BIG DATA – DATA ANALYSIS

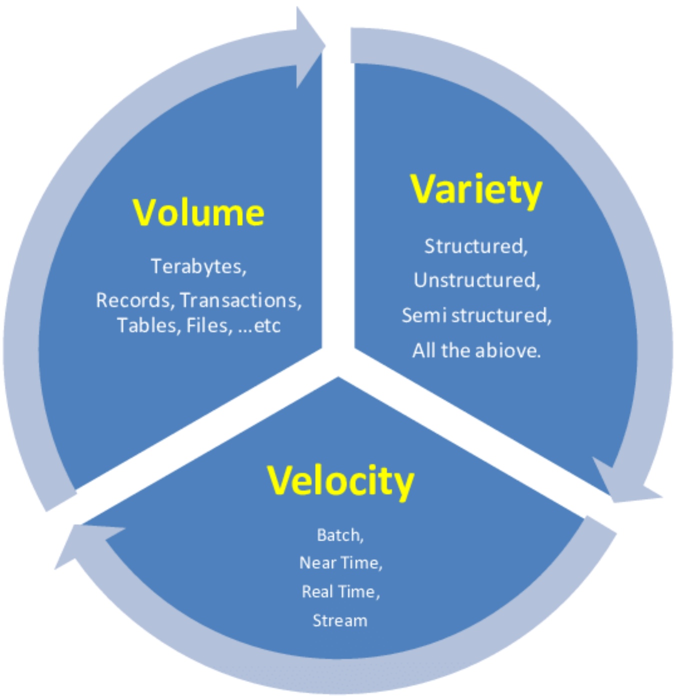
Lê Hồng Hải
UET-VNUH

Big Data Overview

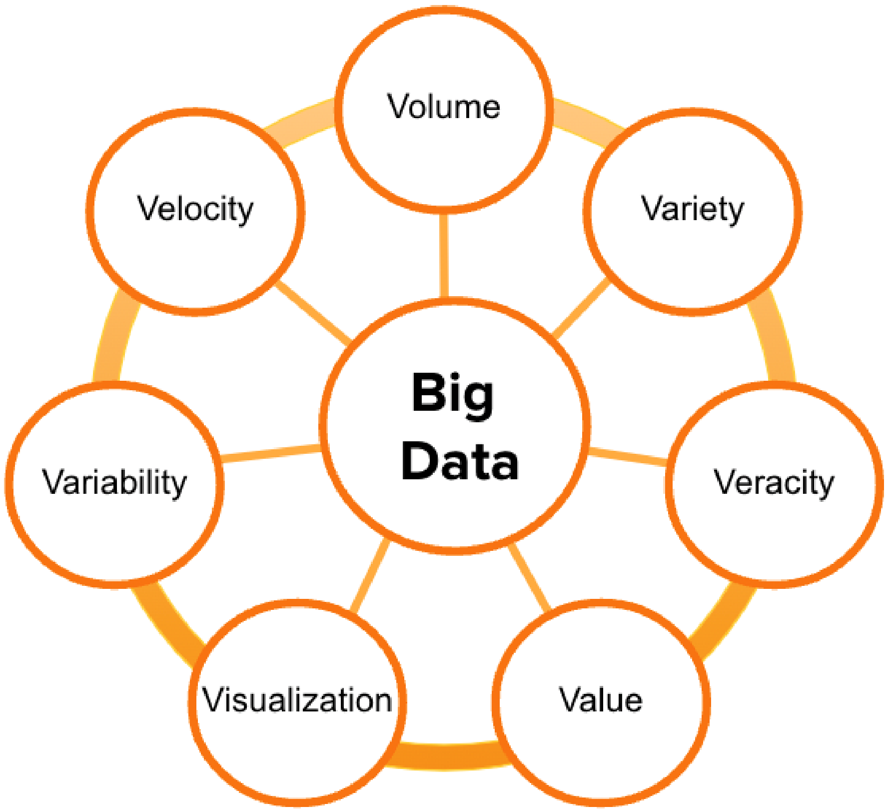
1. Introduction
2. Big Data storages
3. Big Data processing
4. Streaming

Big Data

- The definition of big data is data that contains greater **variety**, arriving in increasing **volumes** and with more **velocity**. This is also known as the 3 Vs



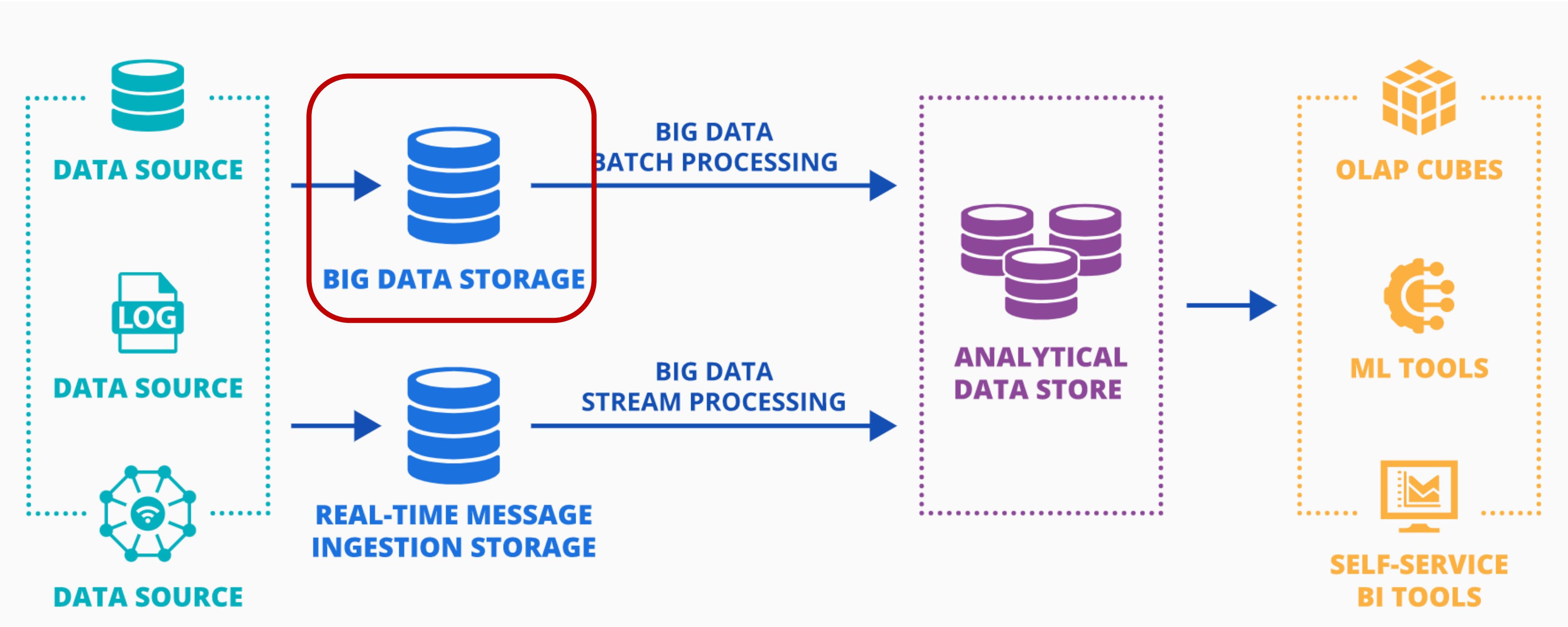
7 V'S OF BIG DATA



Big data architecture components

- **Data sources** – relational databases, files (e.g., web server log files) produced by applications, real-time data produced by IoT devices.
- **Big data storage** –storing high data volumes of different types before filtering, aggregating, and preparing data for analysis.
- **Real-time message ingestion store** – to capture and store real-time messages for stream processing.
- **Analytical data store** – relational databases for preparing and structuring big data for further analytical querying.
- **Big data analytics and reporting**, which may include OLAP cubes, ML tools, BI tools, etc. – to provide big data insights to end users.

Big data architecture



Big Data Storage

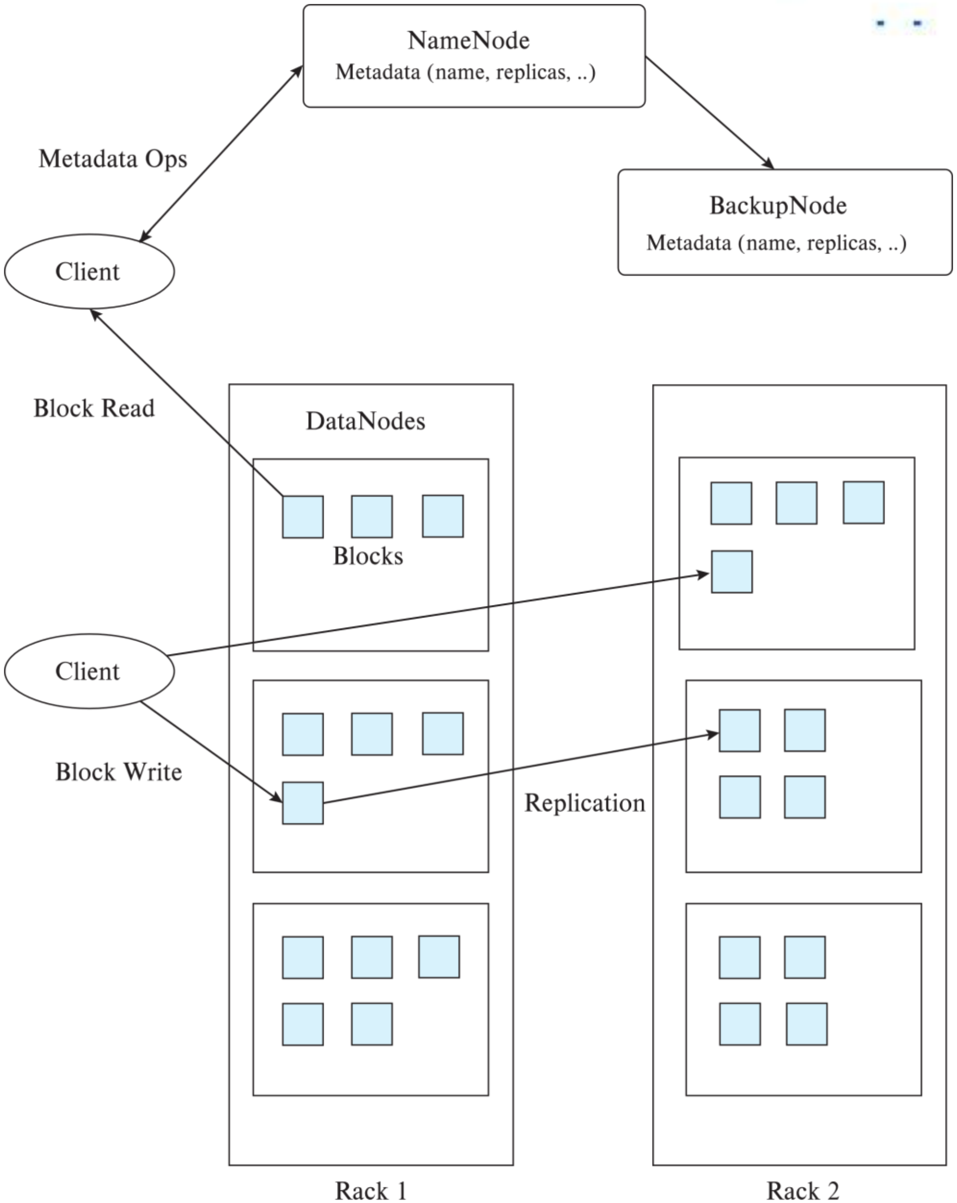
1. Distributed file systems
2. Sharding across multiple databases
3. Key-value storage systems
4. Parallel and distributed databases

Distributed File Systems

- A distributed file system stores data across a large collection of machines, but provides a single file-system view
- Provides redundant storage of massive amounts of data on cheap and unreliable computers
  - Google File System (GFS)
  - Hadoop File System (HDFS)

Hadoop File System Architecture

- Single Namespace for entire cluster
- Files are broken up into blocks
  - Typically 64 MB block size
  - Each block replicated on multiple DataNodes
- Client
  - Finds the location of blocks from NameNode
  - Accesses data directly from DataNode



Hadoop Distributed File System (HDFS)

- Data Coherency
  - Write-once-read-many access model
  - Client can only append to existing files
- Distributed file systems good for millions of large files

1. Distributed file systems
2. Sharding across multiple databases
3. Key-value storage systems
4. Parallel and distributed databases

Sharding

- **Sharding**: partition data across multiple databases
- Partitioning usually done on some ***partitioning attributes*** (also known as ***partitioning keys*** or ***shard keys*** e.g. user ID
  - E.g., records with key values from 1 to 100,000 on database 1,
    records with key values from 100,001 to 200,000 on database 2, etc

Key Value Storage Systems

- Key-value storage systems store large numbers (billions or even more) of small (KB-MB) sized records
- Records are **partitioned** across multiple machines and
- Queries are routed by the system to appropriate machine
- Records are also **replicated** across multiple machines, to ensure availability even if a machine fails
  - Key-value stores ensure that updates are applied to all replicas, to ensure that their values are **consistent**

Key Value Storage Systems

- Key-value stores may store
  - **uninterpreted bytes**, with an associated key
    - E.g., Amazon S3, Amazon Dynamo
  - **Wide-table** (can have arbitrarily many attribute names) with associated key
    - Google BigTable, Apache Cassandra, Apache Hbase, Amazon DynamoDB
  - JSON
    - MongoDB, CouchDB (document model)
- **Document stores** store semi-structured data, typically JSON
- Some key-value stores support multiple versions of data, with timestamps/version numbers

- An example of a JSON object is:

```
{
    "ID": "22222",
    "name": {
        "firstname: "Albert",
        "lastname: "Einstein"
    },
    "deptname": "Physics",
    "children": [
        { "firstname": "Hans", "lastname":
  "Einstein" },
        { "firstname": "Eduard", "lastname":
  "Einstein" }
    ]
}
```

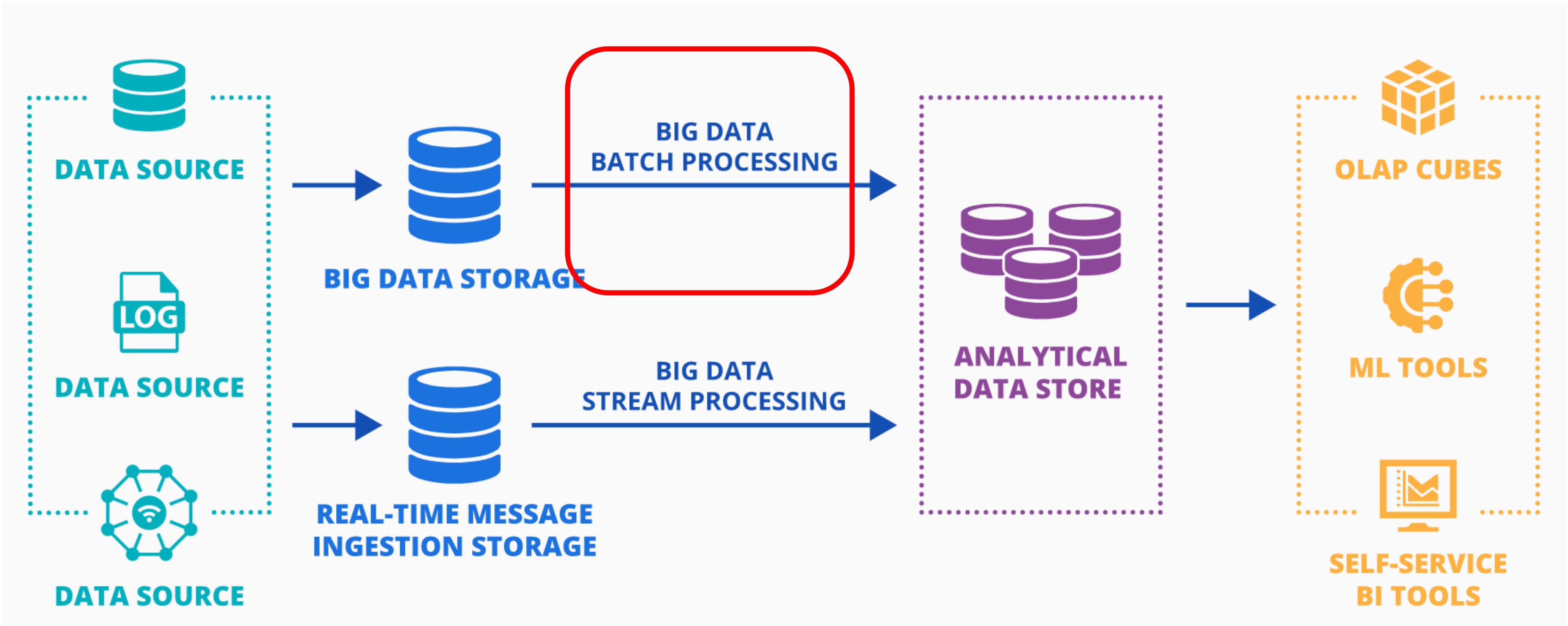
Key Value Storage Systems

- Key-value stores support
  - ***put***(key, value): used to store values with an associated key,
  - ***get***(key): which retrieves the stored value associated with the specified key
  - ***delete***(key) -- Remove the key and its associated value
- Some systems also support ***range queries*** on key values
- Document stores also support queries on non-key attributes
  - See book for MongoDB queries
  - Also called **NoSQL** systems

Replication and Consistency

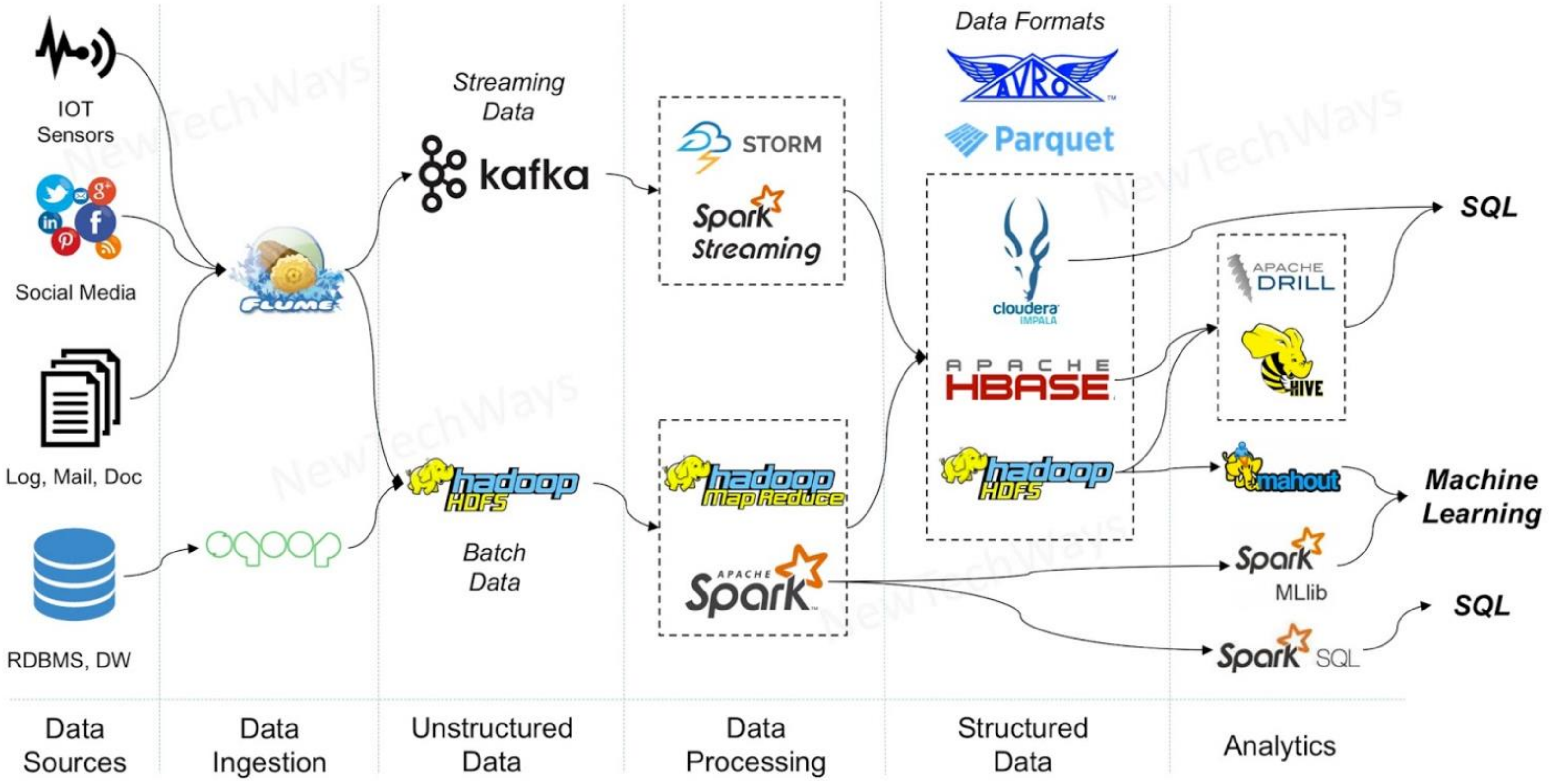
- **Availability** (system can run even if parts have failed) is essential for parallel/distributed databases
  - Via replication, so even if a node has failed, another copy is available
- **Consistency** is important for replicated data
  - All live replicas have same value, and each read sees latest version
- **Network partitions** (network can break into two or more parts, each with active systems that can't talk to other parts)
- In presence of partitions, cannot guarantee both availability and consistency
  - **Brewer's CAP "Theorem"**

Big data architecture



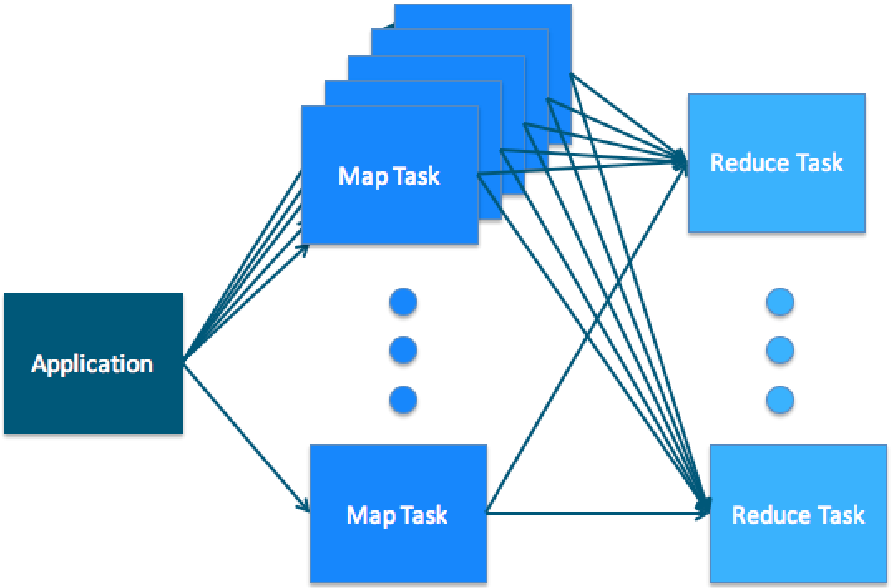
Big Data Processing

- Map-Reduce
- Spark
- Streaming

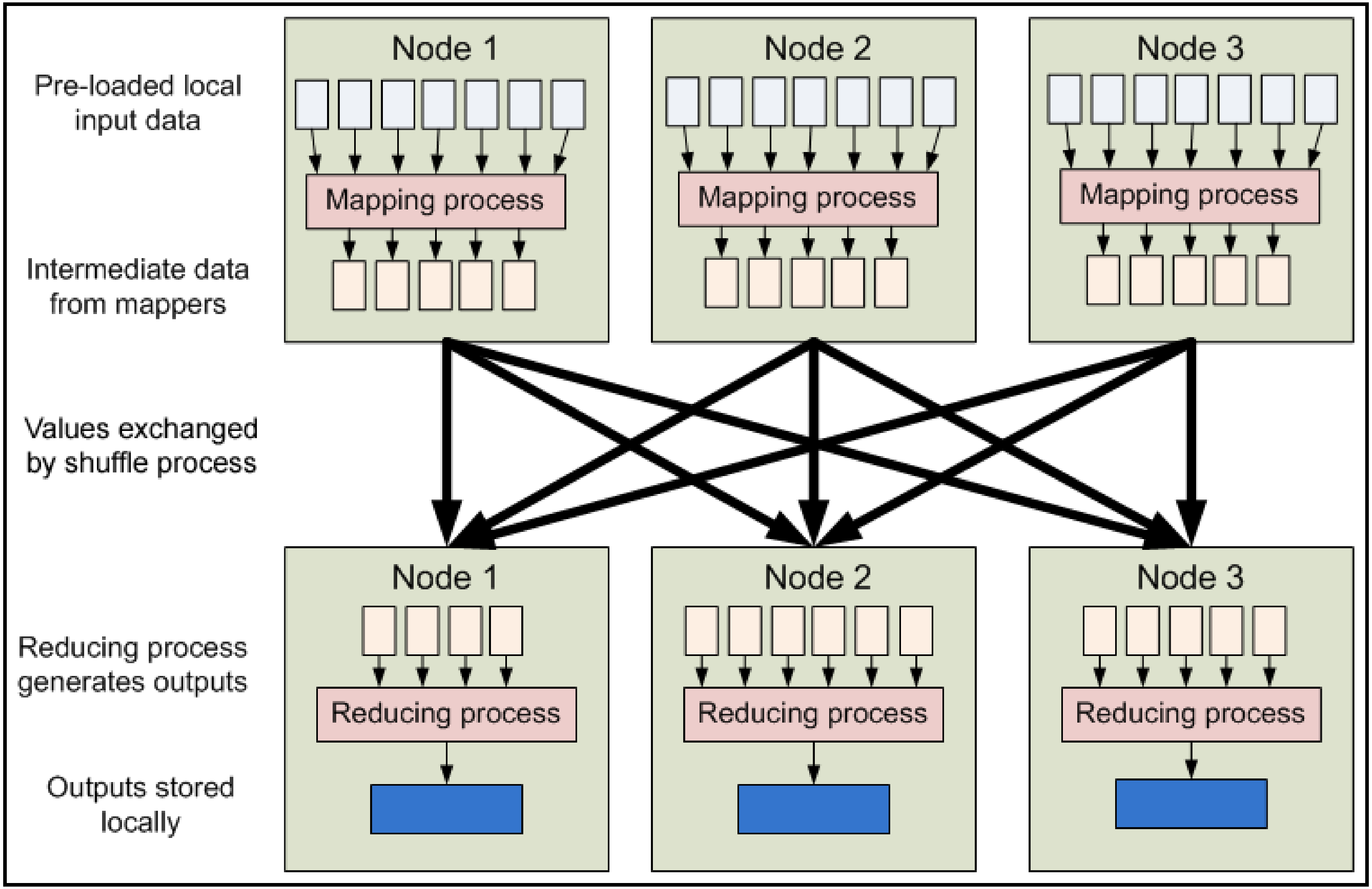


The MapReduce Paradigm

- Platform for reliable, scalable parallel computing
- Abstracts issues of distributed and parallel environment from programmer
  - Programmer provides core logic (via map() and reduce() functions)
  - System takes care of parallelization of computation, coordination, etc



MapReduce - Dataflow



The MapReduce Paradigm

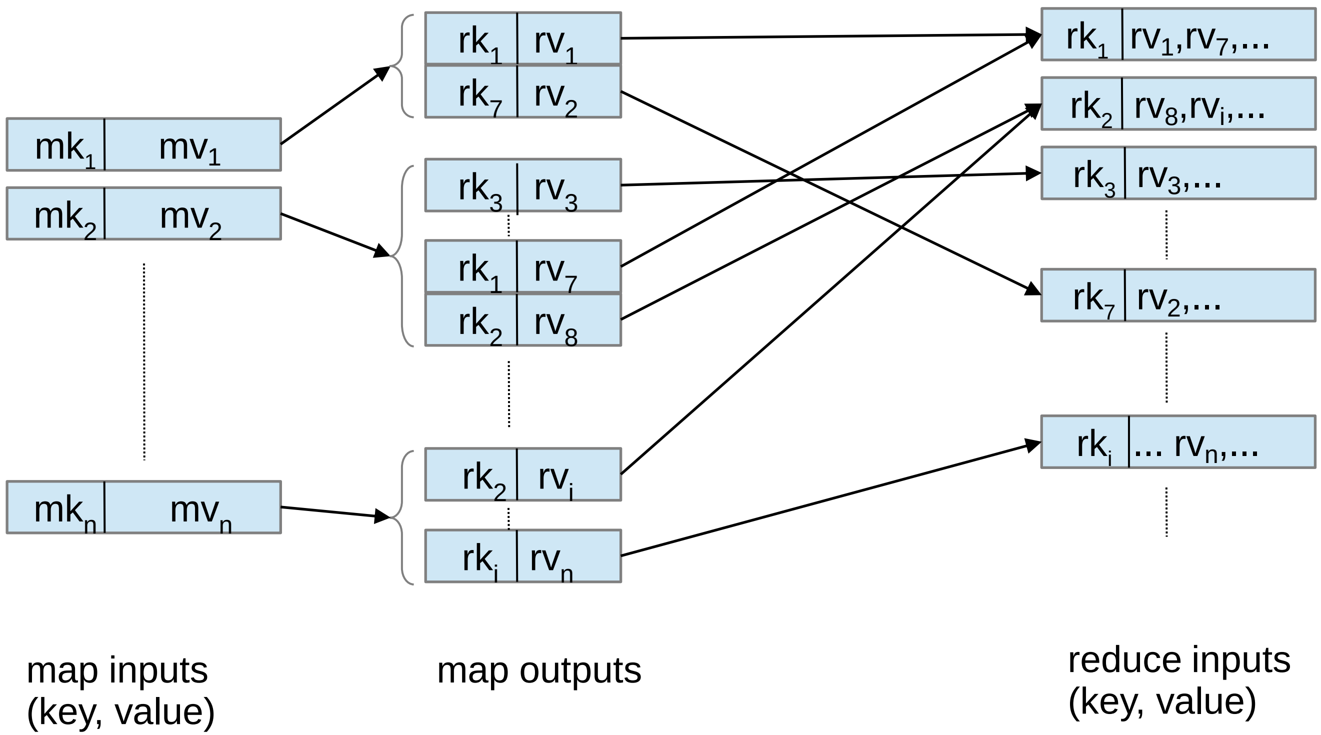
- Paradigm dates back many decades
  - But very large scale implementations running on clusters with 10^3 to 10^4 machines are more recent
  - Google Map Reduce, Hadoop, ..
- Data storage/access typically done using distributed file systems or key-value stores

MapReduce Programming Model

- Input: a set of key/value pairs
- User supplies two functions:
  - **map**(k,v) → list(k1,v1)
  - **reduce**(k1, list(v1)) → v2
- (k1,v1) is an intermediate key/value pair
- Output is the set of (k1,v2) pairs

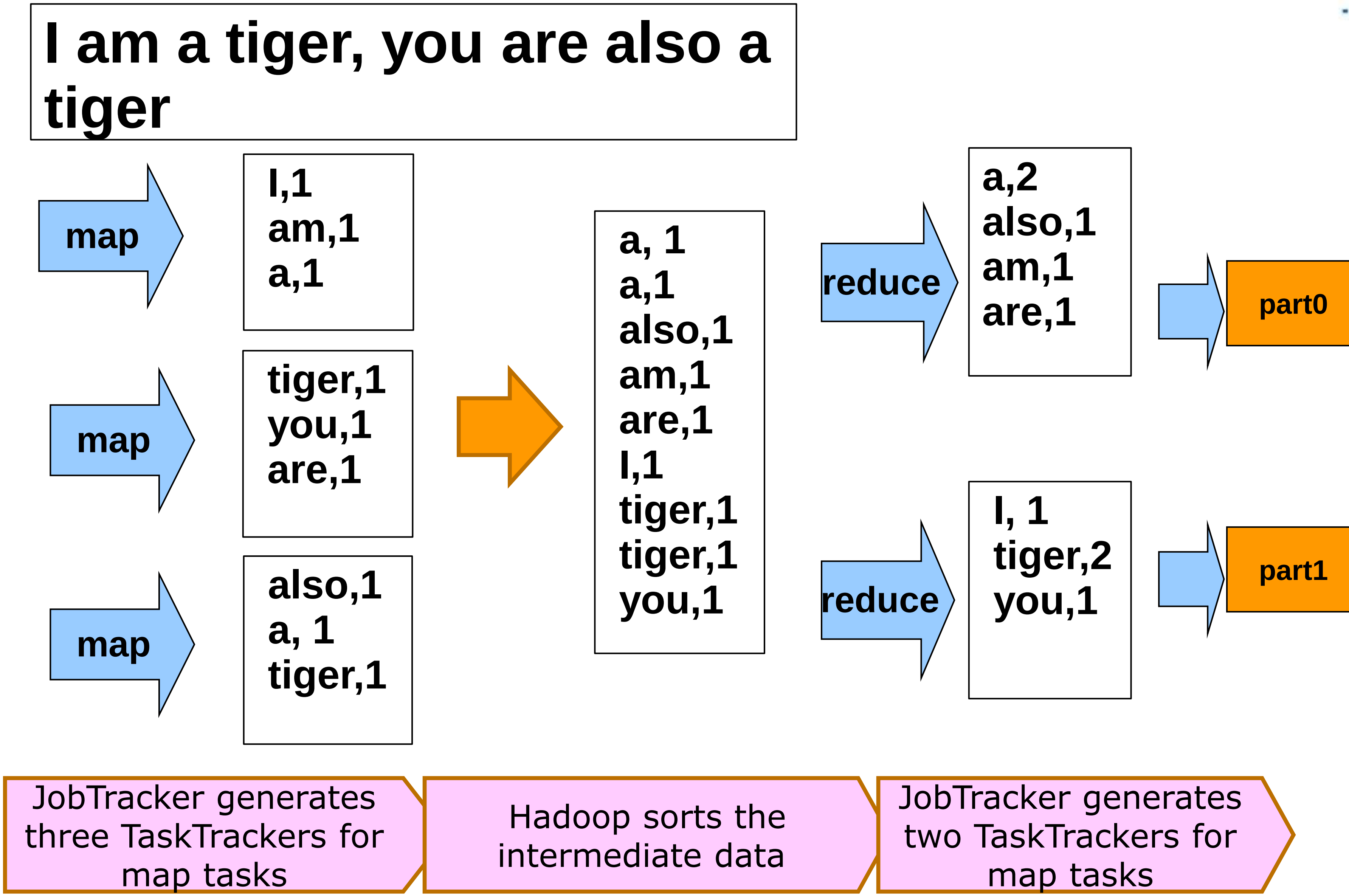
Flow of Keys and Values

- **Flow of keys and values in a map reduce task**



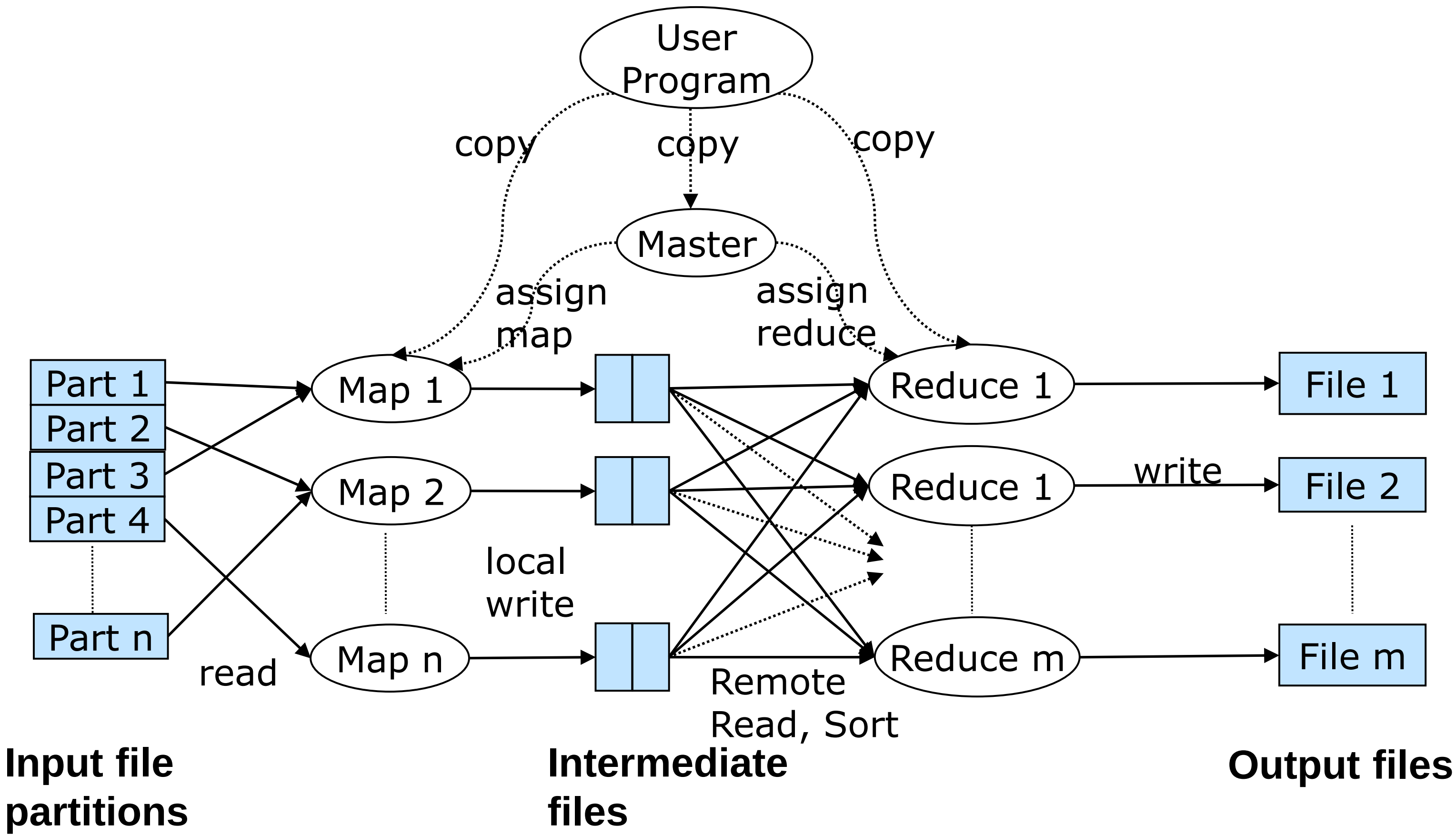
https://www.geeksforgeeks.org/how-to-execute-wordcount-program-in-mapreduce-using-cloudera-distribution-hadoop-cdh/

Example



I am a tiger, you are also a tiger
map
map
map
I,1
am,1
a,1
tiger,1
you,1
are,1
also,1
a, 1
tiger,1
a, 1
a,1
also,1
am,1
are,1
I,1
tiger,1
tiger,1
you,1
reduce
reduce
a,2
also,1
am,1
are,1
I, 1
tiger,2
you,1
part0
part1
JobTracker generates three TaskTrackers for map tasks
Hadoop sorts the intermediate data
JobTracker generates two TaskTrackers for map tasks

Parallel Processing of MapReduce Job



Map Reduce vs. Databases

- Map Reduce widely used for parallel processing
  - Google, Yahoo, and 100's of other companies
  - Example uses: compute PageRank, build keyword indices, do data analysis of web click logs, ...
- Many real-world uses of MapReduce cannot be expressed in SQL
- But many computations are much easier to express in SQL

Map Reduce vs. Databases (Cont.)

- Relational operations (select, project, join, aggregation, etc.) can be expressed using Map Reduce
- SQL queries can be translated into Map Reduce infrastructure for execution
  - Apache Hive SQL, Apache Pig Latin, Microsoft SCOPE

Where is MapReduce Inefficient?

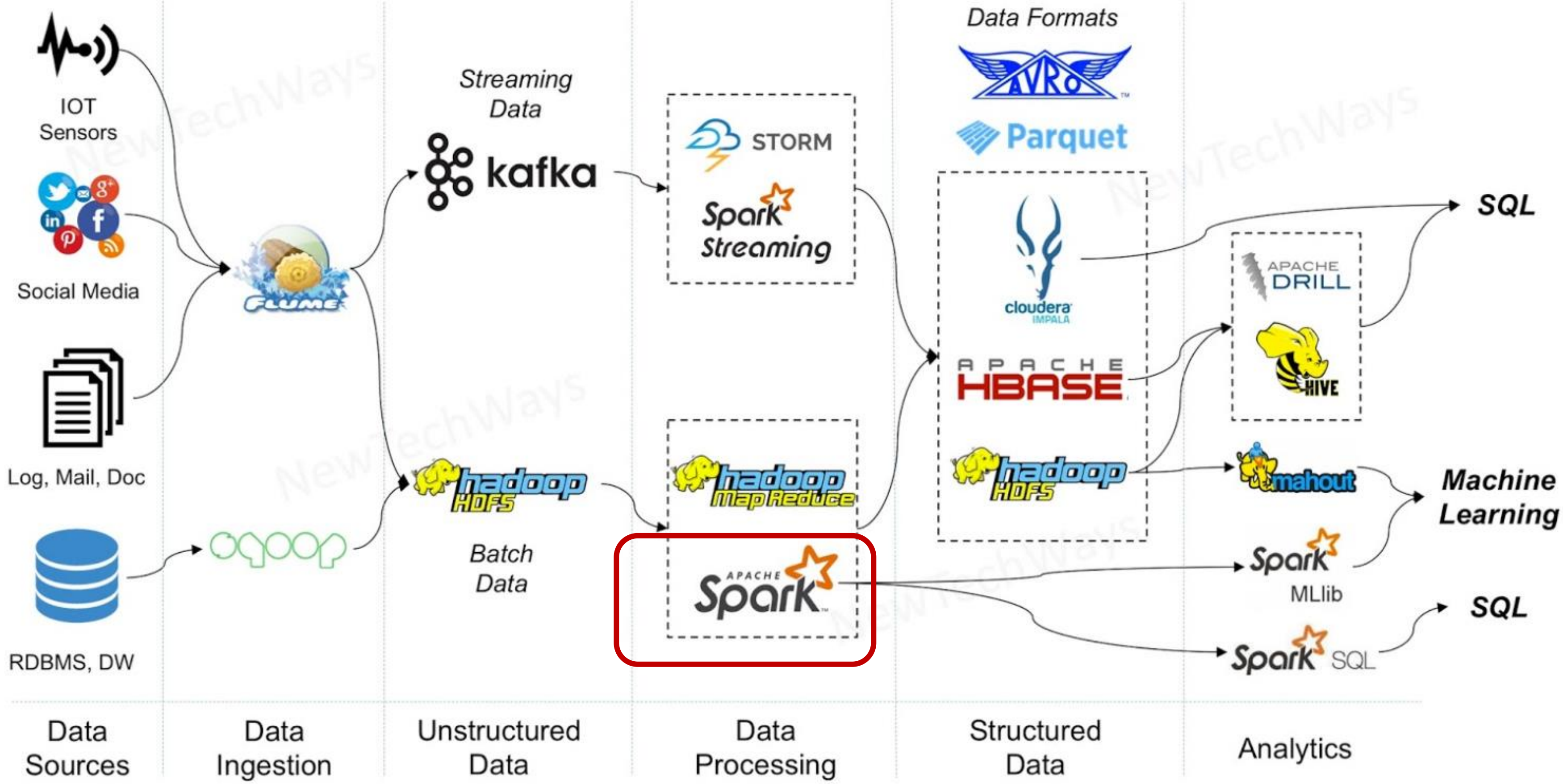
- Long pipelines sharing data
- Interactive applications
- Streaming applications

DFS ⇨ Map ⇨ LocalFS ⇨ Network ⇨ Reduce ⇨ DFS ⇨ Map ⇨ ...

(MapReduce would need to write and read from disk a lot)

Spark

- The key idea of Spark is **R**esilient **D**istributed **D**atasets (RDD)
- It supports in-memory processing computation



- **Resilient Distributed Dataset** (**RDD**) abstraction
  - Collection of records that can be stored across multiple machines
- Read-only partitioned collection of records (like a DFS) but with a record of how the dataset was created as a combination of transformations from other dataset(s)

Word Count in Spark

```
import java.util.Arrays;
import java.util.List;
import scala.Tuple2;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.sql.SparkSession;
public class WordCount {

    public static void main(String[] args) throws Exception {

        if (args.length < 1) {
            System.err.println("Usage: WordCount <file-or-directory-name>");
            System.exit(1);
        }
        SparkSession spark =
            SparkSession.builder().appName("WordCount").getOrCreate();

        JavaRDD<String> lines = spark.read().textFile(args[0]).javaRDD();
        JavaRDD<String> words = lines.flatMap(s -> Arrays.asList(s.split(" ")).iterator());
        JavaPairRDD<String, Integer> ones = words.mapToPair(s -> new Tuple2<>(s, 1));
        JavaPairRDD<String, Integer> counts = ones.reduceByKey((i1, i2) -> i1 + i2);

        counts.saveAsTextFile("outputDir"); // Save output files in this directory

        List<Tuple2<String, Integer>> output = counts.collect();
        for (Tuple2<String,Integer> tuple : output) {
            System.out.println(tuple);
        }
        spark.stop();
    }
}
```

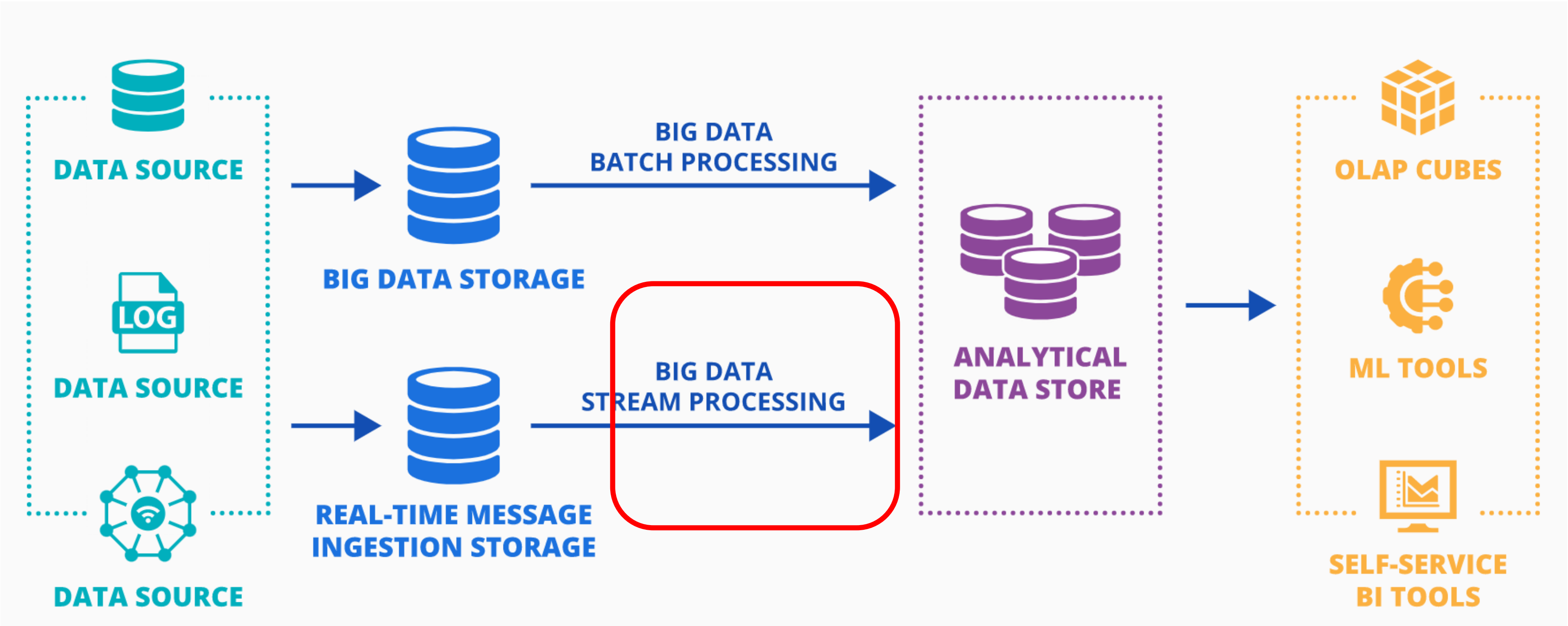
Spark DataFrames and DataSet

- RDDs in Spark can be typed in programs, but not dynamically
- The DataSet type allows types to be specified dynamically
- Row is a row type, with attribute names
  - In code below, attribute names/types of instructor and department are inferred from files read

Spark DataFrames and DataSet

- Operations filter, join, groupBy, agg, etc defined on DataSet, and can execute in parallel
- Dataset<Row> instructor = spark.read().parquet("...");
  Dataset<Row> department = spark.read().parquet("...");
  instructor.filter(instructor.col("salary").gt(100000))
  .join(department, instructor.col("dept name")
  .equalTo(department.col("dept name")))
  .groupBy(department.col("building"))
  .agg(count(instructor.col("ID")));

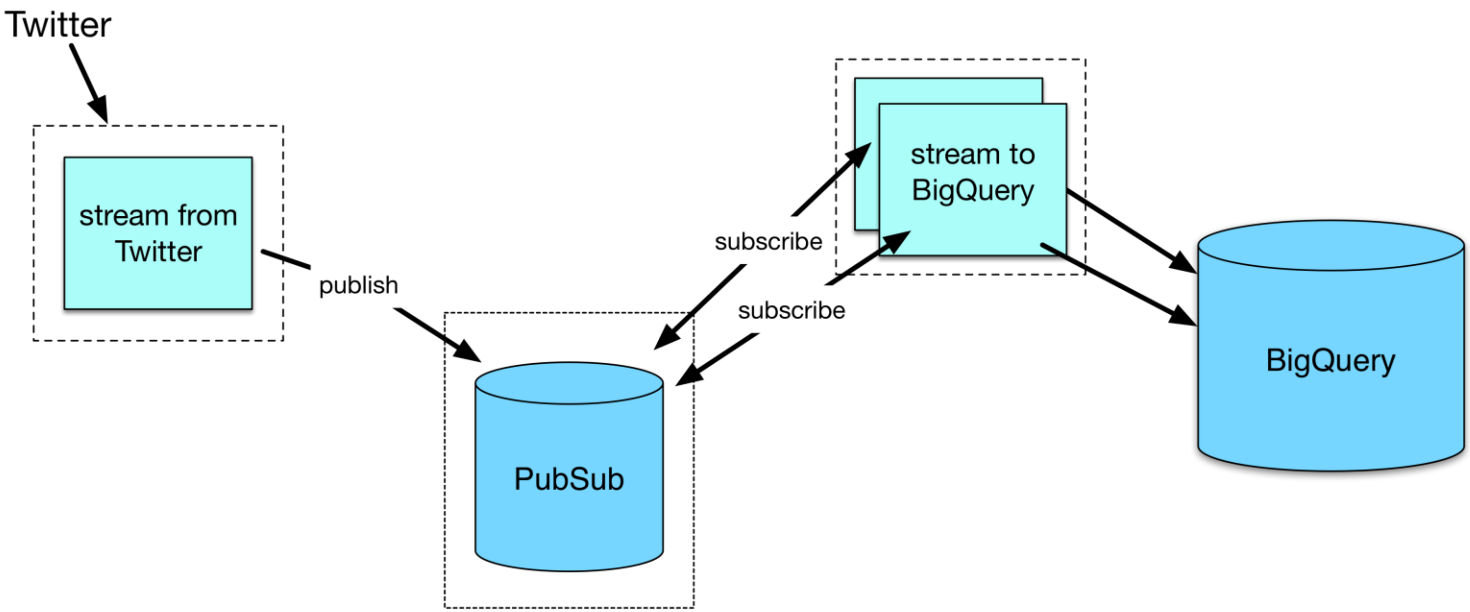
Streaming Data



Streaming Data and Applications

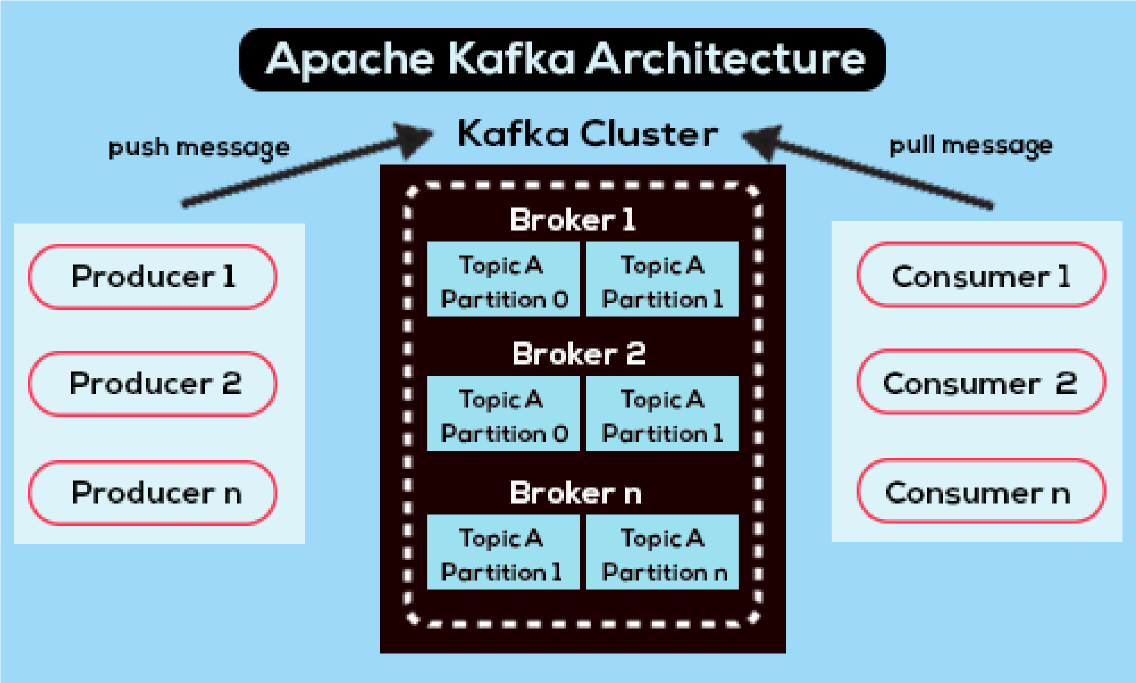
- **Streaming data** refers to data that arrives in a continuous fashion
- Applications include:
  - Stock market: stream of trades
  - Sensors: sensor readings
    - Internet of things
  - Network monitoring data
  - Social media: tweets and posts can be viewed as a stream
- Queries on streams can be very useful
  - Monitoring, alerts, automated triggering of actions

- **Publish-subscribe** (**pub-sub**) systems provide a convenient abstraction for processing streams
  - Tuples in a stream are published to a topic
  - Consumers subscribe to topic

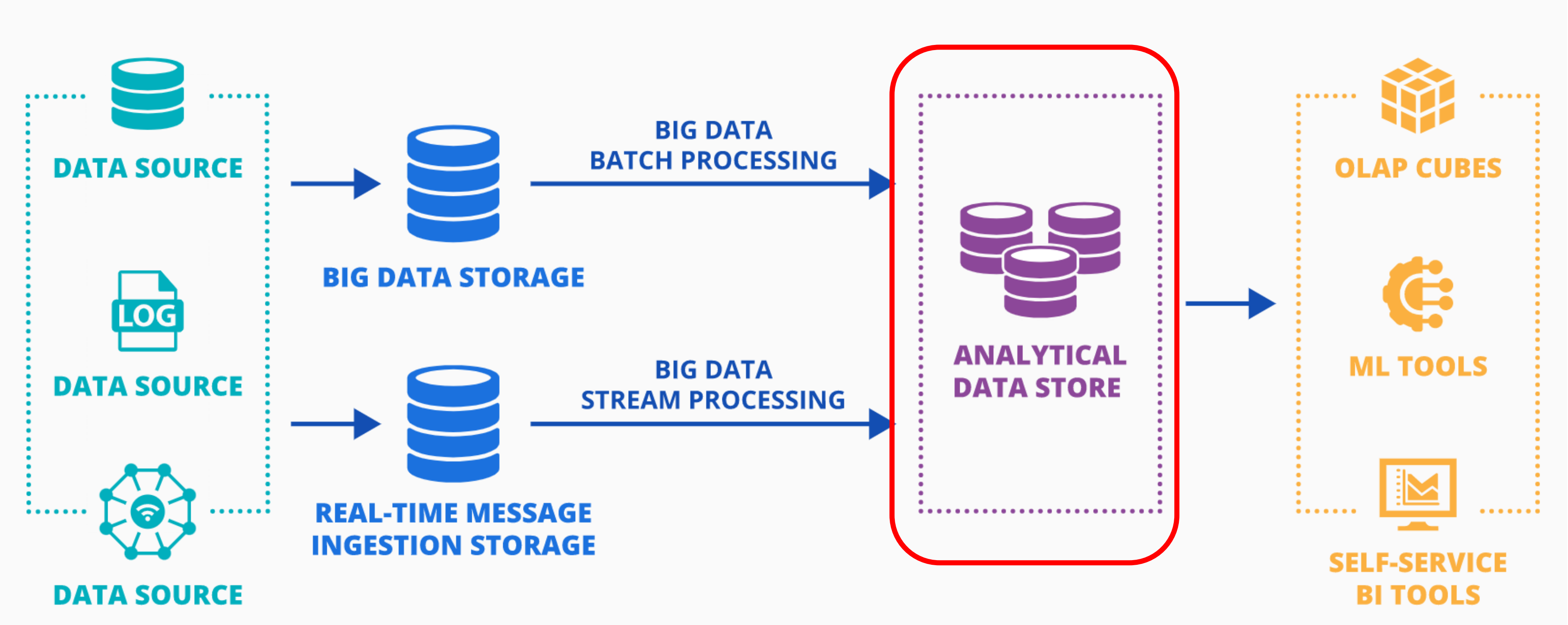


Apache Kafka

- **Apache Kafka** is a popular parallel pub-sub system widely used to manage streaming data
- Parallel pub-sub systems allow tuples in a topic to be partitioned across multiple machines



Big data architecture



1. Overview
2. Data Warehousing (DW)
3. Online Analytical Processing (OLAP)
4. Data Mining

Overview

- **Data analytics**: the processing of data to infer patterns, correlations, or models for prediction
- Primarily used to make business decisions
  - E.g., what product to suggest for purchase
  - E.g., what products to manufacture/stock, in what quantity
- Critical for businesses today

Common steps in data analytics

- Gather data from multiple sources into one location
- Data warehouses also integrate data into a common schema
- Data often needs to be **extracted** from source formats, **transformed** into common schema, and **loaded** into the data warehouse (**ETL**)

- **Generate aggregates and reports summarizing data**
  - Dashboards showing graphical charts/reports
  - **Online analytical processing (OLAP) systems** allow interactive querying
  - Statistical analysis using tools such as R/SAS/SPSS
- Build **predictive models** and use the models for decision making

Overview (Cont.)

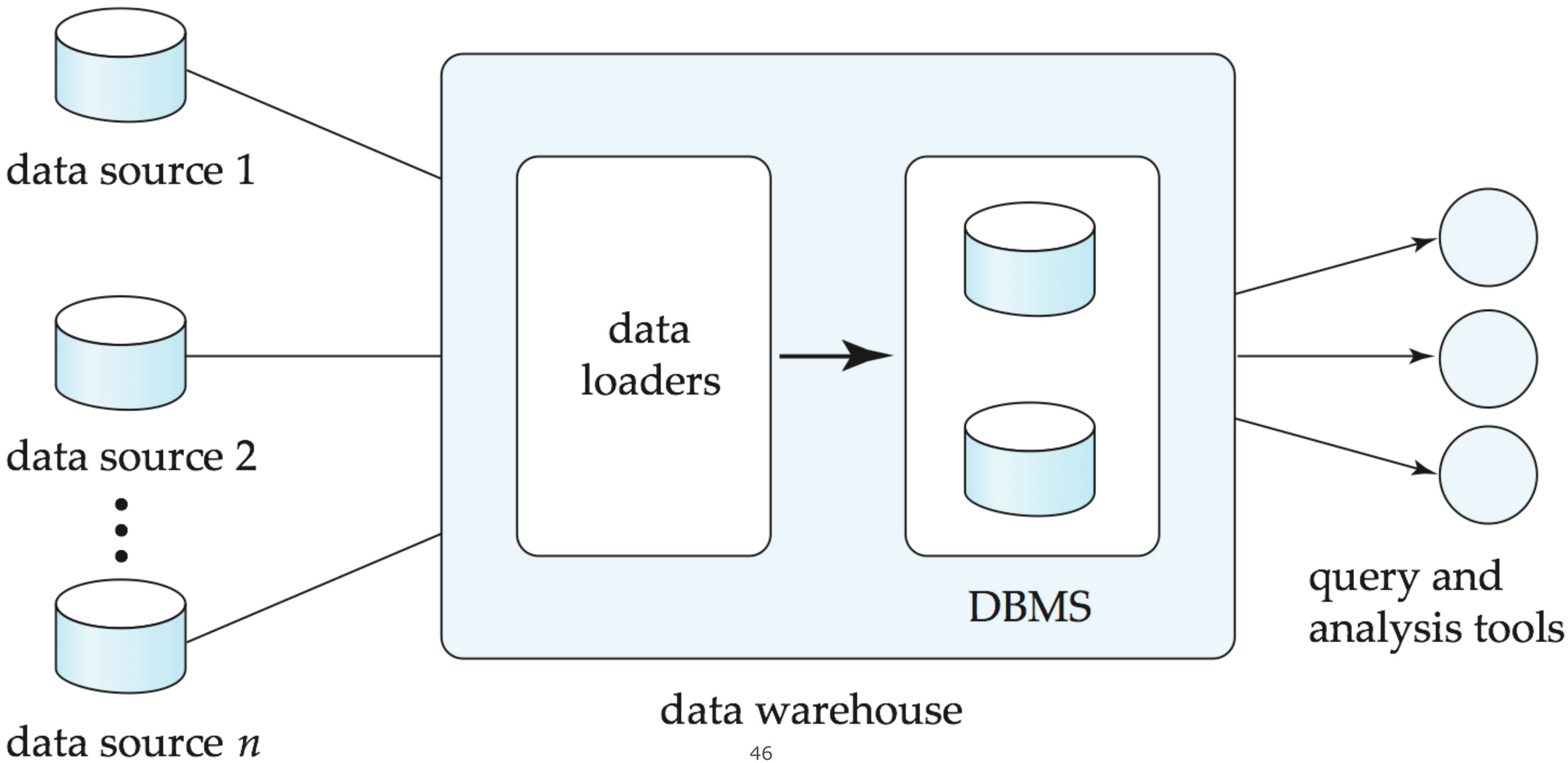
- Predictive models are widely used today
  - E.g., use customer profile features and the history of a customer to predict the likelihood of default on a loan
  - E.g., use history of sales to predict future sales
- Other examples of business decisions:
  - What items to stock?
  - What insurance premium to change?
  - To whom to send advertisements?

Overview (Cont.)

- **Machine learning** techniques are key to finding patterns in data and making predictions
- **Data mining** extends techniques developed by machine-learning communities to run them on very large datasets
- The term **business intelligence (BI)** is synonym for data analytics

Data Warehousing

- A **data warehouse** is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site



Warehouse Design issues

- Data transformation and data cleansing
  - E.g., correct mistakes in addresses (misspellings, zip code errors)
- How to propagate updates
- What data to summarize

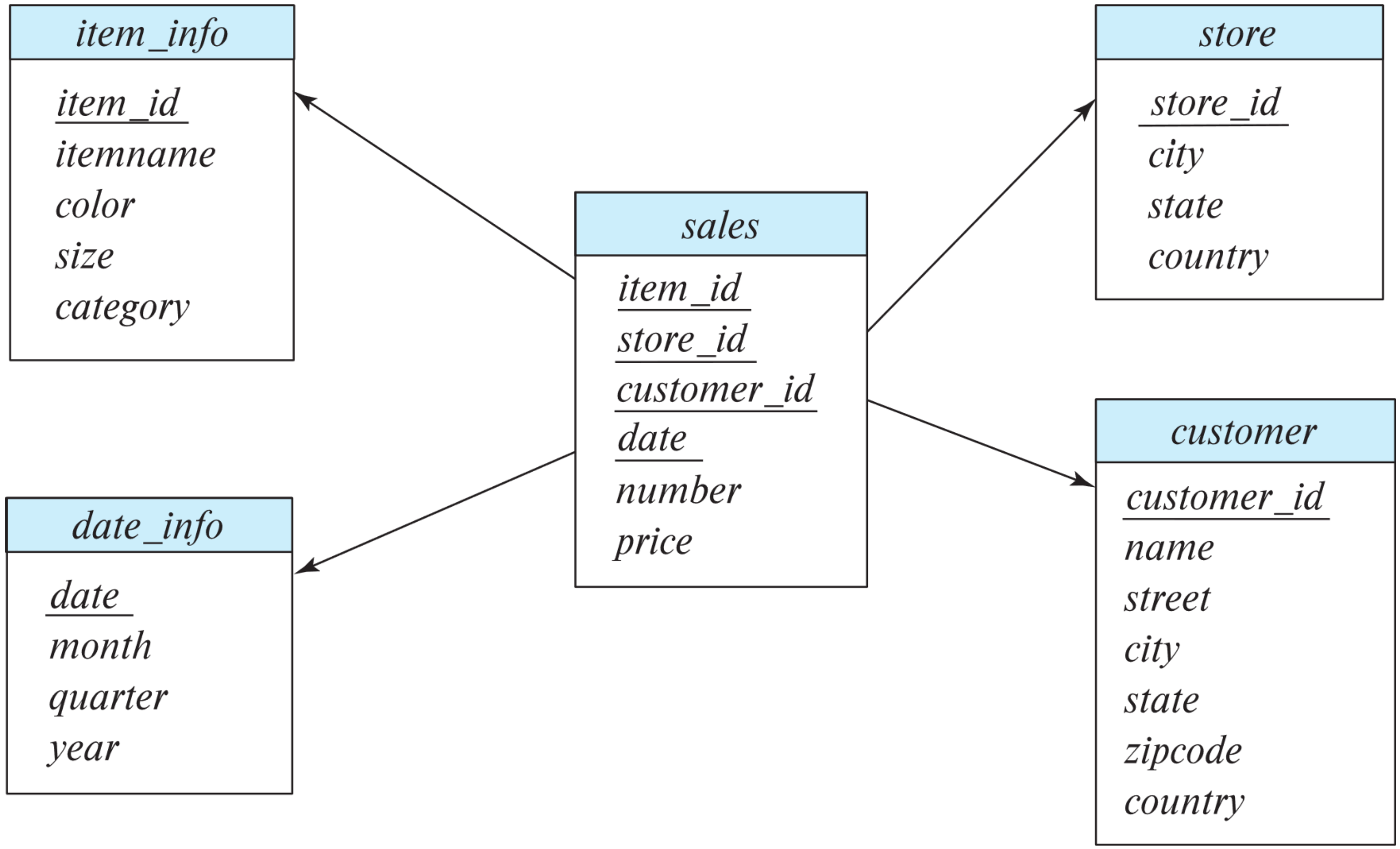
Multidimensional Data

- Data in warehouses can usually be divided into
  - **Fact tables**, which are large
    - E.g, *sales*(*item_id, store_id, customer_id, date, number, price*)
  - **Dimension tables**, which are relatively small
    - Store extra information about stores, items, etc.

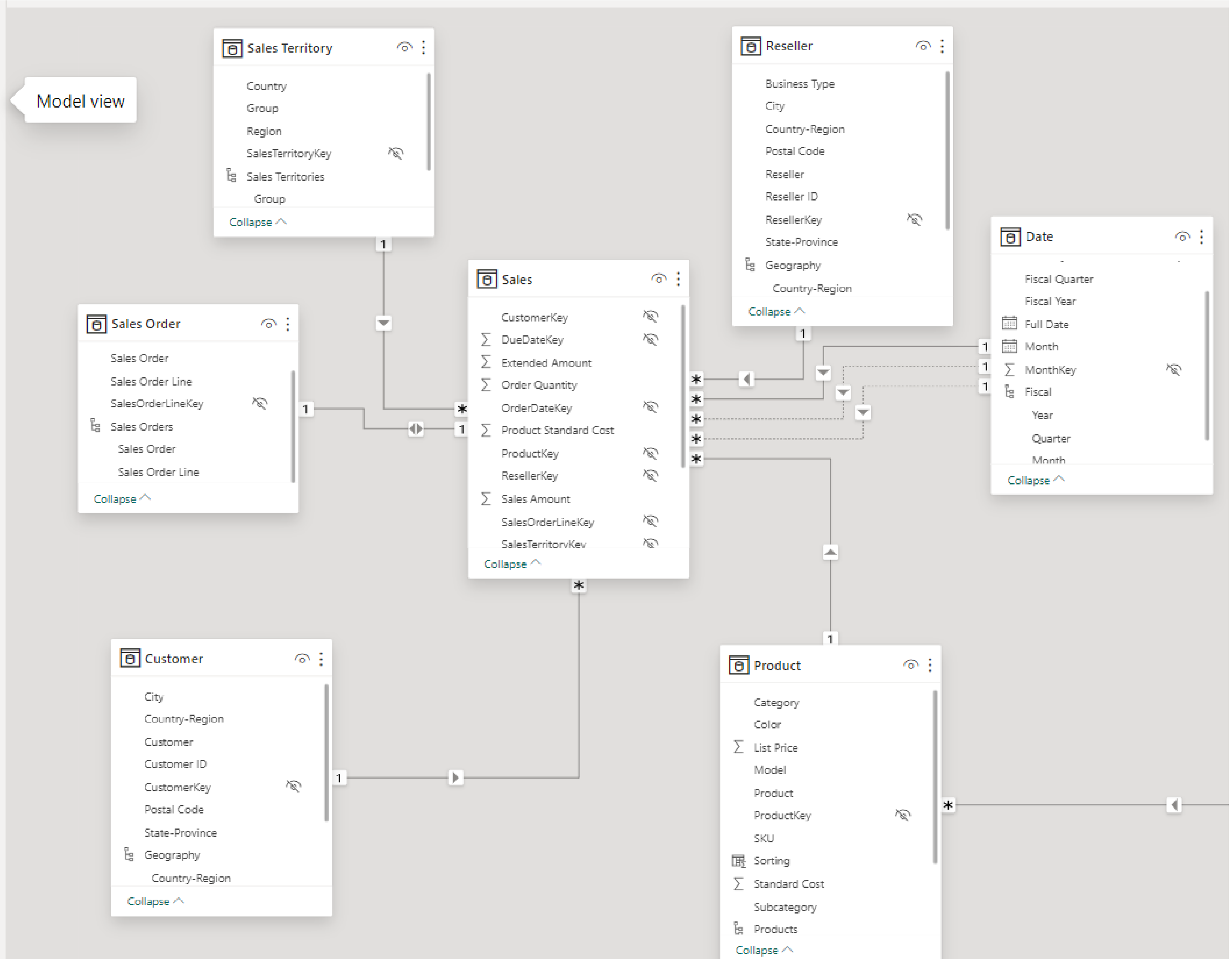
Fact Tables

- Attributes of **fact tables** can be usually viewed as
  - **Measure attributes**
    - measure some value, and can be aggregated upon
    - e.g., the attributes *number* or *price* of the *sales* relation
  - **Dimension attributes**
    - dimensions on which measure attributes are viewed

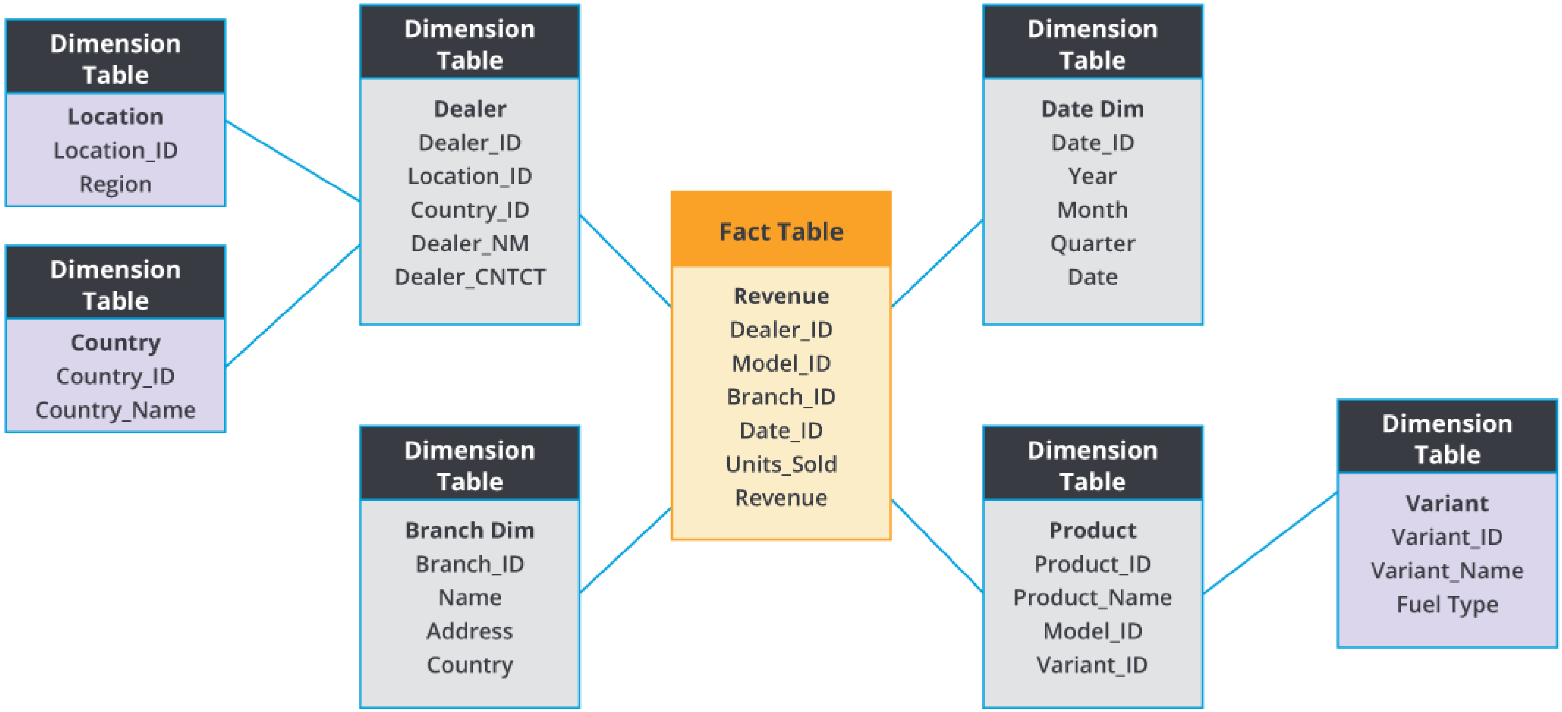
Data Warehouse Star Schema



More on Data Warehouse Star Schema



- More complicated schema structures
  - **Snowflake schema**: multiple levels of dimension tables

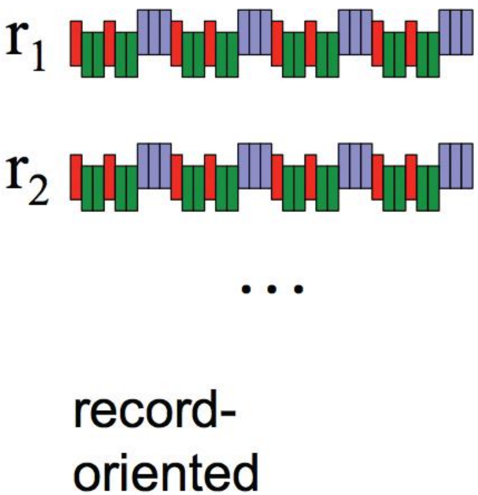


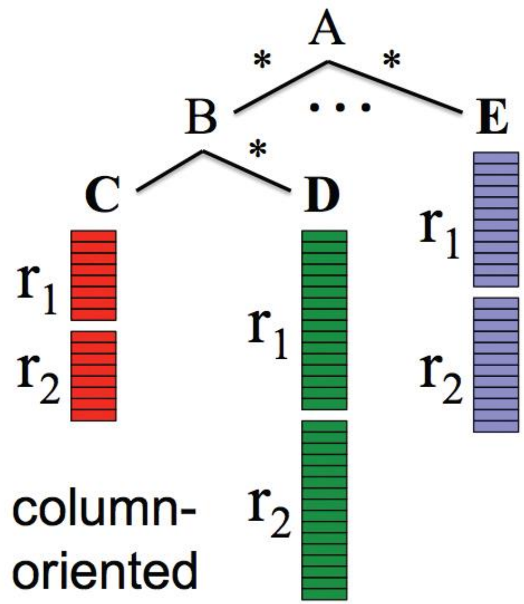
Example of Snowflake Schema

Data lakes

- Some applications do not find it worthwhile to bring data to a common schema
  - **Data lakes** are repositories which allow data to be stored in multiple formats, without schema integration
  - Less upfront effort, but more effort during querying

- Data in warehouses usually append-only, not updated. Can avoid concurrency control overheads
- Data warehouses often use **column-oriented storage**





Column-oriented storage

- Arrays are compressed, reducing storage, IO and memory costs significantly
- Queries can fetch only attributes that they care about, reducing IO and memory cost
- Data warehouses often use parallel storage and query processing infrastructure

- **Online Analytical Processing (OLAP)**
- Interactive analysis of data, allowing data to be summarized and viewed in different ways in an online fashion (with negligible delay)

Cross Tabulation

- The table below is an example of a **cross-tabulation** (**cross-tab**), also referred to as a **pivot-table**

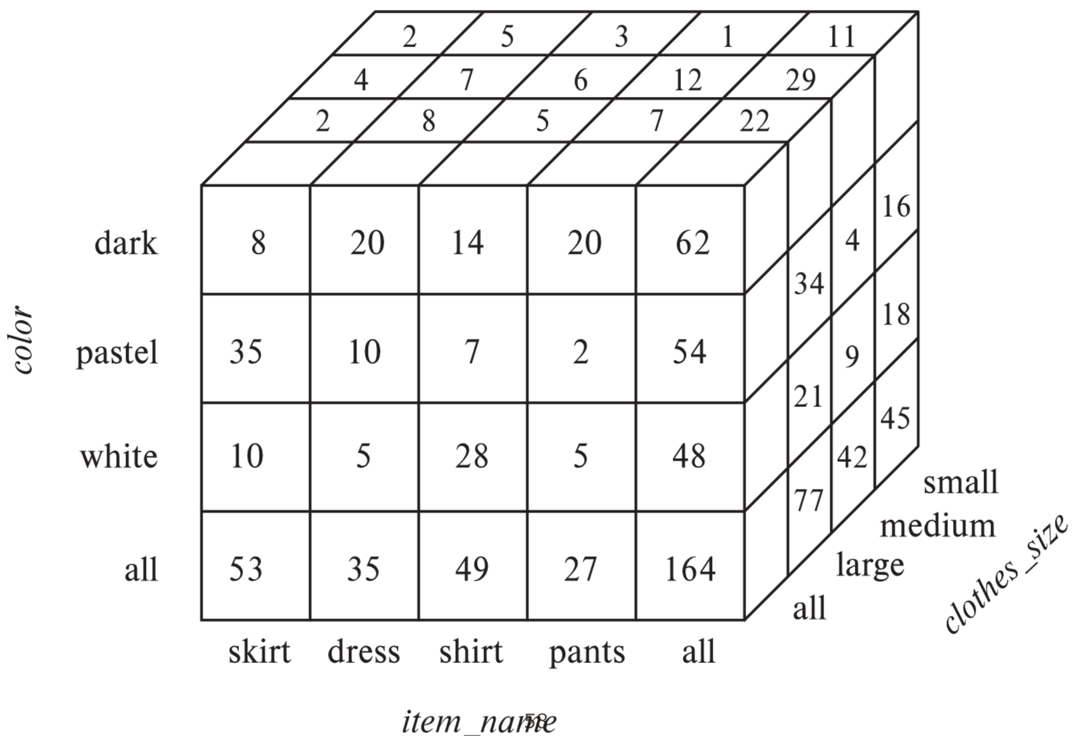
clothes_size **all**

color

item_name		dark	pastel	white	total
	skirt	8	35	10	53
	dress	20	10	5	35
	shirt	14	7	28	49
	pants	20	2	5	27
	total	62	54	48	164

Data Cube

- A **data cube** is a multidimensional generalization of a cross-tab
- Can have n dimensions; we show 3 below
- Cross-tabs can be used as views on a data cube

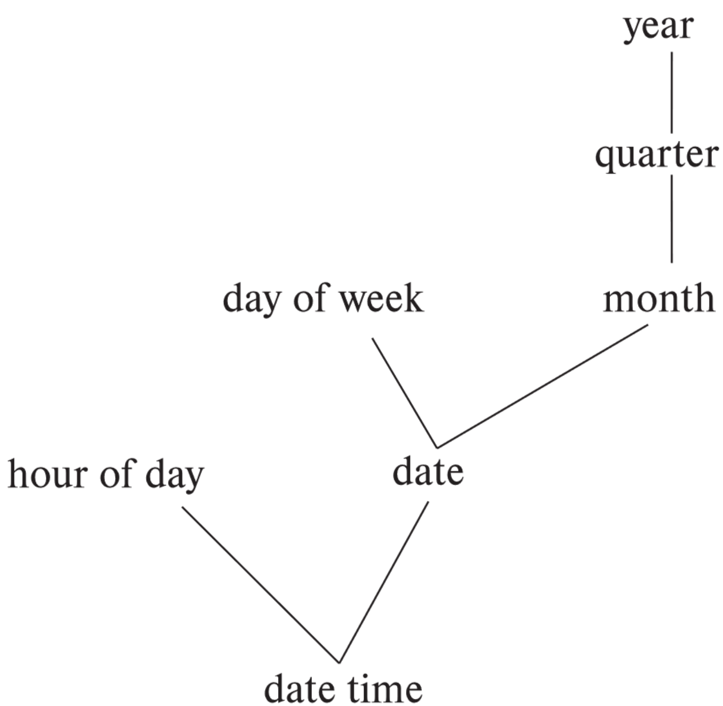


Online Analytical Processing Operations

- **Pivoting:** changing the dimensions used in a cross-tab
- **Slicing:** creating a cross-tab for fixed values only
- **Rollup:** moving from finer-granularity data to a coarser granularity
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data

Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions be viewed at different levels of detail



(a) time hierarchy

(b) location hierarchy

Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
- Can drill down or roll up on a hierarchy
- E.g. hierarchy: *item_name* → *category*

clothes_size: **all**

category	*item_name*	*color* dark	pastel	white	total	
womenswear	skirt	8	8	10	53	
	dress	20	20	5	35	
	subtotal	28	28	15		88
menswear	pants	14	14	28	49	
	shirt	20	20	5	27	
	subtotal	34	34	33		76
total		62	62	48		164

Reporting and Visualization

- **Reporting tools** help create formatted reports with tabular/graphical representation of data
- **Data visualization** tools help create interactive visualization of data
  - E.g., PowerBI, Tableau, FusionChart, plotly, Datawrapper, Google Charts, etc.

Acme Supply Company, Inc.
Quarterly Sales Report

Period: Jan. 1 to March 31, 2009

Region	Category	Sales	Subtotal
North	Computer Hardware	1,000,000	
	Computer Software	500,000	
	All categories		1,500,000
South	Computer Hardware	200,000	
	Computer Software	400,000	
	All categories		600,000
		Total Sales	2,100,000

- **Data mining** is the process of semi-automatically analyzing large databases to find useful patterns
- Some types of knowledge can be represented as rules
- More generally, knowledge is discovered by applying **machine learning techniques** to past instances of data to form a **model**

- **Prediction** based on past history
  - Predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, ..) and past history
- Some examples of prediction mechanisms:
  - **Classification**
    - Items (with associated attributes) belong to one of several classes
    - **Training instances** have attribute values and classes provided
  - **Regression** formulae
    - Given a set of mappings for an unknown function, predict the function result for a new parameter value

THANKS YOU