CM2606 Data Engineering

Big Data Storage 2

Week 04 | Piumi Nanayakkara

Learning Outcomes

- Covers LO1 and L02 for Module

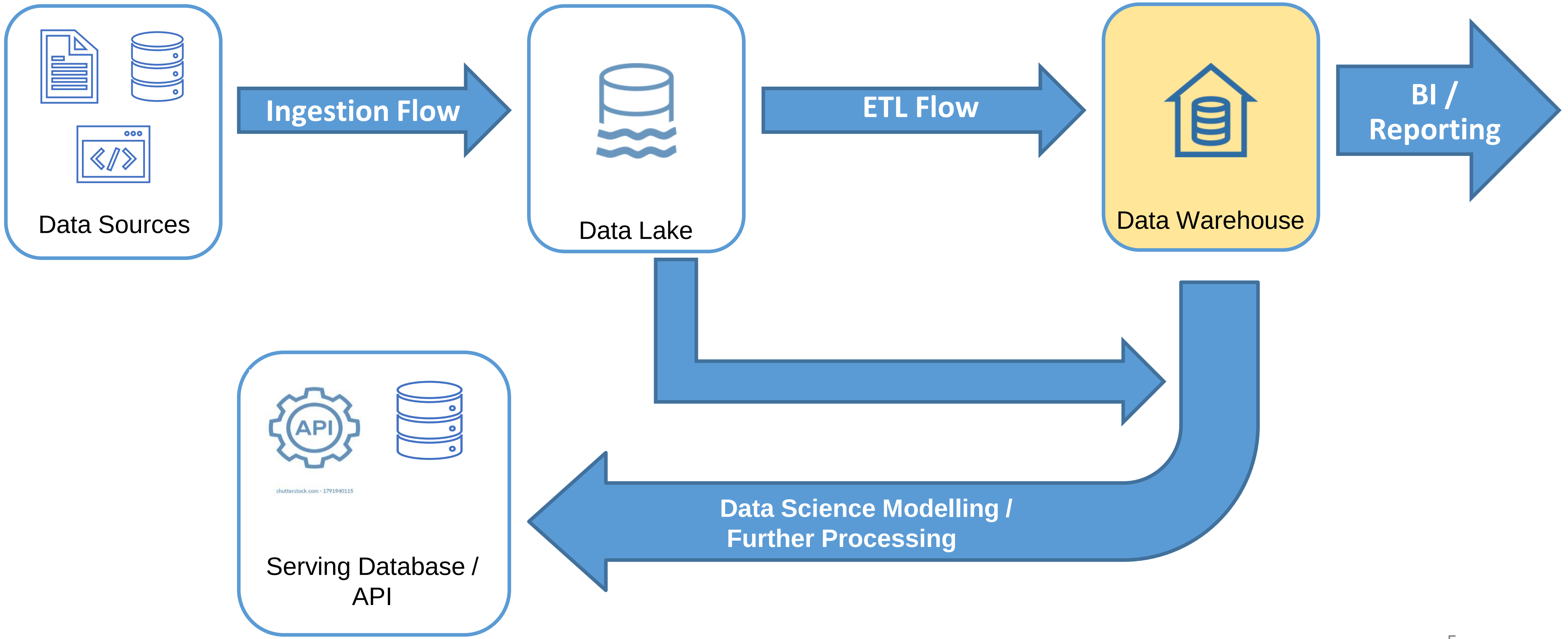
- On completion of this lecture, students are expected to be able to:
  - Explain the concept of a Datawarehouse
  - Understand how an open source Datawarehouse setup would work
  - Evaluate different options available to be used as a serving storage and choose the appropriate one

Content

- Data Warehouse
  - Hive
- Serving / Staging Databases
  - Database Sharding
  - NOSQL Databases: Theory
  - NOSQL Database: Types and Examples
- Considerations

Data Warehouse

Data Pipeline: Common Usage



Data Warehouse

- A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process.
- **Subject-Oriented**: used to analyze a particular subject area. E.g., Sales
- **Integrated:** Integrates data from multiple data sources.
- **Time-Variant:** Historical data is kept
- **Non-volatile:** Once data is in the data warehouse, it will not change.

Solutions

- Oracle Data Warehouse – Fully autonomous data warehouse. Can be used cloud or on-prem
- Snowflake – Cloud based data platform providing elastic scaling
- Teradata – RDBMS enables parallel processing based on MPP (Massively Parallel Processing) architecture
- Vertica – Column based storage with MPP architecture
- Hive – Open source / Hadoop Eco System

Open-Source Implementation – Based on Hive

- Apache Hive is a data warehouse software project built on top of Hadoop for providing data query and analysis.
- Designed to work quickly on petabytes of data
- Access to files stored either directly in HDFS
  - Support multiple data formats.
  - Built in connectors for CSV/TSV, text files, Parquet, ORC and other formats.

Open-Source Implementation – Based on Hive

- Tools to enable easy access to data via SQL
  - SQL-like query language called HiveQL
  - Query execution via Apache Tez, Apache Spark, or MapReduce
  - Hive's SQL can also be extended with user code via user defined functions (UDFs), user defined aggregates (UDAFs), and user defined table functions (UDTFs).

Open-Source Implementation – Based on Hive

- Driver
  - Acts like a controller which receives the HiveQL statements. The driver starts the execution of the statement by creating sessions.
  - It monitors the life cycle and progress of the execution.
  - Driver stores the necessary metadata generated during the execution of a HiveQL statement

- Meta store
  - Stores metadata for
    - each of the tables like their schema and location
    - Attribute level data like data types
  - Helps the driver to keep a track of the data

Hive: Components

- Compiler
  - It performs the compilation of the HiveQL query. This converts the query to an execution plan. The plan contains the tasks. It also contains steps needed to be performed by the MapReduce to get the output as translated by the query
- Optimizer
  - It performs various transformations on the execution plan to provide optimized DAG
- Executor
  - Once compilation and optimization complete, the executor executes the tasks. Executor takes care of pipelining the tasks.

Hive: Components

- Hcatalog
  - Table and storage management layer that reads data from the Hive meta store to facilitate seamless integration between Hive and query processing frameworks such as Apache Pig, and MapReduce.
- WebHCat
  - A RESTful API for HCatalog to access and reuse Hive metadata.
  - Custom applications or third-party integrations can use WebHCat

Hive: Partitioning Vs Bucketing

- When partitioned for every unique value in the partitioned column a folder will be created in the underlying storage system.

- There could be multiple files inside that folder. So, when filtering data based on that column only that folder needs to be read.

- When there is high cardinality in the column, instead of making partitions for each of the values we specify the number of buckets we require
  - Then buckets will be created based on a hash function

Hive: Partitioning Vs Bucketing

- For every bucket HDFS will create a file and each file would then be divided into 128MB blocks by default.

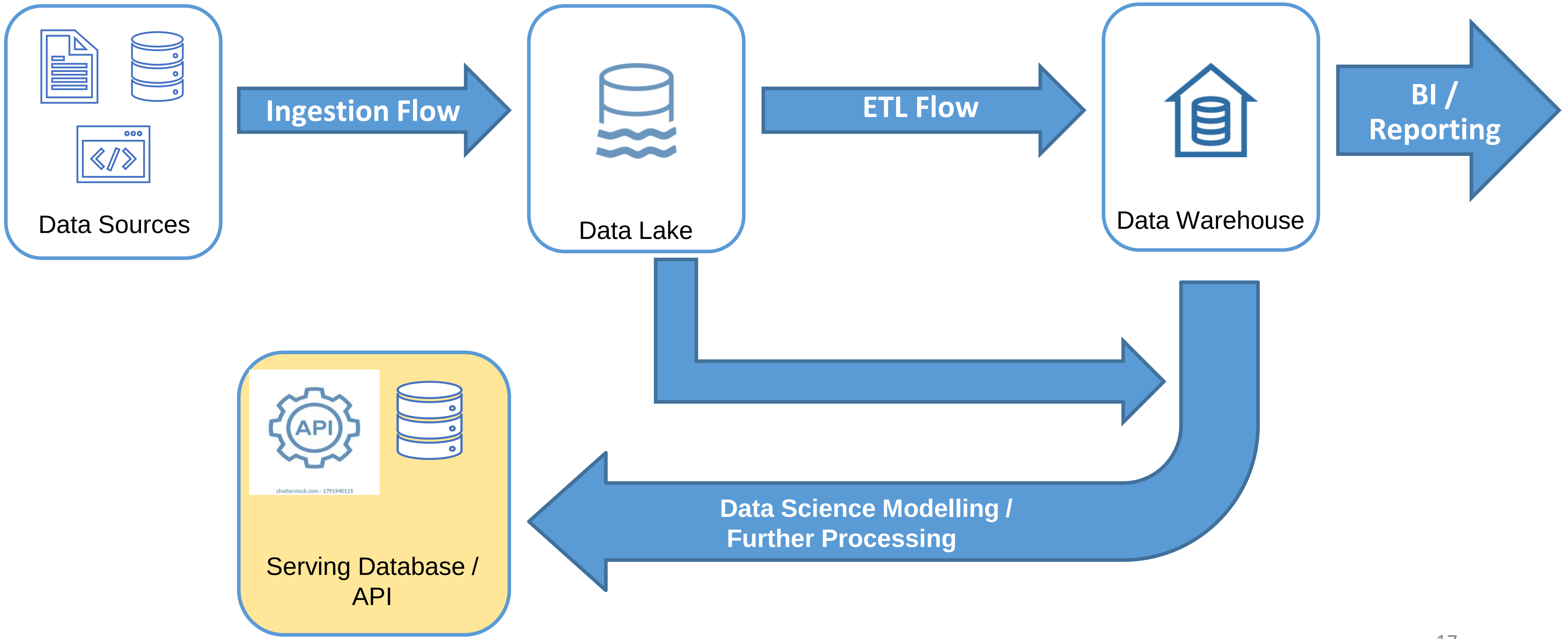
- Deciding no. of buckets
  - Bucket size should be large enough such that
    - It avoids small file problem
  - Bucket size should be small enough such that
    - one bucket should be able to easily fit in memory on the executor
  - It is best to make the bucket size approximately close to or bigger than the block size (128mb).

Open-Source Implementation – Other Features

- Security
  - Apache Ranger – Authorization, Audit and Data encryption
  - Apache Knox – Authentication
- Support for BI/ Data Cubes / Dimensional Modelling
  - Apache Druid – Data Cubes
  - Apache Zeppelin, Apache Spark – Data Science Applications
- Monitoring and Management
  - Apache Ambari
- Replication and Disaster Recovery
  - Apache Falcon

Staging / Serving DBs

Data Pipeline: Common Usage

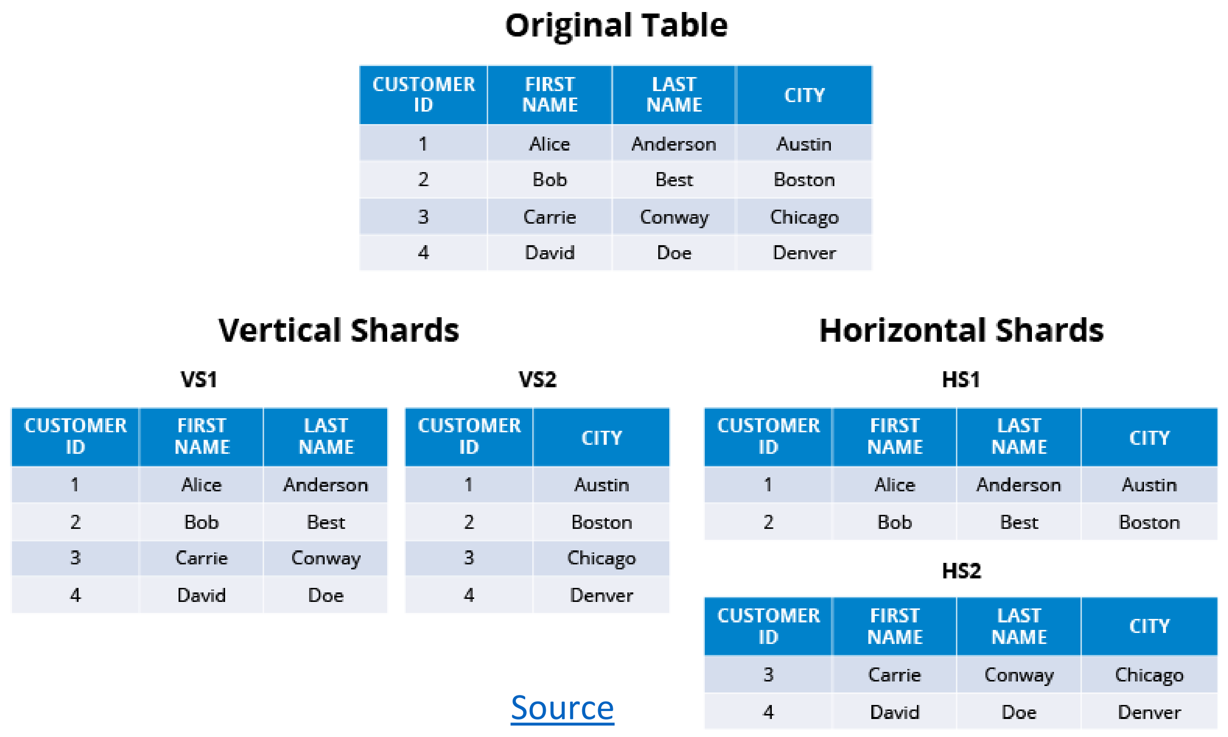


Serving / Staging DBs: Why not RDBMS

- Need to store different data types – Structured, Semi/Un Structured
- Cost of running complex queries with normalized data
- Analytical workloads normally does not require ACID properties
- Difficult to scale horizontally

Database Sharding

- Sharding is the practice of optimizing database management systems by separating the rows or columns of a larger database table into multiple smaller tables.



Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical Shards

VS1

CUSTOMER ID	FIRST NAME	LAST NAME
1	Alice	Anderson
2	Bob	Best
3	Carrie	Conway
4	David	Doe

VS2

CUSTOMER ID	CITY
1	Austin
2	Boston
3	Chicago
4	Denver

Horizontal Shards

HS1

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston

HS2

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Source

Database Sharding

- Different shards could be stored in multiple machine → supporting horizontal scaling.
- Avoid single point of failure
- When multiple DBs used, routing the requests to the DB having required shard could be handled either at application level or database level.

NOSQL Databases

- If Sharding doesn’t work next option is to use NOSQL databases.

RDBMS	NOSQL Databases
Schema on Write - strict	Schema on Read - relaxed
Data is normalized to avoid duplicates. But this makes queries requiring multiple joins,	Data is stored to support faster queries
Storage model is always tables with fixed rows and columns	Different storage models such as documents, key-value pairs, graphs etc.
Need to be scaled vertically (changing configurations of the server)	Can be scaled horizontally, by adding more commodity servers
Maintain ACID Properties	Maintain BASE Properties according to CAP Theorem
Stable, dependable and high vendor support	Mostly open-source projects.

BASE Properties

- **B**asically **A**vailable - Guarantees the availability of the data
- **S**oft state – Data will vary over a period
- **E**ventually consistent – may not be consistent when writing, but eventually consistent when reading
- BASE properties are much looser than ACID guarantees

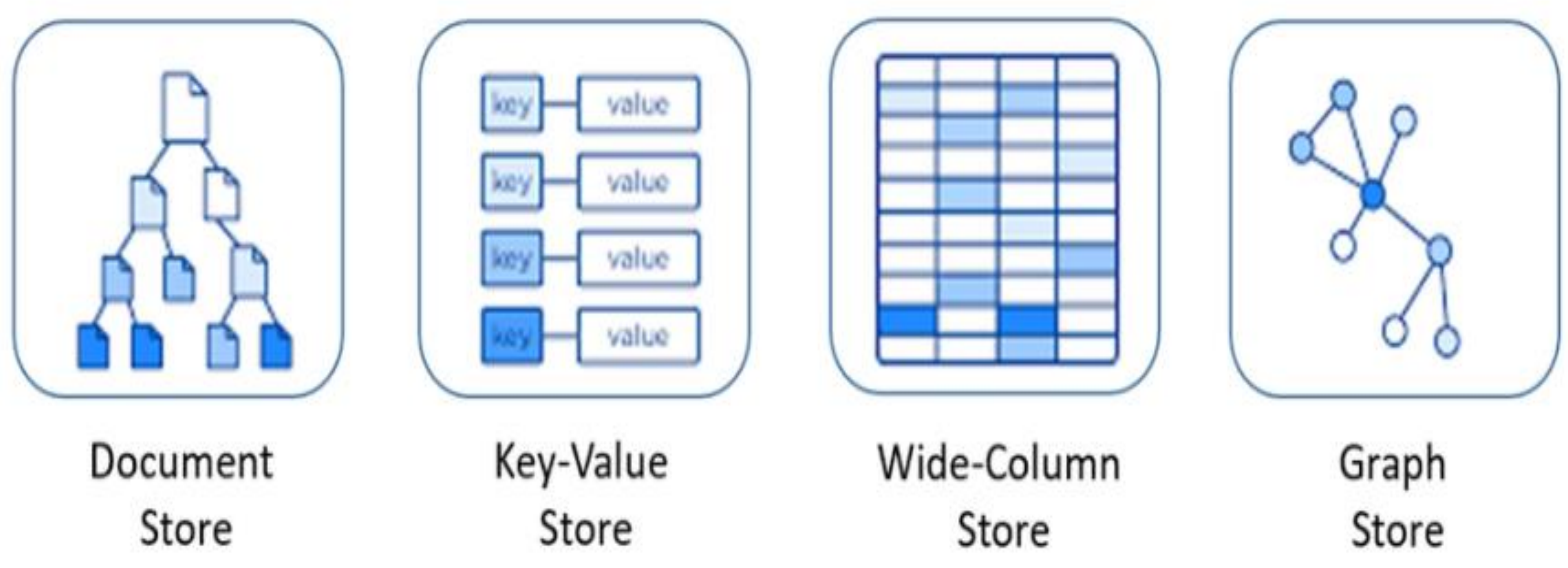
CAP Theorem

- States that any distributed data store can only provide two of the following three guarantees
  - **Consistency** - Every node in the cluster responds with the most recent data, even if the system must block the request until all replicas update.
  - **Availability** - Every node returns an immediate response, even if that response isn't the most recent data.
  - **Partition tolerance** - The system continues to operate despite a data node failure

- No distributed system is 100% free from network failures, thus network partition generally needs to be tolerated. So, it's a trade off between consistency and availability.

- In the case of a network failure, needs to decide whether to:
  - Cancel the operation and thus decrease the availability but ensure consistency or to
  - Proceed with the operation and thus provide availability but risk inconsistency.

NOSQL Database Types



Database Type	Document Store	Key-Value Store	Wide-Column Store	Graph Store
Examples:	MongoDB, CouchDB	Riak, Redis	Cassandra, HBase	Neo4J and HyperGraphDB

Image Source

Document Stores

- Store data in JSON, XML or BSON formats
- Contain any number of fields of any data type (number, string, binary, document)
- Support nested documents
- Fields can be indexed for fast query results.
- Can be converted to an object at the application layer
- Use Cases: Content Management, E-Commerce Search

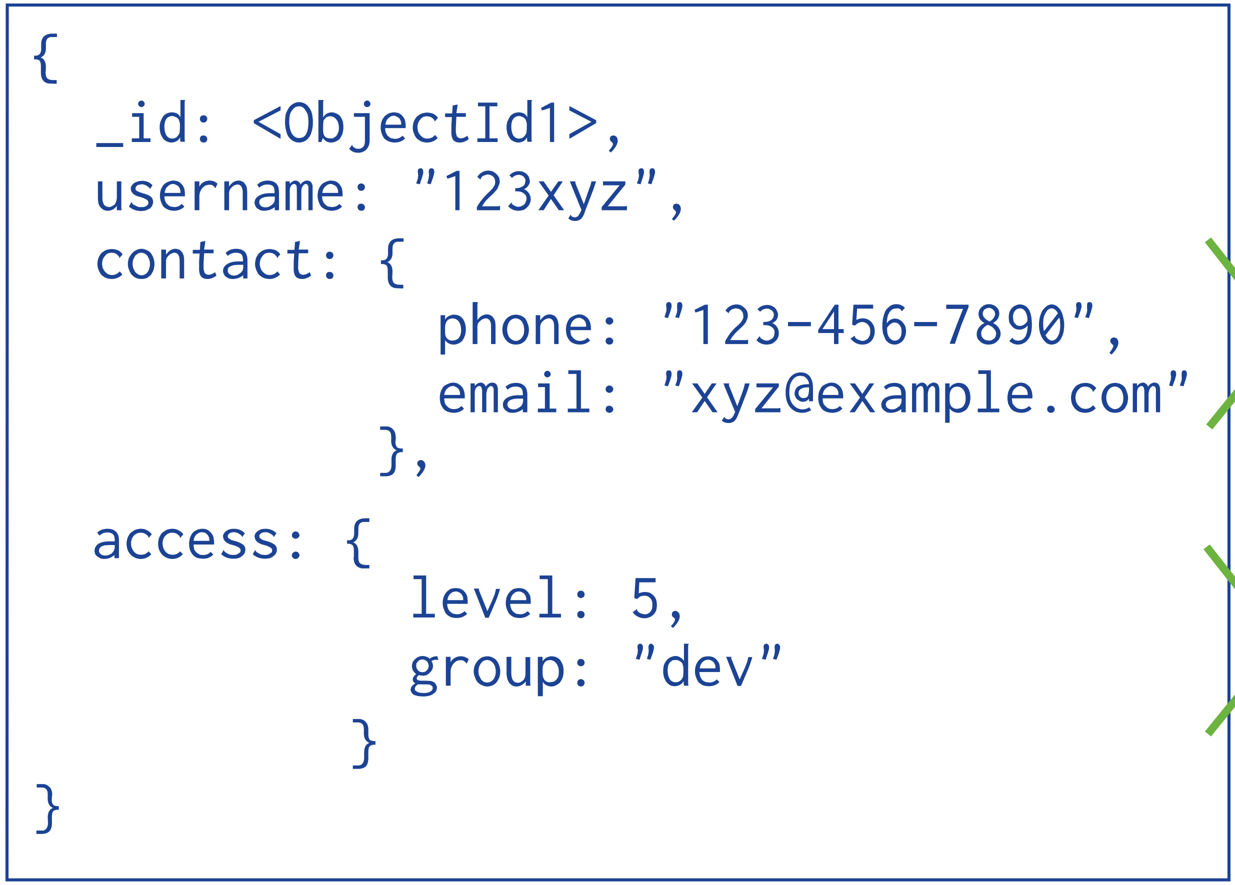
Document Stores: MongoDB

- Open-source, document-oriented, cross-platform database system written using C++

- Compared with RDBMS - Terminology

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)

Document Stores: MongoDB



```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
              phone: "123-456-7890",
              email: "xyz@example.com"
            },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

Embedded sub-document

Image Source

Column Oriented Storage

- Row-oriented storage is very inefficient if not all columns are needed but a lot of rows need to be read

- Column-oriented databases primarily work on columns
  - Values of a single column are stored contiguously
  - Almost all column stores perform compression further reducing the storage footprint of each column
  - All data within each column datafile have the same type, making it ideal for compression

Column Oriented Storage - Disadvantages

- Reading all columns of a row is an expensive operation in a column store
  - so full row tuple construction is avoided or delayed as much as possible internally

- Updating/deleting or inserting rows may also be very expensive and may cost much more time than in a row store

Column Oriented Storage - HBase

- An open-source implementation of Google’s Bigtable paper on top of HDFS.
- In the HBase data model columns (referred as column family qualifiers) are grouped into ***column families***
- Column families must be defined up front during table creation, however columns do not need to be defined at schema time but can be conjured on the fly while the table is up and running
- Column families are stored together on disk

Column Oriented Storage - HBase

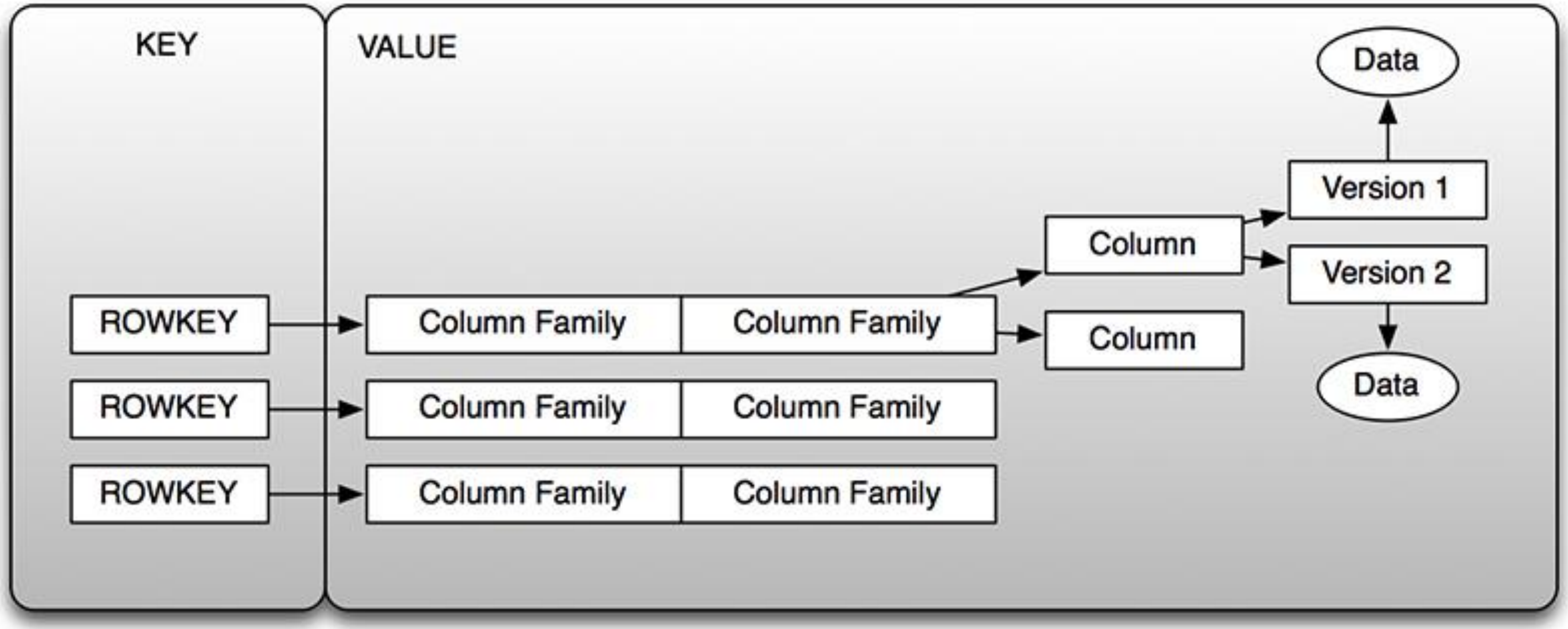
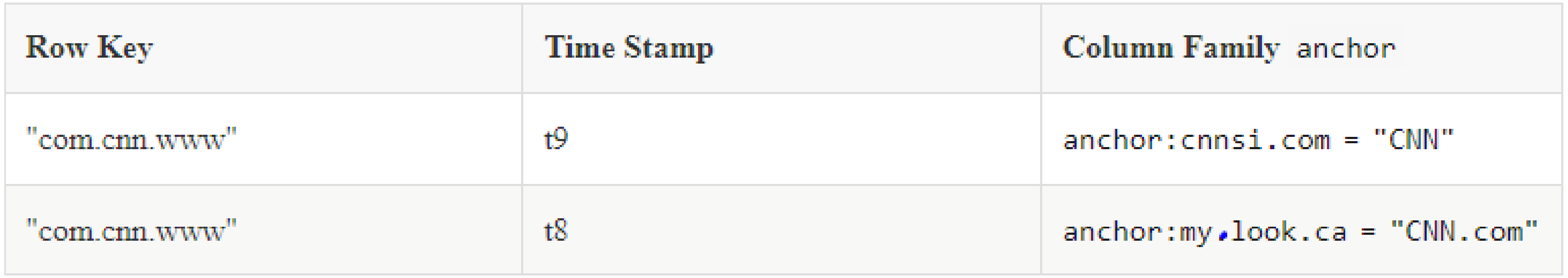
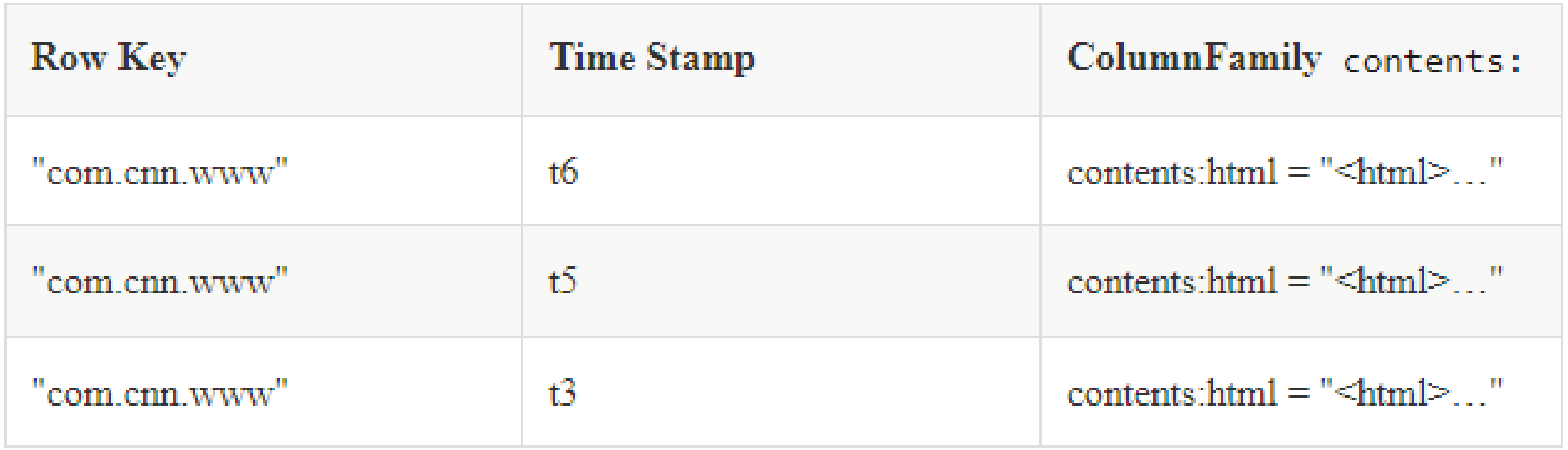


Image Source

Column Oriented Storage - HBase



Row Key	Time Stamp	Column Family `anchor`
"com.cnn.www"	t9	`anchor:cnnsi.com = "CNN"`
"com.cnn.www"	t8	`anchor:my.look.ca = "CNN.com"`



Row Key	Time Stamp	ColumnFamily `contents:`
"com.cnn.www"	t6	contents:html = "<html>…"
"com.cnn.www"	t5	contents:html = "<html>…"
"com.cnn.www"	t3	contents:html = "<html>…"

Image Source

Key-Value Stores

- The data is fetched by a unique key or several unique keys to retrieve the associated value with each key.
- Both keys and values can be anything, ranging from simple objects to complex compound objects.
- Highly partitionable and allow horizontal scaling at scales
- MongoDB and HBase can also be considered as Key-Value stores
- Database type which is optimized for reading and writing that data.
    - Use Case: E-Commerce for session storing, shopping cart and/or recommendations

Graph Databases

- A graph database stores nodes and relationships instead of tables, or documents.
- Useful in modelling connections between entities without having to deal with expensive JOIN operations or cross-lookups.
- Helps to:
  - Navigate down the hierarchies
  - Find hidden relationships
- Use Cases: Social network analysis

Data Storage Considerations

- Data volume
- Structure and format
- Duration and retention
- Query frequency and volume
- Other users and uses of the data
- Governance constraints
- Privacy and security
- Fault tolerance and disaster recovery

READING

- Thusoo, Ashish, et al. "Hive: a warehousing solution over a map-reduce framework." Proceedings of the VLDB Endowment 2.2 (2009): 1626-1629.
- Thusoo, Ashish, et al. "Hive-a petabyte scale data warehouse using hadoop." 2010 IEEE 26th international conference on data engineering (ICDE 2010). IEEE, 2010.
- Leavitt, Neal. "Will NoSQL databases live up to their promise?." Computer 43.2 (2010): 12-14.