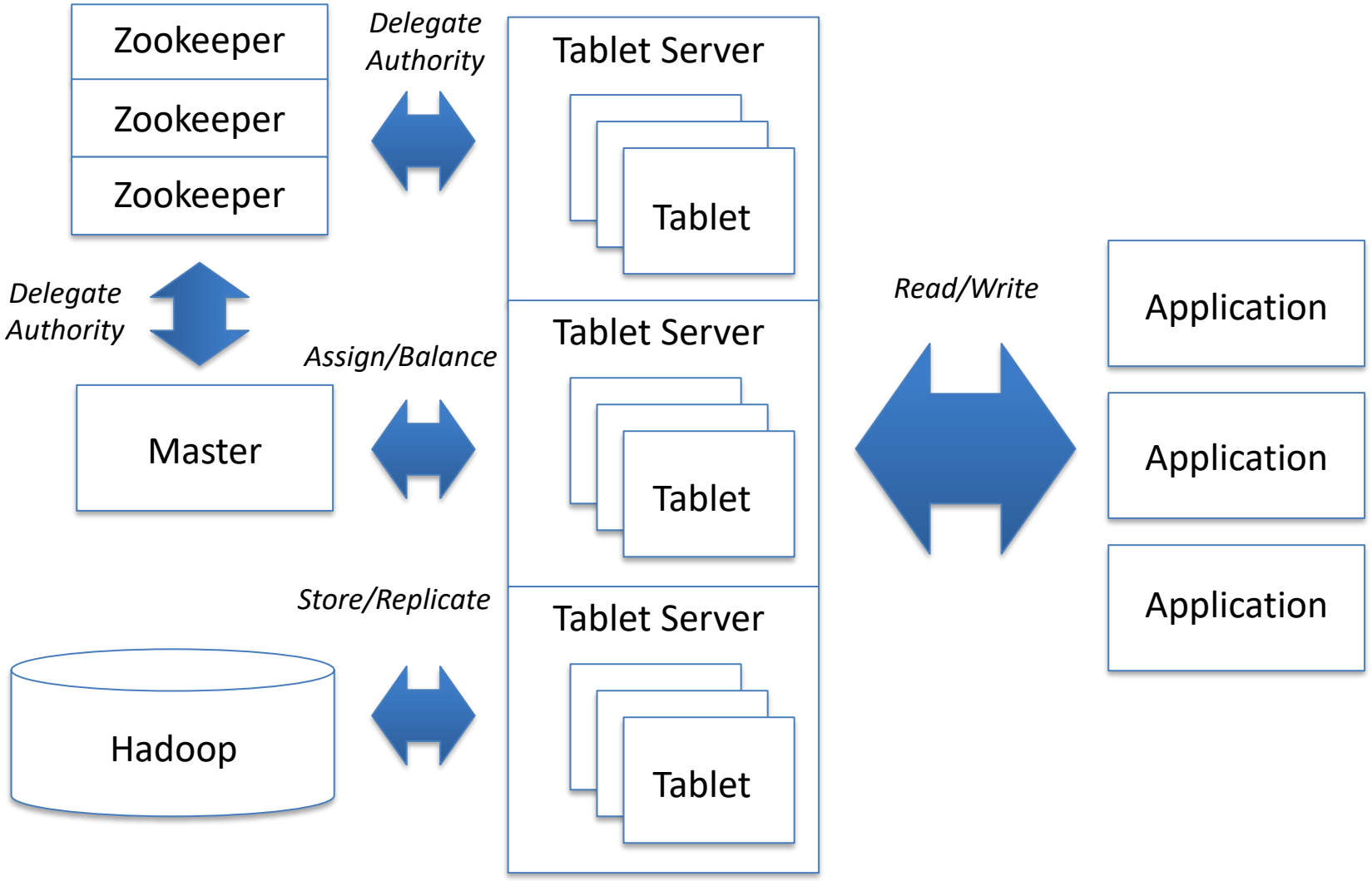
Sorted, distributed key/value store with cell-based access control and customizable server-side processing

- Developed by NSA (National Security Agency), also open source
- Based on BigTable, HBase
- **Iterator framework**: embeds user-programmed functionality into different LSM-tree (log-structured merge-tree) stages
- For example, iterators can operate during minor compactions by using the memstore data as input to generate on-disk store files comprised of some transformation of the input such as statistics or additional indices
- Enables interactive access to Trillions of records or Petabytes of indexed data across 100s-1000s of servers

Multi-dimensional Key

<table>
<tr><th colspan="4">Key</th><th rowspan="2">Value</th></tr>
<tr><th rowspan="2">Row ID</th><th colspan="3">Column</th><th rowspan="2">Timestamp</th></tr>
<tr><th>Family</th><th>Qualifier</th><th>Visibility</th></tr>
</table>

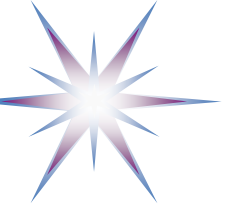
Accumulo Architecture



Graph Databases

Graph databases are used to represent relational graphs and are optimized for finding relations between graph nodes, e.g. shortest paths, or chain of trust.

- Based on Graph Theory
  - Store data in Triplestores or Quad stores
- Scale vertically, no clustering
- Typically achieve/require high consistency (due to feature critical to graph presentation and analysis)
- Graph algorithms can be used naturally

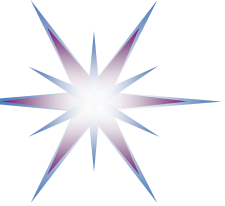


Neo4J

Neo4j is a highly scalable, robust (fully ACID) native graph database, used in mission-critical apps by thousands of leading startups, enterprises, and governments around the world [ref]

- High Performance for highly connected data
  - Traverses 1,000,000+ relationships / second on commodity hardware
- High Availability clustering
- Schema free, bottom-up data model design
- Cypher, a declarative graph query language that allows for expressive and efficient querying and updating of the graph store
- Graph traversal function to visit all nodes on a specified path, updating and/or checking their values
- ETL, easy import with Cypher LOAD CSV
- Hot Backups and Advanced Monitoring
- HTTP/REST protocol for data access and administration
- Embeddable and server

[ref] http://www.neo4j.org/



Modern Cloud Databases and CAP Theorem Challenges

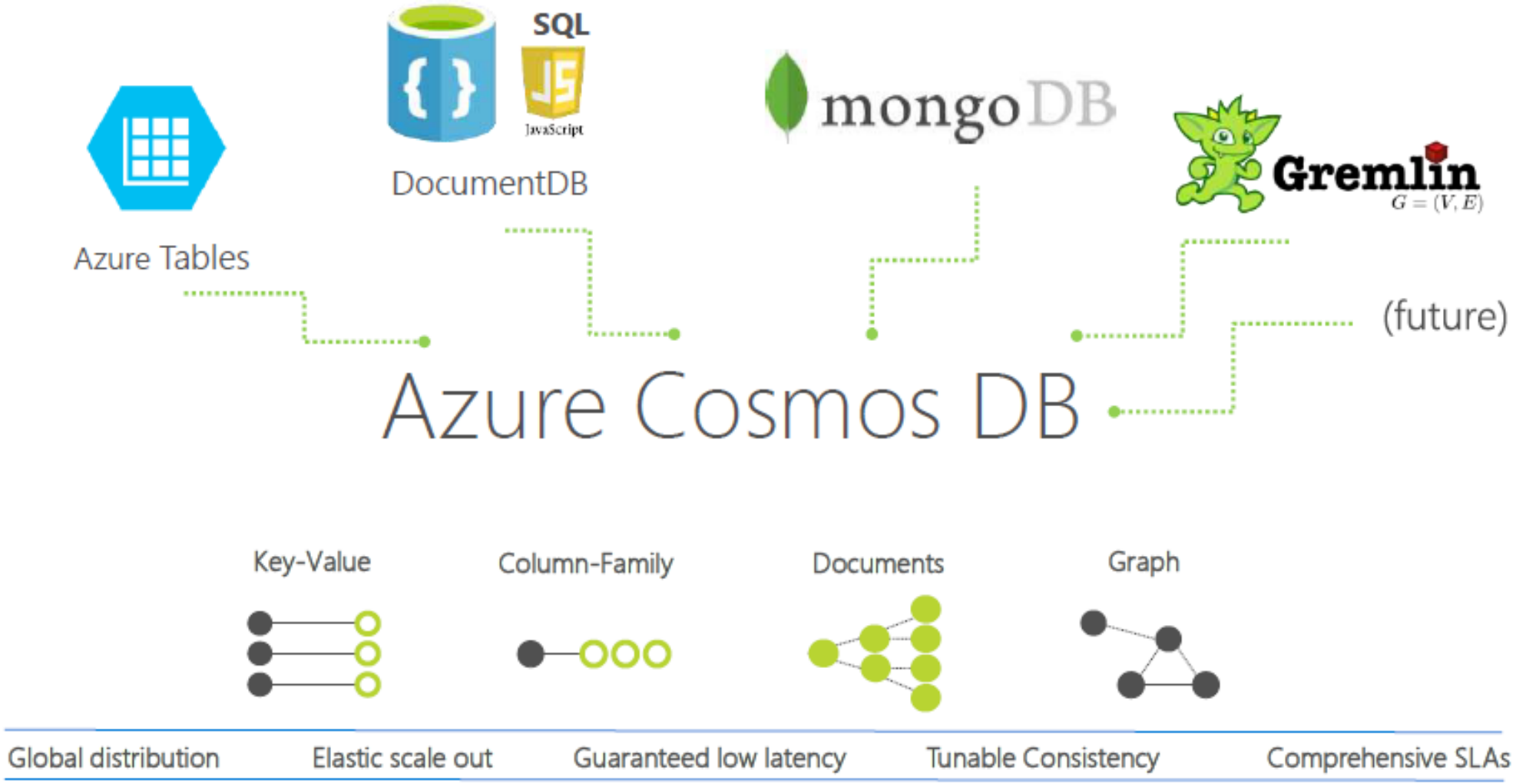
- Google Spanner database
- Azure CosmosDB (with underlying blob storage and HDFS)
- AWS Aurora big SQL database

Google Cloud Spanner - Mission Critical RDBMS

- SQL v/s NoSQL Databases
- Best of both worlds
- Strong consistency and high availability worldwide

	CLOUD SPANNER	TRADITIONAL RELATIONAL	TRADITIONAL NON-RELATIONAL
Schema	✓ **Yes**	✓ Yes	✗ No
SQL	✓ **Yes**	✓ Yes	✗ No
Consistency	✓ **Strong**	✓ Strong	✗ Eventual
Availability	✓ **High**	✗ Failover	✓ High
Scalability	✓ **Horizontal**	✗ Vertical	✓ Horizontal
Replication	✓ **Automatic**	↻ Configurable	↻ Configurable

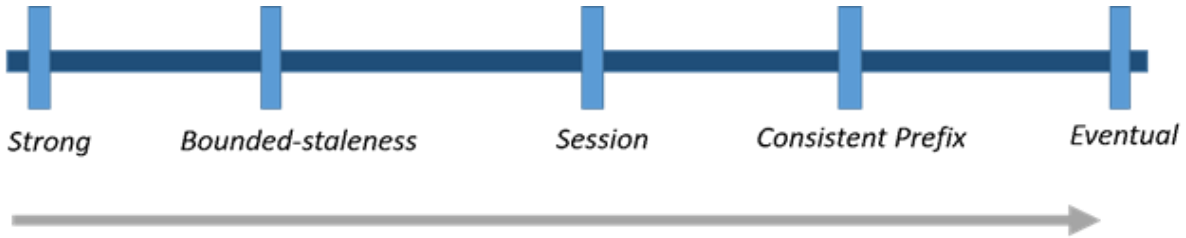
Azure CosmosDB (former DocumentDB) – Multi-model global distributed database



CosmosDB: Consistency and latency

Five Consistency Models

- Helps navigate Brewer's CAP theorem
- Intuitive Programming
  - Tunable well-defined consistency levels
  - Override on per-request basis
- Clear PACELC tradeoffs
  - Partition – Availability vs Consistency
  - Else – Latency vs Consistency



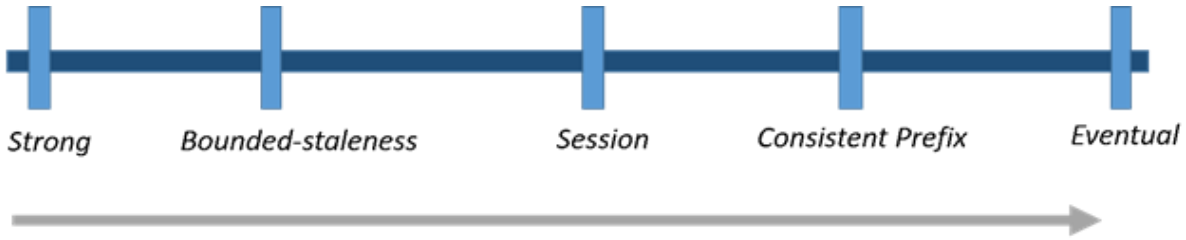
- Guaranteed low latency at P50 and P99 percentile
- Globally distributed with requests served from local region
- Write optimized, latch-free database
- Automatic Indexing

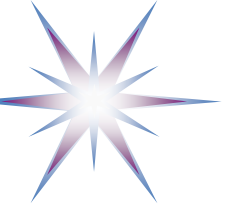
	Reads (1KB)	Indexed Writes (1KB)
P50	<2ms	<6ms
P99	<10ms	<15ms

CosmosDB: Consistency levels and guarantees

Consistency Level	Guarantees
Strong	Linearizability
Bounded Staleness	Consistent Prefix. Reads lag behind writes by k prefixes or t interval
Session	Consistent Prefix. Monotonic reads, monotonic writes, read-your-writes, write-follows-reads
Consistent Prefix	Updates returned are some prefix of all the updates, with no gaps
Eventual	Out of order reads

Lower latency, higher availability, better read scalability





Amazon Aurora – Big mySQL database

- High performance and
- High availability and durability
- High security
- MySQL and PostgreSQL Compatible
- Fully managed
- Migration support

Summary and take away

- Multiple data types require different types of data stores or databases
- SQL or Relational databases serve majority of current business and enterprise purposes
- Abrupt data growth and advent of Big Data technologies motivates new type of databases NoSQL (Not only SQL) to serve different types of data: key-value, columnar, document, graph
- CAP Theorem provides a basis for defining properties of the distributed system i.e. storage and databases such as Consistency, Availability and Partitioning tolerance
- Columnar databases HBase, Cassandra are designed to handle web scale data volume
  - They are specifically adopted to work with such scalable parallel processing systems as Hadoop
- MongoDB is a highly scalable schema free document oriented databases where the each record can have own schema or structure that can be defined by the XML schema or JSON script
- Neo4j is an example of the Graph databases that have growing use for social network and business relations analysis

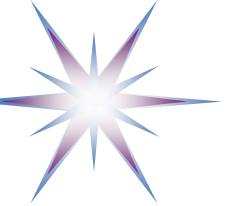
Discussion Questions

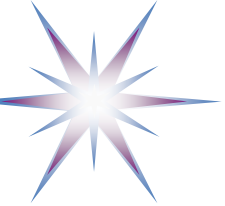
- What is the specifics with the Big Data? We have already 1x TB database and this goes well with current Data Warehouse and analytics. We don’t see needs for any NoSQL database or cloud.
- What are the reasons to have so many NoSQL databases?
- How do I select them?
- Why CosmosDB claims such versatility? Why it is needed?
- How can you claim that databases located in LA and in London can be consistent?
- Can you give an example of how CAP theorem can be applied in life?

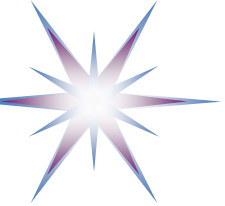
Acknowledgement

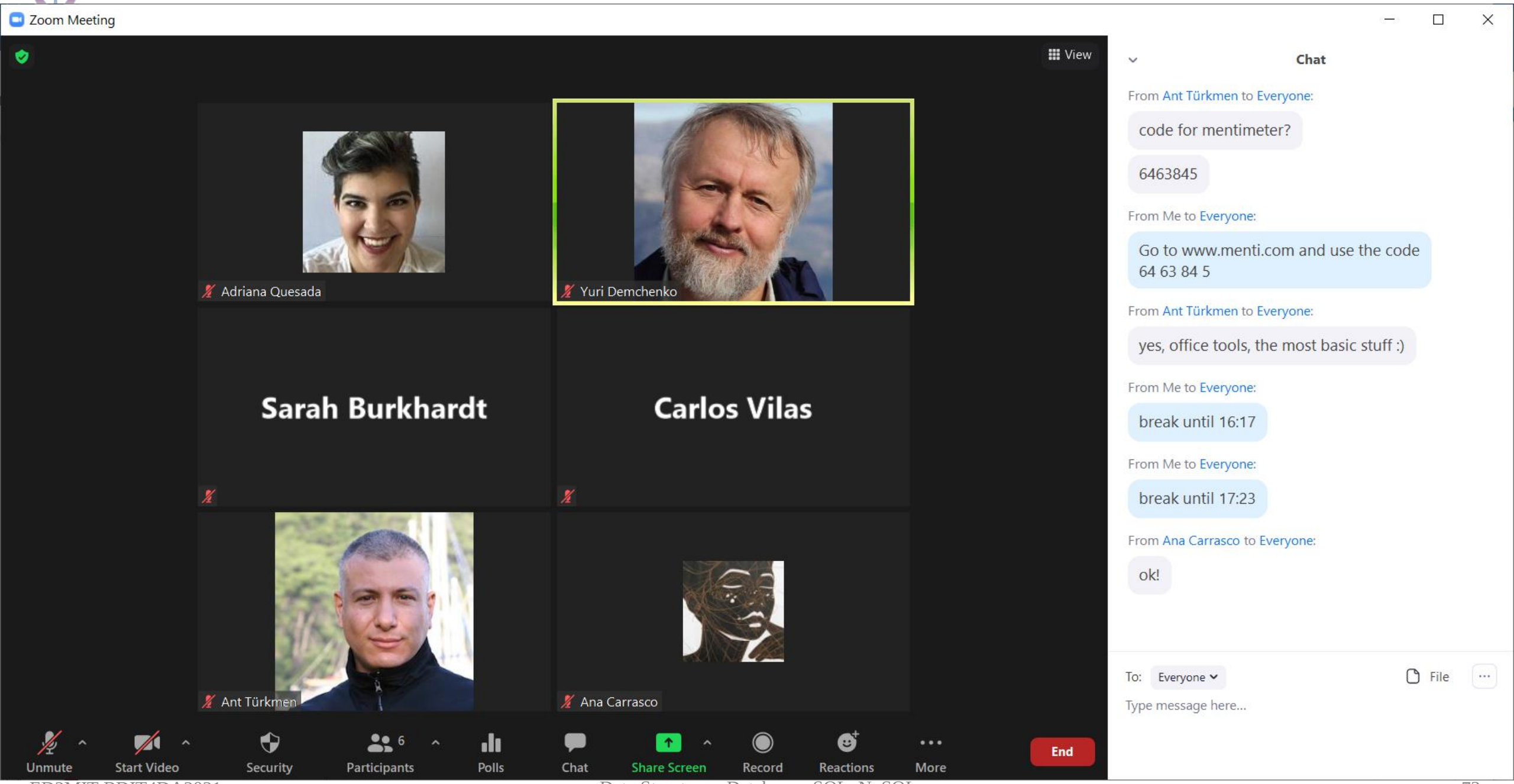
- This work is supported by the ERASMUS+ MATES project

- The work is committed to the Open Source under Creative Commons 4.0 CC BY license








Zoom Meeting
View
Adriana Quesada
Yuri Demchenko
Sarah Burkhardt
Carlos Vilas
Ant Türkmen
Ana Carrasco
Unmute
Start Video
Security
Participants
6
Polls
Chat
Share Screen
Record
Reactions
More
End
Chat
From Ant Türkmen to Everyone:
code for mentimeter?
6463845
From Me to Everyone:
Go to www.menti.com and use the code 64 63 84 5
From Ant Türkmen to Everyone:
yes, office tools, the most basic stuff :)
From Me to Everyone:
break until 16:17
From Me to Everyone:
break until 17:23
From Ana Carrasco to Everyone:
ok!
To: Everyone
File
Type message here...