Big Data stores and tools

Parte 9 (2015) – *Knowledge Management And Protection Systems*

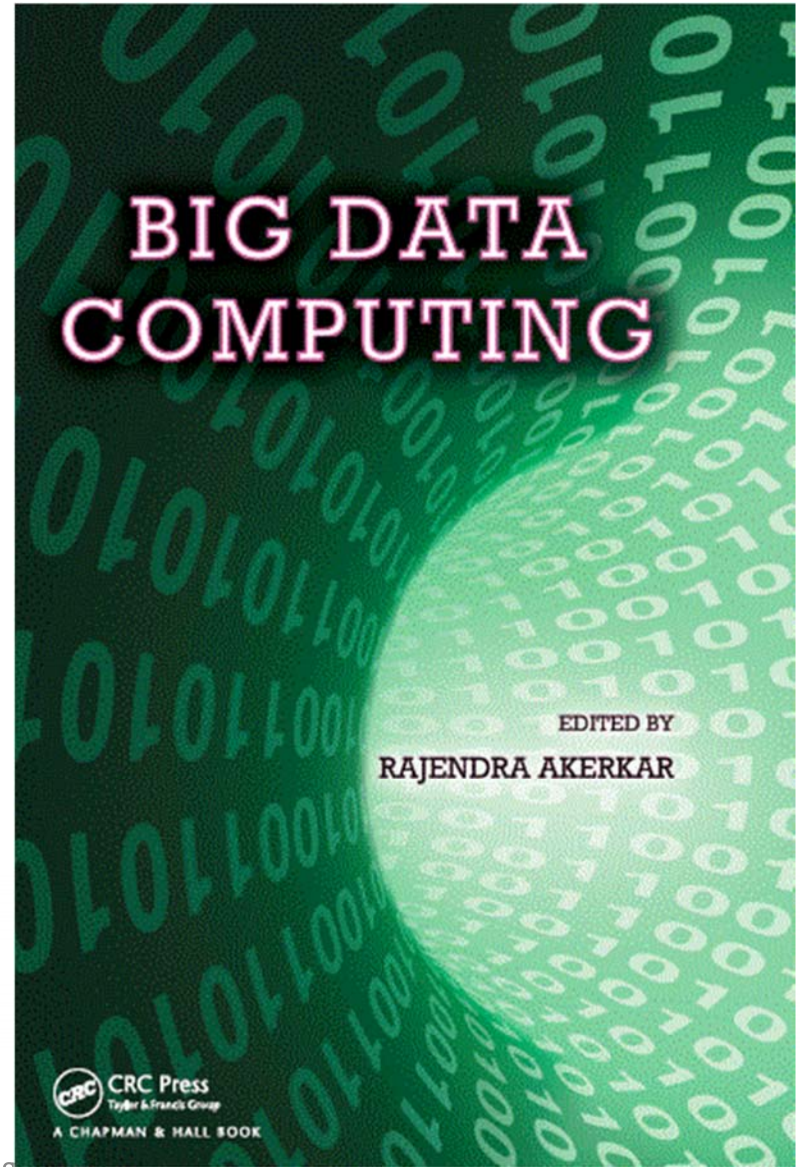
Ing. Nadia Rauch
nadia.rauch@unifi.it

DISIT Lab
Department of Information Engineering
University of Florence
http://www.disit.dinfo.unifi.it/

Slide del corso:
Knowledge Management And Protection Systems
(Prof.Paolo Nesi, paolo.nesi@unifi.it)

P. Bellini, M. Di Claudio, P. Nesi, N. Rauch, "Tassonomy and Review of Big Data Solutions Navigation", in "Big Data Computing", Ed. Rajendra Akerkar, Western Norway Research Institute, Norway, Chapman and Hall/CRC press, ISBN 978-1-46-657837-1, eBook: 978-1-46-657838-8, july 2013, in press.

http://www.tmrfindia.org/bigdata.html



Index

WHAT IS BIG DATA

Index

- **What is Big Data**
- 5V of Big Data
- CAP Principle
- Big Data Application Fields
- Big Data Problems, Criticality and Risk
- NoSQL
- Big Data Analysis Pipeline
- Big Data Solutions

A bit of History…

- In the 60s data was stored on **disks** and **magnetic tapes**. **Static** and limited **analysis** was carried out on them (e.g. number of sales in the last six months).

- In the 80s **relational database** and **SQL** (Structured Query Language) allow to realize a more **dynamic analysis**. The analysis was conducted over **operational DB**, where, for example the daily activity of a company is registered.

Operational Data Base

- **OLTP** (OnLine Transaction Processing).
- They have a highly **normalized** data **model**.
- **Data analysis** is carried out by **different applications**
  - 1 application for orders, 1 for billing, etc.
- Using **different applications** is not possible to guarantee **data uniformity** and **consistency**
  - Data Handling and Replication with different sw.
  - Multiple data formats.
  - Data updates not guaranteed.

Operational Data Base

- A highly **normalized** data **model:**
  - ☺ Facilitates data insertions, deletions and modifications (transactional activities).
  - ☹ Makes reading more difficult.
  - ☹ Increases the number of used tables.
  - ☹ Complicates data extraction (many JOIN to denormalize data).
  - ☹ Limits historical data.
- This data model is not suitable for big data, and their analysis.

Data Warehouse

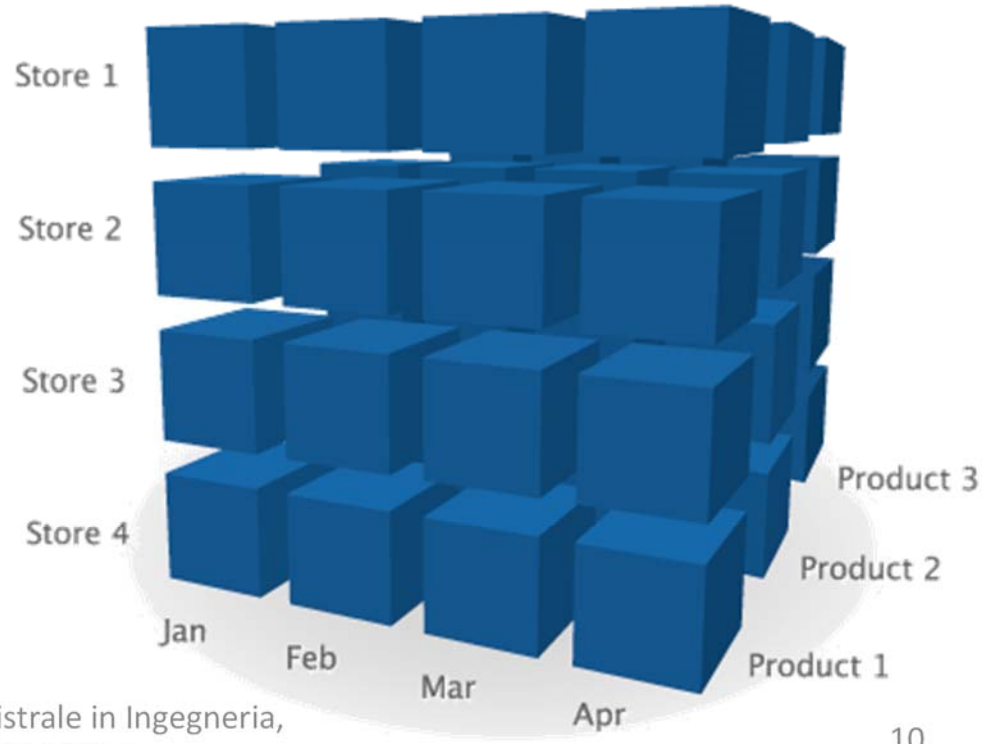
- Introduced in the early 90s.
- Integrates data from different operational systems → database that contains **integrated, consistent** and **certificates data**, concerning all company processes.
- Data can be **processed**, **aggregated**, **analyzed** and transformed (value) into information, which are **stored** and **accessed** in a **simple** and **flexible way**.

OLAP Cube

- In the years after **multidimensional databases** have been introduced that **combine data** and **metadata**, and allows the analyst to focus on data → **OLAP** (OnLine Analytical Processing).

Simplified operations:

- **Drill up** & **Drill down** (go into details)
- **Slicing** (dimensionality reduction)
- **Dicing** (results filtering)



Data Mining

- Since the beginning of 2000s comes the need to obtain **predictions** and **suggestions** from data analysis, to **anticipate events**.
- **Data Mining**: a set of techniques that can "*excavate*" into data, to extract **new information** and **meaning**, not immediately obvious.
- Applications:
  - Customer segmentation
  - Market basket analysis
  - Advertising campaigns
  - Sales forecasts

Why Big Data?

- Since 2010, **new** evolution **trends** have emerged:
  - Business Analytics
  - Collaboration and Information Sharing
  - Cloud Computing
- **Data Sources** used increasingly :
  - Operational Database
  - Sensors and Scientific Instruments
  - Non-structural data

Traditional databases are **not enough**!!!

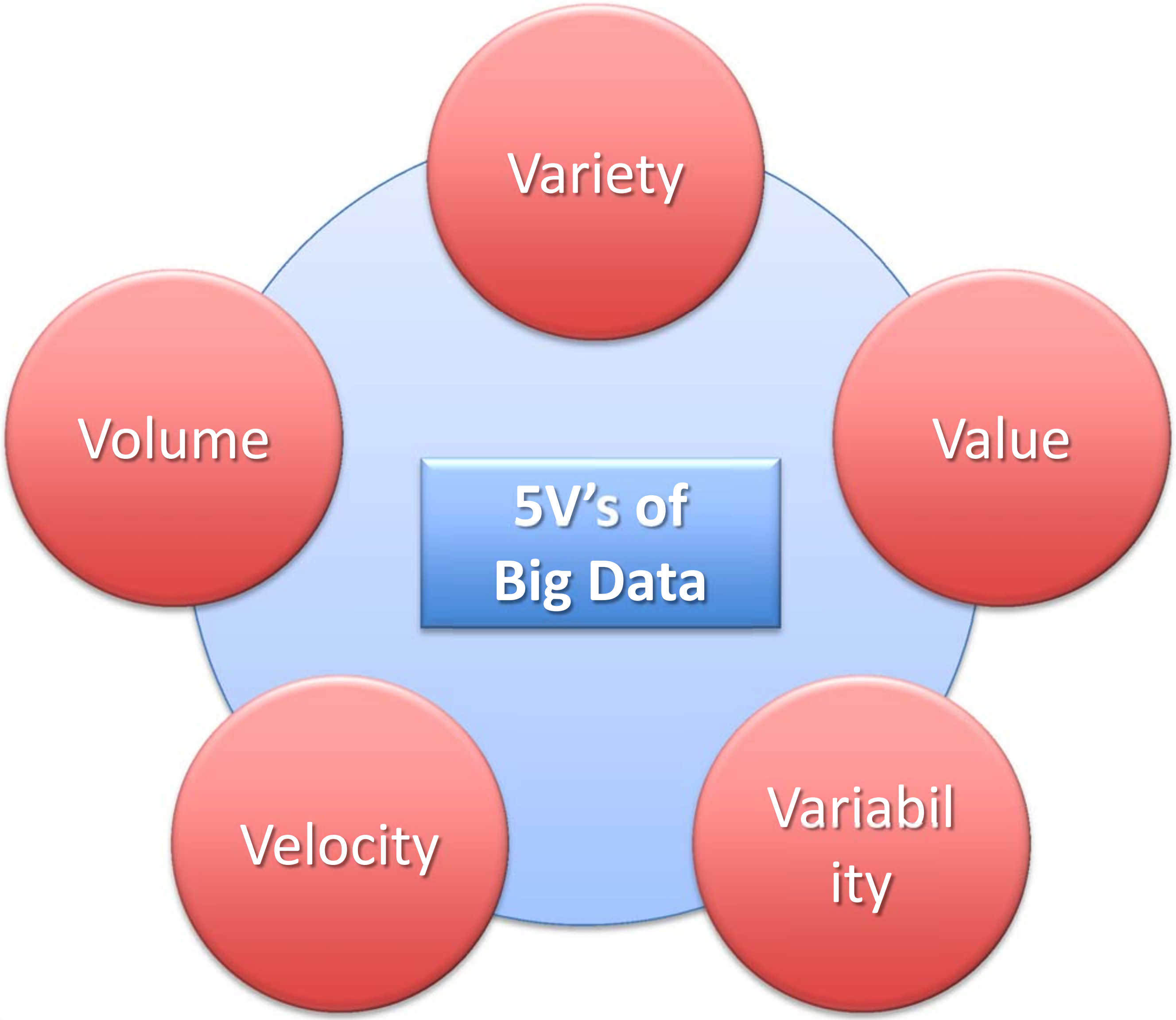
5V OF BIG DATA

Index

Big Data: a definition

- Data usually available in large **volumes**, which is presented in **different formats** (often without any structure) and with **heterogeneous characteristics**, it is produced and distributed generally with a **high frequency**, and it **often changes** over time.

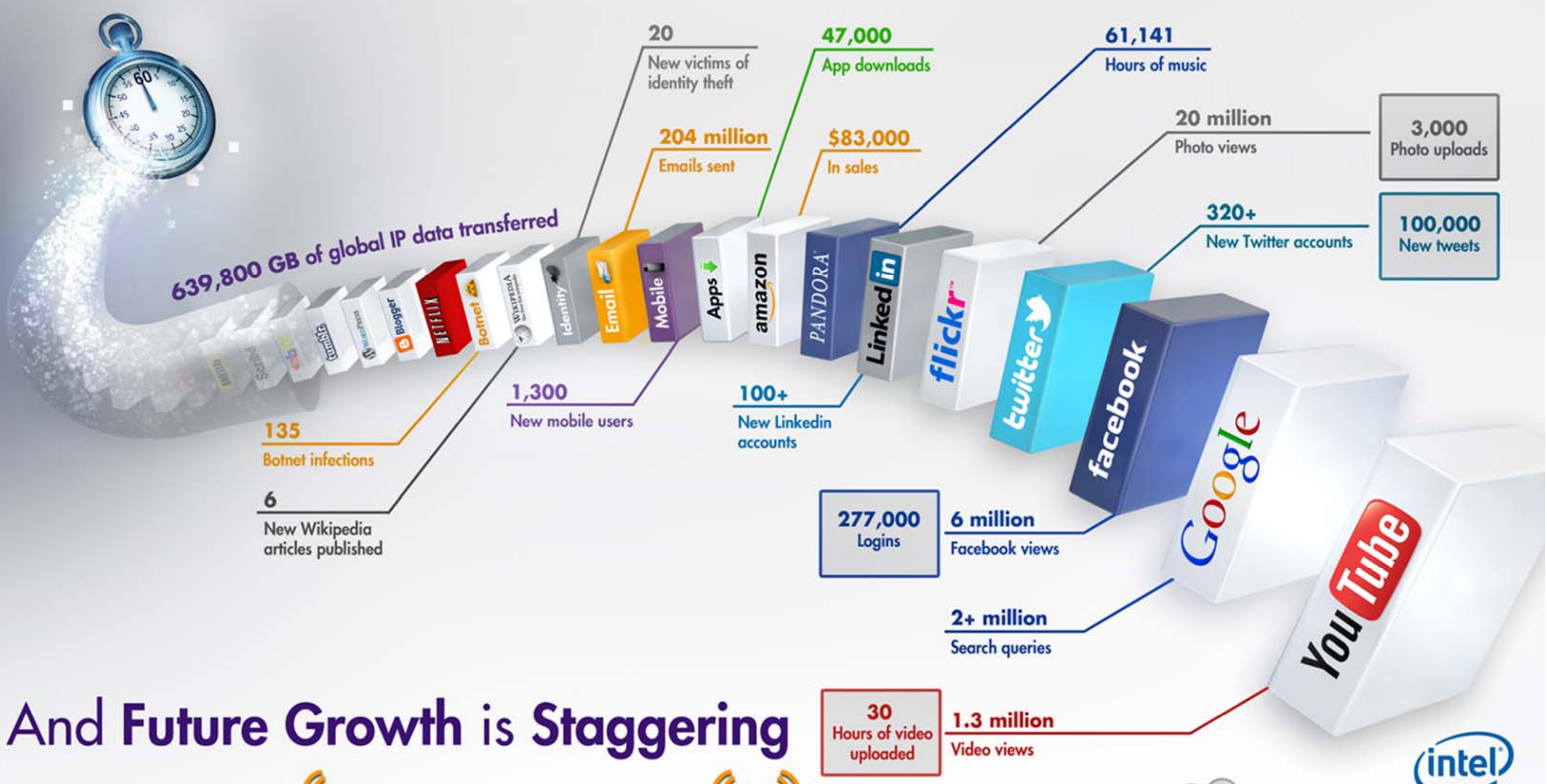
5V of Big Data



5V of Big Data - Volume

- Companies amassing **terabytes/petabytes** of information, and they always look for faster, more efficient, and **lower-cost solutions** for data management.
- In **1 minute**:
  – **100 000** tweets sent around the world.
  – **35 000** FB "Like" on official organization's websites.
  – **204 million** of emails sent.
  – **2000** check in on 4square.
- The **first step** in working with big data, is **storage**. **Analysis** (and **cleaning**) are performed at a **later** stage (to avoid losing potential information).

What Happens in an Internet Minute?



And Future Growth is Staggering

5V of Big Data - Velocity

- 2 meanings:
  - It refers to the **high frequency** at which **data** are **generated** and affects the amount (**volume**).
  - It refers to the **speed** at which new technologies allow **to access and analyze** this **data**.
- For **time-sensitive processes**, Big Data must be used as **Data Streams** in order to maximize its **value**.

➡ Higher speed in data access

➡ Higher speed in decision-making

➡ Higher market competitiveness

5V of Big Data - Velocity

- A Distributed Architecture is recommended:
  - Management of complex data structures.
  - Access to real-time data.
  - Good processing speed through techniques of distributed computing.
  - Non-relational databases such as DB column and key/value database (NoSQL).

5V of Big Data - Variety

- It refers to **the form** in which **data** are provided.
- Big Data includes any type of data: **structured** and **unstructured data** such as **text, sensor data**, **audio, video, click streams, log files** and more.
- **Not suitable** to be processed with **traditional techniques** of **relational databases**: email, images, video, audio, text strings that give meaning **can not be stored in a table**.
- A **NoSQL database** is recommended: do not impose a rigid scheme to organize data (**schemaless** database)

5V of Big Data - Variability

- 2 meanings:
  - Refers to **variance** in **meaning** and in **lexicon**, that is the **data contextualization**.
  - Refers to the **variability** in **data structure**.
- Example: "**read the book**“
  - **positive meaning** in a blog about literature
  - **negative connotation** in a blog for movie fans.
- It is important to **find mechanisms** that are able to **give a semantics** to the data **based on the context** in which they are expressed.

5V of Big Data - Value

- Big Data **hiding** a great **value**.
- With the primary use you can extract only a part, the remaining **value remains "dormant"** until their secondary use.
- **Value is** all that you can gain from all possible modes of use of the data, the sum of many small parts that are **slowly discovered**.
- **Changing direction**: once **data were eliminated** after the first use.

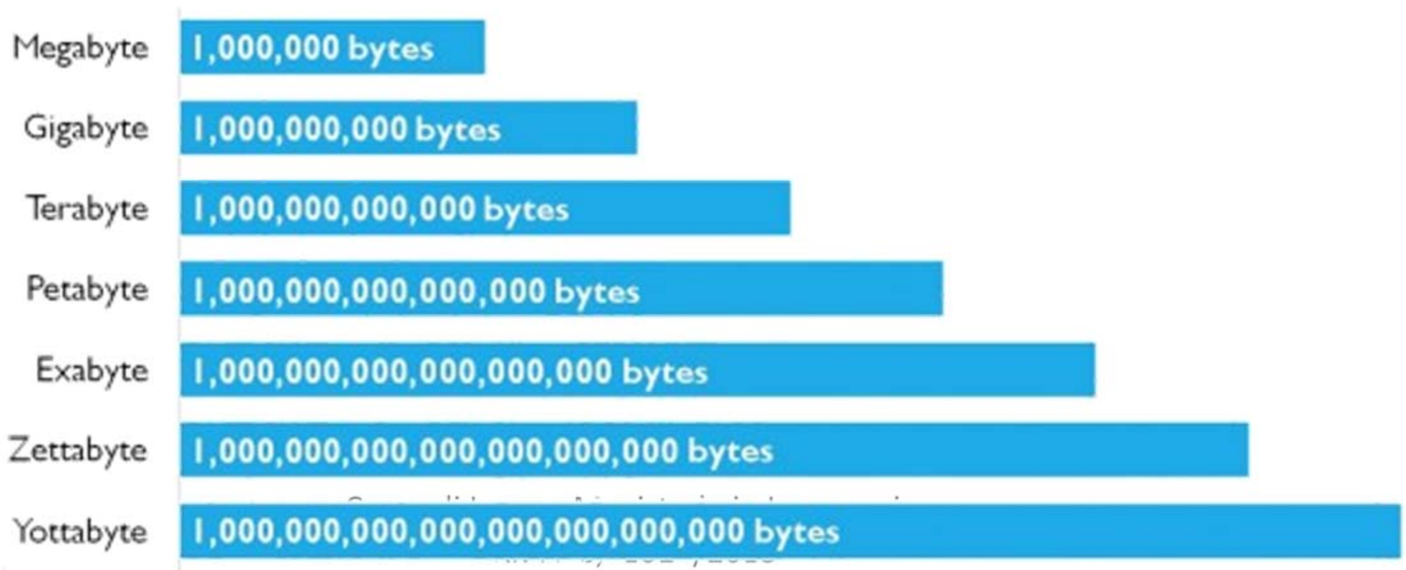
It is important to adopt methods and technologies that allow **a continuous integration of new information**, following a repeated use, with the goal of **building a knowledge base** always wider

5V of Big Data…or more?

- Initially there were 3V, then 4 and now 5V of Big Data. Different meanings are attributed to the 5V.
- **Viral**: refers to how much and how the data are spread (data propagation).
- The **large amount** of data and the **high speed** at which they are produced involves a **viral spread** of information.
- **Viral** is the **volume growth** of data generated by users digital activities (**user-generated content**)

5V of Big Data…or more?

- In **2010** it was estimated a production of **1.2 zettabytes** of data (**1ZB = one trillion GB**).
- In **2011**, grew up in **1,8ZB**.
- In **2013** came to **2,7ZB**.
- The prediction for **2015** is about **4,8ZB**.



Why a content became viral?

- A data analysis company sought to understand what are the characteristics that make a viral content:
  - **Dimensions**: greater length of content → more shares.
  - **Emotions**: people love to share elements that cause laughter and amazement (42%). Emotions to avoid: sadness and fear (7%).
  - **Images**: visual contents attract attention, encourage understanding → increase shares on social. 65% of people use Facebook to share posts with at least 1 image. More than 20% of Twitter users prefer to publish content with an image.

Why a content became viral?

- A data analysis company sought to understand what are the characteristics that make a viral content:
  - **Bulleted lists**: web users love bulleted lists, infographics, and how-to, because they allow to summarize salient aspects in visual form, making them easy to understand.
  - **Influencer**: content shared by people considered "experts" reach a greater number of users, "targeted" and interested.

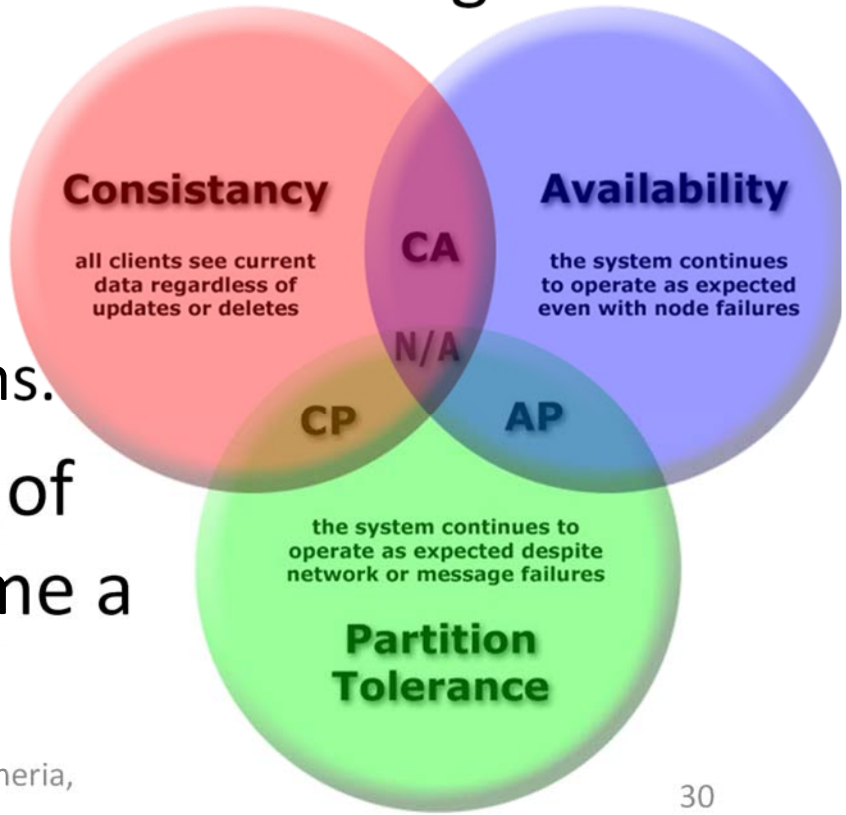
CAP PRINCIPLE

Index

- What is Big Data
- 5V of Big Data
- **CAP Principle**
- Big Data Application Fields
- Big Data Problems, Criticality and Risk
- NoSQL
- Big Data Analysis Pipeline
- Big Data Solutions

CAP theorem

- The **CAP theorem** (Consistency - Availability - Partition tolerance) is essential to **understand** the **behavior** of **distributed SW systems**, and **how** to **design** the **architecture** in order to meet stringent requirements, such as:
  - High **performance**.
  - Continued **availability**.
  - **Geographically distributed** systems.
- Working on billions and trillions of data every day, **scalability** became a key concept.



CAP theorem

It is **highly desirable** for a distributed SW system **simultaneously** provide:

- **Consistency**
- Continued **Availability**
- **Partitions Tolerance**

…but this is **not possible**!!

You can satisfy **at most 2** out of this 3 **requirements**, so it is necessary to determine in each case which of these three characteristics sacrifice.

Consistency

- A distributed system is **fully consistent** if a **data** written on a **Node A** is **equal** to the value read from another **Node B**.
- The system will return the last value written (consistent).
- Example - **Cache Memory**:
  - In a **single node** the total **consistency is guaranteed**, as well as the tolerance to the partitions. There is not enough availability (fault tolerance) and good performance.
  - If the cache is **distributed** on two or more nodes, the **availability increases**, but **complex mechanisms** must be provided, which allow each node to access a **virtual** distributed **repository** (to read the same value).

Availability

- A distributed system is **always available** if **each** working **node** is **always** able to **respond** to a query or provide its services.

- Example – **Cache Memory**:
  - A cache on a **single node** does **not guarantee** continuous **availability**.
  - A **distributed cache** keeps, on various nodes, some areas to store backup data of other nodes.
  - In order to realize the **continuous availability** , **data redundancy** is required (multiple nodes). This requires mechanisms to ensure the consistency and to avoid problems regarding partitions tolerance.

Partitions Tolerance

- It is the ability of a system to be **tolerant to add/ remove** a **node** in a distributed system (partitioning) or to the loss of messages on the network.

- Example - **clusters** formed by nodes on **two different data centers**:
  - If data center **lose** their **network connectivity**, nodes in the cluster can **no** longer **synchronize** the system state.
  - **Nodes** of the **same data center**, reorganize themselves into sub-clusters, **cutting off** node of the **other data center**.
  - The system will continue to operate in an uncoordinated manner, with **possible data loss**.

Consistency/Availability (CA)

- By designing a distributed system, you must consider as a compromise solution accept: **CA**, **CP** and **AP**.
- It is the compromise offered by **RDBMS**.
- **Data** is **consistent** on all nodes (active and available).
- Writes/reads are always possible, updated data are propagated across the cluster nodes.
- Possible issues:
  - ☹ performance and scalability
  - ☹ misalignment between the data in the case of partitions of nodes.

Consistency/Partition-Tolerance (CP)

- **Preferred** compromise **solutions** by HBase, MongoDB, BigTable.
- **Data** is **consistent** on all nodes, **partitions** are **guaranteed**, ensuring data synchronization.

- Possible issues:
  - ☹ **Availability**: data are no longer available if a node goes down.

Availability/Partition-Tolerance (AP)

- Compromise solutions used by **CouchDB**, **Riak**, Apache **Cassandra**.
- Nodes remain online even if unable to talk to each other.
- It **requires** a process of **data re-synchronization** to eliminate any conflicts when the partition is resolved.
- The system is still available under partitioning, but some of the **data returned** may be **inaccurate**.

☺ **Good performance** in terms of latency and scalability.

Which to choose?

- Most of the existing solutions provide **operating mode tuning**, that is they leave to the **developer** the ability to **choose** which guarantee sacrifice.

BIG DATA APPLICATION FIELDS

Index

- What is Big Data
- 5V of Big Data
- CAP Principle
- **Big Data Application Fields**
- Big Data Problems, Criticality and Risk
- NoSQL
- Big Data Analysis Pipeline
- Big Data Solutions

Application Fields

Increasing investments in Big Data can lead to interesting discoveries in **science**, **medicine**, benefits and gains in the **ICT sector** and in **business** contexts, new services and opportunities for digital **citizens** and **web users**.

- Healthcare and Medicine
- Data Analysis – Scientific Research
- Educational
- Energy and Transportation
- Social Network – Internet Service – Web Data
- Financial/Business
- Security

Healthcare and Medicine

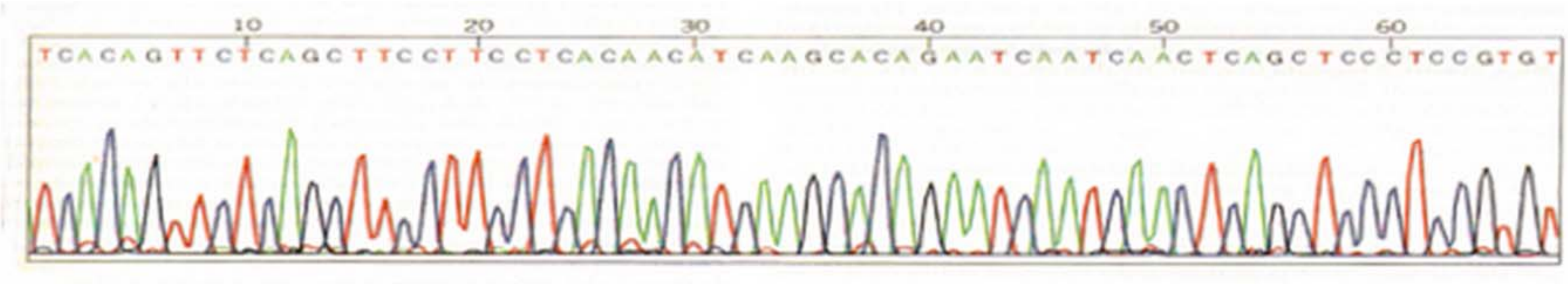
- Large amount of information is collected about :
  – Electronic Patient Record (EPR)
  – Symptomatology
  – Diagnoses
  – Therapies
  – Responses to treatments
- In just **12 days** approximately **5000 patients** entered the emergency department.
- In Medical Research two main application are:
  – **Genomic Sequence Collections** (A single sequencing experiment yield 100 million short sequences)
  – **Analysis of neuroimging data** (Intermidiate data stored ~1.8PetaBytes)

Healthcare and Medicine

- **Data mining techniques:** to **derive knowledge** from data (to identify new pattern in infection control data, to examine reporting practices).
- Hospitals with **Electronic Patient Record** (EPR) have investigated techniques to **fast access** and **extraction of information from event's log**, to produce interpretable models, using partitioning, clustering and preprocessing techniques.
- By **building a predictive model**, it could be possible to provide **decision support** for specific triage and diagnosis or to produce **effective plans** for chronic disease management, **enhancing** the **quality** of healthcare and **lower costs**.

Healthcare and Medicine

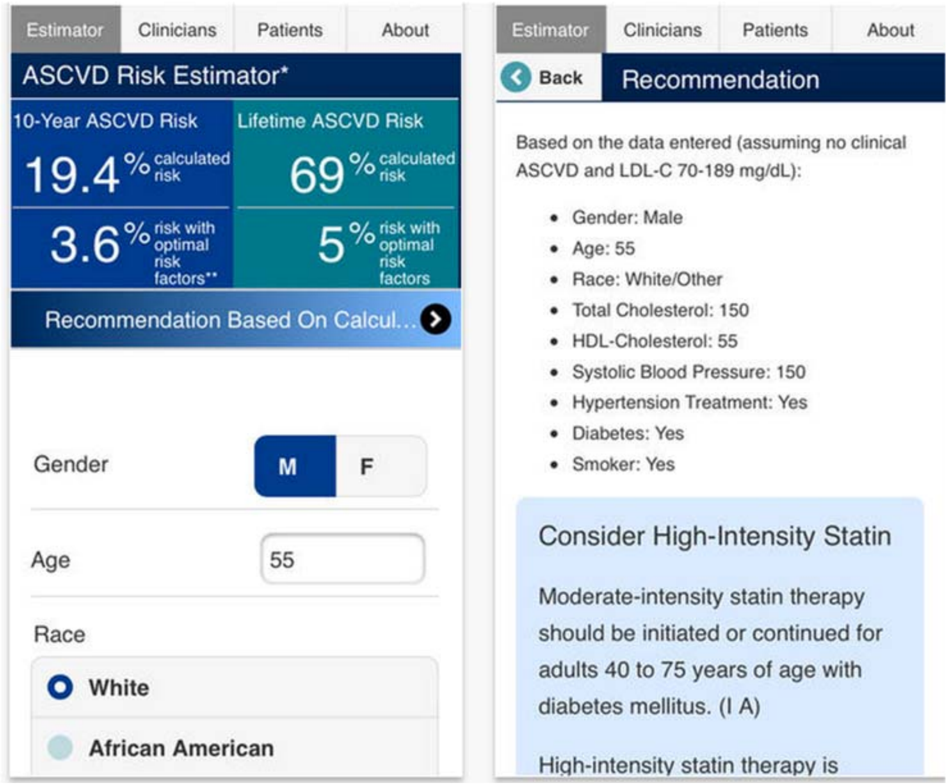
- Techniques of **gene cloning** and **sequencing of DNA** to **know the entire genome** of organisms.



- **Knowledge of the entire genome**:
  - To **identify** the genes involved.
  - To observe **how these interact**, in the case of complex diseases such as **tumors**.

Healthcare and Medicine

- **Ascvd Risk Estimator** is an App launched by **America College of Cardiology** and **The American Heart Association**.
- **Constantly monitor** the risk of heart attack and cardiovascular problems for **ten years**.
- **Collects** patient **information** (age, sex, race, cholesterol, blood pressure, hypertension) .
- Doctors, **analyze data**, estimate the possibility of **risk** and communicate to patients the **care** and **treatment** to follow, according to an **evolutionary process constantly updated**.

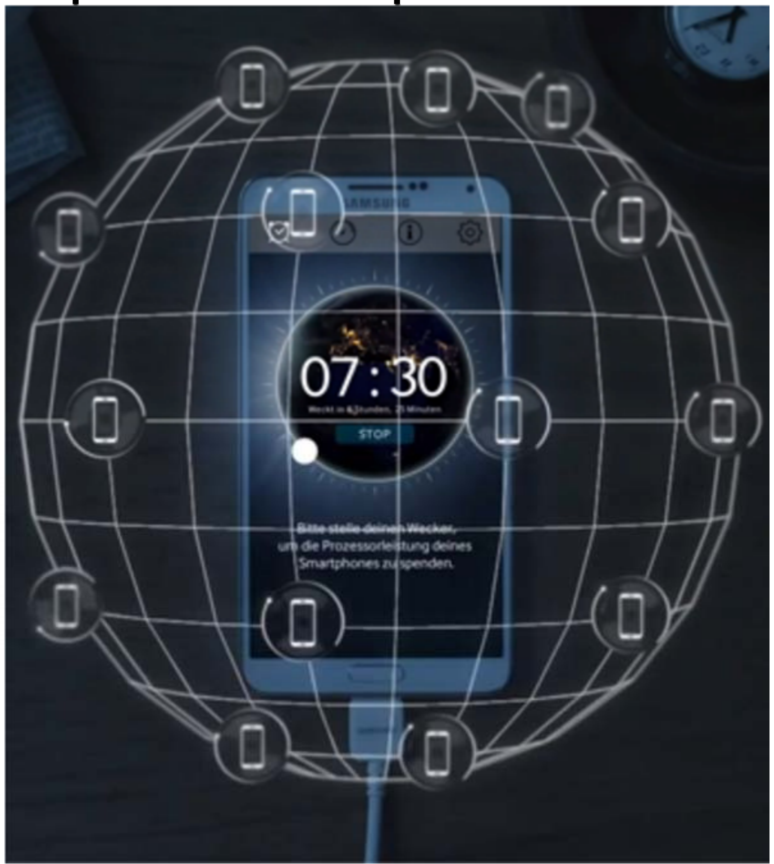


Data Analysis – Scientific Research

- **Big Data analysis** to **extract meaning** from data and **determine** what **actions** take:
  - **Astronomy** (Automated sky survey): **200 GB** of new high resolution optical data are captured **every day** by charge-coupled devices (CCDs) attached to telescopes.
  - **Biology** (Sequencing and encoding genes)
  - **Sociology** (Web log analysis of behavioral data): up to **15 million people** world wide accessing the internet **each day** (10 hours per week on line).
  - **Neuroscience** (genetic and neuro-imaging data analysis)
- Scientific research is **highly collaborative** and involves scientists from different disciplines and from different countries.

Data Analysis – Scientific Research

- **Samsung Power Sleep,** Android App (Samsung Austria and Universität Wien): users set the alarm time on the app and put the phone on charge with Wi-Fi enabled.
- Power Sleep **processes data** and sends it to a database, **Similarity Matrix of Protein** (SIMAP): sequences of proteins are decoded, useful for various scientific research, including **genetics**, **biochemistry**, **cancer** and **Alzheimer's**.



- App connected to the **Berkeley Open Infrastructure Network Computing** (BOINC), which aims to take advantage of pc and mobile device for **processing scientific data**.
- **HTC** Power To Give.

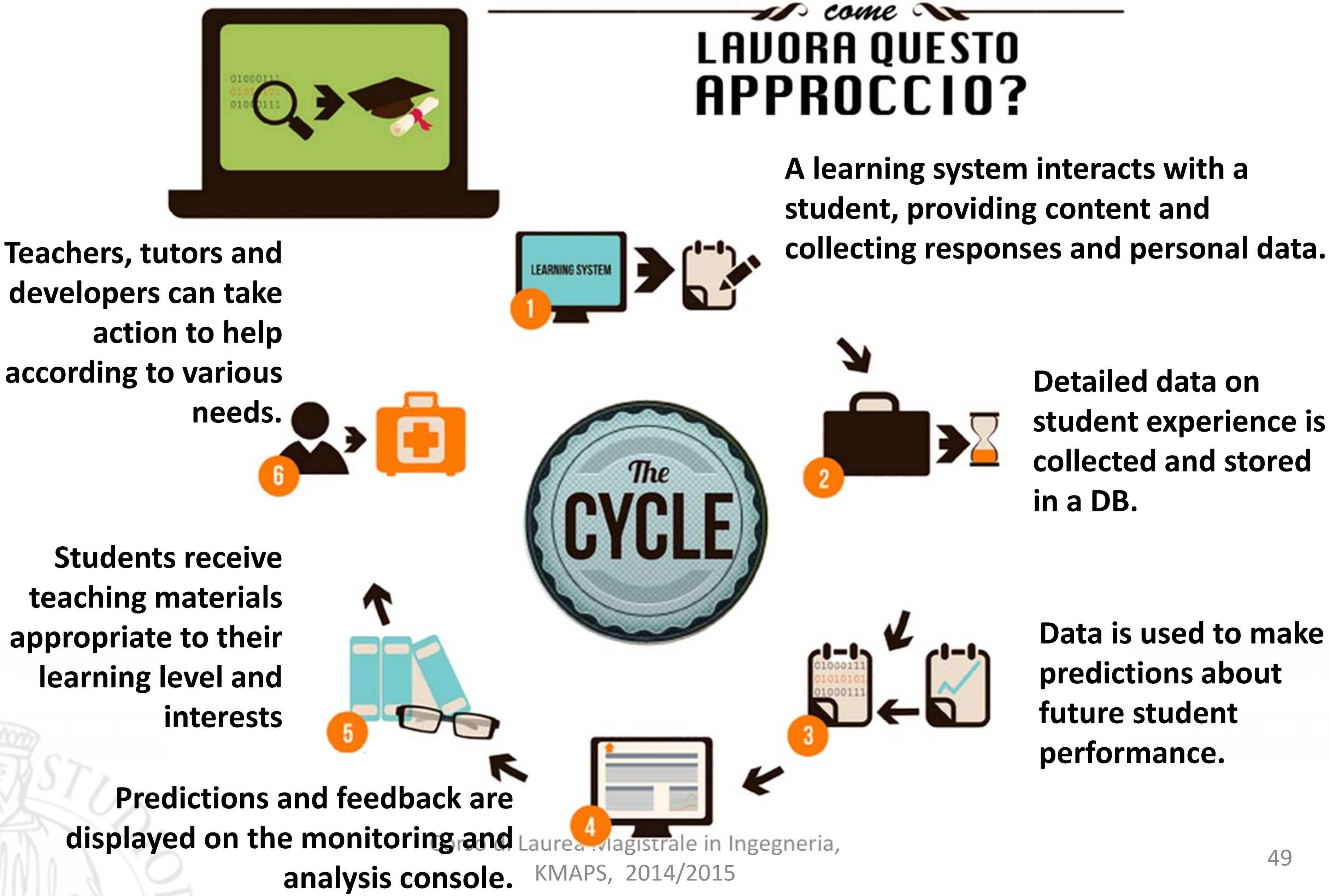
Educational

- Big Data to **revolutionize education**.
- Main educational **data**:
  – students' performance (Project KDD 2010 *)
  – learning mechanism
  – answers to different pedagogical strategies

A **new approach of teaching** can be defined by exploiting the Big Data management!

- Uses data to define models to understand:
  – what **students** actually **know**
  – their **progresses**
  – how to **enrich** this **knowledge**

Educational



Energy and Transportation

- **GPS** information (Buses, taxis, information point, IP,).
- **Traffic** Interruptions (temporary information).
- **Weather** Forecasts.
- **Parking sensors** and sharing mobility services (**RFID**).
- Opening/closing **time** of activities and services.
- **Video** from **security** cameras.

- A **data-centric approach** can also help for **enhancing efficiency** and **dependability** of a transportation system.

- The **optimization** of **multimodal transportation** infrastructure and their **intelligent use** can **improve** travelers **experience** and operational **efficiencies**.

Energy and Transportation

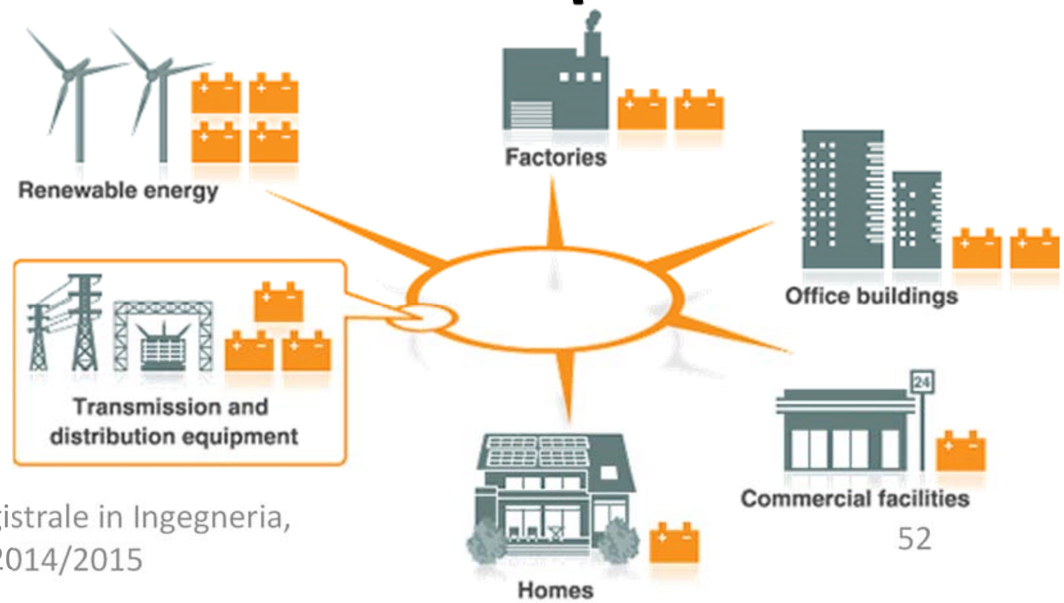
- Merging high-fidelity **geographical data** and **real-time sensor** networks scattered **data → efficient urban planning** system that mix public and private transportation, offering people more flexible solutions (**Smart Mobility**).

- **Data** related to **energy consumption**: help in energy **resources optimization** and environmental monitoring (**electricity**, **gas**, **water**, **CO2 emissions**).

- **Analysis** of **load profiles** and **geo-referenced** information with **data mining** techniques → construction of **predictive models** to define **intelligent** distribution **strategies** (**lower costs** and **improve quality** of **life**).

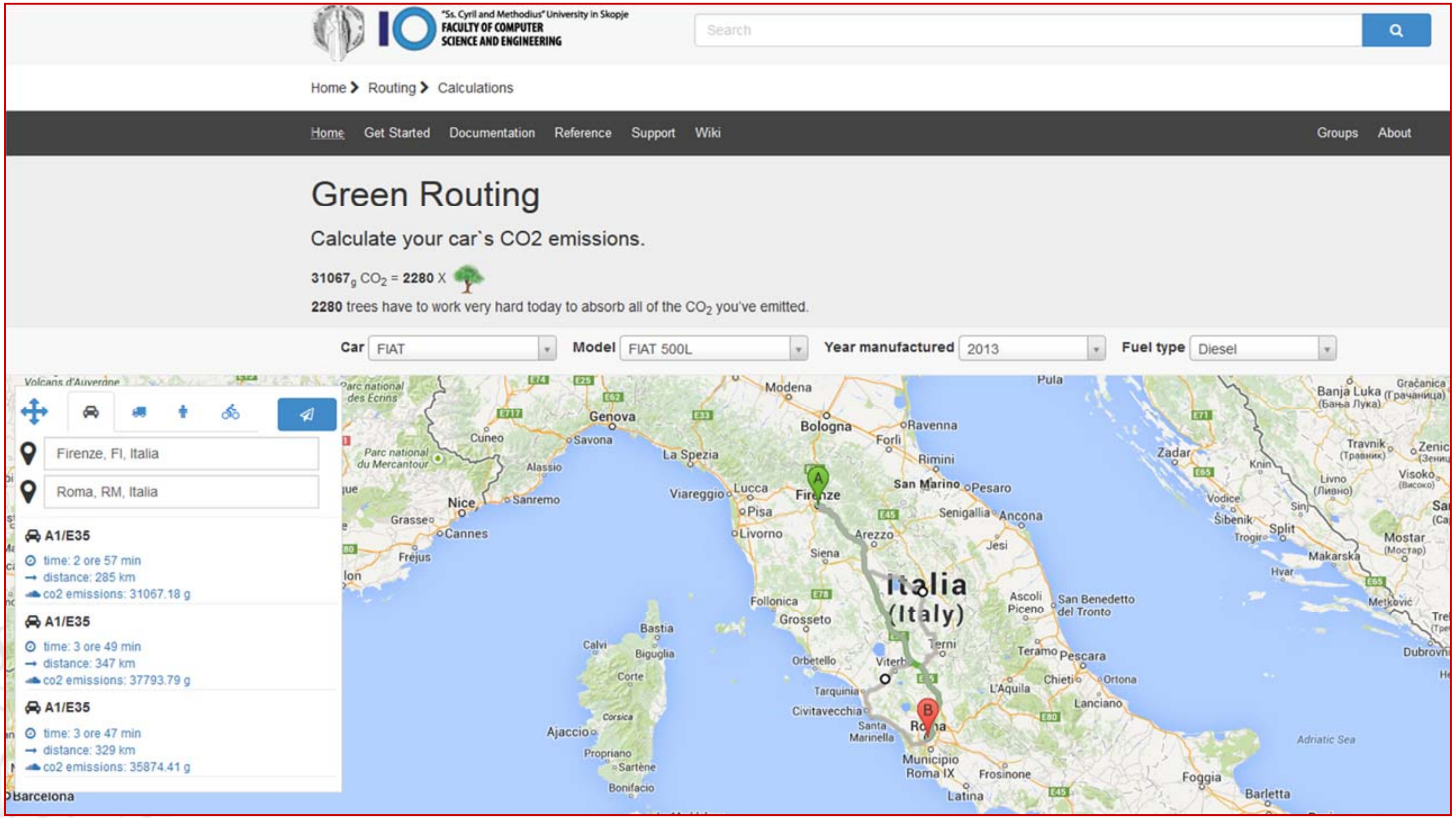
Energy and Transportation

- Instrumenting a home with three sensors:
  - **Electricity**, **Gas** and **Water**, to determine individual resource usage.
- Transform **homes** and residential areas into **Sensor Networks.**
- **Smart Grid**: integrating information about **personal usage** patterns with **energy availability** and **energy consumption** information, to realize an **evidence-based power management**.



Energy and Transportation

Green Routing (University of Skopje): project using Big Data and Google Maps to determine the CO2 emissions of a vehicle during a journey.



Social Network, Internet Service, Web Data

2009:

Facebook: 3 billion photos uploaded per month; "like" button was implemented.

Youtube: all users upload 24h of video per minute;

Twitter: 50 million tweets per day.

Instagram was created in 2010.

2012:

Facebook: more than 10 million photos uploaded per hour, 3billions "like" button/comment per day;

Youtube: 800 million users upload ~1h of video per second;

Twitter: more than 400 million tweets per day.

Instagram: 7.3 million unique users per day

2013:

Facebook: more than 14 million photo uploaded per hour, 4.5 billion Likes per day;

Google Youtube: More than 1 billion unique users visit per month, 100h of video are uploaded every minute;

Twitter: more than 500 million tweets per day.

Instagram: more than 150 million monthly active users

Social Network, Internet Service, Web Data

- **Volume** of data generated by internet services, websites, mobile applications and social network → **high.**
- **Speed** of production → **variable**.

- From **data** collected through **social networks**, researchers try to predict the **collective behavior**, or **trends**. **E.g.** by monitoring the Twitter **hashtag** (#) is possible to identify **patterns of influence**.

- From all this information is possible to **extract** knowledge and **data relationships**, by **improving** the activity of **query answering**.

Social Network, Internet Service, Web Data

- The University of Cambridge has published a study in 2013 which shows how describe personality from the analysis of Facebook Like.
- Attributes analyzed:
  - policy orientation
  - Sexual Orientation
  - Religious Orientation
  - character traits
  - Level of life satisfaction
- The model can be applied to any set of data capable of expressing a user preference.
- Facebook data are public and using Facebook Connect, are easily obtainable (after user authorization).

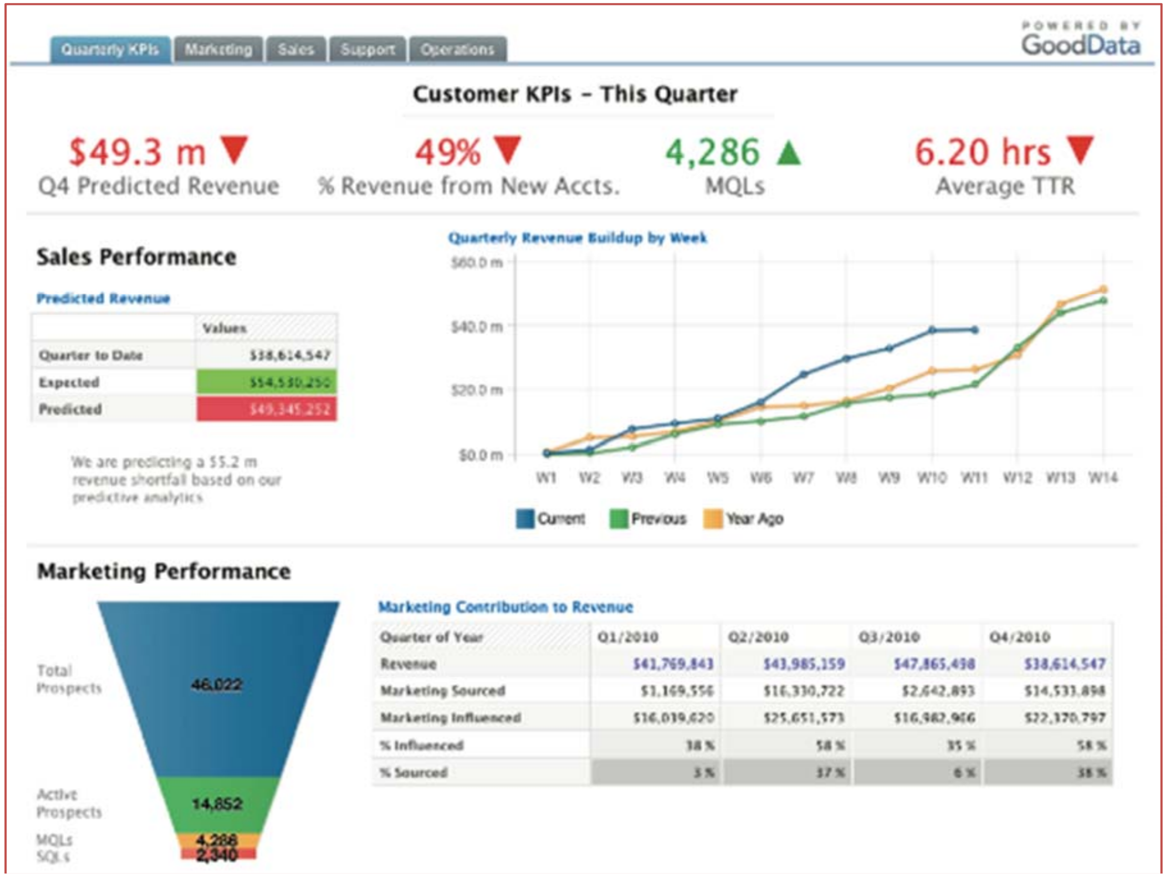
Financial/Business

- Traditionally **business analysts** use **statistical** techniques.
- Today: large electronic repositories store **numerous business transaction**.
- **Data magnitude**: ~ 50-200 PBs at day.
- **Europe Internet** audience in is about **566,3 million** unique visitors (**68.6%** population)
- **40%** of European citizens **buy online**.

- Analyze these data **to obtain**:
  - **Prediction** about the behavior of users
  - **Buying pattern** of individual/group customers
  - New **custom services** to provide.

Financial/Business

- **GoodData** (San Francisco) provides a platform with a set of BI tools, to help companies to **analyze** their **huge** amount of **data** (surveys, account sales, costs, etc.) to facilitate the **decision making** process.



Security

- **Data sources** for intelligence services:
  - Public domain sources (**web sites**, **blogs**, **tweets**, and other Internet data, print media, **television**, and **radio**).
  - **Sensor data (**meteorological, oceanographic, security camera feeds).
  - **Biometric data** (facial images, DNA, iris, fingerprint, gait recordings).
  - **Structured** and **semi-structured** information supplied by companies and organizations: airline flight logs, credit card and bank transactions, phone call records, employee personnel records, electronic health records, police and investigative records.

Security

- **Data sources** for intelligence services:
  - **Satellite** and **UAV**s Image.
  - **Wiretaps**: civilian and military, including voice, email, documents, transaction logs, and other electronic data - 5 billion mobile phones in use worldwide.
  - **Radar** tracking data.
- **1zettabyte** (10^21 bytes or 1billion terabytes) of digital data are generated each year.
- Need to sophisticated **methods** to **identify** accurate **models**, without generating a large number of false positives in a way that does **not reveal conspiracies** or alarms **where none exist**.

Security

- Philadelphia (December 2012) has released a dataset with the list of crimes from January 1, 2006.
- Every crime (theft, robbery, murder) is tagged in the exact position in which it was committed.
- With these data it is possible to create tools and useful statistics to both the citizen to the government.

BIG DATA PROBLEMS, CRITICALITY AND RISK

Index

- What is Big Data
- 5V of Big Data
- CAP Principle
- Big Data Application Fields
- **Big Data Problems, Criticality and Risk**
- NoSQL
- Big Data Analysis Pipeline
- Big Data Solutions

Big Data’s problems

- How is it possible to discover their “**value**”?
- The ecosystem of the data is **highly fragmented**:
  - The **large number** of **application areas**, so different from each other
  - The **different channels** through which data are daily collected.
- Analysis tools should serve as a new "**refinery**", **addressing** part of this **fragmentation** by **increasing connectivity**, **reliability** and **efficiency**.

Big Data's problems

Architectures to collect, refine and analyze all this huge amount of data collected, **are inefficient**:

There is need for **innovation**!

Other critical aspects that should be consider:

- **Issues** related to the **data quality** and **reliability**.
- **Issues** related to **privacy** and **data ownership**.

Data quality and reliability

Data quality is determined by :

- **Completeness**: presence of **all** information needed to describe an object, entity or event (eg. Identifyng).
- **Consistency**: data must not be **contradictory**. For example, the total balance and movements.
- **Accuracy**: data must be **correct**, i.e. conform to actual values. For example, an email address must not only be well-formed *nome@dominio.it*, but it must also be valid and working.

Data quality and reliability

- **Absence of duplication**: tables, records, fields should be stored **only once**, avoiding the presence of copies. Duplicate information involve double handling and can lead to **problems** of **synchronization** (consistency).

- **Integrity** is a concept related to **relational databases**, where there are tools to implement integrity constraints. Example a control on the types of data (contained in a column), or on combinations of identifiers (to prevent the presence of two equal rows).

Data quality and reliability

- The overall **data quality** can be **undermined** by:
  - Errors in **data entry** operations (fields and missing information, incorrect or malformed).
  - Errors in **data management** software (query and incorrect procedures).
  - Errors in the **design** of databases (conceptual and logical errors).

Data quality and reliability

- In the world of Big Data instead:
  - **Operational data**: the quality problems are known and there are several tools for **automatic** data **cleansing**.
  - **Data automatically generated**: scientific data and data from sensors, have **no entry errors**, but they are "**weak**" in terms of information content: there is the need to **integrate data** from other systems and then **analyze** them.
  - **Data on the Web**: Social networks, forums, blogs generate **semi-structured data**. The most reliable are the **metadata** (if present) shall instead be subject to errors, abbreviations, etc.

Data quality and reliability

- **Disambiguate information**: the **same data** can have **different meanings**. The challenge is trying to find the most relevant to the present context. Help are tags, tagging the data you are trying to highlight the scope of relevance.
- **Truth**: News, statements, documents do **not always** correspond to **reality** or real.
- The quality of the data, however, is also **linked** to the **context** in which they are analyzed. Transactions of filtering and **cleaning** must be done **by degrees** to **avoid removing** potentially useful **data**.

Privacy and data ownership

- The Big Data problems are connected to **privacy**, ownership and **use** of **data** by third parties.
- **Data of the Web**: the user-generated-content are shared for all. It is **ethical** usage?
- **Sensitive data**: the data in the DB hospitals regarding the **medical history** of the patients, are properly protected?
- **Location Data**: use of smartphones, GPS, electronic payment systems, but also social networks **leave traces** of where you can get the movements of the user.

NOSQL

Index

- What is Big Data
- 5V of Big Data
- CAP Principle
- Big Data Application Fields
- Big Data Problems, Criticality and Risk
- **NoSQL**
- Big Data Analysis Pipeline
- Big Data Solutions

NoSQL: definition

- "**Not Only SQL**": subset of structured storage software, designed for **increased optimization** for **high-performance** operations on **large dataset**.

Why NoSQL?

- **ACID** (**A**tomicity, **C**onsistency, **I**solation, **D**urability) doesn't scale well.
- Web apps have **different needs**: High **Availability**, Low Cost **Scalability** & **Elasticity**, Low **Latency**, **Flexible Schemas**, Geographic **Distributions**.
- Next Generation Databases mostly addressing some of this needs being non-relational, distributed, open-source and horizontally scalable.

NoSQL: PROs & CONs

PROs:

- Schema-free;
- High Availability;
- Scalability;
- Easy replication support;
- Simple API;
- Eventually consistent/BASE (not ACID);

CONs:

- Limited query capabilities;
- Hard to move data out from one NoSQL to some other system (but 50% are JSON-oriented);
- No standard way to access a NoSQL data store.

NoSQL: BASE Approach

- **B**asically **A**vailable, **S**oft state, **E**ventual consistency approach;
- To give up **consistency** to provide greater **scalability** and **availability**.
- **Basically Available**: the system must ensure the **availability** of information.
- **Soft State**: the system can **change** its **status** in time, even if **no writings and readings** happenend.
- **Eventual consistency**: The system can become **consistent** over time (even without writings) thanks to the **consistency recovery systems**.

- According to this approach **inconsistencies are temporary**, i.e. each DB node in the cluster, at the end, get the latest major changes to the data.

NoSQL: BASE Approach

- **BASE** model has **3 operation modes:**
  - **Casual Consistency**: At any modification, application notifies the other sessions that they will **see** the **updated data** from that moment.
  - **Read your own writes**: The session that performs data modification, will see **changes immediately**, while other sessions will see them with a **slight delay**.
  - **Monotonic Consistency**: A session will **never** see data of a **previous version** than the one **read**. Data will always of the readed version or of a **more recent version** (based on **Vector Clock**).

NoSQL: Vector Clock

- The goal is to **apply changes** to the data in the **correct sequence**.
- Each cluster node maintains a sequential number that identifies the changes (**change number, CN**).
- The **Vector Clock** is a list of **CN**, from all nodes.
- At each change, the **new vector** is sent to **all nodes** with the update.
- Each node analyzes the vector received and compares it with the previous one, to determine if the **update** is the **next in sequence**.
- Otherwise, the update is not applied immediately, but is stored, **waiting** for **previous changes**.

NoSQL: Main Features

Key features of NoSQL DB:

- **Multi-Node Environment**: usually a model is created, consisting of a set of **multiple distributed nodes** (cluster), which can be added or removed.
- **Data Sharding**: using appropriate algorithms data is divided and **distributed to multiple nodes**, retrieving them when needed (e.g. Gossip Algorithm).
- **Replica of Information**: data distributed on the various nodes, is often copied **several times** to ensure the **availability** of information.

NoSQL: Main Features

NoSQL DB are subject of studies to improve:

- **Performance**: **new algorithms** are designed and implemented to **increase** the overall **performance** of NoSQL systems.
- **Horizontal scalability**: it is important to be able to **increase/decrease** the size of a cluster by **adding/removing** nodes "**invisibly**"; that is, the entire system must not stop when happening.
- **Single Machine Performance**: it is important to be able to **increase** the **performance** of **each machine**, because they are responsible for **finding information** in the cluster to **external applications**.

Types of NoSQL db

- Key-Value DB
- Col-Family/Big Table DB
- Document DB
- XML DB
- Object DB
- Multivalue DB
- ACID NoSQL

Cassandra riak Neo4j the graph database HBASE redis mongoDB hadoop cloudera HIVE Hortonworks MAPR TECHNOLOGIES GREENPLUM HD DataStax HADAPT amazon web services

Key-Value Database

- High scalable database, not suitable for large data sets. Allows to obtain **good speed**.
- It is a kind of big **hash table**, used for **large lists of elements**, such as stock quotes, or shopping carts.
- The design of **key** is **crucial**, because the **storage** is based on direct addressing.
- Research is done through key.

Key-Value DB

- Some Key-Value DB allow the definition of **secondary indexes** (e.g. B + tree) **for keys →** performance **decrease**, **problems** in clustering managing.
- **Data Model**: collection of kay-value pairs
- Based on Amazon's Dynamo Paper
- **Example**: Dynomite, Voldemort, Tokyo

Column Family/Big Table Database

- Key-value stores can become column-family database (**grouping columns**);
- Very useful with **time series** or with data coming from multiple sources, **sensors**, **devices** and website, with **high speed**;
- Data is organized more like a hash table, but using **two or more levels of indexing**;
- They require **good** performance in **reading** and **writing** operations;
- The table stores **a column at time** and eventually proceeds with the storage of the next column.
- In addition to the column, even a **surrogate key** is stored, necessary to **reconstruct** the **record**.

Column Family/Big Table Database

Id	Name	Last Name	Points
1	Joe	Smith	40000
2	Mary	Jones	50000
3	Cathy	Johnson	44000

Id	Name	Last Name	Points
1	Joe	Smith	40000
2	Mary	Jones	50000
3	Cathy	Johnson	44000

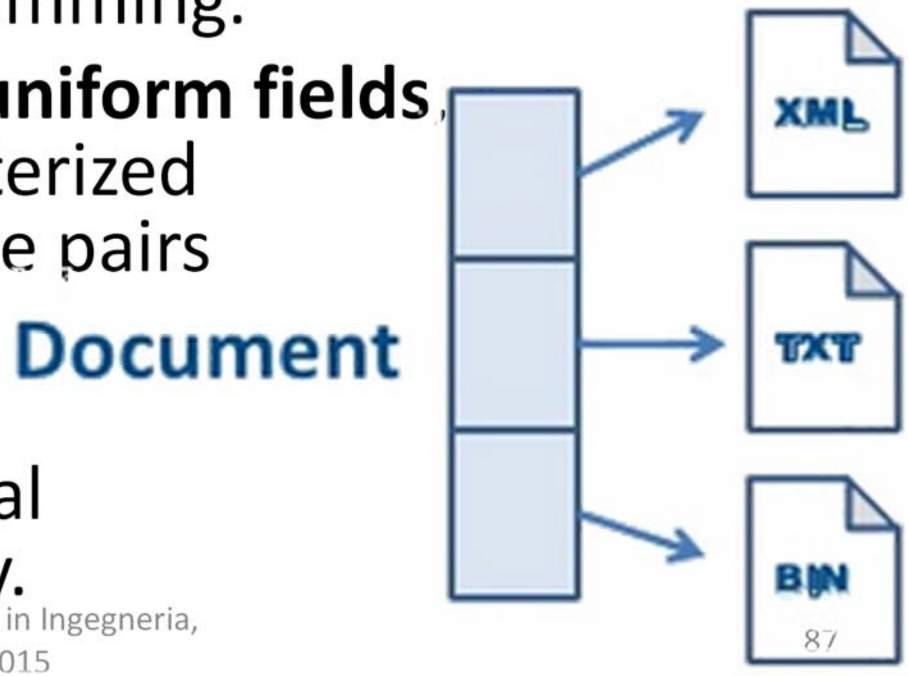
- **Columns** that make up the column-family, should **not be defined in advance**.
- **Each record** can have **different information** in a column-family.
- **Empty columns**, for which there is no value, will **not be included** in the column-family.
- Considerable **gain** in terms of **memory**, especially on **large amounts of data**.

Column Family/Big Table Database

- **Adding** or **deleting** a **column** does **not imply** a **redefinition** of the scheme
- They are **not suitable** for datasets where **data** has the **same importance** of **relationship**.
- Suitable for **data warehousing** and **read/only reporting systems**, mainly **OLAP** (On Line Analytical Processing) → suitable for **interactive** and **fast analysis** on **large** amounts of **data**, using techniques software.
- **Based on** *Google BigTable Paper.*
- **Example: Hbase**, Hypertable, Cassandra.

Document Database

- Designed for storing, retrieving, and managing semi-structured data.
- "**Documents**" are collections of **key/value pairs** organized in **JSON** or **XML** format (self-describing formats). They are a kind of sophistication of **key/value DB**.
- Fit perfectly to the OO programming.
- Data is stored in **tables** with **uniform fields**, but each document is characterized by **specific features** (key/value pairs simply structured).
- Useful when data are hardly representable with a relational model due to high complexity.



Document Database

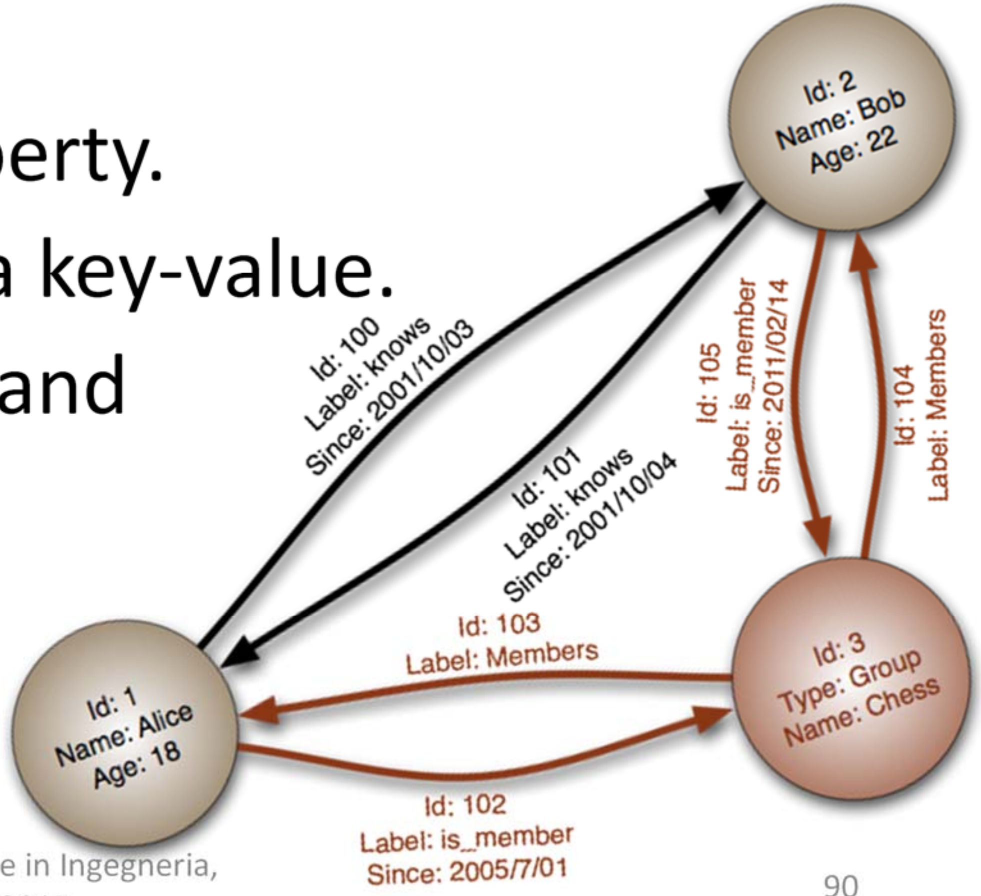
- Used with medical records or with data coming from social networks.
- Used to respond to the **variety of Big Data**. Data has a dynamic structure, or structures very different from each other, or have a large number of optional data.
- Used encoding: XML, YAML, JSON, e BSON;
- Each document is identified by a **unique key** (string, URI or path). There is a **key index** to speed up searches.
- Often there are **APIs** or a specific **query language**, to retrieval information based on content, i.e. value of a specific field.
- **Example**: CouchDB, MongoDB.

Graph Database

- Born from the need to manage **strongly-connected data** to each other (non-tabular), in which the concept of **relationship** between data has a **high information potential**.
- They take up **less space** than the **volume of data** with which they are **made**, and **store** a **lot** of **information** on **relationship** between data.
- Used in field like **geospatial**, **bioinformatics**, **network analysis** and recommendation engines.
- Inspired by **Graph's Theory**;
- Not easy execute query on this database type;

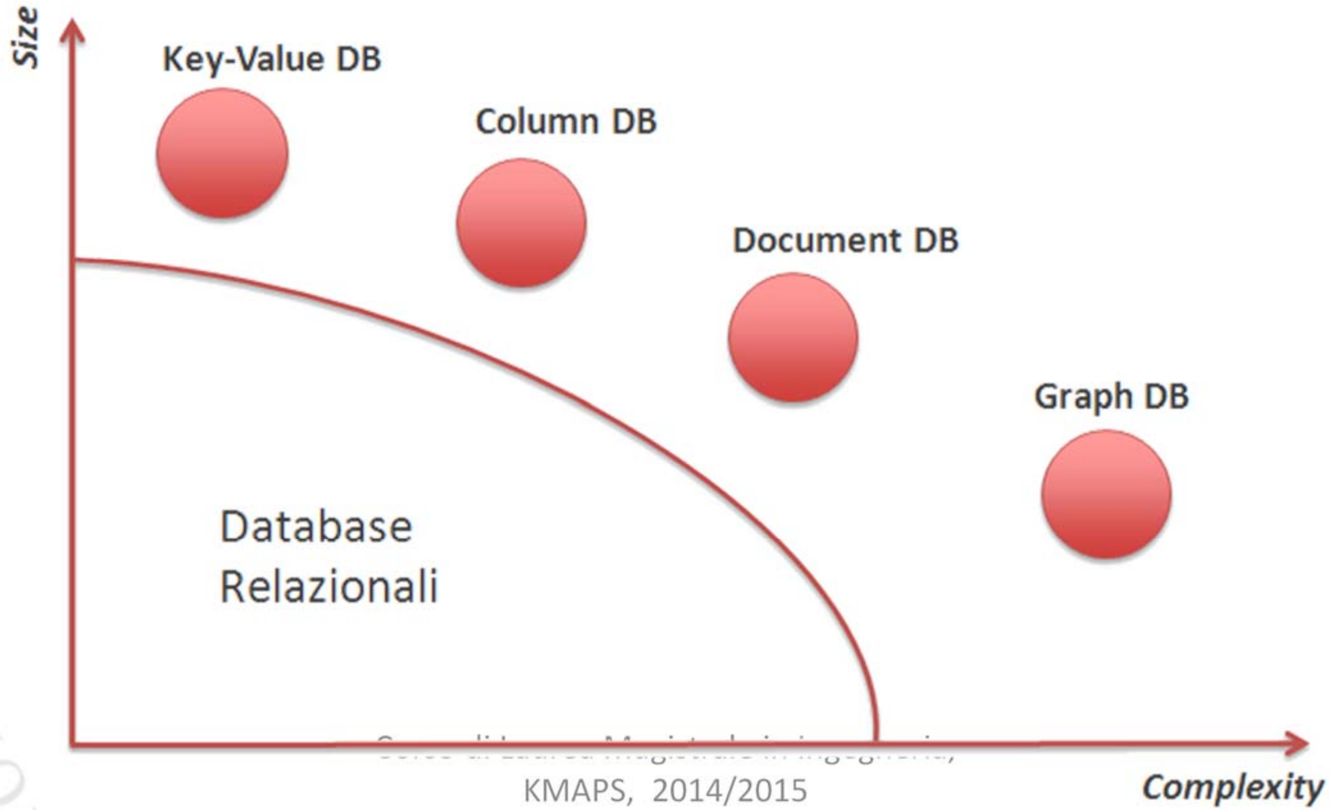
Graph Database

- The most used model to implement graph database, is based on "**property graph**".
- Suitable to manage ad hoc and changing data with evolving schemas;
- **Nodes**: containers of property.
- **Properties**: expressed as a key-value.
- **Relationships**: link nodes and sometimes they contain properties.
- **Examples**: AllegroGraph, Objectivity.



NoSQL Comparison

- **Evaluation** of NoSQL DB solutions, based on size and operational **complexity**.
- The **increased complexity** implies a **decrease** in storage capacity (size**).**



Object Database

- Created with the aim to **model complex data** (e.g. cartographic program for the management of maps) and **interconnected** in **scientific-technological** fields
- Allow the creation of **very reliable** storage system.
- It should be a **DBMS**, and it should be an **object-oriented** system;
- They directly **integrate** the **object model** of **programming language** in which they were written (not much popularity);
- They **cannot provide** a **persistent access** to data, due to the **strong binding** with specific platforms;

Object Database

- The defined **data structures** can be **saved** directly in the database **without suffering** any change-adaptation.
- The **data model** is **stored** in mass **memory** and the **DBMS** assigns to each object **a unique identifier** (**Oid**), as long as the object will be stored (**object lifecycle**).
- **Oid** is used to **retrieve** an **object**, its **properties** and for managing **relationships** with other objects.
- **PROs**: persistence, secondary storage management, concurrency, recovery and an ad hoc query facility;
- **CONs**: **lack** of **standardization** and interoperability between **different OODBMS**.
- **Example**: Objectivity, Versant.

XML Database

- **XML Database** have emerged as a solution to **management data hardly** represented in **tabular** format
- They are usually **associated** with **document-oriented** databases;
- **XML format** is widely used for **data exchange**; the **XML view** is **preferred** by users and applications.
- Large **XML document** optimized to contain **large** quantities of **semi-structured data**;
- **XML documents** are well suited to contain **hierarchical data structures**.

XML Database

- The data model is **flexible**;
- **Queries** are **simple** to perform, but **not** very **efficient**;
- XML-enabled (map XML to traditional database), Native XML (uses XML documents as the fundamental unit of storage);
- **Example**: eXist, BaseX.

XML Database

The **XML-DB** organize data in **XML documents** that can be:

- Easily **queried** via **XPath** queries.
- **Transformed** by **XSLT** in tabular output.

- **XPATH**: Language that allows to **extract** and **manipulate** XML **nodes** or **values**. Allows the **creation** of **indexes** and **queries** on **XML data** type, managed **in** relational **DBMS**.

- **XSLT** (e**X**tensible **S**tylesheet **L**anguage **T**ransformation): language that allows to transform XML documents in
  - Other XML documents, but with a different structure.
  - In HTML or text documents (TXT, CSV ...)

Multivalue Database

- Synonymous with **Pick Operating System**;
- Support use of **attributes** that can be a **list of value**, rather than a single value (RDBMS);
- The **data model** is well suited to **XML**;
- They are classified as **NoSQL but** is possible to **access** data both **with or without SQL**;
- They have **not been standardized**, but simply classified in pre-relational, post-relational, relational and embedded.
- **Example**: OpenQM, Jbase.

Multivalue Database

- **Database** = “Account”, **Table** = “File”, **Column** = “Attribute” (row) or “Dictionary“ (trasformed row).
- In the “person” **File** there is a **Dictionary** called “emailAddress” where we can store a variable number of email address values in the single record.
- Data is stored using **2**separate **files**:
  - The first file is used to store **data raw**.
  - The second, called "**Dictionary**" is used to store the **display format** of data.

ACID NoSQL

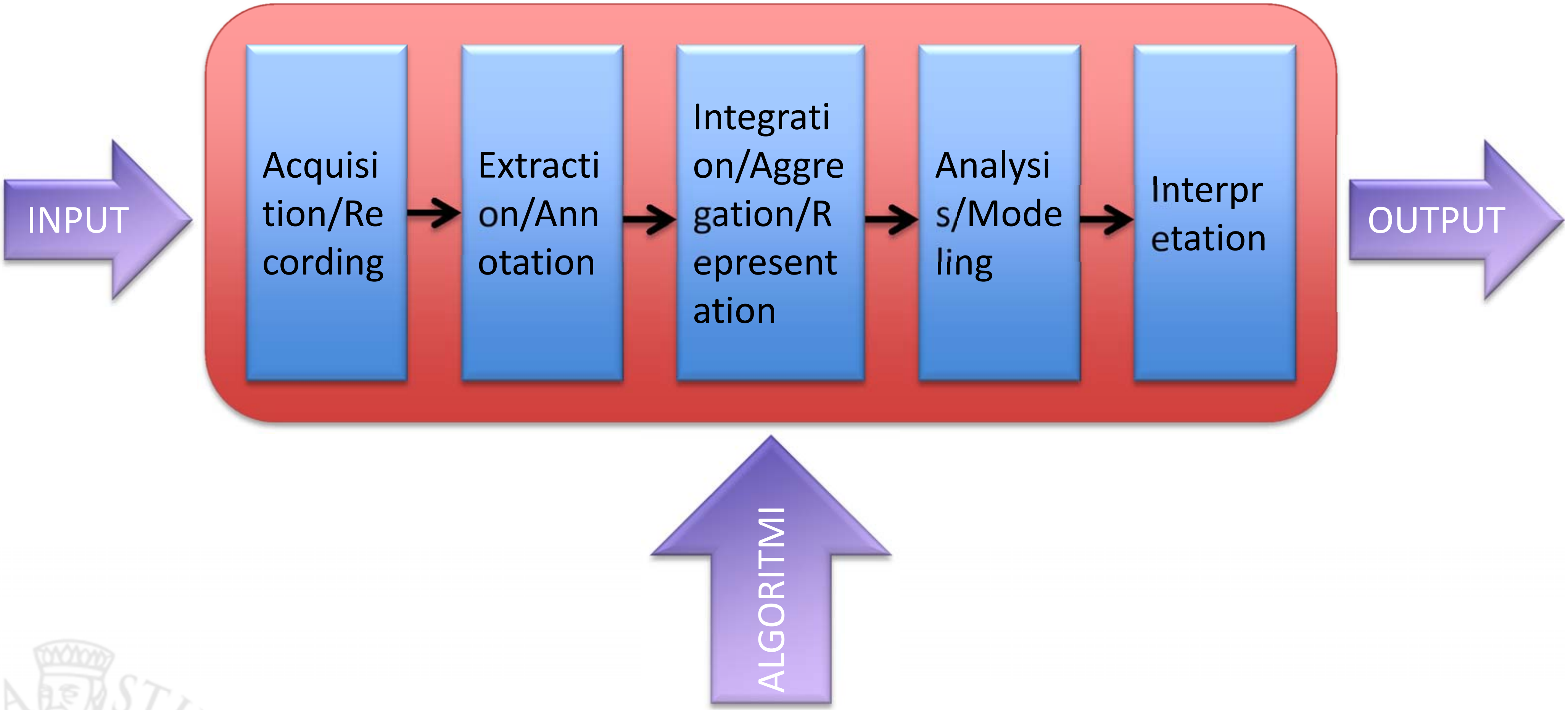
- Try to **combine** the more **desirable features** of **NoSQL** databases and **RDBMS.**
- **FoundationDB** and **OracleNoSQL**: distributed, replicated **key-value store** with a shared-nothing architecture;
- All **reads** and **writes** in **FoundationDB** provide **ACID** guarantees (Atomic, Consistent, Isolated, and Durable);
- **OracleDB** supports **atomic operations** on the same key, and allows **atomic transactions** on sets of keys that share the same major key path.

BIG DATA ANALYSIS PIPELINE

Index

- What is Big Data
- 5V of Big Data
- CAP Principle
- Big Data Problems
- Big Data Application Fields
- Big Data Criticality and Risk
- NoSQL
- **Big Data Analysis Pipeline**
- Big Data Solutions

Pipeline


INPUT
Acquisition/Recording
Extraction/Annotation
Integration/Aggregation/Representation
Analysis/Modeling
Interpretation
OUTPUT
ALGORITMI

Acquisition/Recording

- **Huge** amount of **data** can be **filtered** and **compressed** by orders of magnitude.
  - **Challenge**: define these filters in such a way that they do not discard useful information.
- Detail regarding **experimental conditions**, procedures may be **required to interpret** the results correctly.
  - **Challenge**: automatically generate the right metadata.
- Must be possible to **research** both into **metadata** and into **data** systems.
  - **Challenge**: create optimized data structures that allow searching in acceptable times.

Information Extraction & Cleaning

- The **information collected** will **not** be **in** a format ready for **analysis** (surveillance photo VS picture of the stars).
  - **Challenge**: realize an information extraction process that pulls out information, and express it in a form suitable for analysis.

- **Big Data** are **incomplete** and **errors** may have been committed during **Data Acquisition** phase.
  - **Challenge**: define constraints and error models for many emerging Big Data domains.

Data Integration, Aggregation, Representation

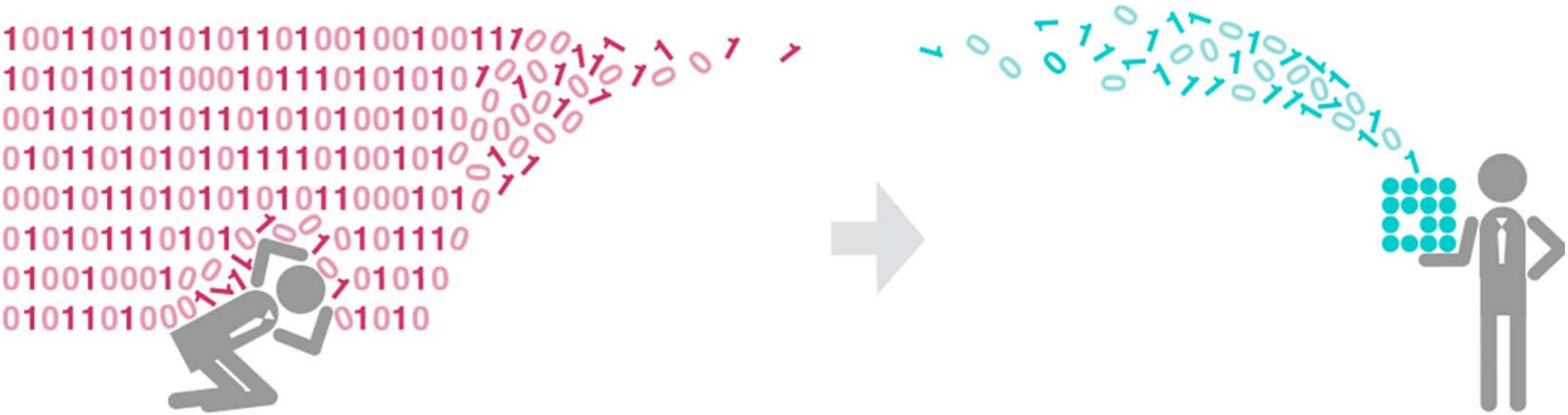
- Data is **heterogeneous**, it is not enough throw it into a repository.
  - **Challenge**: create a data record structure that is suitable to the differences in experimental details.
- **Many ways to store** the same information: some designs have advantages over others, for certain purposes.
  - **Challenge:** create tools to assist in database design process and developing techniques.

Query Processing, Data Modeling & Analysis

- Methods for querying and **mining Big Data** are **different** from traditional **statistical analysis**.
  - **Challenge**: create a scaling complex query processing techniques to terabytes while enabling interactive response times.
- **Interconnected Big Data** forms large **heterogeneous networks**, with which **information redundancy** can be explored to **compensate** for **missing data**, to **crosscheck conflicting** cases and to uncover hidden relationships and models.
  - **Challenge**: add coordination between database systems and provide SQL querying, with analytics packages that perform various forms of non-SQL processing (data mining, statistical analysis).

Datification

- **Datification**: to taking information about all things and transforming it into a data format to make it quantified.
- Use this information in new ways to **unlock** the implicit, latent **value** of this information.
- When the data were "**few**", it was desirable they were **accurate** (Random Sampling). BigData have changed the expectations of precision: to deal with these **large quantities** of data as something imprecise and imperfect allows us to make superior forecasts (**Predictive Analisys**).



BIG DATA SOLUTIONS

Index

- What is Big Data
- 5V of Big Data
- CAP Principle
- Big Data Application Fields
- Big Data Problems, Criticality and Risk
- NoSQL
- Big Data Analysis Pipeline
- **Big Data Solutions**

Big Data Solutions

- Considering technologies and solutions for Big Data, **4 classes** of **fundamental aspects** can be identify:
  - Data Management aspects;
  - Architectural aspects;
  - Access/Data Rendering aspects;
  - Data Analysis and Mining/Ingestion aspects.

Data Management aspects

- **Scalability**: Each storage system for big data must be scalable, with common and cheap hardware (increase the number of storage discs.)
- **Tiered Storage**: To optimize the time in which we want the required data.
- **High availability**: Key requirement in a Big Data architecture. The design must be distributed and optionally lean on a cloud solution.
- **Support to Analytical and Content Applications**: analysis can take days and involve several machines working in parallel. These data may be required in part to other applications.
- **Workflow automation**: full support to creation, organization and transfer of workflows

Data Management aspects

- **Integration with existing Public and Private Cloud systems**: Critical issue: transfer the entire data. Support to existing cloud environments, would facilitate a possible data migration.
- **Self Healing**: The architecture must be able to accommodate component failures and heal itself without customer intervention. Techniques that automatically redirected to other resources, the work that was carried out by failed machine, which will be automatically taken offline.
- **Security**: The most big data installations are built upon a web services model, with few facilities for countering web threats, while it is essential that data are protected from theft and unauthorized access.

Architectural aspects

- **Clustering size**: the number of nodes in each cluster, affects the completion times of each job: a greater number of nodes, correspond to a less completion time of each job.
- **Input data set**: increasing the size of the initial data set, the processing time of the data and the production of results, increase.
- **Data node**: greater computational power and more memory, is associated with a shorter time to completion of a job.
- **Date locality**: it is not possible to ensure that the data is available locally on the node. we need to retrieve the data blocks to be processed, and the time of completion be significantly higher.

Architectural aspects

- **Network**: The network affects the final performance of a Big Data management system; connections between clusters make extensive use during read and write operations. Highly available and resiliency network, which is able to provide redundancy.
- **Cpu**: more processes to be carried out are CPU-intensive, greater will be the influence of the CPU power on the final performance of the system.
- **Memory**: For applications memory-intensive is good to have the amount of memory on each server, able to cover the needs of the cluster. 2-4GB of memory for each server, if memory is insufficient performance would suffer a lot.

Riak

- Open source NoSQL, written in C++;
- **Key/Value DB** implementing the principles from Amazon's Dynamo paper;
- **Masterless** system (eventually consistent);
- Data is automatically distributed across nodes using **consistent hashing**;
- **Riak Control**, an open source graphical console for monitoring and managing Riak clusters;
- **RiakCS** (cloud storage built on a distributed database Riak).

Riak

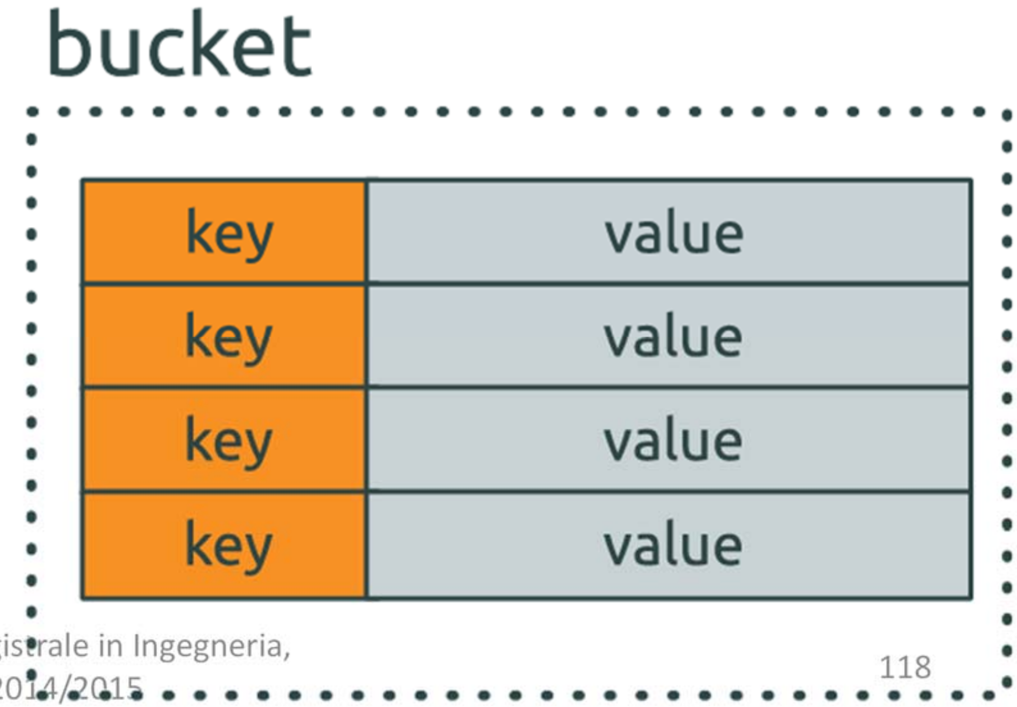
- **Riak Control**: an open source graphical console for monitoring and management of the Riak cluster.
- **RiakCS**: a simple **cloud storage** system **open source**, built on a distributed database Riak.

Riak - Features

- **Scalability:** data is rebalanced automatically with no downtime, when add/remove machine; data is automatically distributed around the cluster and yields a near-linear performance increase as capacity is added.
- **Availability:** a neighboring node will take over write and update responsibilities for a node becoming unavailable.
- **Operational Simplicity**: Add new machines to a Riak cluster is easily without larger operational burden.

Riak - Features

- **Simple Data Model:** key/value pairs stored in flat namespace (bucket). All object are stored on disk as binaries. Developing code for this model is simpler and more efficient; perfect for applications that require rapid interactions;
- **Masterless design:** any node can serve any incoming request, because all data is replicated across nodes;



Riak - Features

- **MapReduce:** allows operation like filtering documents by tag, counting words in documents, extracting links to related data (javascript support);
- **Riak Search**: distributed full-text search engine that provide support for various MIME type and robust querying (exact matches, wildcards, range queries, proximity search).
- **Secondary Indexing (2i):** each object can be tagged with 1 ore more queryable values, integers or strings (exact matches, range queries).

Riak - Features

- **Fault-Tolerance:** due to network partition or hardware failure, access can be lose to many nodes without losing data;
- **Consistent hashing:** ensures data is distributed *evenly* around the cluster. New nodes can be added with automatic, minimal reshuffling of data.
- **Active anti-entropy:** self-healing property in background; it uses a hash tree exchange to compare replicas of objects and automatically repairs/update any divergence.

Riak – CRUD Operations

- **Create**: the key can be predetermined or self-generated by Riak.
- **Read**:
  - Direct access to key.
  - Secondary indexes.
  - Inverted indexes (text search).
  - List of keys to the bucket.
- **Update**: update must always contain all data.
- **Delete**: to remove a bucket each contained value must delete.

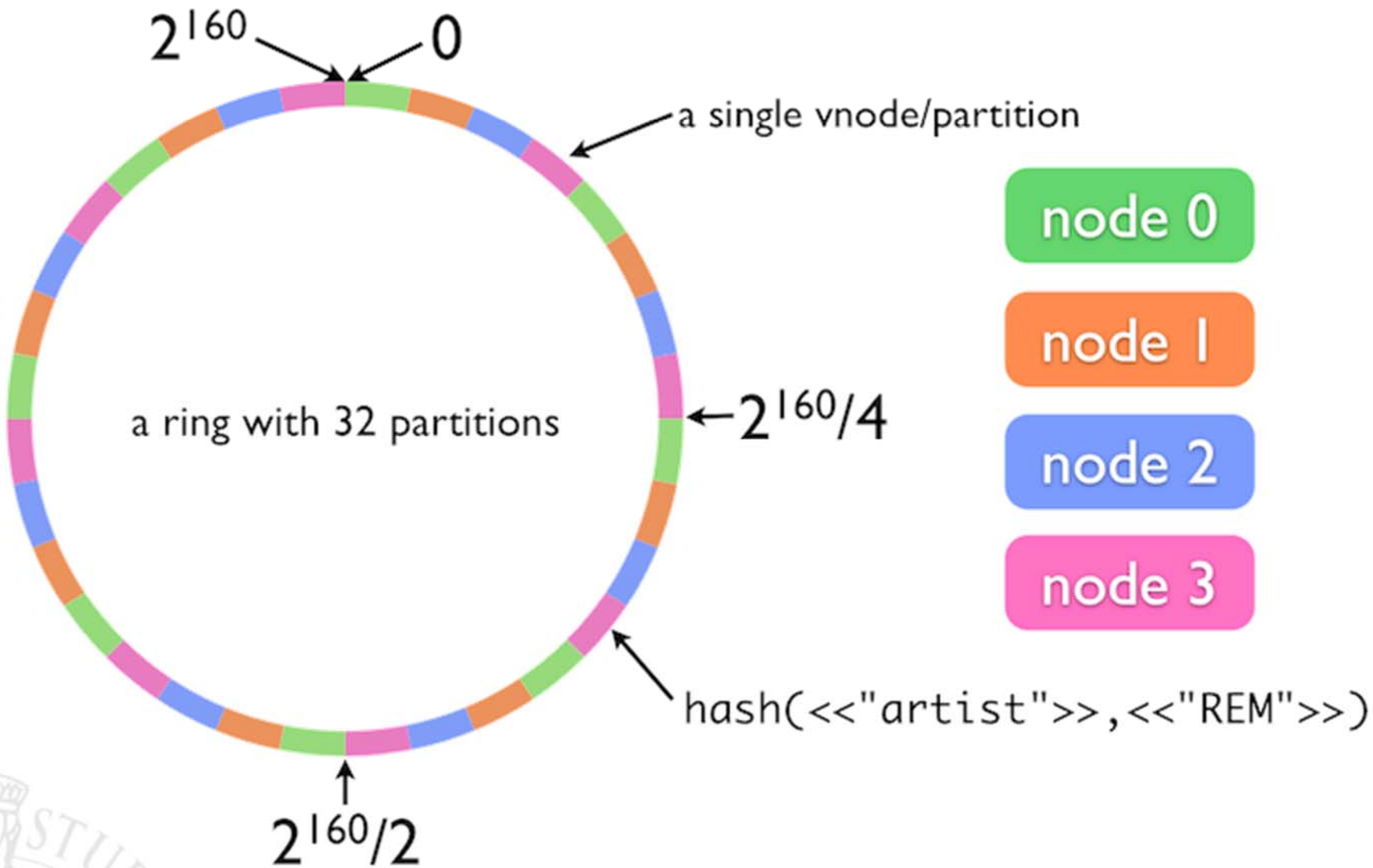
Riak

- Riak: developed to be **used in a cluster**, i.e. a set of nodes that are physically host.
- Each node has an internal set of virtual nodes (**vnodes**), each of which deals with data storage in a partition of the **key space** (= space of entire **bucket-key**)
- Nodes are **not clones**, **not all** nodes participate to satisfy the **same request**. Their behavioral model is configurable, in fact, for example you can set via the **API** Riak:
  - **W** value: identifies **number of nodes** that must take action to complete an **Update**.
  - **R** value: indicates **number of nodes** that need to **respond positively** to consider a **reading as valid** .

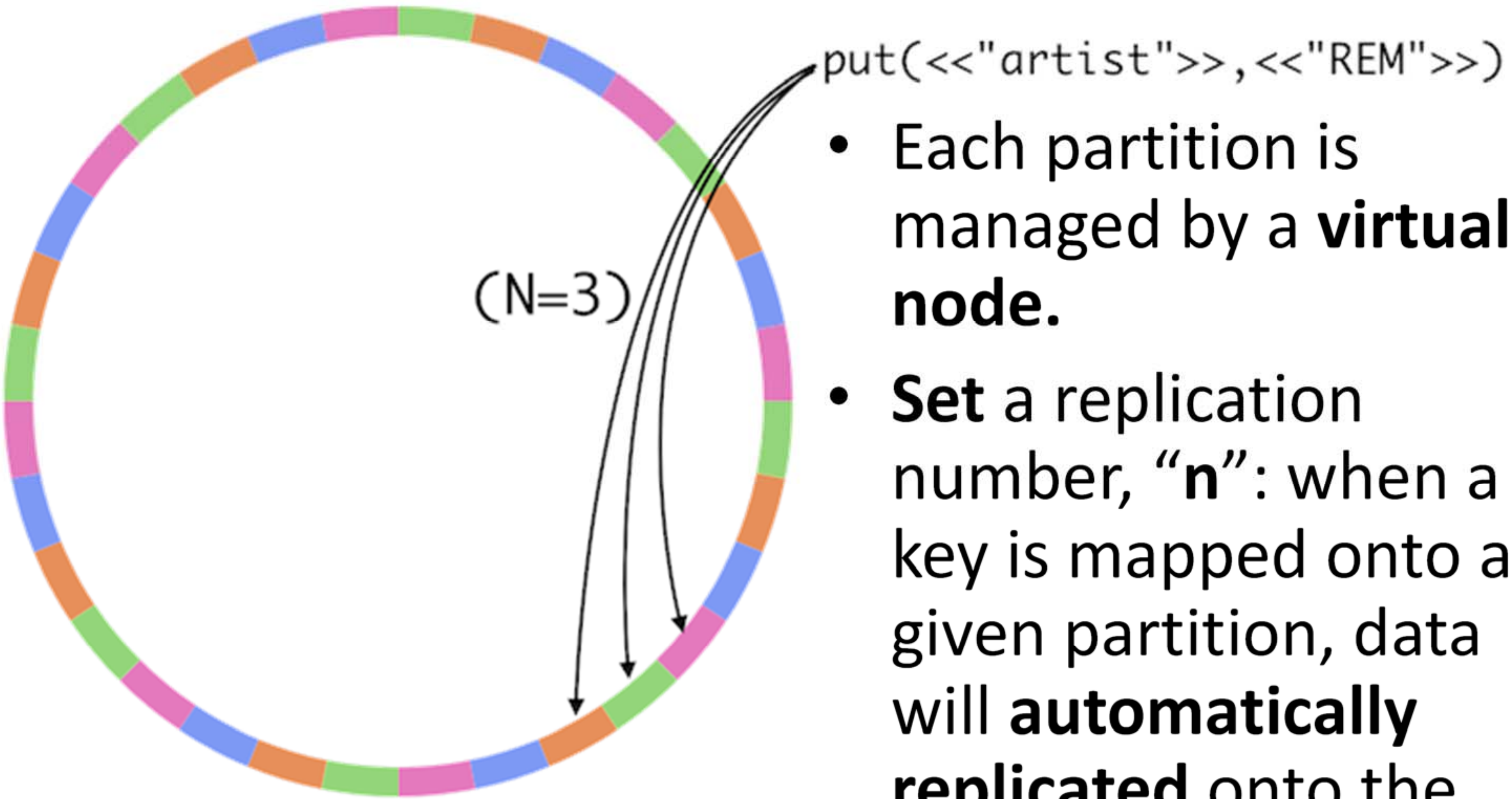
Riak – Consistent Hashing

- Makes possible the **automatic redistribution** of data across nodes;
- Ensures data is evenly distributed;
- Bucket and key combination is hashed to an **hash maps** onto a 160-bit integer space (ring)
- The ring is used to determine **what data** to put on **which** physical **machines**
- The integer space is divided into **equally-sized partitions**. Each partition correspond to a **range of values** on the ring, and is responsible for all buckets/keys couple that, when hashed, fall into that range.

Riak – Consistent Hashing



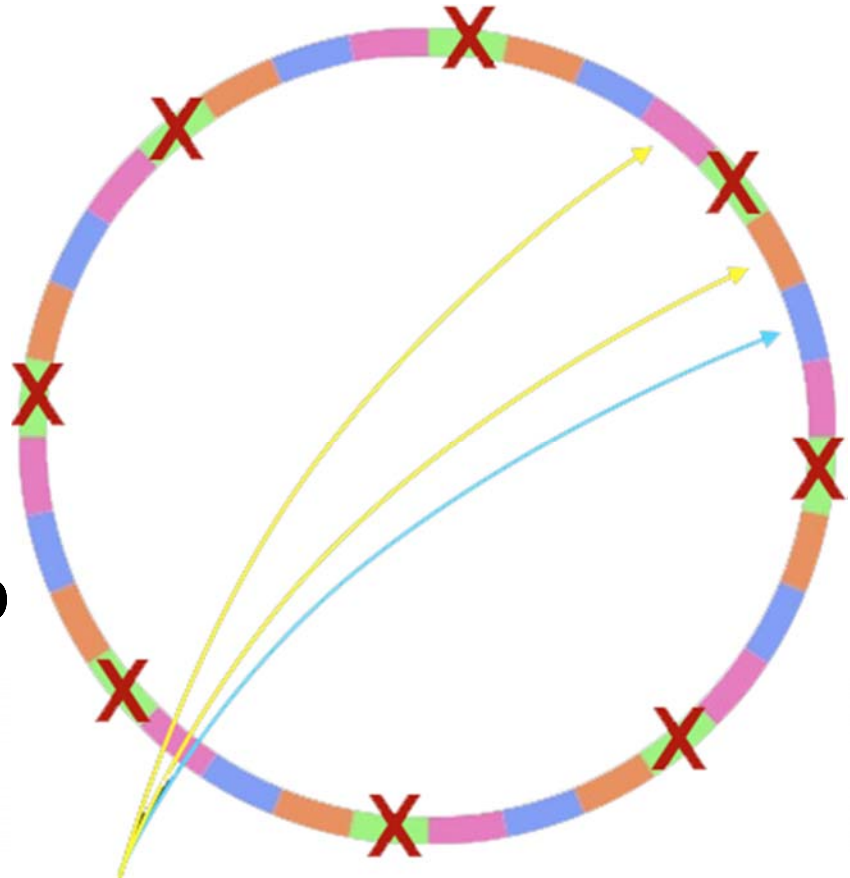
Riak – Consistent Hashing



- Each partition is managed by a **virtual node.**
- **Set** a replication number, "**n**": when a key is mapped onto a given partition, data will **automatically replicated** onto the next **n-1** partitions.

Riak – Consistent Hashing

- When a **node fails**, the **requests** are sent to a "**secondary replica**", in which **data**, from other replicas primary still active, **has been copied**.
- If the node is **restored**, thanks to **handoff**, data can be transferred back to the original node.
- The system recovers its **normal functioning.**



Riak

- A running node can be added to an existing cluster. To stage a join request:

  riak-admin cluster **join** riak@192.168.2.2

- If successful, node receives the **new cluster state** and starts claiming partitions until even distribution (primary replica for the partition);
- It recalculates a new cluster state and **gossips** it to a random node;
- **Partition handoff** starts to transferring data from existing nodes to the new one (already ready to serve requests).

Riak – Vector Clock

- A method for keeping track of which version of a value is current;
- When a value is stored, it is **tagged** with a vector clock, that it is **extended** for each update;

a85hYGBgzGDKBVIcR4M2cgczH7HPYEpkzGNlsP/VfYYvCwA=

- **Auto-repair** out-of-sync data.
- Enable clients to **always write** to the database in exchange for consistency conflicts being resolved at read time by either application or client code.
- It can be configured to store copies of a given datum based on size and age.
- It can be disable to fall back to simple **time-stamp** based “last-write-wins”.

Riak – Vector Clock

- On each update **vector clock** is **extended** with a specific mechanism, so **Riak** can **compare two replicas** of an object **to determine**:
  - If an object is a **direct descendant** of another object.
  - If multiple objects are **direct descendants** of a common object.
  - If multiple objects are **uncorrelated**.
- Allows to **understand** if there were **conflicts during writing** process.
- Using this knowledge, **self-repair mechanisms** on **unsynchronized data** can be applied; or provide to **clients** the ability to **reconcile changes** with divergent mechanisms.

Access and Data Rendering aspects

- **User Access**: thanks to an Access Management System, is possible to define different types of users (normal users, administrator, etc.) and assign to each, access rights to different parts of the Big data stored in the system.
- **Separation of duties**: using a combination of authorization, authentication and encryption, may be separate the duties of different types of users.

These separation provides a strong contribution to safeguard the privacy of data, which is a fundamental feature in some areas such as health/medicine o government.

Access and Data Rendering aspects

- **Efficient Access**: defining of standard interfaces (specially in business and educational application), managing concurrencies issues, multi-platform and multi-device access it is possible not decrease data's availability and to improve user experience.
- **Scalable Visualization**: because a query can give a little or enormous set of result, is important a scalable display tools, that allow a clear vision in both cases (i.e. 3D adjacency matrix for RDF)

RDF- Resource Description Framework

- **RDF** is a standard for describing resources on the web, using statements consist of **subject** - **predicate** – **object**

- An **RDF model** can be represented by a **directed graph**
- An RDF graph is represented physically by a serialization **RDF/XML** - **N-Triple** - **Notation3**

RDF- Resource Description Framework

- RDF triples are collected in **triplestore**.
- These data structures can be realized, for example using **Owlim-SE, Virtuoso**.
- **RDF triples** must be **indexed**: the main problem is to define **indices** that allow to **obtain** a query **response**, in an **acceptable time**.
- Different frameworks allow to define **different types of index**.

RDF - OwlimSE

- A **high-performance semantic repository** created by Ontotext, implemented in **Java** and packaged as a Storage and Inference Layer (SAIL) for the **Sesame RDF** framework.
- The **memory** required for the **indices** (cache types) depends on which indices are being used.
- The **SPO** (Subject - Predicate - Object) and **PSO** (Predicate - Subject - Object) indices are always used.
- **Optional indices** include:
  - Predicate Lists
  - Context indices PCSO/PSOC
  - Full-Text Search (FTS) indices (Node Search, Search with Lucene)
  - Geo-Spatial

RDF – Predicate Lists

- 2 indices (**SP** and **OP**) to improve performance in 2 separate situations:
  - **Loading/querying datasets** that have a large number of predicates
  - **Executing queries** or **retrieving statements** that use a wildcard in the predicate position, for example using the statement pattern: dbpedia:Human ?predicate dbpedia:Land
- A dataset with **more** than about **1000 predicates** will **benefit** from **using these indices**.
- Predicate list indices are **not enabled** by **default**, but can be switched on using the **enablePredicateList** configuration parameter.

RDF – PCSO PCOS

- 2 indices used for providing better performance when **executing queries** that use **contexts**.
- Thanks to context, the core RDF model can be extended **from a triple to a quad** (**Named graphs**), provide a useful **extra degree** of **freedom** managing an RDF dataset.
- Enabled using **theenable-context-index** configuration parameter.
  - Predicate-context-subject-object (**PCSO**)
  - Predicate-context-object-subject (**PCOS**)

RDF - FTS

- **Full-text search** (FTS) concerns **retrieving text** documents out of a large collection
  - **by keywords** or,
  - **by tokens** (represented as sequences of characters).
- A query represents an unordered **set of tokens** and the **result** is **set of documents**, relevant to the query.
- 2 type of full text search in OwlimSE:
  - **Node Search** - Proprietary Full-Text Search
  - **RDF Search** - Full-Text Search using **Lucene**

RDF - Lucene

- **Apache Lucene** is a high-performance, **full-featured text search engine** written entirely in Java.
- Owlim-SE supports **full text search capabilities** using **Lucene**
- **RDF molecule**: a text document created for each node in the RDF graph to be indexed.
- Some Parameters:
  - **Exclude:** Provides a regular expression to identify nodes that will be excluded from to the molecule.
  - **Include:** Indicates what kinds of nodes are to be included in the molecule.
  - **Index:** Indicates what kinds of nodes are to be indexed.

RDF – FTS Node

- Resembles functionality **similar** to typical FTS implementations in **relational DBMS**.
- **Some algorithm Predicates:**
  - **fts:exactMatchMatches:** searching for <United:States> will match "The president of the United States", but not "United Statesless", "united states" or "notUnited notStates.".
  - **fts:matchIgnoreCaseSimilar**: <United:States> will match "The president of the United States", "united states" but not "United Statesless" or "notUnited notStates."

RDF - GEO

- **Owlim-SE** allows to use queries involving constraints such as '**nearby point**' and '**within region**‘, thanks to special-purpose indices .
- The position information in **GeoNames** is provided using standard RDF and can be queried using **SPARQL**.
- Includes special support for **2-Dimensional geo-spatial data**.
  - To express constraints reminiscent of a function call, e.g.omgeo:nearby (lat long distance)
  - To find points within a circle, rectangle or polygon
  - To compute distance

RDF - Exastore

- **H2RDFplus**: maintaining all permutations of RDF elements, namely **spo**, **pso**, **pos**, **ops**, **osp** and **sop** indexes (**Exa-store**)
- All **SPARQL** triple patterns can be answered efficiently using a **single index scan** on the corresponding index.
- All six indexes guarantees that **every join** between triple patterns can be done using **merge joins**.
- Allows to create a **distribute indexing scheme** for storing RDF data implemented in HBase (with MapReduce jobs to load/index large RDF datasets).

RDF-HDT Library

Publication

- No Recommendations/methodology to publish at large scale
- Use of Vocabulary of Interlinked Data and Semantic Map

Exchange

- Main RDF format (RDF/XML – Notation3 – Turtle…) have a **document-centric view**
- Use of universal compressor (gzip) to reduce their size

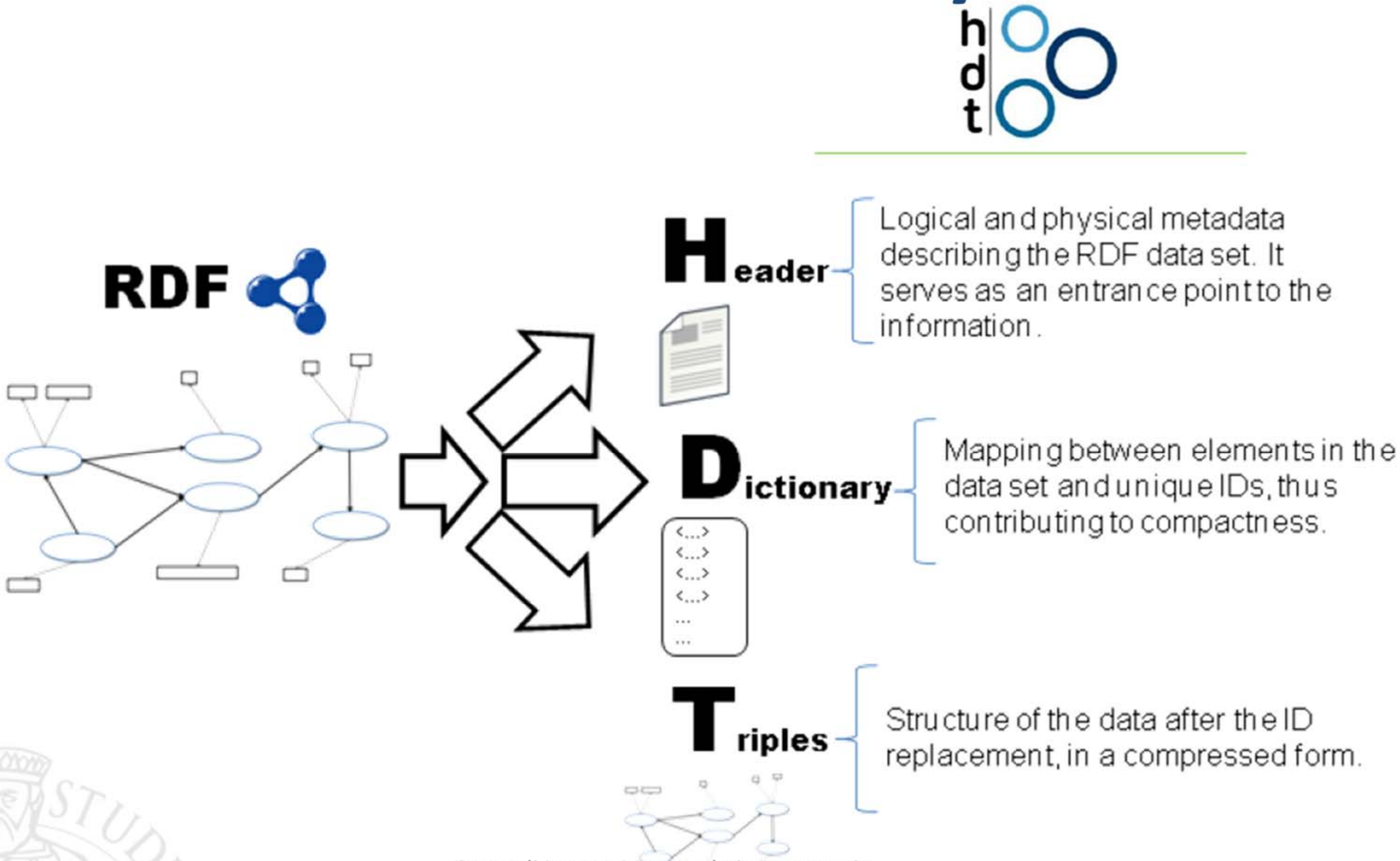
Consumption (query)

- Costly Post-processing due to **Decompression** and **Indexing** (RDF store) issues

RDF-HDT Library

- **HDT** (Header, Dictionary, Triples) is a **compact data representation** for RDF dataset to reduce its verbosity.
- RDF graph is represented with 3 logical components.
  - **H**eader
  - **D**ictionary
  - **T**riples
- This makes it an ideal format for storing and sharing RDF datasets on the Web.

RDF-HDT Library



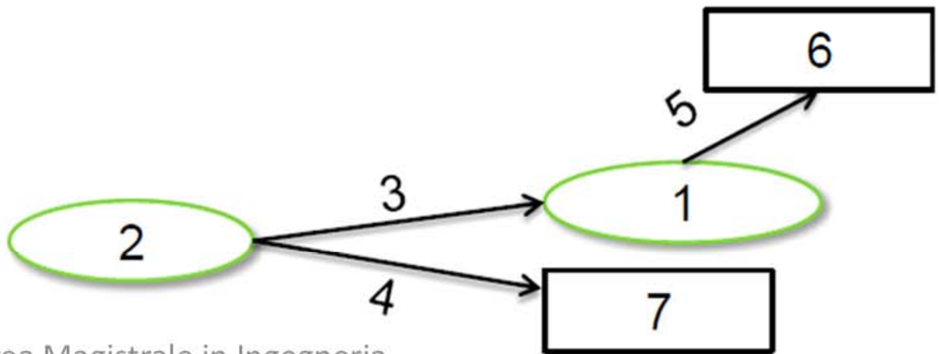
RDF-HDT Library

- HDT improve the value of metadata;
- Header is itself an RDF graph containing information about:
  - **provenance** (provider, publication dates, version),
  - **statistics** (size, quality, vocabularies),
  - **physical organization** (subparts, location of files)
  - **other types of information** (intellectual property, signatures).

RDF-HDT Library

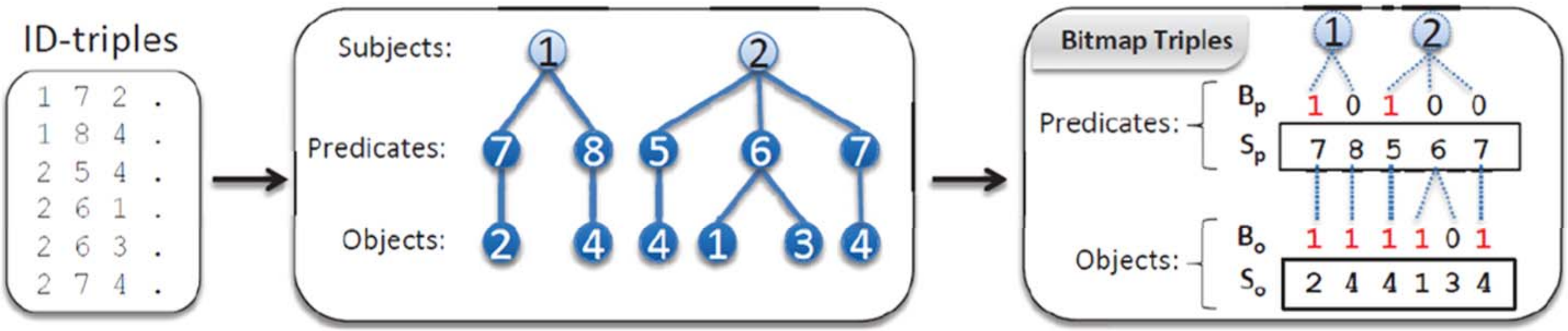
- **Mapping** between **each term** used in a dataset and **unique IDs.**
- **Replacing** long/redundant **terms** with their **corresponding IDs.** So, Graph structures can be indexed and managed as integer-steam.
- **Compactness** and **consumption performance** with an advanced dictionary serialization.

1	<http://books/author33>
2	<http://books/book21>
3	dc:author
4	dc:title
5	foaf:name
6	“Pablo Neruda”
7	“Spain in the Heart”



RDF-HDT Library

- These **ID-triples** component **compactly** represent **the RDF graph.**
- **ID-triple** is the key component to accessing and querying the RDF graph.
- **Triples component** can provide a succinct index for some basic queries.



To get a **spatial optimization** and **efficient performance in the primitive operations.**

RDF-HDT Library

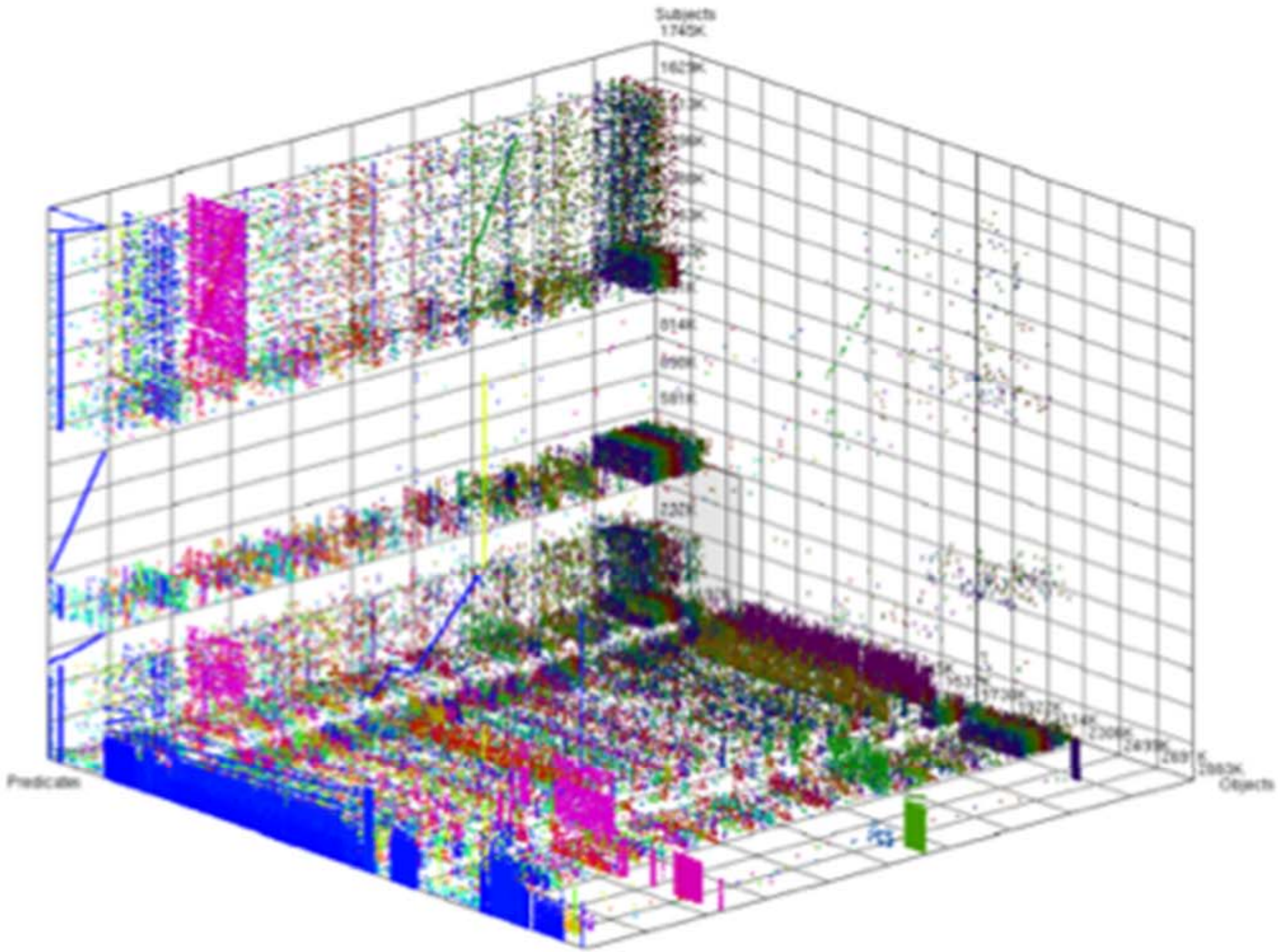
- RDF-HDT improves data exchange.

Dataset	Triples (millions)	Size (GB)	Compression (MB)		
			gzip	bzip2	HDT
wikipedia	47.0	6.88	491.04	360.01	**230.48**
dbtune	58.9	9.34	924.85	630.28	**462.31**
uniprot	72.5	9.11	1233.25	739.76	**481.34**
dbpedia-en	232.5	33.12	3513.58	2645.36	**2176.54**

- With appropriate **compression techniques**, better results can be obtained, that allow a **faster** and more **efficient** data **transfers**.

RDF-HDT Library

HDT-it!



Data Analysis and Mining/Ingestion Aspects

- Mining and Ingestion are **2 key features** in the field of big data, in fact there is a tradeoff between
  - The **speed** of data ingestion
  - The ability to **answer queries quickly**
  - The **data quality** in terms of **update**, **coherence** and **consistency**.
- This compromise impacting the design of any storage system (i.e. OLTP vs OLAP).
- For instance, some **file-systems** are **optimized** for **reads** and others for **writes**, but **workloads** generally **involve** a **mix** of both these **operations.**

Data Analysis and Mining/Ingestion Aspects

- **Type of indexing**: to speed data ingest, records could be written to the cache or apply advanced data compression techniques, meanwhile the use of a different method of indexing can improve speed of data retrieval operations at only cost of an increased storage space.
- **Management of data relationship**: in some context, data and relationships between data, have the same importance. Furthermore new types of data, in the form of highly interrelated content, need to manage multi-dimensional relationships in real-time. A possible solution it is to store relationships in apposite data structures which ensure good ability to access and extraction in order to adequately support predictive analytics tools.

Data Analysis and Mining/Ingestion Aspects

- **Temporary data**: some kinds of data analysis are themselves big: computations and analyses that create enormous amounts of temporary data that must be opportunely managed to avoid memory problems. In other cases, however, make some statistics on the information that is accessed more frequently, it is possible to use techniques to create well-defined cache system or temporary files then optimize the query process.

Hadoop

- **Hadoop** is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- **HDFS**: A distributed file system that provides high-throughput access to application data.
- **MapReduce**: A YARN-based system for parallel processing of large data sets.