A"

Aalto University
School of Science

Big Data Storage and Database Services – common systems & integration problems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Common data models and data storage/database systems

Common data models

- **File**

- **Relational data model**

- **Key-Value data model**

- **Document-oriented model**

- **Column family model**

- **Graph model**

Aalto University
School of Science
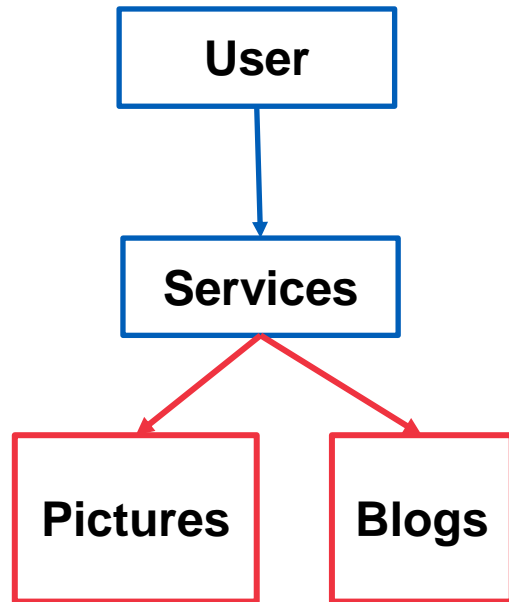
Blob data

Big files:

- Pictures, documents, big log files, images, video, backup data

Storage

- File systems or blob storage

Implementations

- File systems: NFS, GPFS, Lustre (http://lustre.org/)

- Storage: Amazon S3, Azure Blob storage, OpenStack Swift, Cepth, Google Storage

- Simple API for direct access

Aalto University
School of Science

Example - Amazon S3
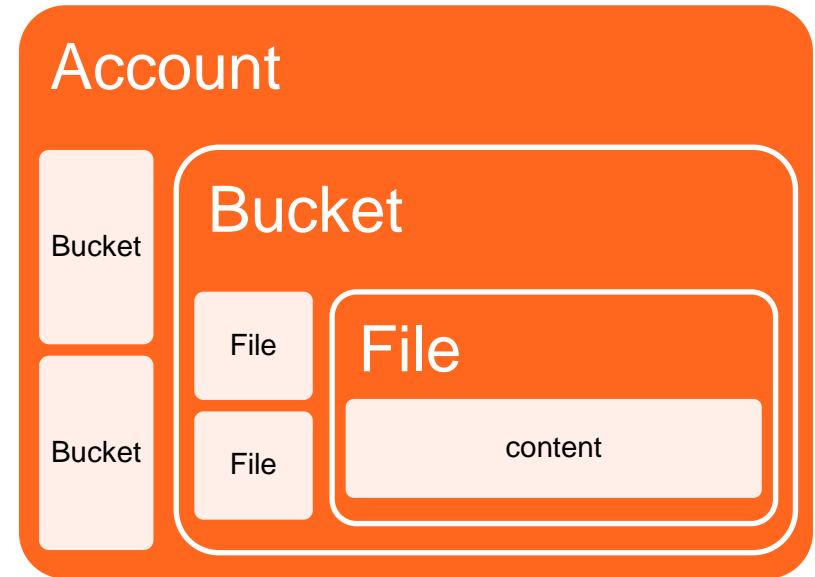
Store blob files and their metadata

- Max 5TB per file
- A File is identified by a key

Structure

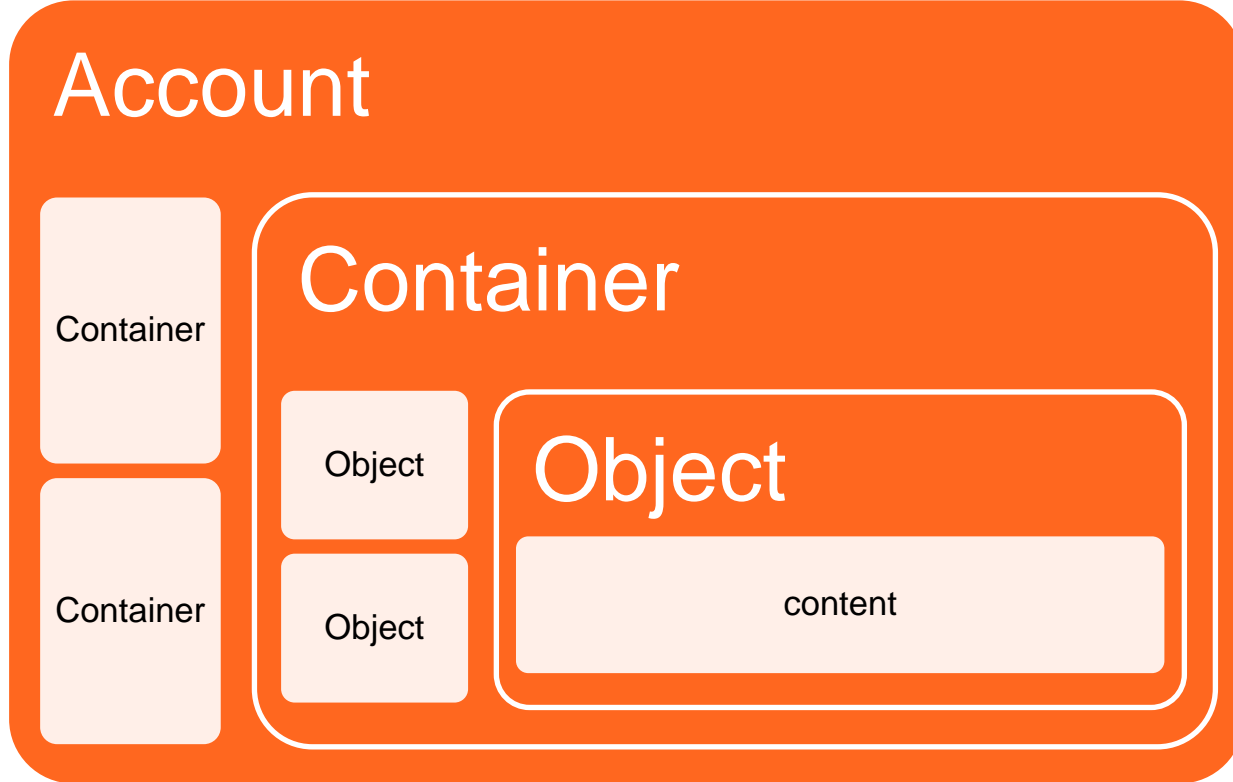- File = Object
- Object: name and metadata
- Objects are organized into Buckets

Simple APIs

- REST

Account

Bucket

Bucket

Bucket

File

File

File

content

Aalto University
School of Science

OpenStack Swift

http://docs.openstack.org/developer/swift/

**Aalto University
School of Science**

Relational Model

- **Well-known, long history**
- **Tables with rows and columns**
  - Strict schema requirements
- **Powerful querying & strong consistency support**
  - E.g.: Oracle Database, MySQL Server, PostgreSQL, MariaDB

Example: Alarm in BigQuery

Relational Databases for big data scenarios

- **Relational database at very large-scale**
  - Amazon Aurora, Microsoft Azure SQL Data Warehouse

- **We said ACID is hard with big data**
  - relational big database must address replication, distribution, and scalability issues

- **Examples of Amazon Aurora (reading list)**
  - based on MySQL/InnoDB but change the architecture, separate storage from engine, support cloud scale and replication, etc.

Key-Value Model

- **Tuple = (key, value)**
  - Values can be base on different structures
- **Scalable and performance**
- **Primary use case: caching (pages, sessions, frequently access data, distributed lock)**
  - Simple, very efficient but limited querying capabilities
- **Implementation:**
  - Memcached, Riak, Redis

Example: Redis

- **http://redis.io/**

- **In-memory cache service**
  - *Store (key,value) tuples in memory but persistent back to database*
- Simple APIs
  - Well support with many programming languages
  - Widely used in big data ecosystems
- **Learning**
  - *https://redislabs.com/ provides a free account*

Example: Redis

http://redis.io/topics/benchmarks

Aalto University
School of Science

Document-oriented model – simple analogy
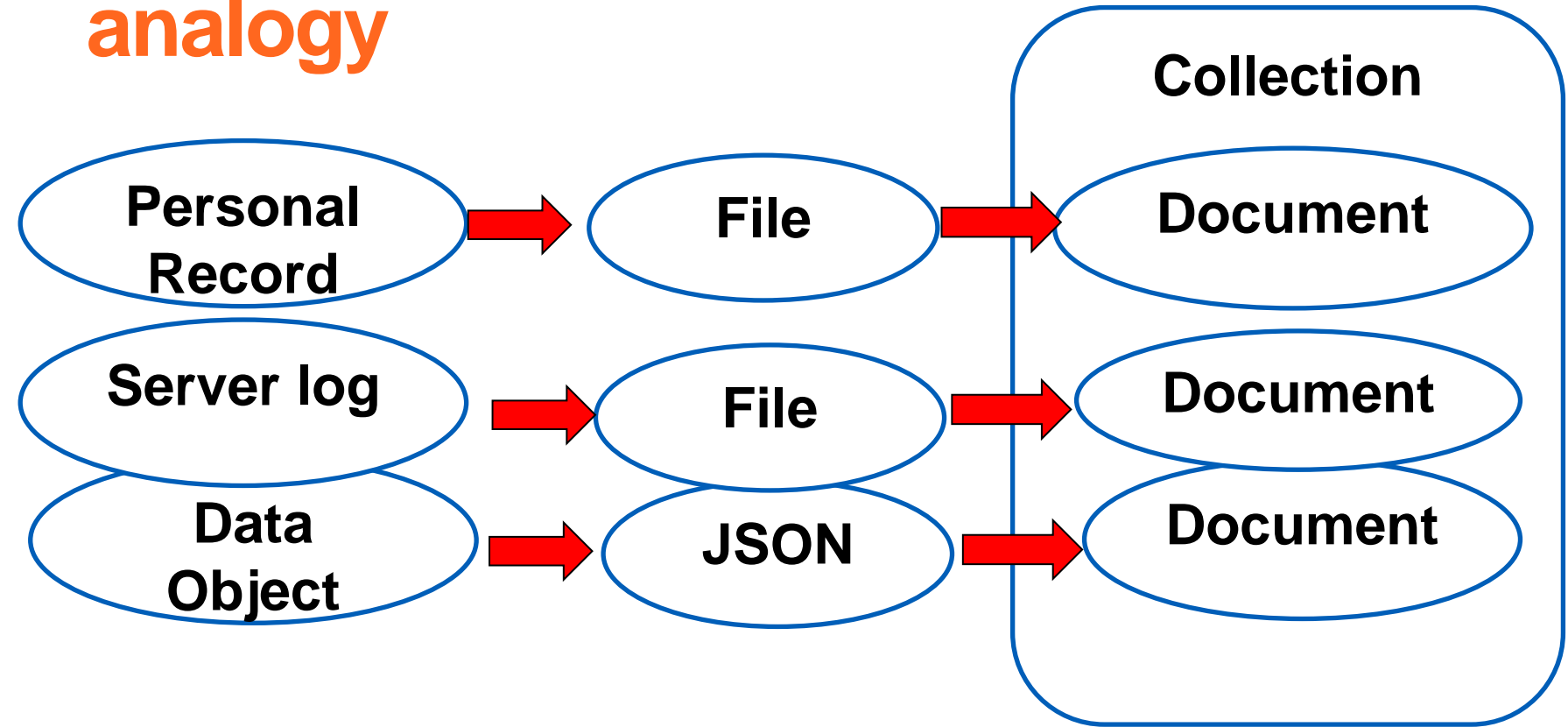
Document-oriented Model

- **Documents**
  - flexible schema (schemaless) with flexible content
  - data fields can be complex for sub documents
  - use collections, each collection is a set of documents
- **Primary use cases**
  - large amounts of semi-structured data
  - collection of data with different structures

Examples: MongoDB.Atlas

https://www.mongodb.com/cloud/atlas

Graph-oriented model

- **Data is represented as a graph**
    - nodes or vertices represent objects
    - an edge describes a relationship between nodes
    - properties associated with nodes and edge provide other information
- **Use cases**
    - when searching data is mainly based on relations (social networks, asset relationship, knowledge graph)

Working with graph databases

- **Graph databases**
  - Auze CosmosDB, ArgangoDB, Titan, Grakn.AI, Neo4J, OrientDB
- **Query languages:**
  - Gremlin, SPARQL, Cypher
- **Graph computing frameworks (analysis)**
  - Apache TinkerPop, Apache Spark GraphX

Example

https://grakn.ai/

Column-family data model

Motivation: scalable, distributed storage for multi-dimensional sparse sorted map data
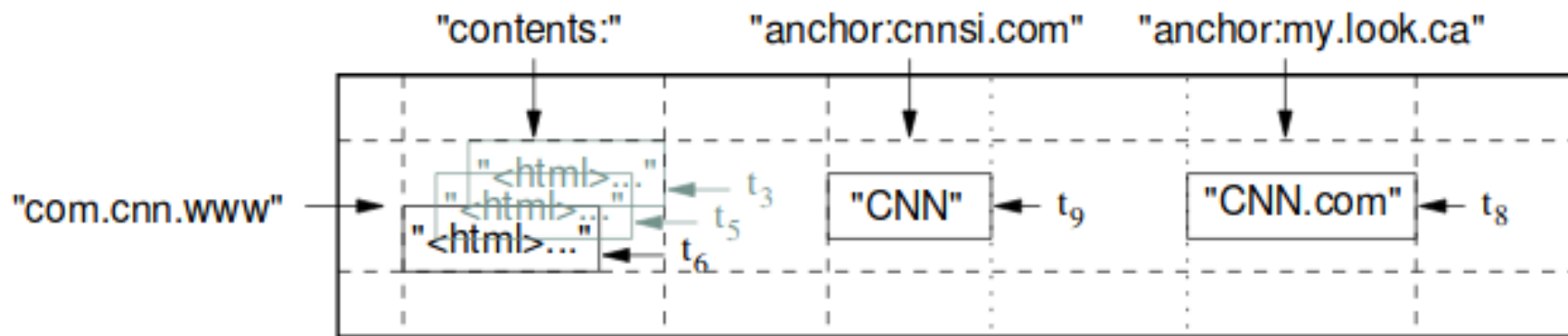
Figure source: Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. 2006. Bigtable: a distributed storage system for structured data. In Proceedings of the 7th symposium on Operating systems design and implementation (OSDI '06). USENIX Association, Berkeley, CA, USA, 205-218.

Column-family data model

- **Data Model**
  - Table consists of rows
  - Row consists of a key and one or more columns
  - Columns (column name, value, timestamp)
  - Columns are grouped into column families
  - Column families can be grouped into a row

Example of a data model in Cassandra

Column (key,value, timestamp)

Column family

Row key/Partition key

1	species	country	Duration_seconds	English_cname					
	Aberti	US	3	…					

Row

2	species	country	Duration_seconds	English_cname	name	url	latitutde	longitude	Text

Row
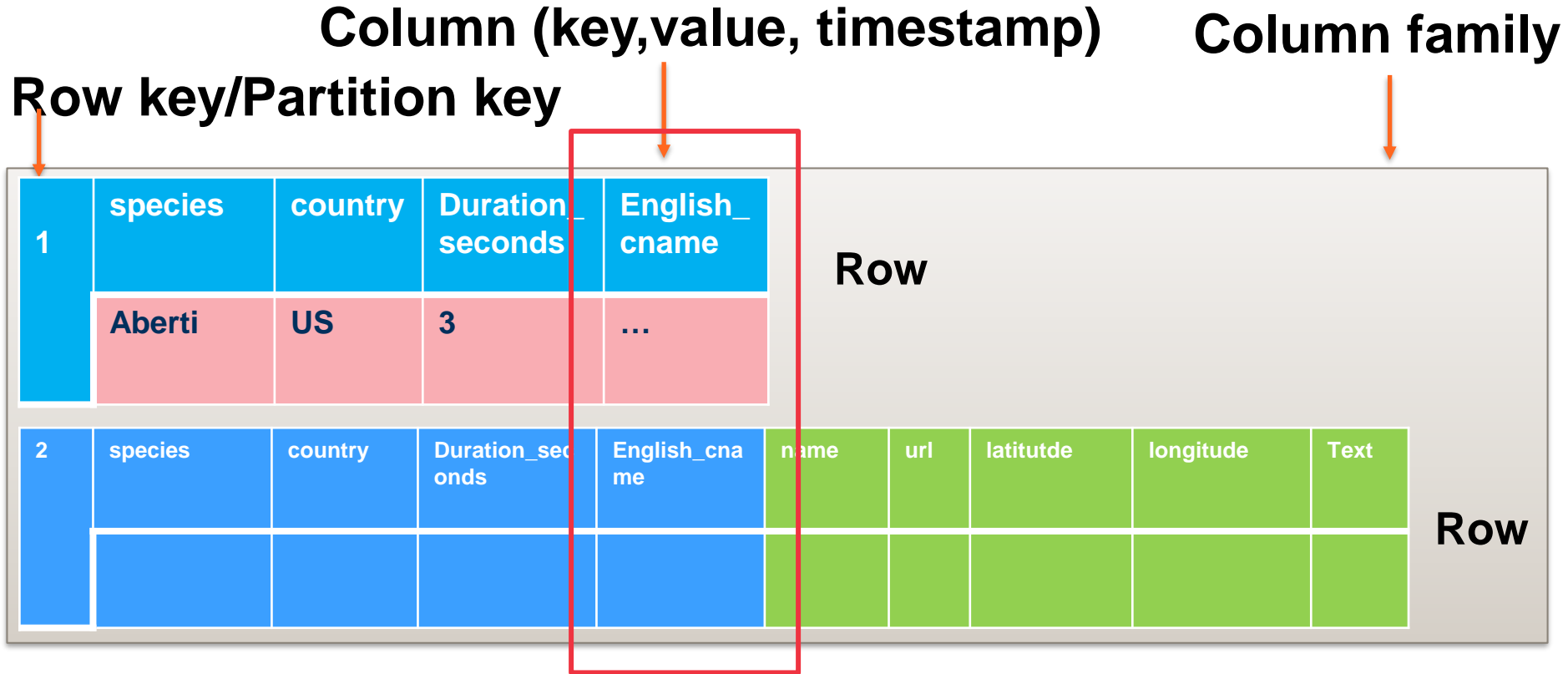
Examples

Examples of rows

Column (name, value, timestamp)

```
cassandra@cqlsh> select * from tutorial12345.bird2;

@ Row 1
------------------+------------------------------
 species          | melanura
 country          | Mexico
 duration_seconds | 29
 english_cname    | Black-tailed Gnatcatcher
 file_id          | 71907
 latitude         | 32.156
 longitude        | -115.793

@ Row 2
------------------+------------------------------
 species          | melanura
 country          | United States
 duration_seconds | 29
 english_cname    | Black-tailed Gnatcatcher
 file_id          | 358907
 latitude         | 33.7329
 longitude        | -115.8023
```

```
 english_cname            | writetime(english_cname)
--------------------------+--------------------------
 Black-tailed Gnatcatcher |         1569966171073228

(1 rows)
```

Time Series Database

- **So many types of data in big data are time series**
  - IoT measurements, session data, log, etc.
- **Of course you can also use other databases**
  - e.g., Cassandra, ElasticSearch, BigTable
- **Time Series Databases specially designed for time series data**
  - *examples: Riak TS (Time Series), InfluxDB, Apache Druid*

Example: InfluxDB

- **https://www.influxdata.com/**

- **High-level query, SQL-alike Language**

- **Retention policy for data storage, sharding and replication**

```
                    ┌──────────────┐
                    │   Database   │
                    └──────────────┘
                   ↙              ↘
    ┌──────────────────┐      ┌──────────┐
    │ Retention Policy │      │  Point   │
    └──────────────────┘      └──────────┘
                        ↙    ↓   ↓    ↘
        ┌─────────────┬──────┬────────┬───────────┐
        │ measurement │ tags │ fields │ timestamp │
        └─────────────┴──────┴────────┴───────────┘
```

Aalto University
School of Science
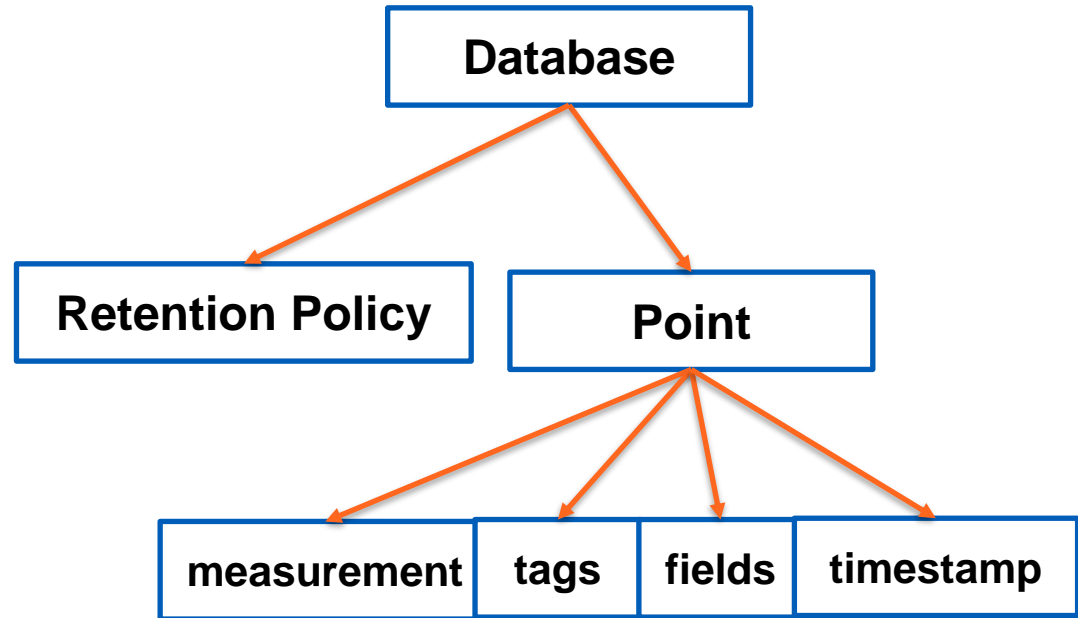
An example of InfluxDB

```
> show measurements
name: measurements
name
----
stationalarm
stationaparameter
> select * from stationalarm;
name: stationalarm
time            alarm_id  datapoint_id   station_id   value   valueThreshold
----            --------  -----------    ----------   -----   --------------
1487444343000000  308       121          1161115016   240     240
```

**Aalto University
School of Science**

In-memory databases

- **Databases use machine memory for storage**
  - Persist data on disks
  - Require very powerful machines
- **In principle it is not just about data models but also data management, data processing, software and hardware optimization, e.g.,**
  - SAP HANA, VoltDB: in memory relational databases
- **Why are in-memory databases important?**

Aalto University
School of Science

Interfaces between a data storage/databases system and its external analysis systems
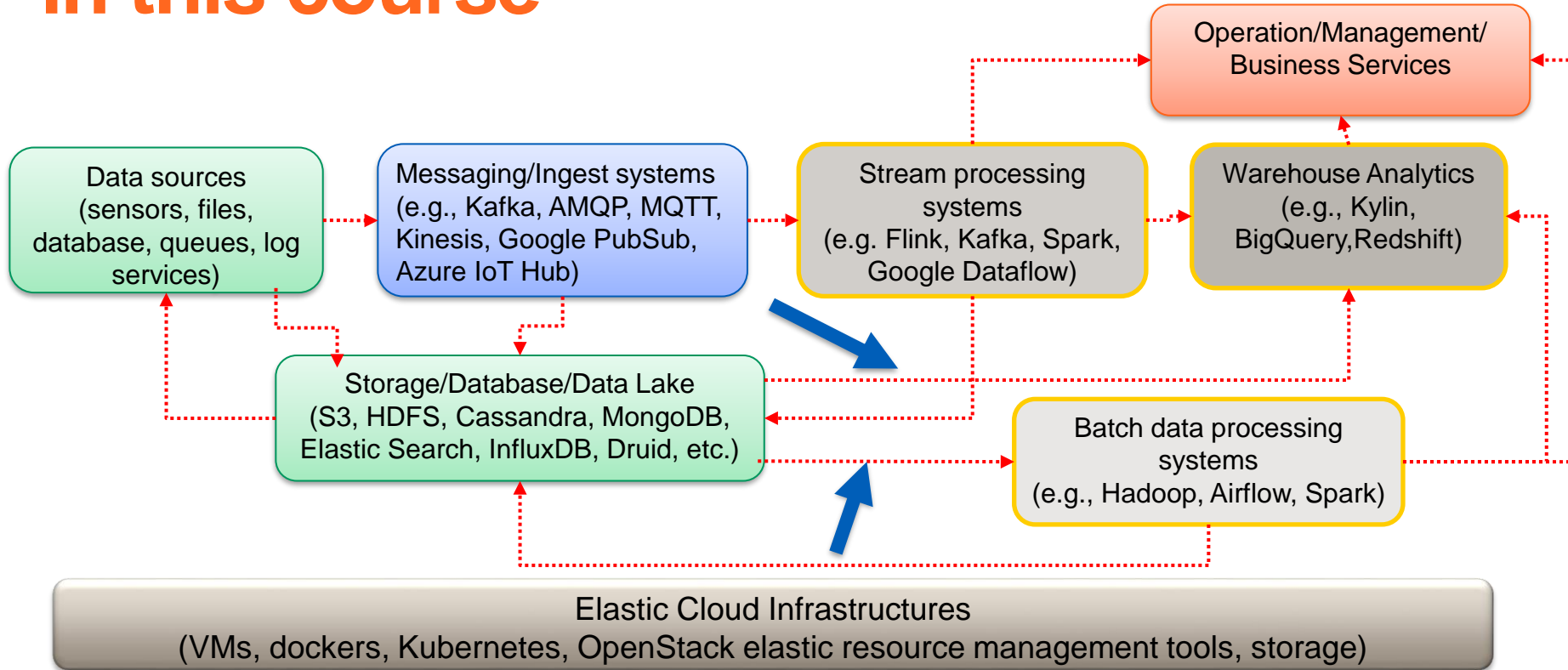
In many cases: the data in data storage/database service must be made available for large-scale analysis:

large-scale analytics and data are managed by different systems

an important consideration in big data platforms design!

Big data at large-scale: the big picture in this course

Data sources (sensors, files, database, queues, log services)

Messaging/Ingest systems (e.g., Kafka, AMQP, MQTT, Kinesis, Google PubSub, Azure IoT Hub)

Stream processing systems (e.g. Flink, Kafka, Spark, Google Dataflow)

Warehouse Analytics (e.g., Kylin, BigQuery,Redshift)

Operation/Management/ Business Services

Storage/Database/Data Lake (S3, HDFS, Cassandra, MongoDB, Elastic Search, InfluxDB, Druid, etc.)

Batch data processing systems (e.g., Hadoop, Airflow, Spark)

Elastic Cloud Infrastructures (VMs, dockers, Kubernetes, OpenStack elastic resource management tools, storage)
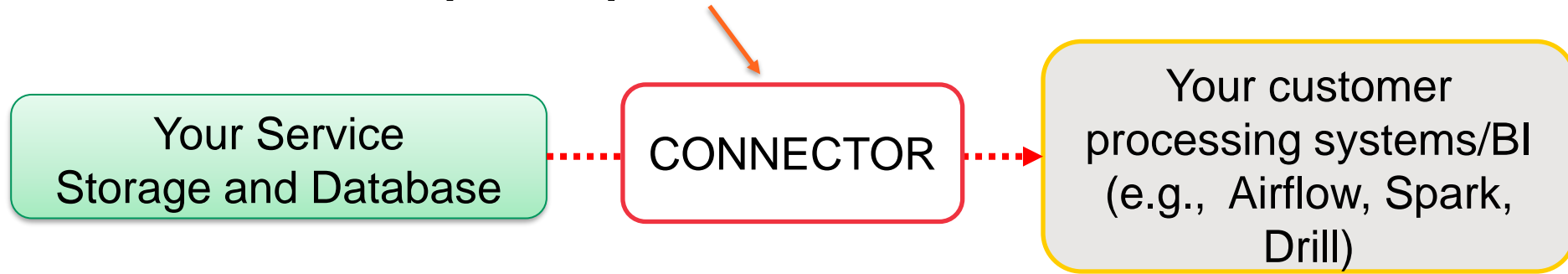
Making data available to the analytics

- **Data layer must map/provide data to processing layer**
  - maximize the analytics possibilities
- **Key issues**
  - avoid data movement as much as possible
  - avoid contention between the data management and the data analytics system
- **Techniques**
  - "mount", specific connectors/drivers, copy-process-remove activities

Mount/"Fuse"

- **Mapping a remote storage as a local file system**
  - Blobfuse (Microsoft Azure), gcsfuse (Google Storage)
  - the network performance is important

Connectors

ODBC or other specific protocol connectors

| Your Service Storage and Database | ···→ | CONNECTOR | ···→ | Your customer processing systems/BI (e.g., Airflow, Spark, Drill) |

Example

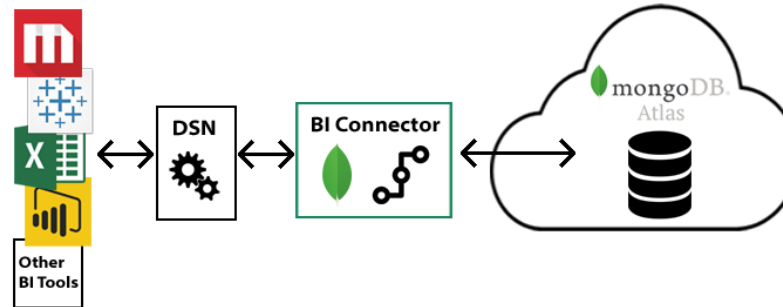**Figure source: https://docs.mongodb.com/bi-connector/master/**

Aalto University
School of Science

Analytics and Cloud Storage

- **Various connectors for making data in cloud storages available for analytics**

- **Apache Hadoop/Spark (data analysis) can work with Amazon S3, OpenStack Swift, Google Cloud Storage**

- **Examples:**
  - https://github.com/GoogleCloudDataproc/hadoop-connectors
  - https://spark.apache.org/docs/2.3.0/cloud-integration.html

"Copy and Process"

Client libraries are used to move data from storages and databases to processing places

Examples:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from cassandra.cluster import Cluster

cluster = Cluster(contact_points=hosts, port=9042,auth_provider=auth_provider)
session = cluster.connect("tutorial12345")
sql_query = "SELECT * FROM tutorial12345.bird1234;"
df = pd.DataFrame()
rows= session.execute(sql_query)
df = rows._current_rows
print(df)
```

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io