Paradigmi big data

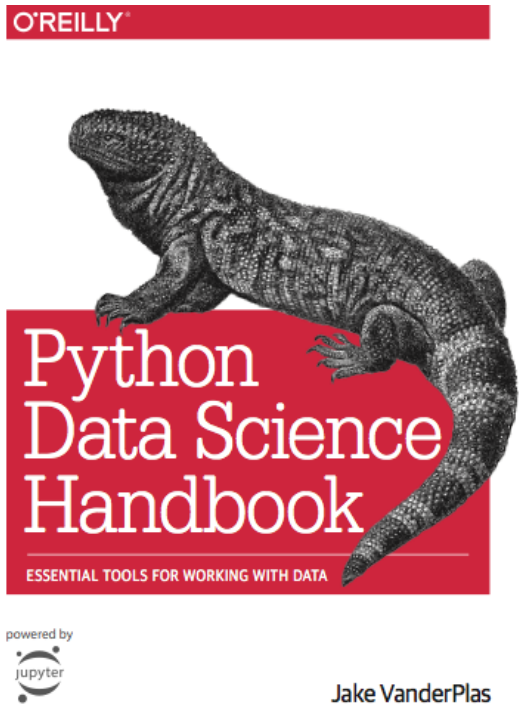
Lorenzo Stacchio

Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Materiale di approfondimento



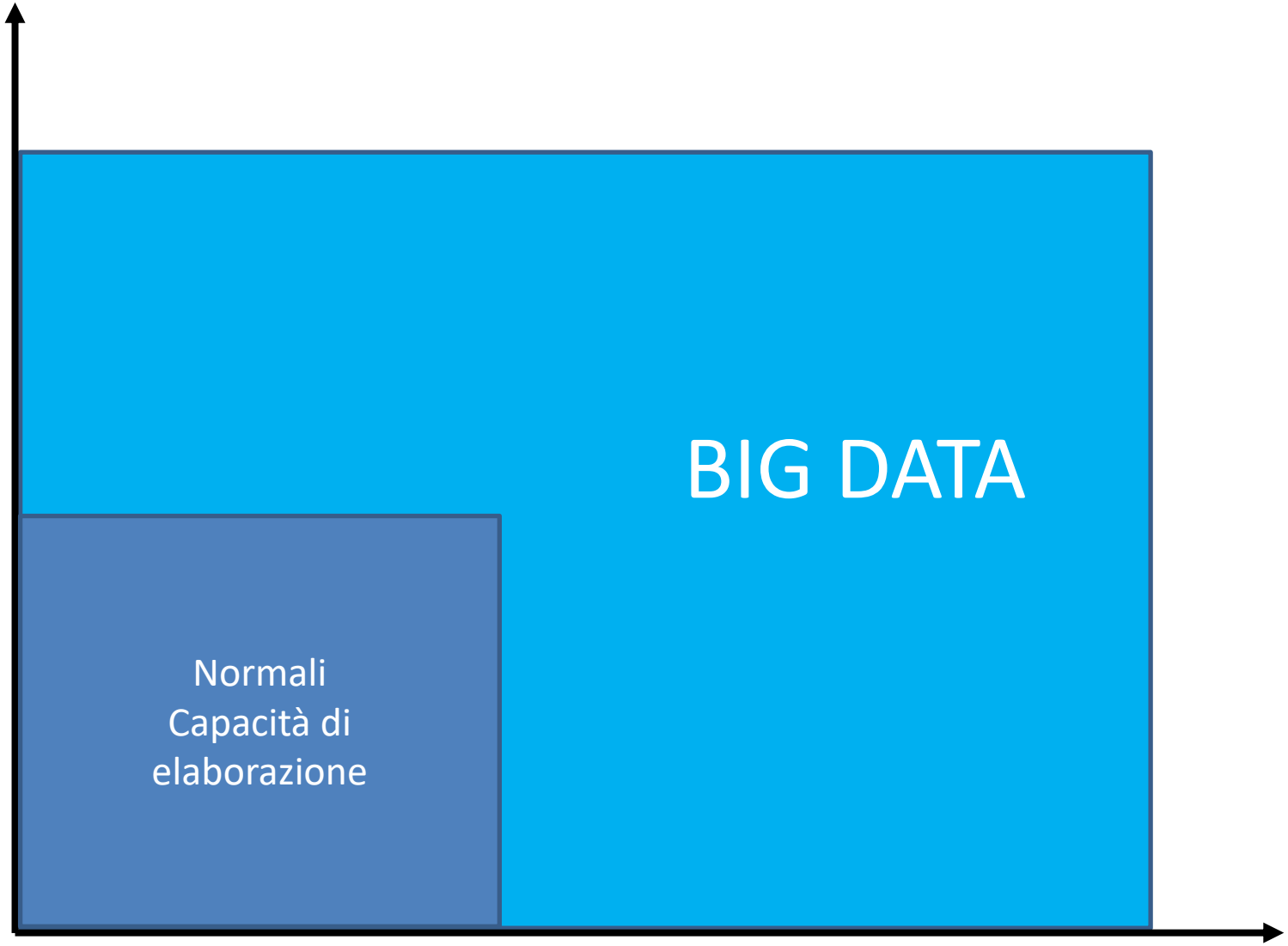
- Purtroppo il materiale di approfondimento per i big data è inglese;
- «L'informatica e in generale il mondo della ricerca ha una lingua ben predefinita»

Perché Big data? *[1]*

- Il termine Big data si riferisce ad un insieme di dati massicci con una struttura più ampia, varia e complessa rispetto a dati normali.
- Questo livello di mole produce **difficoltà di archiviazione, analisi e visualizzazione** per **processi** eseguiti su essi.
- La ricerca su enormi quantità di dati e la scoperta di **pattern** e correlazioni tra essi viene spesso detta **analisi dei big data**.
- I dati e i risultati che possiamo trarre da esse sono fondamentali sia per le aziende (vantaggio rispetto alla concorrenza) ma anche per l'avanzamento nella ricerca in vari campi delle scienze (biologia, chimica, arte etc.)

[1] Sagiroglu, Seref, and Duygu Sinanc. "Big data: A review." 2013 international conference on collaboration technologies and systems (CTS). IEEE, 2013.

Quando i data diventano big?

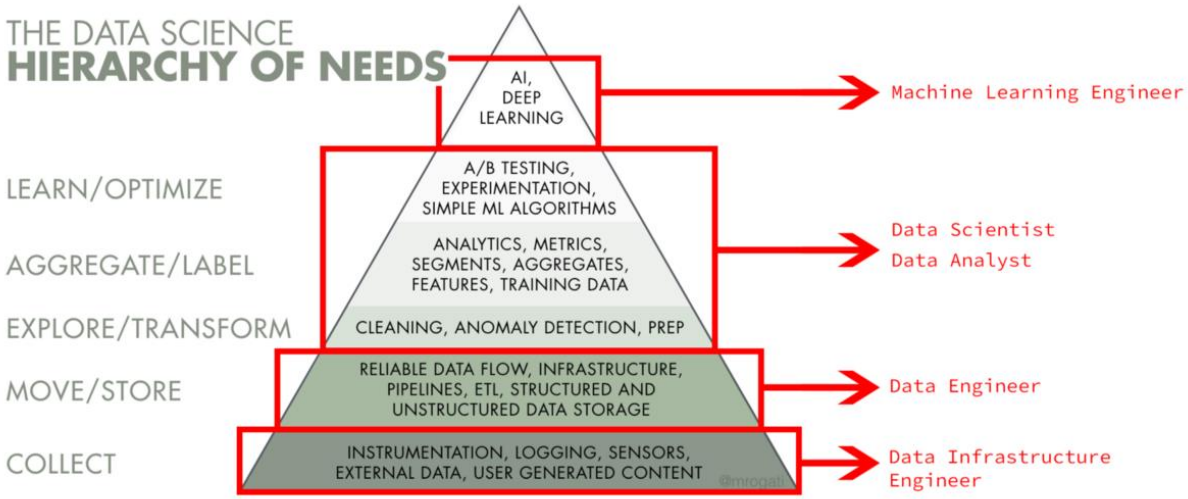


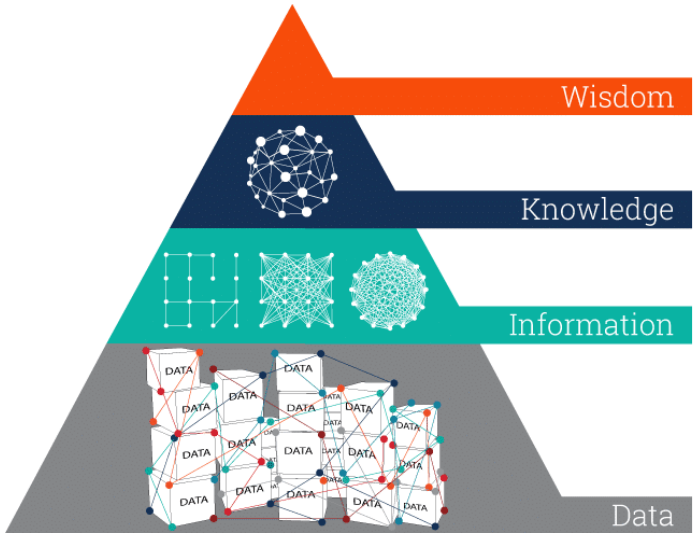
Le 4 V dei big data

- **Volume:**
  - I big data sono tanti!
- **Velocità:**
  - I big data vengono prodotti velocemente e devono essere elaborate velocemente!
- **Varietà:**
  - I big data hanno diversa natura: **strutturata, non strutturata, dati multimediali**
- **Veracità:**
  - I big data potrebber contenere **fonti di dati inattendibili**.
  - Ad esempio, i sentimenti delle persone espressi nei social media sono incerti per natura, dal momento che implicano il giudizio umano (eppure contengono informazioni preziose).

Le 8V dei big data

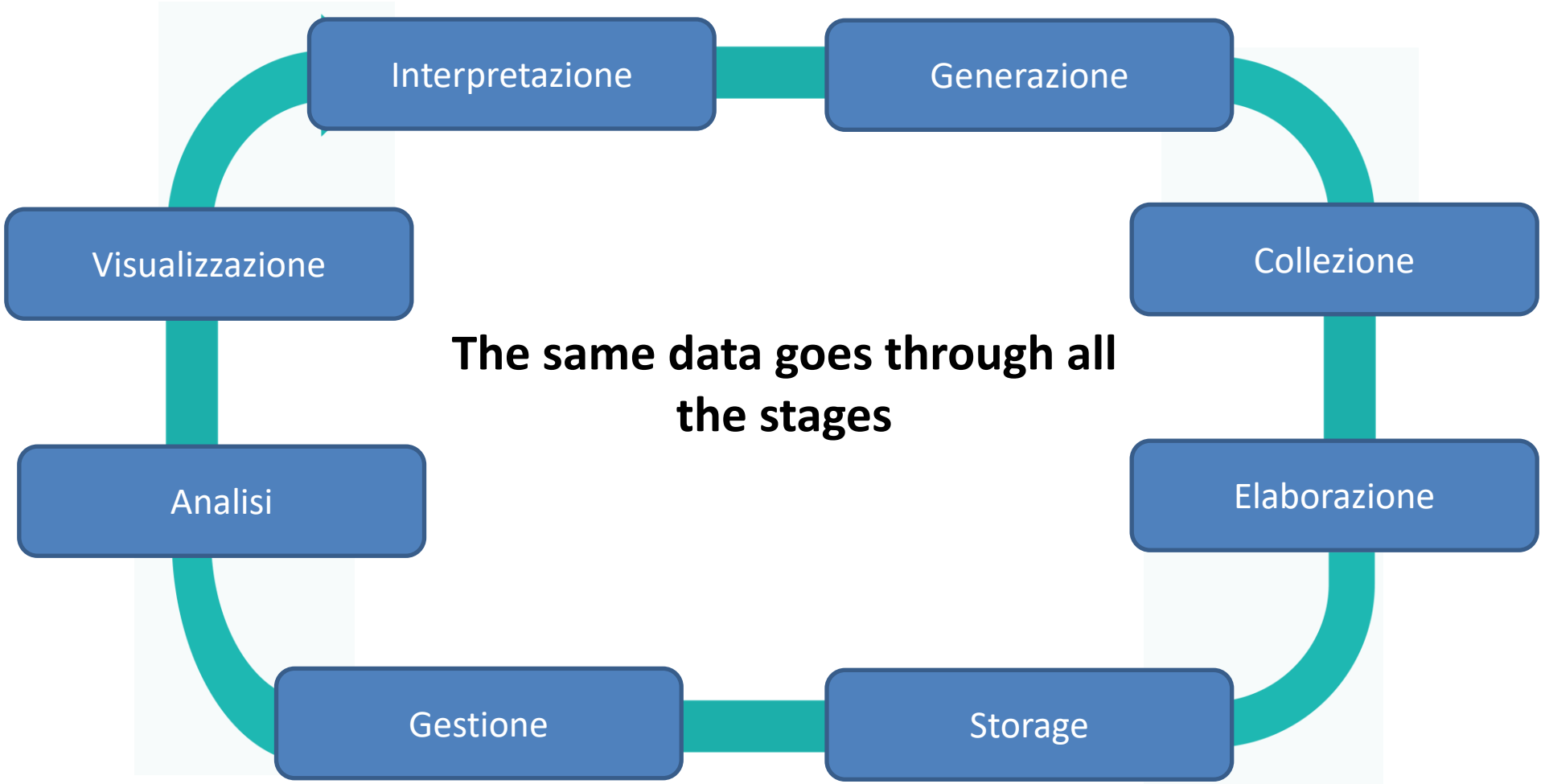
Era egizia dei big data: le piramide Data Science e DIKW

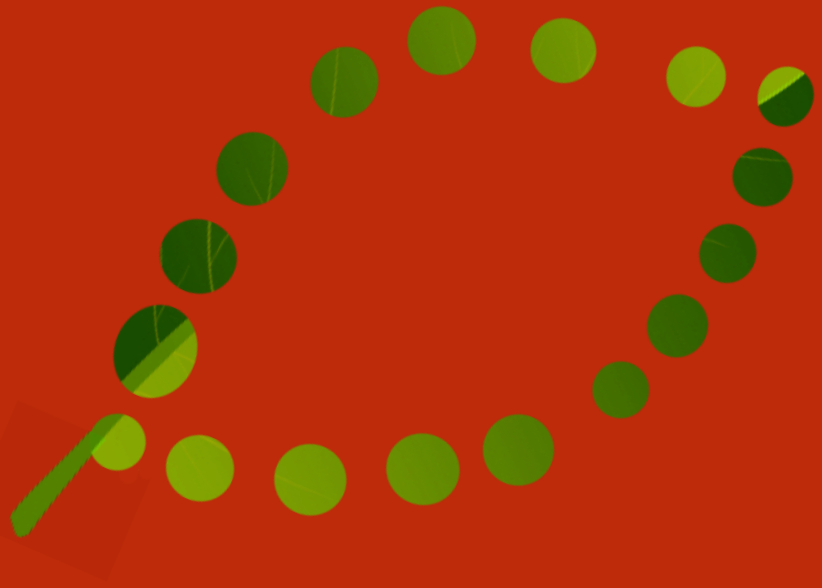




Each step up the pyramid answers questions about and adds value to the initial data.

Il ciclo dei (big) data





La natura dei dati

Lorenzo Stacchio

Studente di dottorato in Computer Science

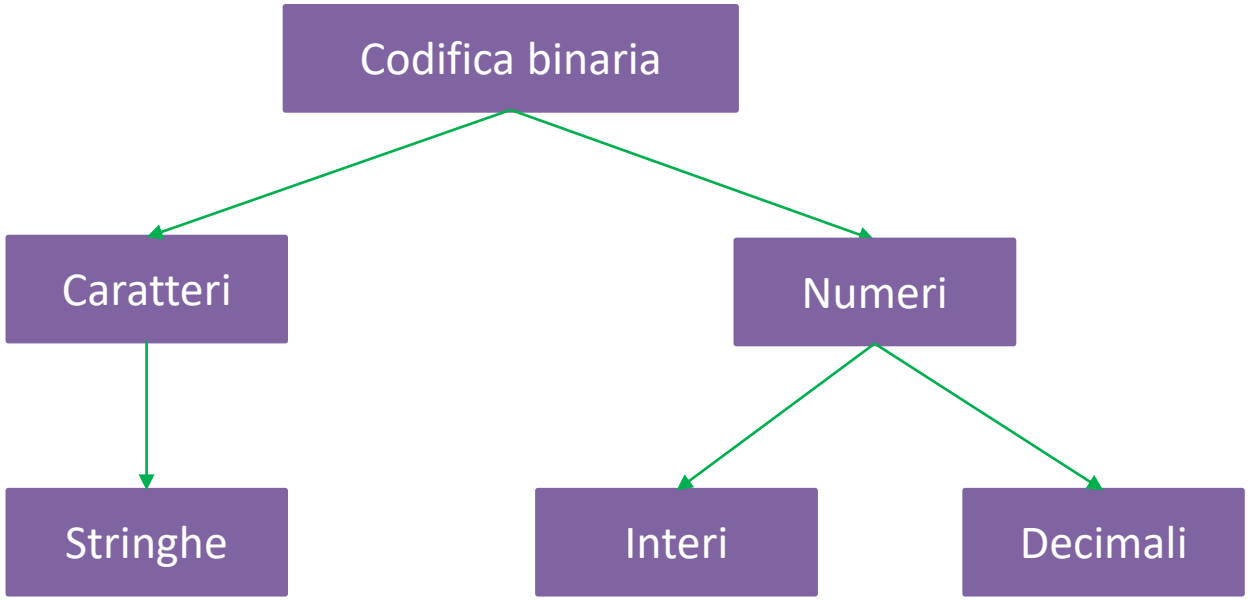
Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

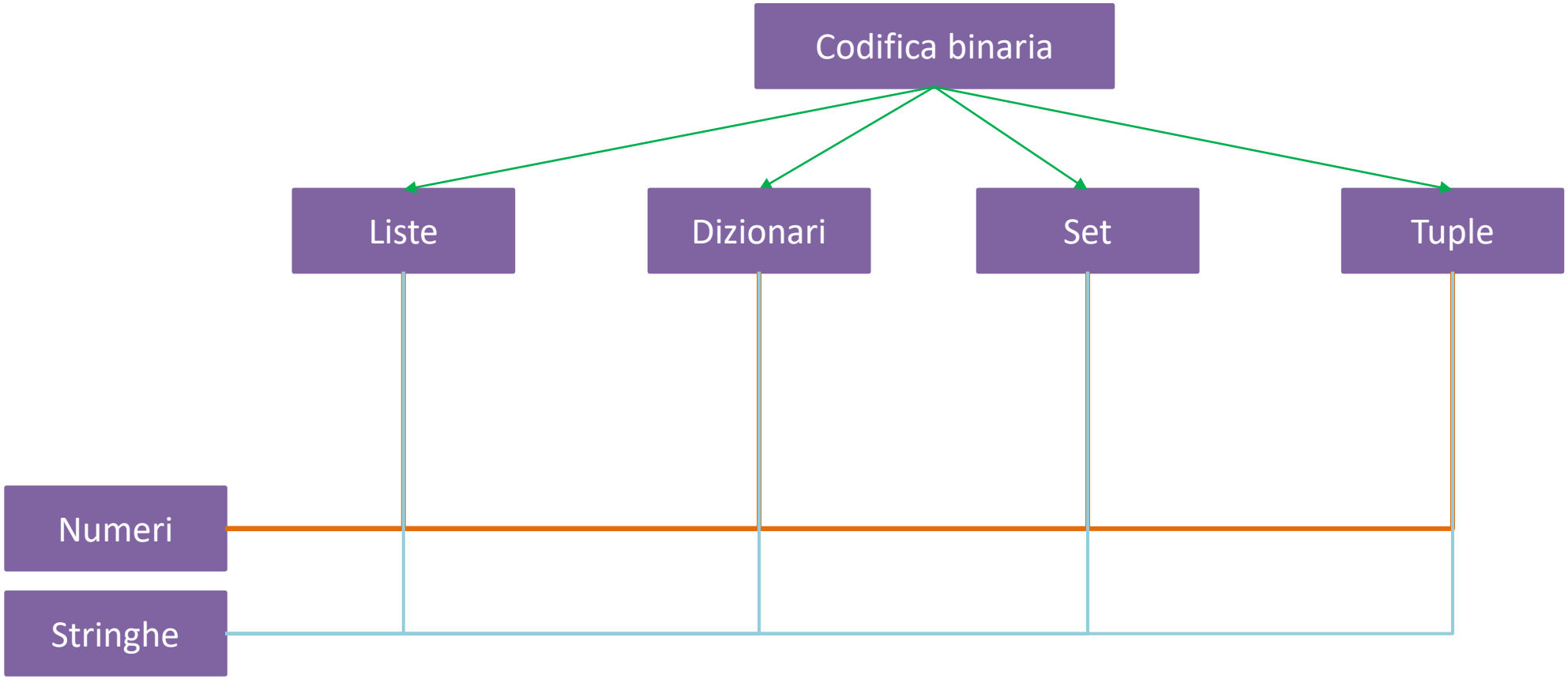
La natura dei dati

- Ormai sappiamo che ogni tipo di dato trattato computazionalmente è codificato come un insieme di elementi binari;

- Tuttavia, in base all'utilizziamo della codifica binaria, siamo in grado di rappresentare diverse tipologie di dato;

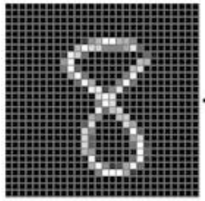


- Possiamo usare la codifica binaria per creare rappresentazioni astratte delle ormai famose **strutture dati**!



Dati multimediali

- I dati multimediali sono tipicamente rappresentati da testo, immagini, audio e video.
- I dati multimediali, non sono altro che codifiche molto più complesse delle strutture dati viste finora!
- Ad esempio possiamo pensare ad una immagine come una matrice di pixel (numeri):



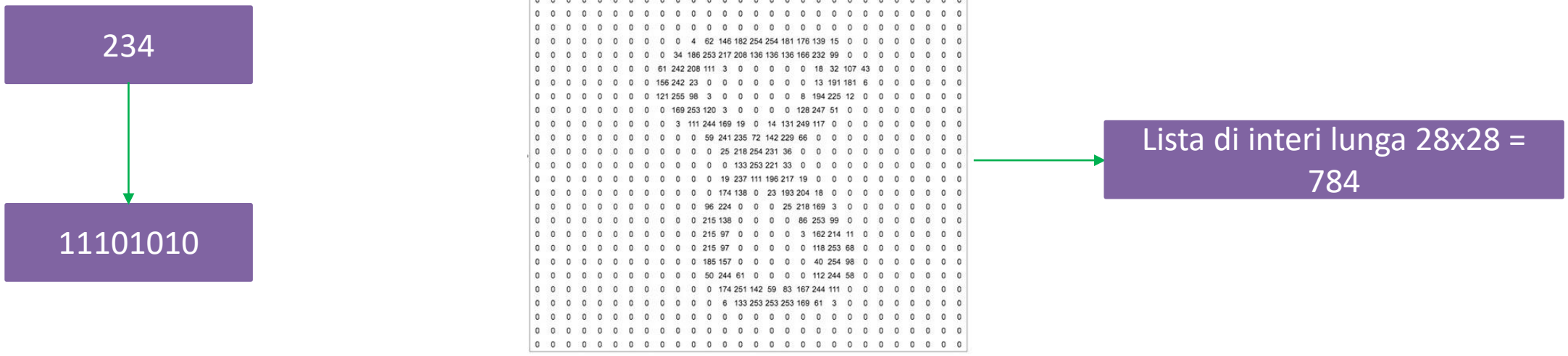
28 x 28
784 pixels

→

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Dati semplice e complessi

- L'intuzione che vorrei passarvi è che esistono dati che hanno una diversa complessità;
- Un **numero intero** può essere codificato come una semplice stringa di bit;
- Una **immagine** è rappresentata con una struttura matematica (matrice composta da interi) codificata tramite **strutture dati** che sono codificate come stringhe di bit!



- Morale: possiamo costruire dati sempre più complessi!

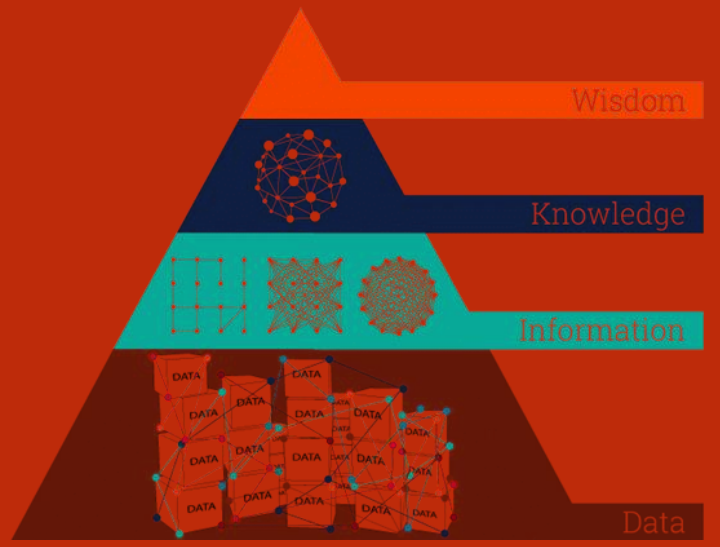
Tipologia di dati interessanti per il mondo big data

- Dati multimediali (immagini, video, suoni…);
- Dati testuali (txt, csv..);
- Di particolare rilevanza sono i file CSV (**Comma-separated values**), **ossia** file di testo utilizzato per l'importazione ed esportazione di una tabella di dati (ad esempio da fogli elettronici o database);
- Vedremo che in realtà, i file CSV sono utilizzati in maniera leggermente più complessa rispetto a questa prima definizione;

OPERA	AUTORE	CASA EDITRICE
I Robot e l'Impero	Isaac Asimov	Mondadori
Il lungo meriggio della Terra	Brian W. Aldiss	Minotauro
Absolute OpenBSD "2d Edition"	Michael W. Lucas	No Starch Press
I mercanti dello spazio	Frederik Pohl; C. M. Kornbluth	Mondadori

CSV

```
OPERA;AUTORE;CASA EDITRICE\n
I Robot e l'Impero;Isaac Asimov;Mondadori\n
Il lungo meriggio della Terra;Brian W. Aldiss;Minotauro\n
"Absolute OpenBSD ""2d Edition""";Michael W. Lucas;No Starch Press\n
I mercanti dello spazio;"Frederik Pohl; C. M. Kornbluth";Mondadori\n
```



Dati, informazioni, conoscenza e saggezza

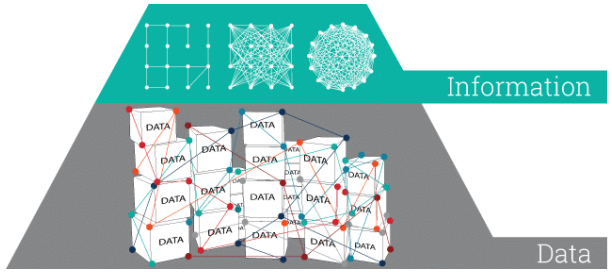
Lorenzo Stacchio

Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

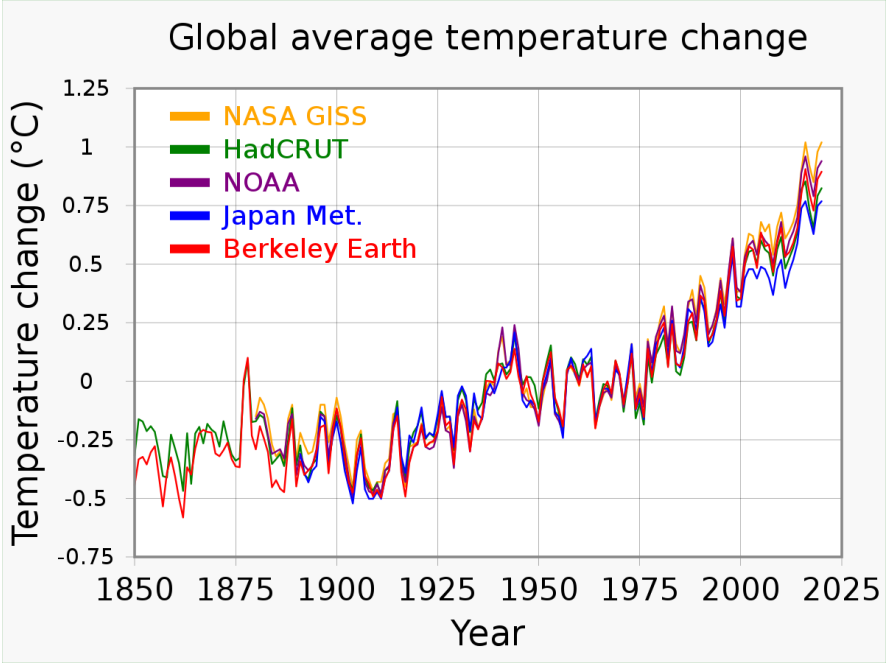
Dati, informazioni, conoscenza e saggezza



- I **dati sono una collezione di fatti grezzi e non organizzati**, come numeri e caratteri. Senza contesto, I dati hanno poco senso. **Ad esempio**, 1 è un numero, ma possiamo vederlo come **un grado di temperatura** (**1**°). Abbiamo quindi trasformato il dato in **informazione**!

- **L'informazione** è un insieme di dati «**pulito ed elaborato**» in un modo che renda più facile la manipolazione, analisi e visualizzazione; **Ad esempio**, potremmo fare un grafico per analizzare i valori della temperatura media globale negli ultimi 170 anni!

- Ponendo domande pertinenti su "chi", "cosa", "quando", "dove", ecc., possiamo ricavare informazioni preziose dai dati e renderle più utili per noi. Ma quando arriviamo alla domanda sul "**come**", siamo costretti a fare il salto dall'informazione alla **conoscenza**!

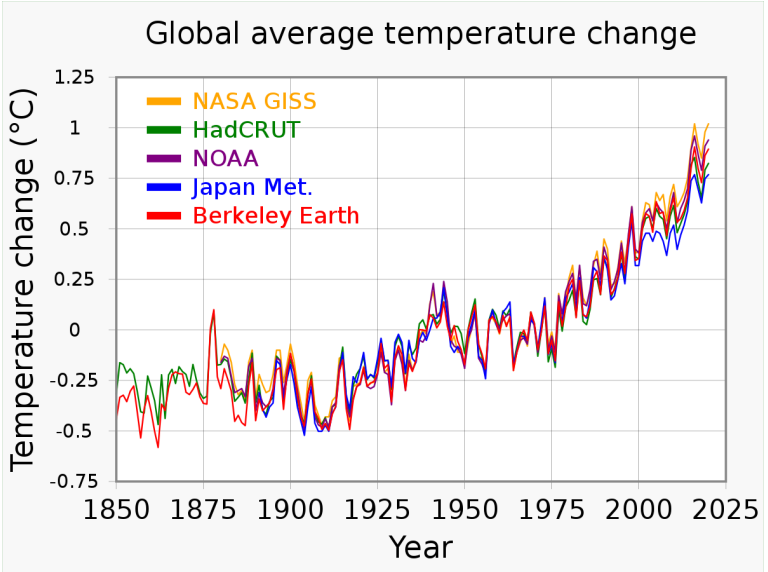


- "In che modo" le informazioni, derivate dai dati raccolti, sono rilevanti per i nostri obiettivi? “Come” i pezzi di queste informazioni sono collegati ad altri pezzi per aggiungere più significato e valore? E, **cosa più importante**, "come" possiamo applicare le informazioni per raggiungere il nostro obiettivo?

- Quando comprendiamo come applicare le informazioni per raggiungere i nostri obiettivi, le trasformiamo in **conoscenza**.

- Quando usiamo le conoscenze e le intuizioni acquisite dalle informazioni per prendere decisioni proattive, possiamo dire di aver raggiunto il gradino finale della piramide: la **saggezza**.

- La **saggezza** è il vertice della gerarchia e non è altro che conoscenza applicata per prendere la miglior decisione possibile!



- Quali sono le cause del riscaldamento globale? (**conoscenza)**

- Cosa possiamo fare per fermarlo? (**saggezza)**

I software manipolano dati.. la data science permette di creare informazione e conoscenza!

- Ogni software può essere visto come un manipolatore di dati;
- Le applicazioni che usiamo tutti i giorni sono fondate sui dati:
  - Social network (**instagram**);
  - Shopping su **amazon**;
  - Transazioni bancarie;
  - Cartelle cliniche digitali;
  - …
- Alcune tra queste applicazioni elaborano dati e informazioni per creare **conoscenza**!

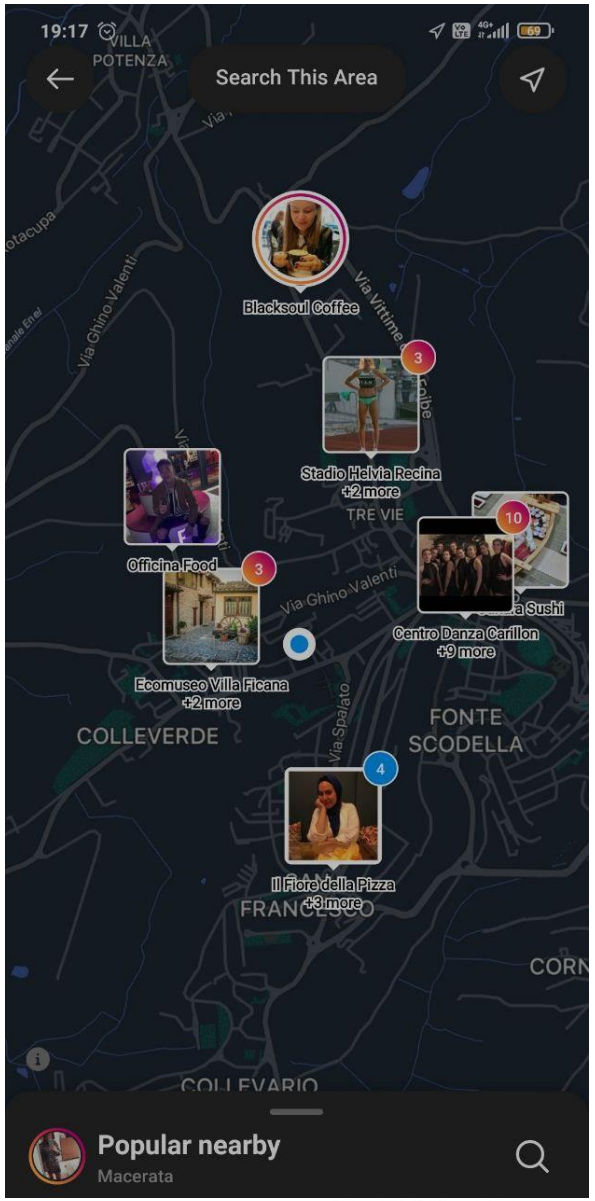
Instagram

- Un social nato per condividere i tuoi scatti migliori!

- Quando carichi una foto su instagram, il caro Zuckerberg non salva solamente i dati relativi al file della foto ma anche:

    - Chi ha caricato la foto;
    - Chi è stato taggato;
    - Quando la foto è stata caricata;
    - La caption;
    - Gli hashtag;
    - Il luogo;
    - …

- Con queste info, è in grado di capire cosa possa piacere all'utente, in base al luogo in cui si trova! (**conoscenza**)



Amazon

- Il tuo profilo amazon è associato con migliaia (se non milioni) di informazioni:
  - Anagrafica (username, password etc..);
  - Storico ordini:
    - Tutti i prodotto acquistati;
    - Chi li ha spediti;
    - Compagnia di spedizione;
    - Tempi di spedizione;
  - Hai il prime? File di accesso a prime video, music etc…
- Tutte queste informazioni, vengono usate per: **predire articoli interessanti per te, prossima serie tv da vedere, prossima canzone da ascoltare**… (**conoscenza**)

Database: superare i limiti delle strutture dati

- Tutto quello di cui abbiamo appena discusso, è possibile avendo dei dati e delle informazioni ben organizzate e gestibili!

- Noi abbiamo visto una forma rudimentale, le strutture dati, che non sono sicuramente sufficienti per organizzare dati così complessi e vari (tranne i dizionari ma ne riparleremo);

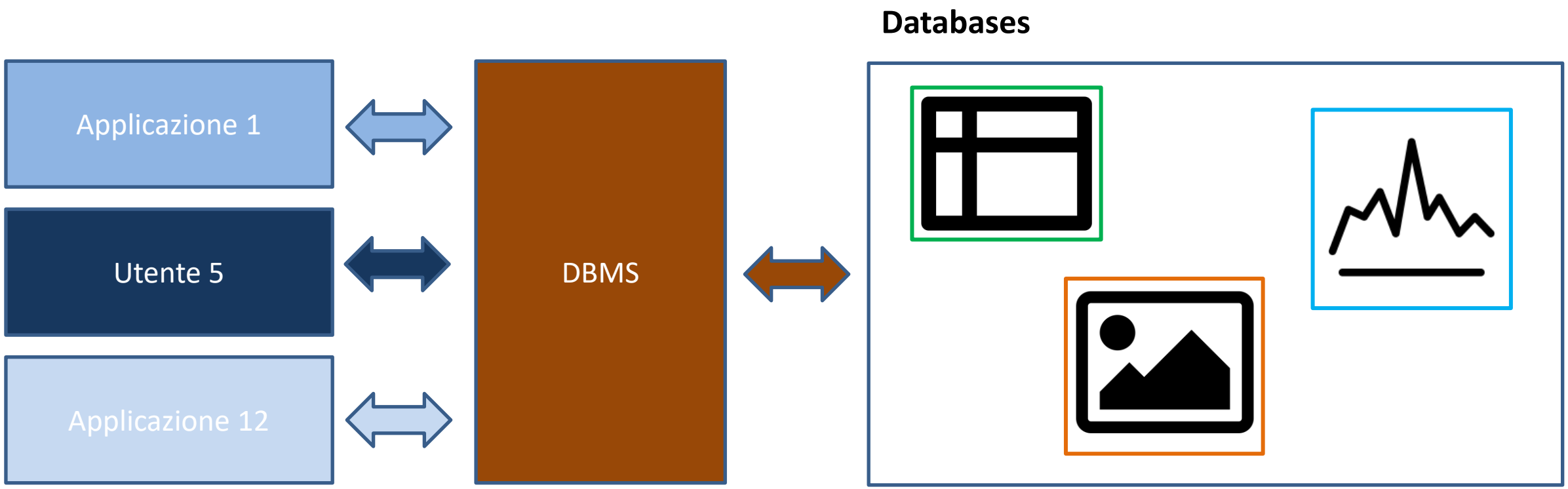
- Dobbiamo quindi trovare **delle strutture più complesse per gestire** una enorme, complessa e variegata mole di dati;

- A questo scopo esistono i **database, ossia una collezione organizzata di dati;**

- Ci sono poi dei software che permettono la gestione dei **database** chiamati **database management system (dbms):** i dbms permettono a utenti, applicazioni e al database stesso di interagire con i dati all'interno del database, permettendone la cattura e l'analisi;

- Assieme, database e dbms creano quello che viene definito **sistema database**;

Applicazioni – DBMS – Databases



Il ciclo dei (big) data

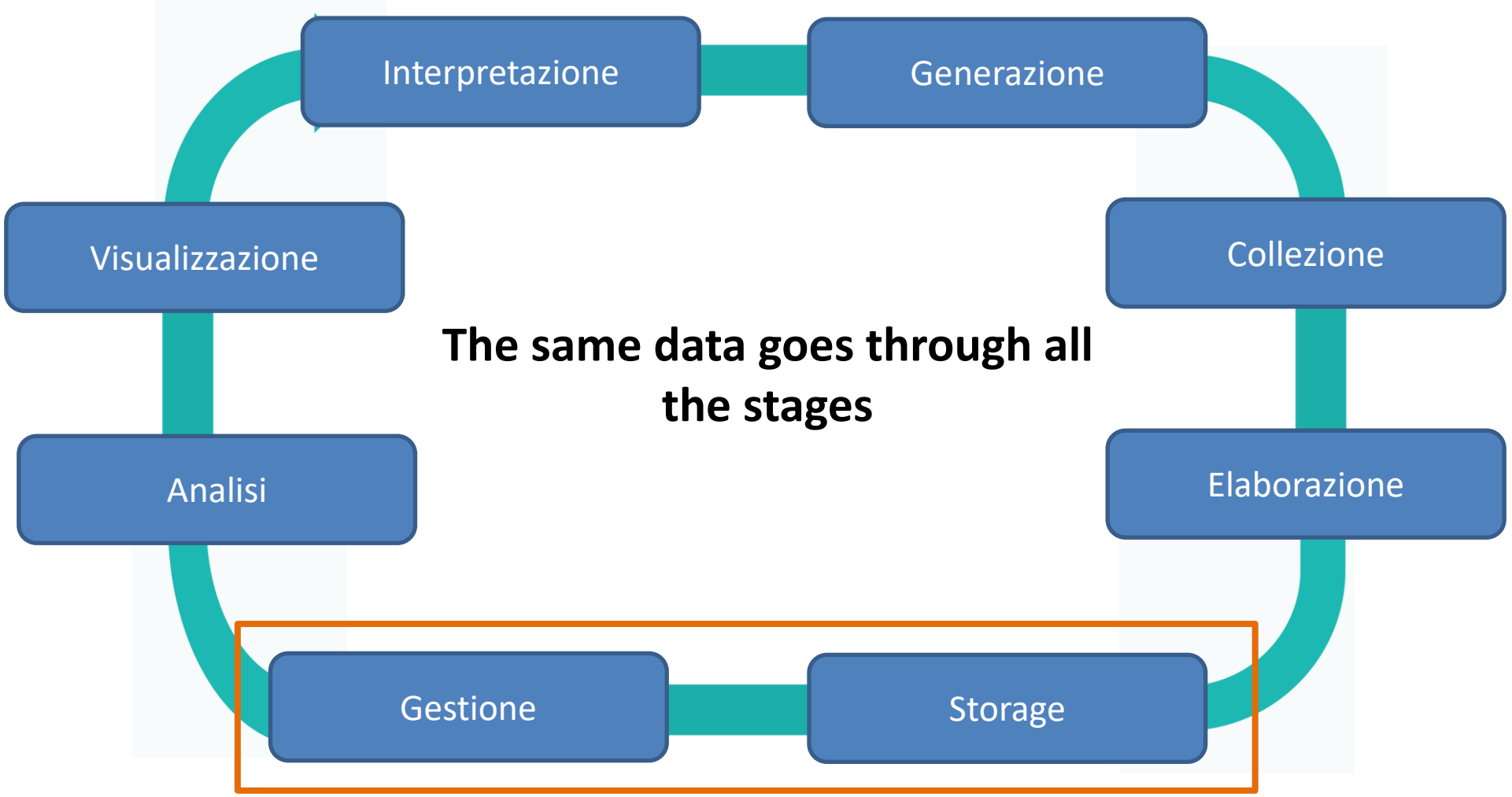
Lorenzo Stacchio

Studente di dottorato in Computer Science

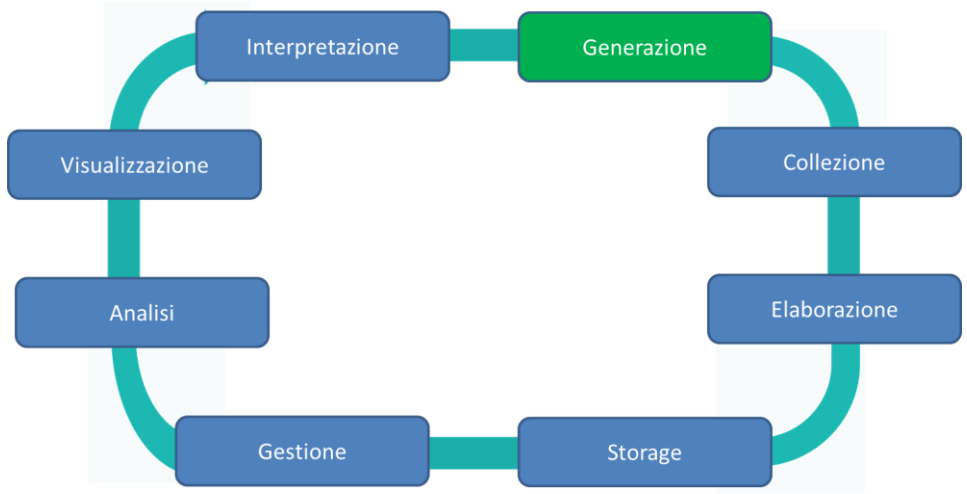
Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il ciclo dei (big) data

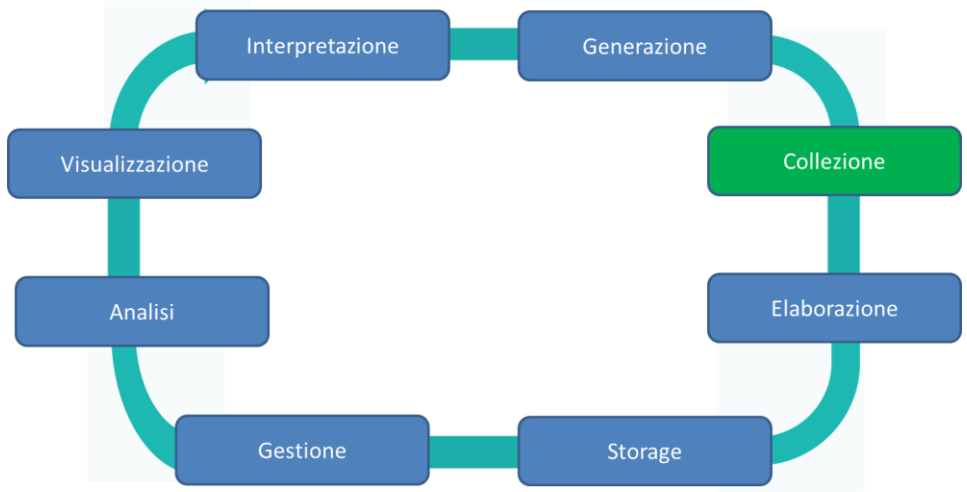


Generazione



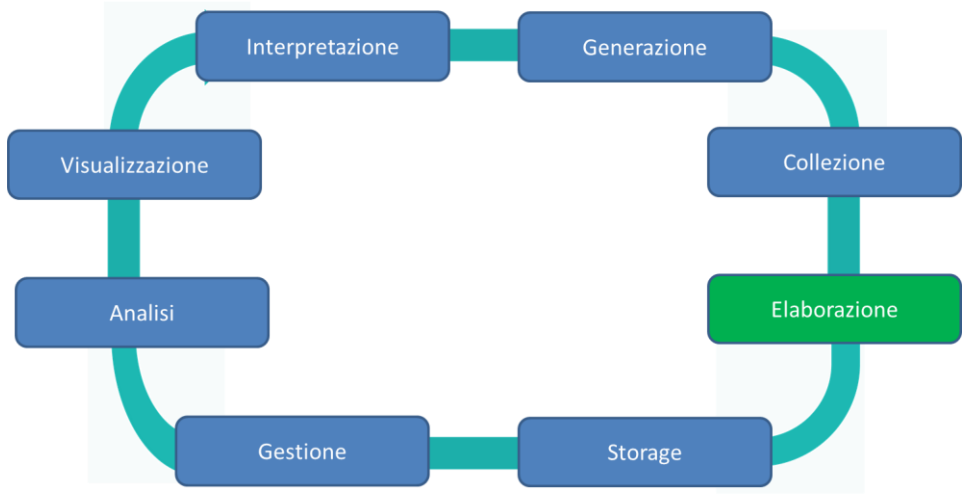
- Affinché il ciclo di vita dei dati abbia inizio, i dati devono prima essere **generati**;
- La generazione di dati avviene indipendentemente dal fatto che tu ne sia consapevole;
- Alcuni di questi dati sono generati dalla tua scuola, università, azienda e da terze parti di cui potresti essere a conoscenza o meno ;
- Ogni vendita, acquisto, noleggio, comunicazione, interazione: **tutto genera dati**;
- I dati generati vengono detti **grezzi.**

Collezione



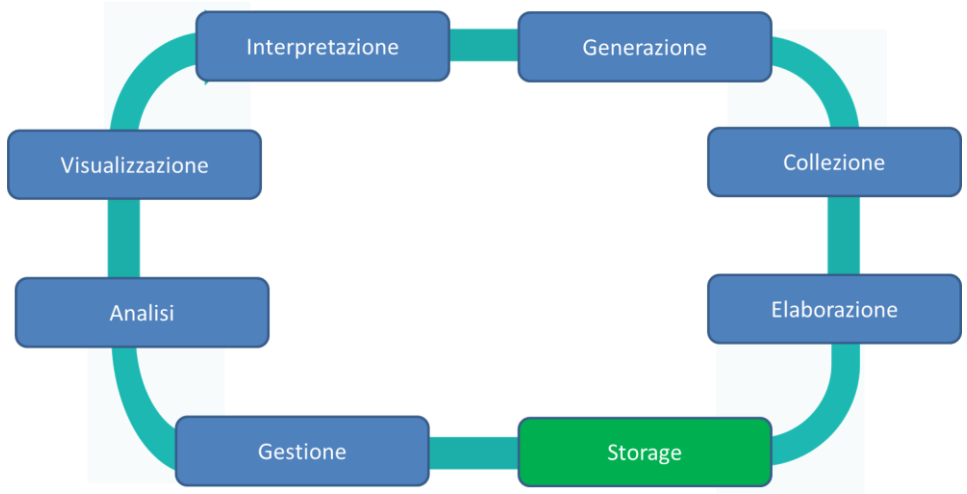
- Non tutti I dati generati ogni giorno vengono però **collezionati o usati**;
- A seconda dello scopo che vogliamo ottenere decideremmo le informazioni da tenere;
- I dati possono essere collezionati in vari modi, come ad esempio:
  - **Sondaggi e interviste**: I sondaggi e le interviste possono essere un modo efficace per raccogliere grandi quantità di informazioni da un gran numero di soggetti;
  - **Osservazione e misurazione**: osservare con l'occhio umano eventi naturali ed etichettarli (es. Cellule normali vs cellule tumorali) oppure usare sensori atti allo scopo (es. misurare la temperatura e umidità)!
- **Capire quali variabili collezionare diventa fondamentale!**

Elaborazione



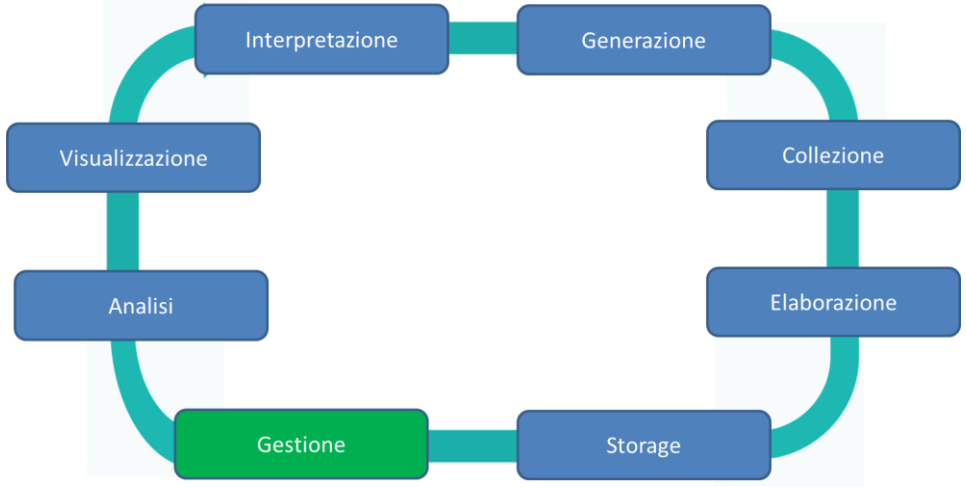
- Una volta che i dati sono stati raccolti, devono essere elaborati. Il trattamento dei dati può riguardare diverse attività, tra cui:
  - **Data wrangling:** processo di pulizia e trasformazione di un dato dalla sua forma grezza in qualcosa di più accessibile e utilizzabile;
  - **Compressione dei dati**, in cui i dati vengono trasformati in un formato che può essere archiviato in modo più efficiente ;
  - **(eventualmente) Crittografia dei dati**, in cui i dati vengono tradotti in un'altra forma di codice per proteggerli da problemi di privacy.

Storage



- Dopo che i dati sono stati raccolti ed elaborati, devono essere archiviati per un uso future;
- Questo processo si implementa attraverso l'uso **di database o set di dati;**
- uesti set di dati possono quindi essere archiviati nel cloud, su server o utilizzando un'altra forma di archiviazione fisica (es. disco rigido);
- Quando si determina come archiviare al meglio i dati per la propria organizzazione, è **importante creare un certo livello di ridondanza** per garantire che una copia dei dati sia protetta e accessibile, anche se la fonte originale viene danneggiata o compromessa;

Gestione

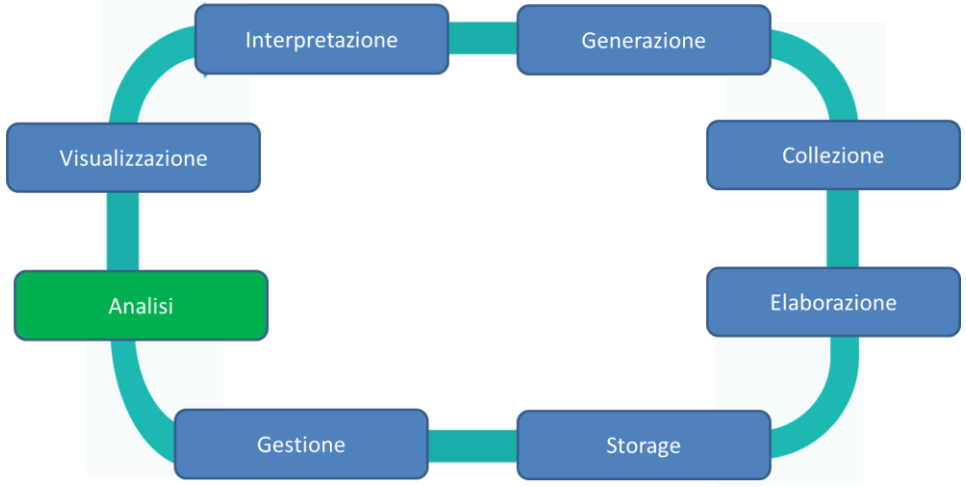


- La gestione dei dati (gestione di database), **implica l'organizzazione, l'archiviazione e il recupero dei dati** secondo necessità durante la vita di un progetto di dati.

- Sebbene qui sia indicato come un "passo", è un processo continuo che si svolge dall'inizio alla fine del ciclo di vita dei big data.

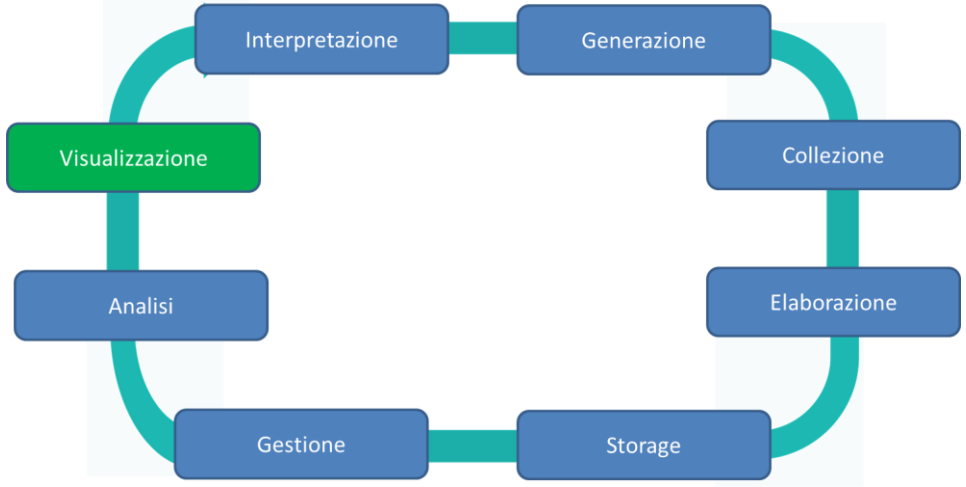
- La gestione dei dati include tutto:
  - Archiviazione ed (eventuale) crittografia;
  - Registri di accesso e modifiche che tengono traccia di chi ha avuto accesso ai dati e delle modifiche effettuate (I cosiddetti **file di log**).

Analisi



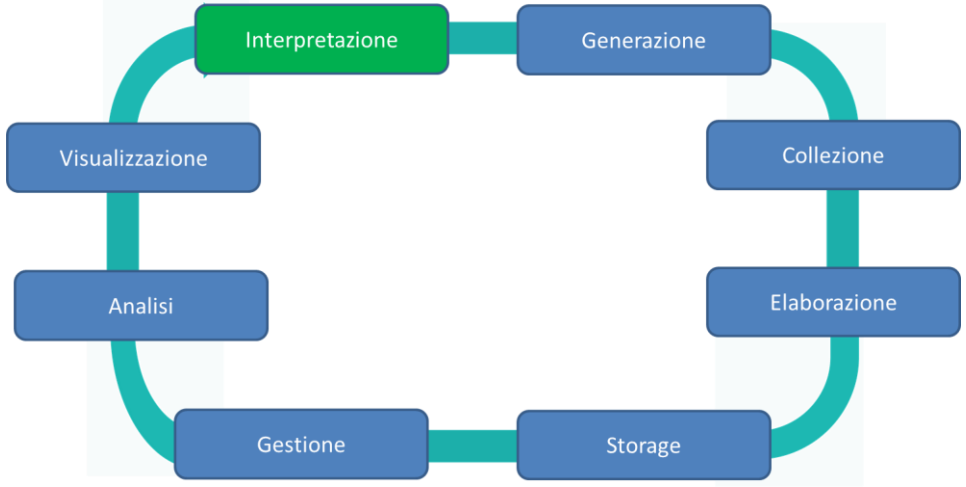
- L'analisi dei dati si riferisce a processi che tentano di raccogliere informazioni significative dai dati grezzi;
- Analisti e data scientist utilizzano diversi strumenti e strategie per condurre queste analisi;
- Alcuni dei metodi più comunemente usati includono **modelli statistici, algoritmi di data mining e machine learning;**
- Le azioni da intraprendere in questa fase, dipendono dalla sfida che stiamo affrontando!
- Ad esempio, se volessimo classificare la tipologia del fiore in base a una foto, di certo non mi metto a studiare un catalogo dell'ikea! **(o forse si?)**

Visualizzazione



- La visualizzazione dei dati si riferisce al processo di creazione di rappresentazioni grafiche delle informazioni, in genere attraverso l'uso di uno o più strumenti di visualizzazione;
- La visualizzazione dei dati semplifica la comunicazione rapida della tua analisi ad un pubblico più ampio;
- La forma che assume la visualizzazione dipende dai dati con cui stai lavorando e dalla storia che vuoi comunicare;
- Sebbene tecnicamente non sia un passaggio obbligatorio per tutti i progetti di dati, la visualizzazione dei dati è diventata una parte sempre più importante del ciclo di vita dei dati!

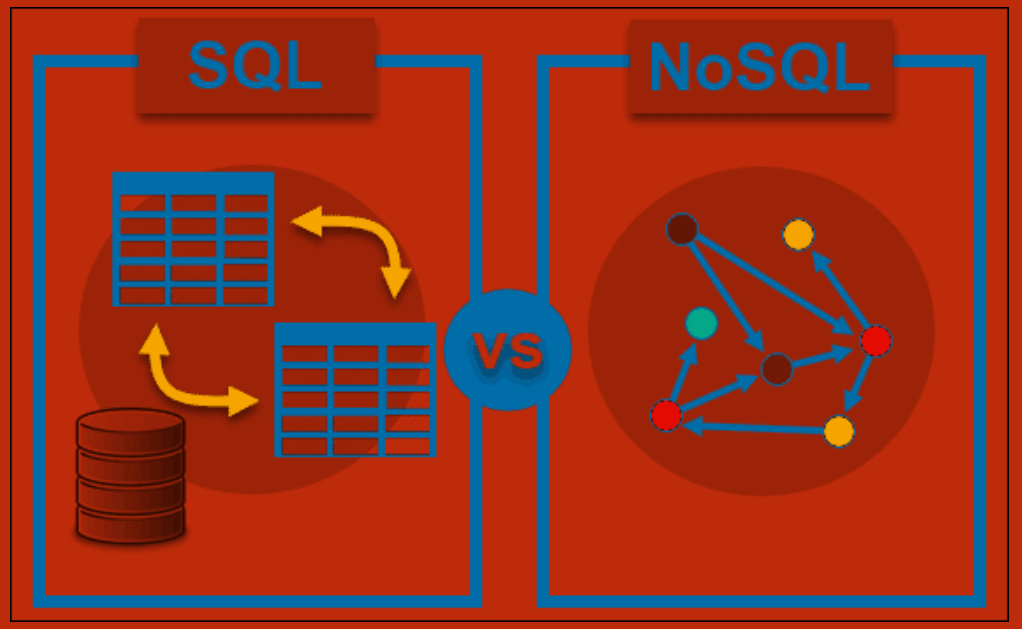
Interpretazione



- Infine, la fase di interpretazione del ciclo di vita dei dati offre l'opportunità di dare un senso all' analisi e alla visualizzazione.

- Oltre a presentare semplicemente i dati, questo step riguarda l'indagazione di tutti gli step precedenti attraverso la lente della propria esperienza (**domain knowledge**);

- **Esempio**: se un algoritmo di Machine Learning validasse un farmaco come **attivo contro le cellule tumorali**, la nostra cara **Valentina** (farmacologa) **potrebbe sintetizzarla e testarla in laboratorio**!



Database SQL e NOSQL

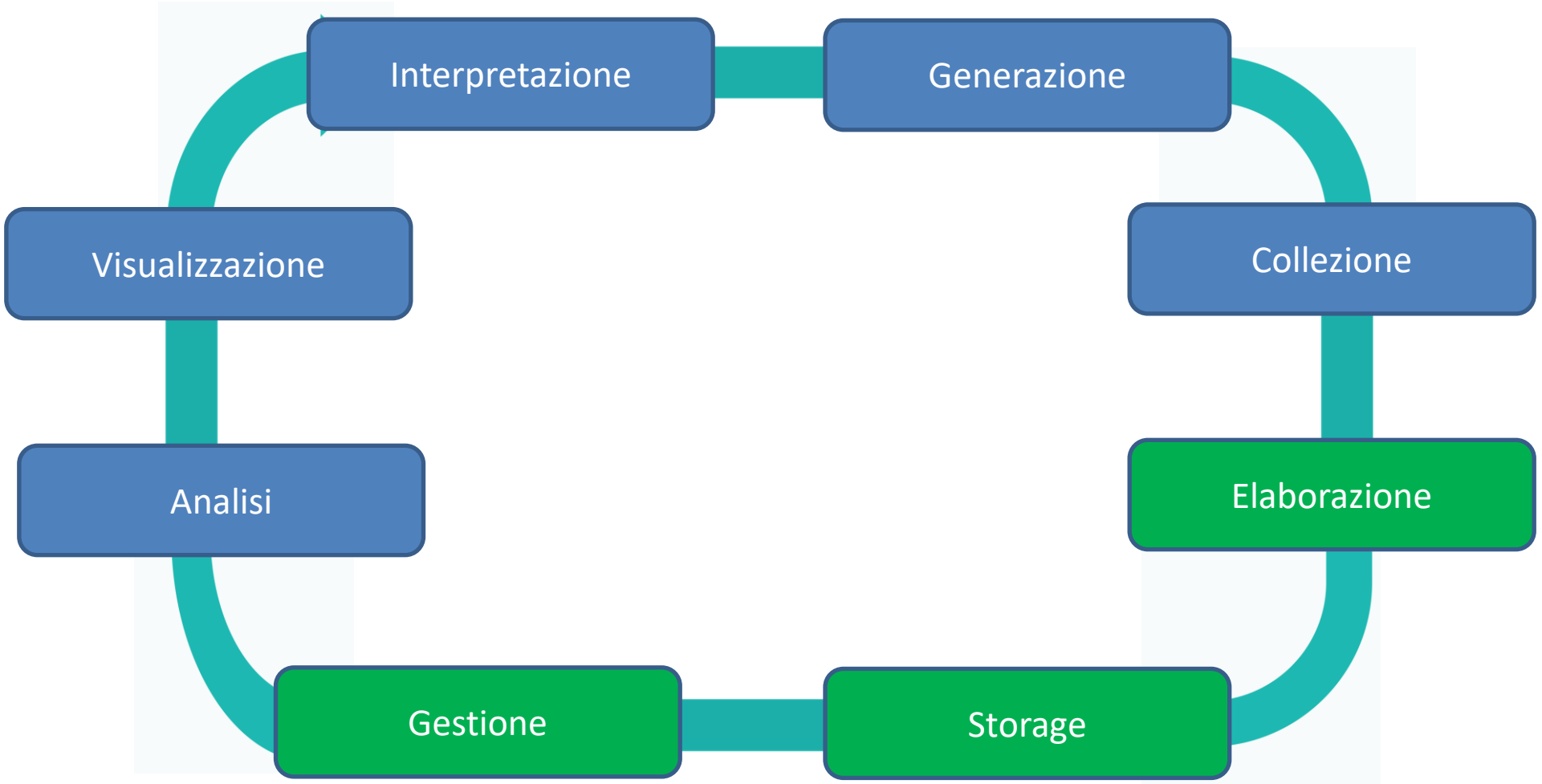
Lorenzo Stacchio

Studente di dottorato in Computer Science

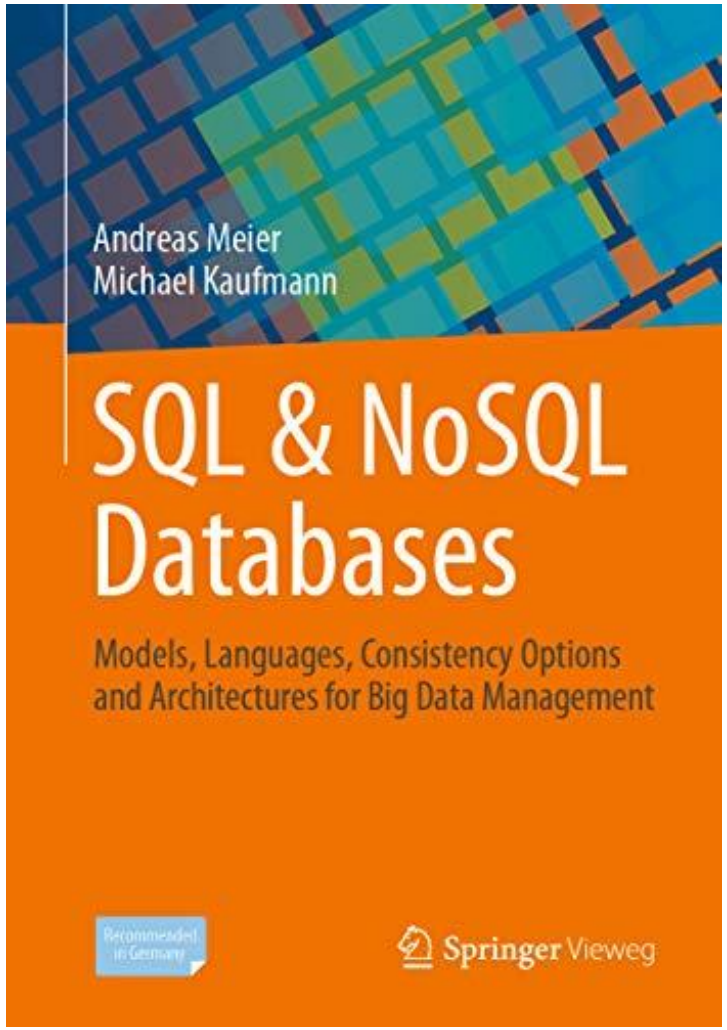
Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Cosa approfondiremo in questa lezione?

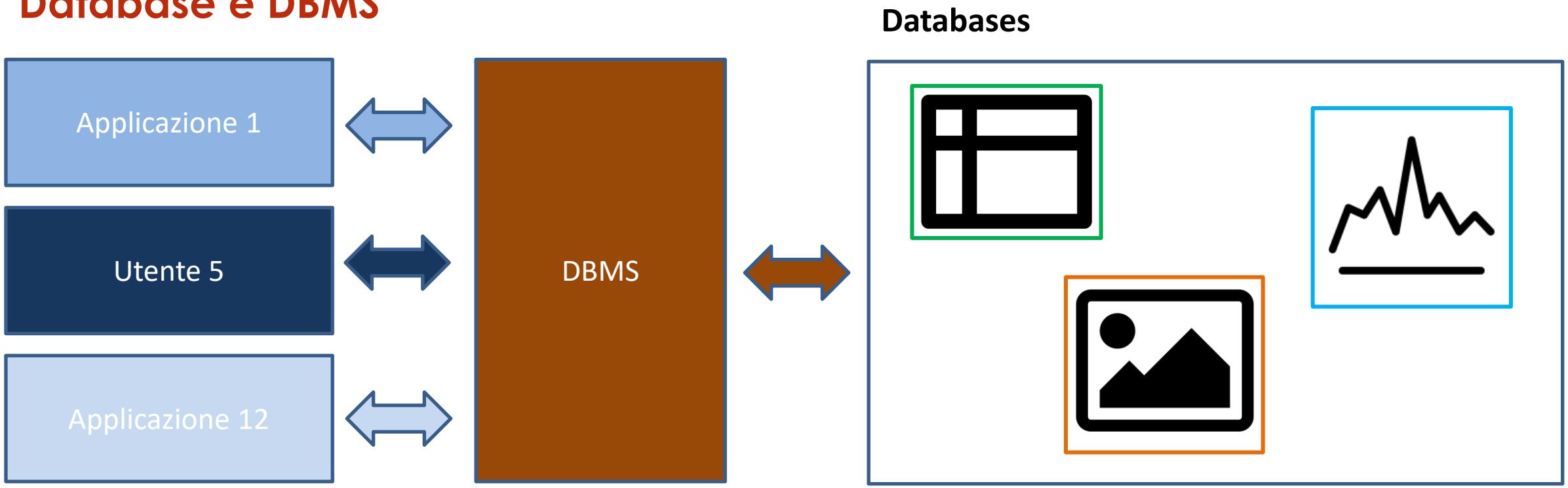


Materiale di approfondimento



- Capitolo 1 e 2;
- Purtroppo ancora in inglese;

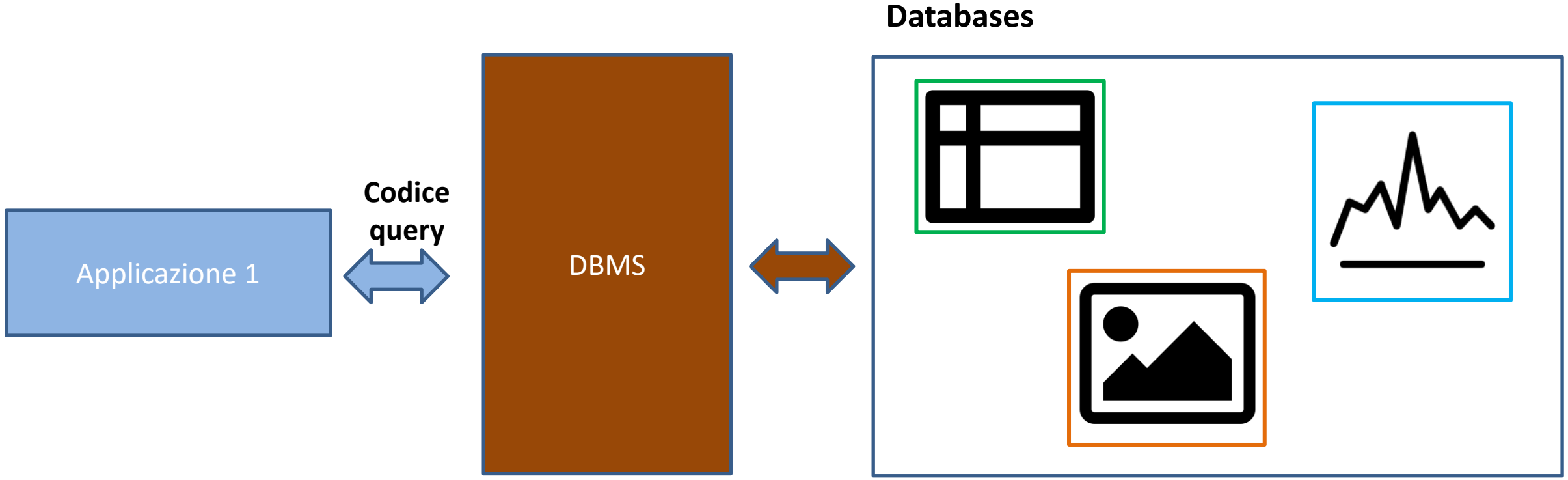
Database e DBMS



- **I database, ossia una collezione organizzata di dati;**
- Ci sono poi dei software che permettono la gestione dei **database** chiamati **database management system (dbms):** i dbms permettono a utenti, applicazioni e al database stesso di interagire con i dati all'interno del database, permettendone il salvataggio e l'analisi;
- Assieme, database e dbms creano quello che viene definito **sistema database**;

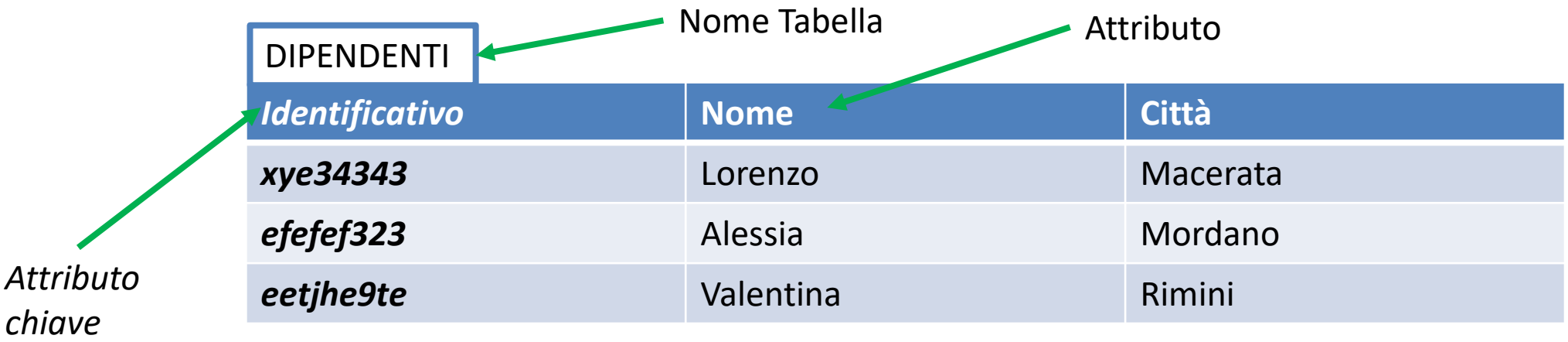
DBMS e linguaggi di query

- Tutti i sistemi di gestione del database contengono le funzionalità per archiviare i dati e per gestirli;
- La parte di archiviazione include tutti i dati salvati più la loro descrizione.
- La parte di gestione contiene un linguaggio cosiddetto di **query, ossia un linguaggio** per manipolare, valutare o modificare i dati e le informazioni all'interno del database;
- Formalmente, un linguaggio di query è un **linguaggio di programmazione specializzato per la ricerca e la modifica dei contenuti di un database**.
- I linguaggi di query moderni come il **SQL** sono linguaggi generali per l'interazione con il **DBMS**, che permettono di :
  - Definire e modificare lo schema del database;
  - Popolare il contenuto di il database;
  - Ricercare i contenuti del database;
  - Aggiornare i contenuti del database;
  - …

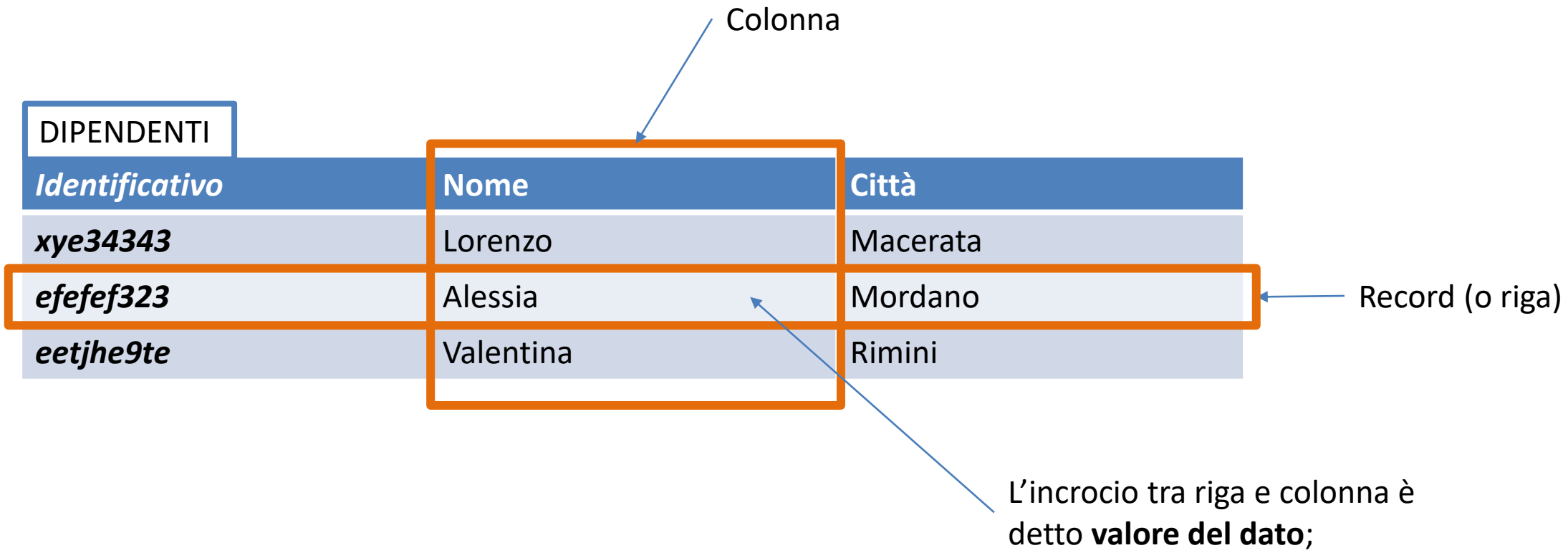


Database SQL-Tabelle

- Uno dei modi più semplici e intuitivi per raccogliere e presentare i dati è in una tabella.
- Ad esempio, per raccogliere informazioni sui dipendenti possiamo usare la seguente struttura tabellare;



- Il nome della tabella in maiuscolo **DIPENDENTI** si riferisce all'intera tabella;
- Gli altri nomi invece definiscono gli **attributi** di una tabella. Un attributo è caratterizzato da un certo intervallo di un certo tipo di dato chiamato **dominio.**
- **Ogni riga della tabella, avrà un valore di quel dominio assegnato** (questo per tutti gli attributi);
- Nella tabella DIPENDENTI, l'attributo «***Identificativo***» consente di **identificare i singoli dipendenti univocamente;**



- Le informazioni richieste dei dipendenti possono ora essere facilmente inserite riga per riga;

- Nelle colonne, i valori possono apparire più di una volta (es. due Lorenzi);

- Questo vale per tutti gli attributi, tranne che per **l'attributo chiave;**
- città, ma anche i nomi dei dipendenti possono esistere più volte. Per questo motivo, il suddetto
- l'attributo chiave E# è necessario per identificare in modo univoco ciascun dipendente nella tabella.

- **L’attributo chiave** deve infatti servire per identificare univocamente una riga;

- La chiave, può essere **un attributo** o una **combinazione** di attributi (es. tutti gli attributi che compongono il codice fiscale);

- Questa breve definizione ci permette di dedurre due importanti proprietà delle chiavi:

    - **Unicità**: ogni valore chiave identifica in modo univoco un record all'interno della tabella;
    - **Minimalità**: se la chiave è una combinazione di attributi, questa combinazione deve essere **minimale**, cioè, nessun attributo può essere rimosso dalla combinazione senza eliminare l’identificazione univoca.

- Anche **il nome della tabella è univoco;**

- **Il nome dei singoli attributi è univoco all’interno della stessa tabella;**

- Non abbiamo bisogno di introdurre nessun ordinamento nelle **colonne e nelle righe**;

Database SQL - Definizione formale di Tabella e modello Relazionale

- Per riassumere, una tabella o **relazione** è un insieme di tuple dove:
    - Il nome della tabella è univoco.
    - I nomi degli attributi sono tutti univoci all'interno di una tabella ed assegnano a una colonna un certo tipo di valori;
    - Non c'è nessun ordinamento nelle **colonne e nelle righe**;

- Da questa definizione, si definisce il **modello relazionale** come un modello che rappresenti sia i **dati che le relazioni tra i dati come tabelle.**

- Matematicamente parlando, qualsiasi relazione **R** è semplicemente un sottoinsieme di un **Prodotto cartesiano** di domini: $R \subseteq D1 \times D2 \times \ldots \times Dn$, dove, ad esempio, D1 si definisce il dominio del primo attributo;

- Qualsiasi tupla $r \subseteq R$ è, quindi, un insieme di valori o manifestazioni di dati specifici r = (d1, d2, …, dn).

- Una qualsiasi tupla esiste solo una volta all'interno di qualsiasi tabella (**c'è sempre la chiave che distingue le righe!**).

Database SQL – Structured query language (SQL)

- Come spiegato, il modello relazionale presenta le informazioni in forma tabellare, dove ogni tabella è un insieme di tuple (o record) dello stesso tipo.
- **Vedere tutti i dati come insiemi rende possibile l'implementazione di un linguaggio di query per la manipolazione basate su operazioni insiemistiche;**
- Ad esempio, il risultato di **un'operazione** di **selezione** è **un insieme**, infatti ogni risultato di ricerca è restituito dal DBMS come tabella; Se non ci sono tuple che posseggano certe proprietà otterremo una tabella vuota;
- Le operazioni di **manipolazione lavorano con gli insiemi** e influiscono su un'intera tabella o alcune sezioni;
- Il linguaggio principale di query e manipolazione dei dati per le tabelle si chiama Structured Query Language, solitamente abbreviato in SQL
- SQL è definito come linguaggio descrittivo, poiché le **istruzioni descrivono il risultato desiderato** invece delle operazioni da eseguire per effettuare un calcolo;

Il costrutto Select – From – Where

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini
efrgr4233	**Gustavo**	**Bologna**

Query di esempio:
«SELEZIONA il NOME dei DIPENDENTI che abitano a BOLOGNA»

In SQL:

```
SELECT Nome
FROM DIPENDENTI
WHERE Città==Bologna
```

Nome
Lorenzo
Gustavo

SELECT Nome
FROM DIPENDENTI
WHERE Città==Bologna

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini
efrgr4233	**Gustavo**	**Bologna**

SELECT Nome
FROM DIPENDENTI
WHERE Città==Bologna

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**
efrgr4233	**Gustavo**	**Bologna**

SELECT Nome
FROM DIPENDENTI
WHERE Città==Bologna

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**
efrgr4233	**Gustavo**	**Bologna**

Tabella di ritorno!

Nome
Lorenzo
Gustavo

Il costrutto Insert Into

Query di esempio:
«**INSERISCI** dentro **DIPENDENTI Gustavo** con id **efrgr4233** che abita a **Bologna**»

In SQL:

INSERT INTO DIPENDENTI
VALUES (efrgr4233, Gustavo, Bologna);

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**

DIPENDENTI

Identificativo	Nome	Città
xye34343	**Lorenzo**	**Bologna**
efrgr4233	**Gustavo**	**Bologna**

Database relazione a più tabelle

- Un database relazionale è di solito formato da più di una tabella!

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

RUOLO

Identificativo	Nome	Stipendio
xye34343	Barista	1300
efefef323	Cameriere	1300
eetjhe9te	Cuoco	1300

- Come vedete, entrambe le tabelle hanno degli identificativi comuni!
- Posso quindi costruire una relazione tra queste due tabelle!
- L'insieme di tabelle e le relazioni tra esse definisce **lo schema del database**;

Relazioni tra tabelle

- Posso costruire una **relazione** che leghi la tabella DIPENDENTI con quella dei RUOLI;
- Una relazione viene stabilita quando il valore **della chiave da un lato della relazione è paragonabile a un valore nel campo di confronto dall'altro lato della relazione**, in base ai criteri specificati nella relazione;
- Esistono vari tipi di relazioni: **1 a 1**, **1 a n** ed **n ad n;**
- **In questo caso**, ogni elemento della tabella DIPENDENTI è associato unicamente al record della tabella RUOLO. Per cui questa è una relazione **1 a 1!**

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

1 — 1

RUOLO

Identificativo	mansione	Stipendio
xye34343	Barista	1300
efefef323	Cameriere	1300
eetjhe9te	Cuoco	1300

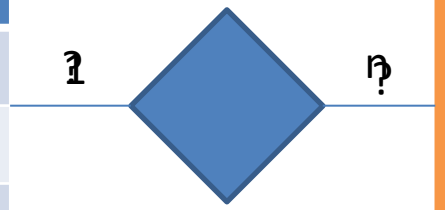
Il gioco delle relazioni

- Con quanto detto finora, supponiamo di avere due tabelle:
  - La prima tabella identifica come al solito i dipendenti;
  - La seconda identifica gli stipendi ricevuti da ogni dipendente durante tutta la sua carriera;

- Che relazione avranno queste due tabelle?

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini



STIPENDI

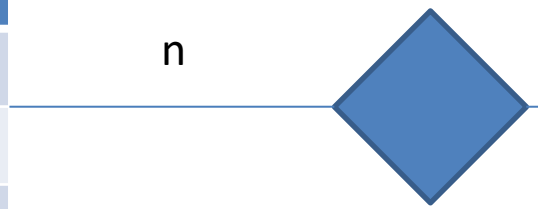
Identificativo	Stipendio ricevuto	Data
xye34343	1300	21/08/2020
xye34343	1300	21/06/2020
eetjhe9te	1300	21/02/2020

Relazione n a n

- Una relazione "molti a molti" si verifica quando più record in una tabella sono associati a più record in un'altra tabella.
- Ad esempio, tra clienti e prodotti è presente una relazione "molti a molti": i clienti possono acquistare diversi prodotti e i prodotti possono essere acquistati da diversi clienti;
- Per collegare le due tabelle ci sarà bisogno di **una terza tabella**!

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini



PRODOTTI

Identificativo	Nome
efes5r43	Piastra
18454	Carne
23232	gioco

Unire due tabelle sulla chiave (inner join)

- Sapendo che esistono delle relazioni tra tabelle viene però da chiedersi quale sia il modo per visualizzare informazioni provenienti da esse!

- Ad esempio: come faccio a sapere lo stipendio di Lorenzo? (**tabella 1 a 1**)

Query di esempio:
«**UNISCI** le tabelle **DIPENDENTI** e **RUOLO** e dimmi quanto prende si **stipendio** il dipendente **Lorenzo**»

In SQL:

SELECT RUOLO**.**stipendio
FROM RUOLO
JOIN DIPENDENTI **ON** DIPENDENTI**.** ***Identificativo*** **=** RUOLO**.** ***Identificativo***
WHERE *DIPENDENTI.Nome = Lorenzo*

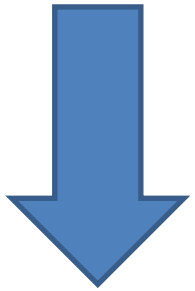
DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

1 — 1

RUOLO

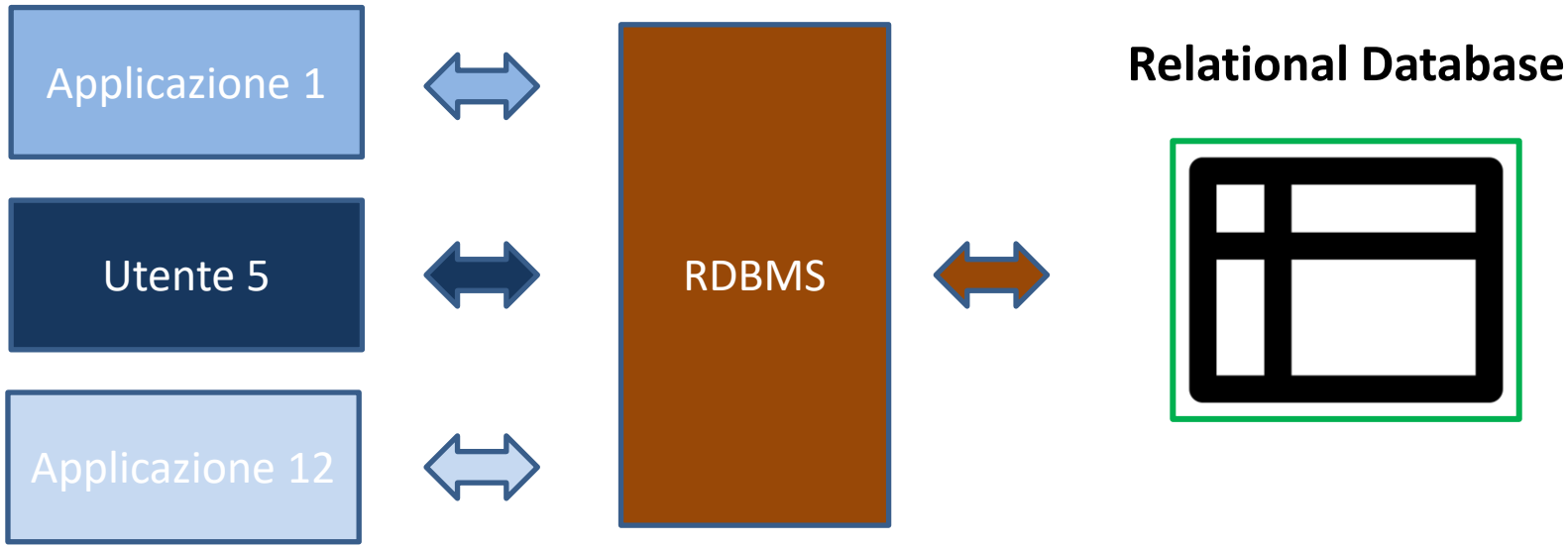
Identificativo	mansione	Stipendio
xye34343	Barista	1300
efefef323	Cameriere	1300
eetjhe9te	Cuoco	1300



RUOLO

Identificativo	mansione	Stipendio	Nome	Città
xye34343	Barista	1300	Lorenzo	Macerata

RDBMS - Relation Database Management system



Gli RDBMS, hanno le seguenti proprietà:

- Gestiscono database che **seguono il modello relazionale;**
- La definizioni di tabelle, **relazioni tra tabelle**, e attributI sono salvate nello schema;
- Utilizzano il **SQL** per la definizione, selezione e manipolazione dei dati;
- Fornisce **indipendenza** dei dati (ossia, dati e applicazioni sono completamente separate, grazie al DBMS);
- Permette operazioni **multi-utente**: **query di modifica** si possono essere effettuate **in parallelo solo su database diversi**;
- **Fornisce consistenza massima e protezione** dei dati, ad esempio, solo i dati corretti e non compromessi vengono salvati.

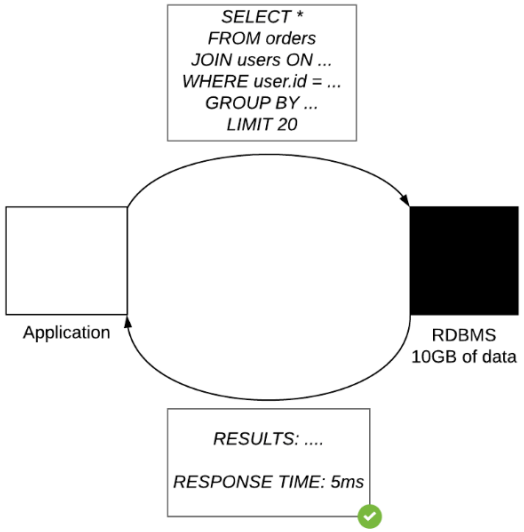
Il teorema ACID

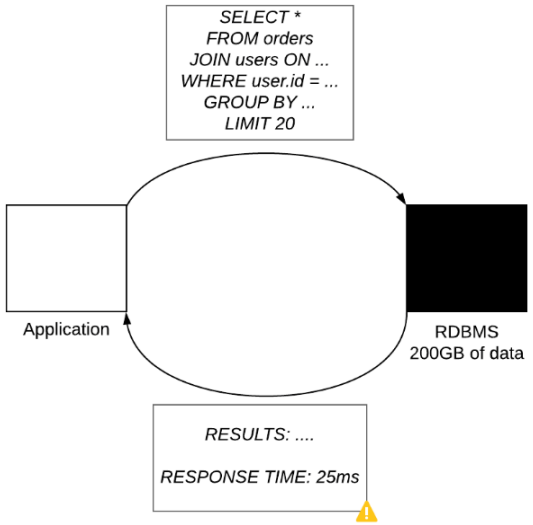
- I database relazionali sottostanno alla legge **ACID**:
  - **Atomicità**: il processo deve essere suddivisibile in un numero finito di unità indivisibili, chiamate **transazioni**. L'esecuzione di una transazione perciò deve essere per definizione o totale o nulla e non sono ammesse esecuzioni parziali;
  - **Consistenza**: il database rispetta i vincoli di integrità, in qualunque momento della **transazione**. Non devono verificarsi contraddizioni (incoerenza dei dati) tra i dati archiviati nel DB;
  - **Isolamento**: ogni transazione deve essere eseguita in modo **isolato e indipendente** dalle altre transazioni, **l'eventuale fallimento di una transazione non deve interferire con le altre transazioni in esecuzione**;
  - **Durabilità**: detta anche persistenza, si riferisce al fatto che una volta che una transazione abbia richiesto un lavoro di impegno (es. bonifico bancario), i cambiamenti apportati non dovranno essere più persi.

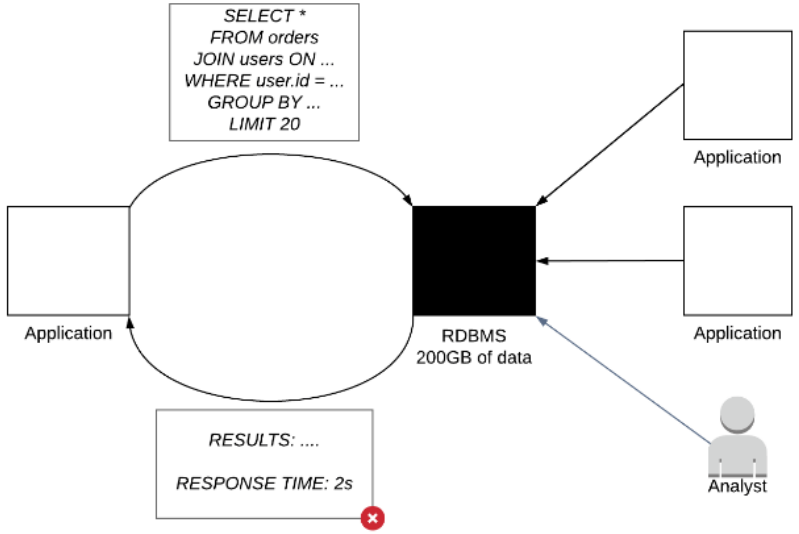
Il teorema ACID è la scalabilità

- La **scalabilità** è la misura della capacità di un sistema di aumentare o diminuire le prestazioni in risposta ai cambiamenti nelle applicazioni e nelle richieste di elaborazione del sistema;
- Gli esempi includono, tra i tanti, **la capacità di un database di resistere a un numero crescente di query;**
- Di base, il teorema ACID non intacca la scalabilità nelle operazioni di **lettura semplici** di un database relazionale. Anche avendo migliaia di lettori posso sempre trovare un modo per **parallelizzare la lettura semplice**;
- **Cosa succederebbe** se migliaia di persone provassero a scrivere all'interno dello stesso database?
- **Cosa succederebbe** se migliaia di persone provassero ad effettuare richiesta di **letture complesse** all'interno dello stesso database?

Database relazionali e letture complesse







Fonte: https://www.alexdebrie.com/posts/dynamodb-no-bad-queries/

Il problema vero è l'ACD

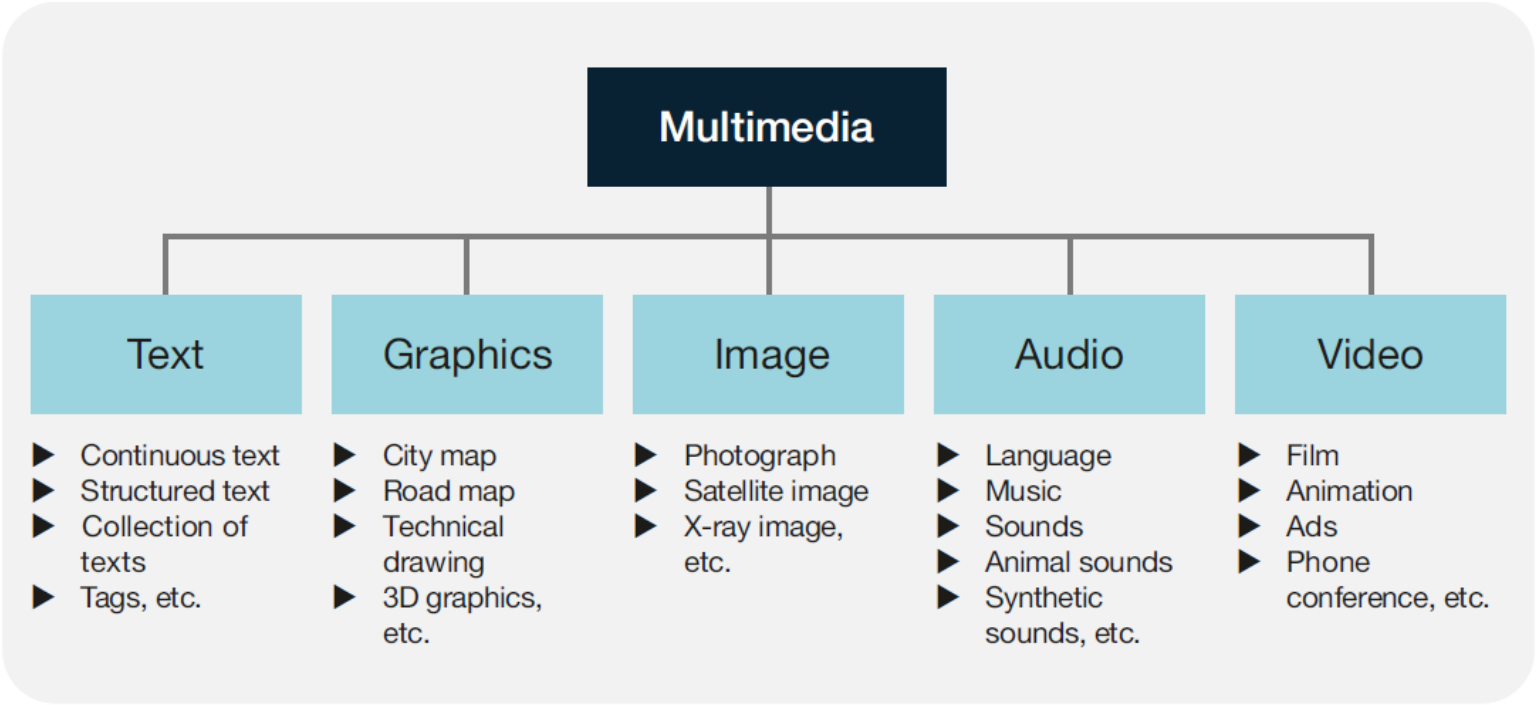
- I database relazionali sottostanno alla legge **ACID. La scalabilità in scrittura non può essere garantita per colpa dell'ACD**:
  - **Atomicità**: richiede che solamente **un processo** possa effettuare operazioni di scrittura nel database (il processo deve essere portato a termine o fallire completamente);
  - **Consistenza:** I vincoli di coerenza indicano che tutti i nodi nel cluster devono essere identici. Se si scrive su un nodo, questa scrittura deve essere copiata su tutti gli altri nodi prima di restituire una risposta al client. Ciò rende difficile la scalabilità di un cluster RDBMS tradizionale;
  - **Durabilità:** I vincoli di durata indicano che per non perdere mai una scrittura è necessario assicurarsi che prima che una risposta venga restituita, la scrittura sia stata scaricata su disco.
- Per scalare dovremmo:
  - **Eliminare** l'atomicità, consentendo di ridurre la durata del blocco dei dati;
  - **Eliminare** la coerenza, consentendo di aumentare le scritture tra i diversi database;
  - **Eliminare** la durabilità, consentendo di rispondere ai comandi di scrittura senza salvare ogni volta lo stato su unità di memoria fissa;

Nota a latere: il mondo dei big data è in continua evoluzione

- I database SQL, non sono adatti a gestire i big data primariamente per la loro impossibilità nello scalare;
- Tuttavia, c'è anche un'altra ragione: il mondo dei big data è in continua evoluzione e abbiamo bisogno di strutture elastiche e pronte ai cambiamenti!
- Una volta definita la struttura del database, quella rimane invariata o richiede ore/giorni/settimane per essere aggiornata!
- La maggior parte dei database NOSQL non è solo scalabile ma è anche elastica ai cambiamenti!
- Infine, i database relazionali nascono per essere **centrali** mentre i database nosql sono **altamente distribuibili**;

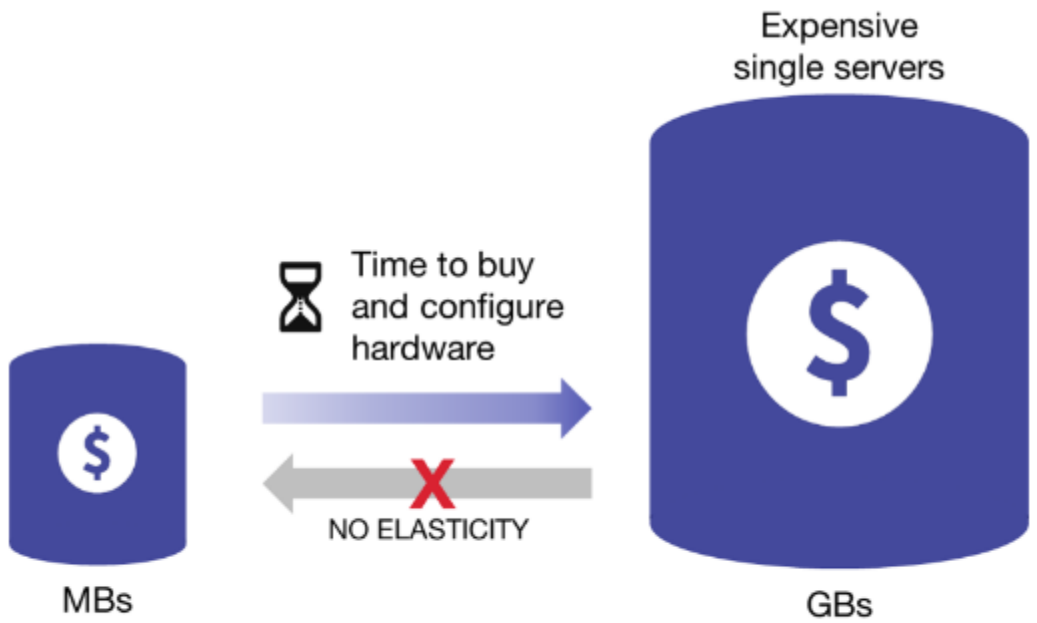
I database SQL non sono adatti ai big data

- La necessità di sviluppare una nuova forma di database nasce dai big data;
- Riprendiamo le principali 3V: *volume* (tanti dati), *varietà* (formati multipli, dati strutturati, semi-strutturati e non strutturati…**figura sotto**) e *velocità* (velocità in produzione ed elaborazione).



Volume is a problem

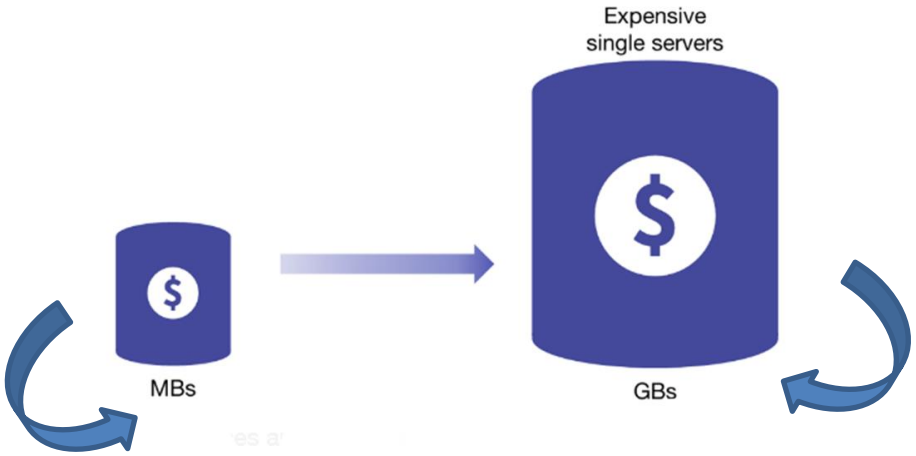
- Perché il volume è un problema? I database relazioni hanno un alto costo di configurazione per scalare anche in termini di memoria!



Relational databases are designed to scale up on expensive single machines

Velocity is a problem

- Perché la velocità è un problema? Come abbiamo visto poco fa, i database relazionali non riescono a scalare in velocità e non riescono a mantenere alte performance all'aumentare del numero dei dati e degli utenti!



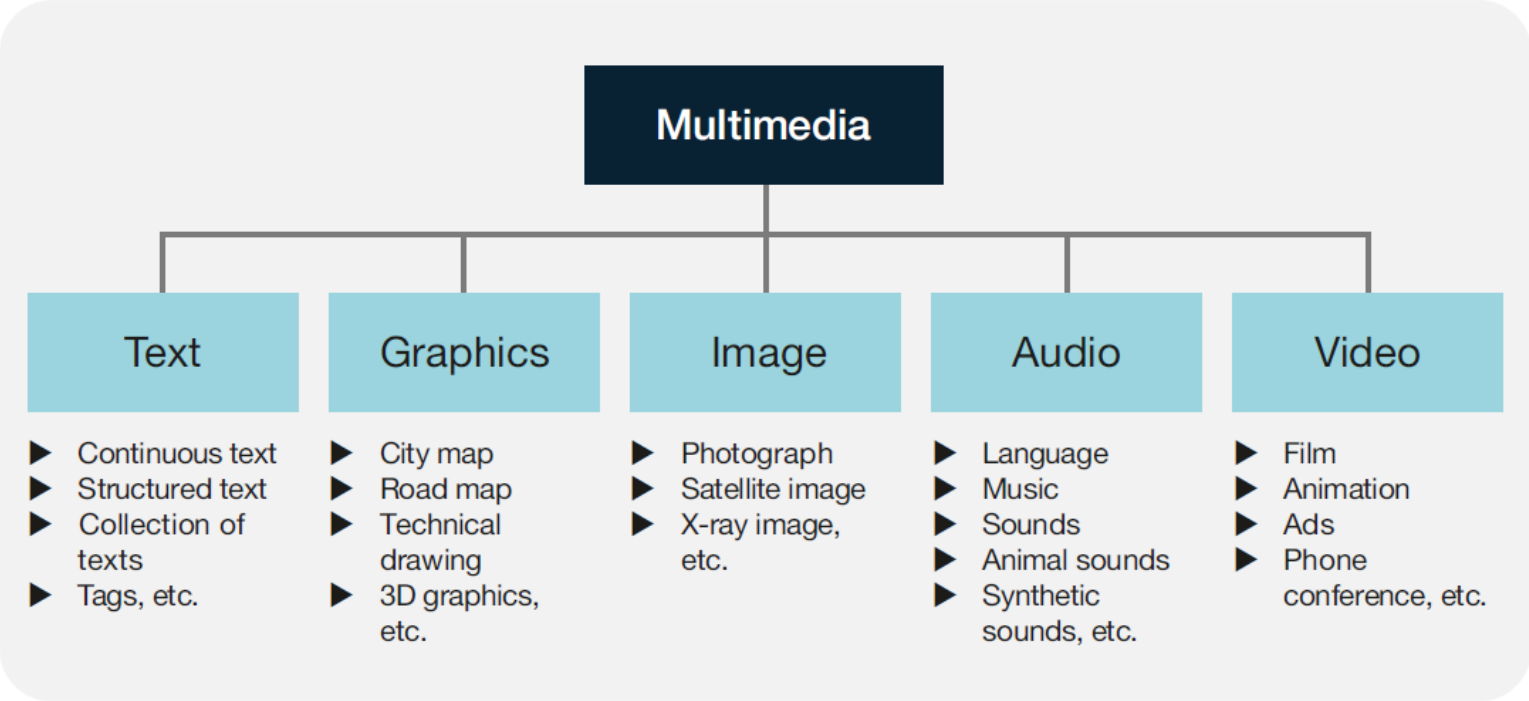
Query time: 2-30 ms

Query time: 2-5 s

Variety is a problem

- Perché la varietà è un problema? Una volta definita, lo schema di un database relazionale rimane com'è e i cambiamenti sono costosissimi.

- Inoltre, i database relazionali **non riescono a gestire tutti i tipi di formati i possibili formati dati (es. non strutturati, semi-strutturati)**;

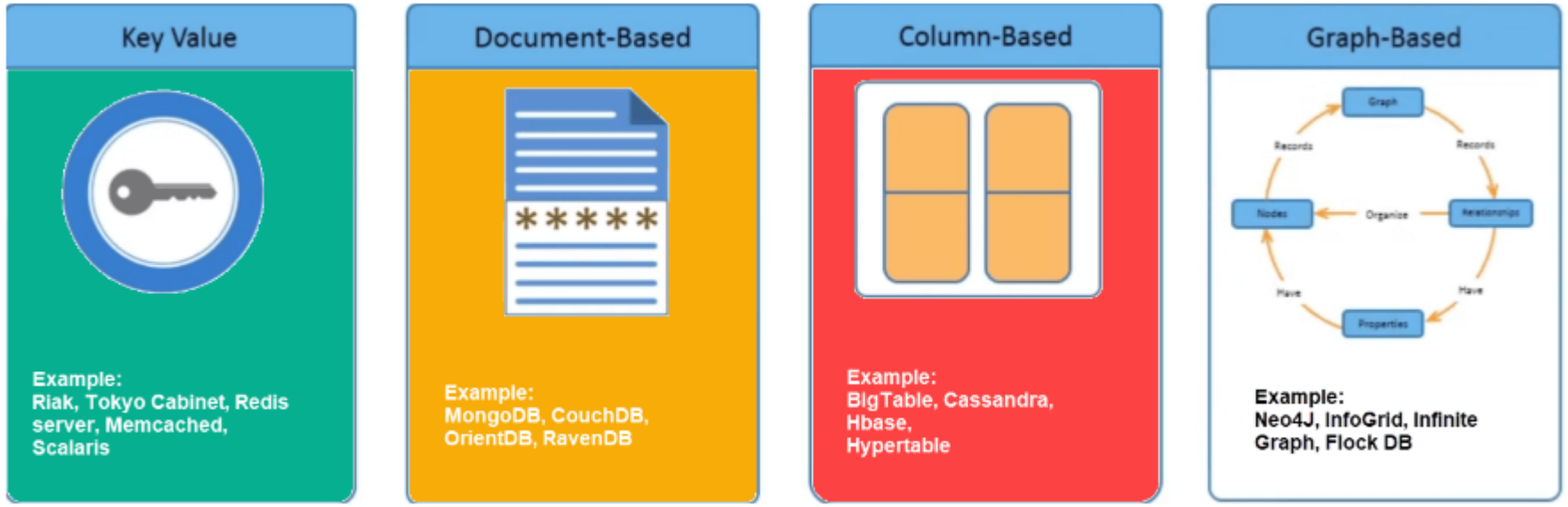


Database NOSQL

- NoSQL Il termine NoSQL è ora utilizzato per qualsiasi gestione di dati non relazionali che soddisfino due criteri:
    - I dati **non** vengono archiviati nelle **tabelle**;
    - Il linguaggio usato per manipolare il database **non** è **SQL**.
- NoSQL viene talvolta interpretato come **"Non solo SQL" per esprimere che altre tecnologie oltre ai database relazionali** vengono utilizzate in applicazioni Web ampiamente distribuite;
- Questa tipologia di database nacque in contesti aziendali, principalmente per soddisfare i seguenti criteri:
    - Gestire volumi di transazioni senza precedenti;
    - Bassa latenza di accesso a insiemi enormi di dati;
    - Fornire **disponibilità del servizio** in maniera ottimale, nonostante l'ambiente possa cambiare in maniera imprevedibile;

Tipi di database NoSql

- I database NoSQL sono principalmente classificati in quattro tipi: **coppie chiave-valore, orientato alle colonne, basato su grafi e orientato ai documenti;**
- Ogni categoria ha attribute, vantaggi e limiti unici;
- Nessuno dei database sopra specificati è migliore per risolvere tutti i problemi;
- Gli utenti dovrebbero selezionare il database in base alle loro esigenze di prodotto;



Database key-value e document - oriented

- I dati vengono archiviati in coppie chiave/valore;
- È progettato in modo tale da gestire molti dati e carichi pesanti;
- Hanno esattamente la stessa forma dei dizionari!

Chiave	Valore
efes5r43	Piastra
18454	Carne
23232	gioco

- Perché sono utili?
  - Permettono di lavorare senza uno schema prefissato: possiamo cambiare lo schema quando vogliamo!
  - Altamente efficienti!
  - Altamente distribuibili!

- I **database document-oriented** sono come i key-value, ma i valori sono dei documenti!

Chiave	Valore
efes5r43	testo.txt
18454	immagine.jpg
23232	…..

Database column oriented

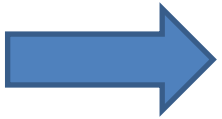
- I database a colonne sono database che **organizzano i dati per campo (o attributo**), mantenendo **in memoria** tutti i dati associati a **campi** vicini;
- I database a colonne sono cresciuti in popolarità e offrono vantaggi in termini di prestazioni per l'interrogazione dei dati;
- Sono ottimizzati per la lettura e il calcolo efficiente **su colonne** (cioè gli attributi);
- Permettono uno schema flessibile e distribuzione dei dati!

TABELLA CLASSICA

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

Aggiungere colonna



- Aggiungere una colonna significa modificare l'intero database!
- Proprio a causa della rigidità dello schema!

COLUMN-ORIENTED

	Nome	
Alessia	Lorenzo	Valentina

	Città	
Mordano	Macerata	Rimini

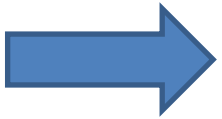
efefef323

TABELLA CLASSICA

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

Aggiungere colonna



- Aggiungere una colonna significa modificare l'intero database!
- Proprio a causa della rigidità dello schema!

COLUMN-ORIENTED

	Nome	
Alessia	Lorenzo	Valentina

	Città	
Mordano	Macerata	Rimini

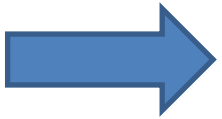
xye34343

TABELLA CLASSICA

DIPENDENTI

Identificativo	Nome	Città
xye34343	Lorenzo	Macerata
efefef323	Alessia	Mordano
eetjhe9te	Valentina	Rimini

Aggiungere colonna



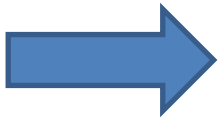
- Aggiungere una colonna significa modificare l'intero database!
- Proprio a causa della rigidità dello schema!

COLUMN-ORIENTED

Nome		
Alessia	Lorenzo	Valentina
Città		
Mordano	Macerata	Rimini

eetjhe9te

Aggiungere colonna



	Nome	
Alessia	Lorenzo	Valentina
	Città	
Mordano	Macerata	Rimini
	Età	
-----	24	----

xye34343

- I sistemi di database NoSQL sono considerati tali se soddisfano i seguenti requisiti:
  - **Il modello di database sottostante non è relazionale;**
  - **Almeno tre V**: il sistema di database include una grande quantità di dati (volume), strutture dati flessibili (varietà) ed elaborazione in tempo reale (velocità);
  - Il **sistema di gestione del database non è vincolato da uno schema di database fisso;**
  - Il sistema di gestione del database **supporta la replica dei dati;**
  - Alcune garanzie di consistenza dei dati: il teorema CAP dice che la consistenza può essere garantita al costo della **disponibilità del servizio** o la **tolleranza ai fallimenti**;

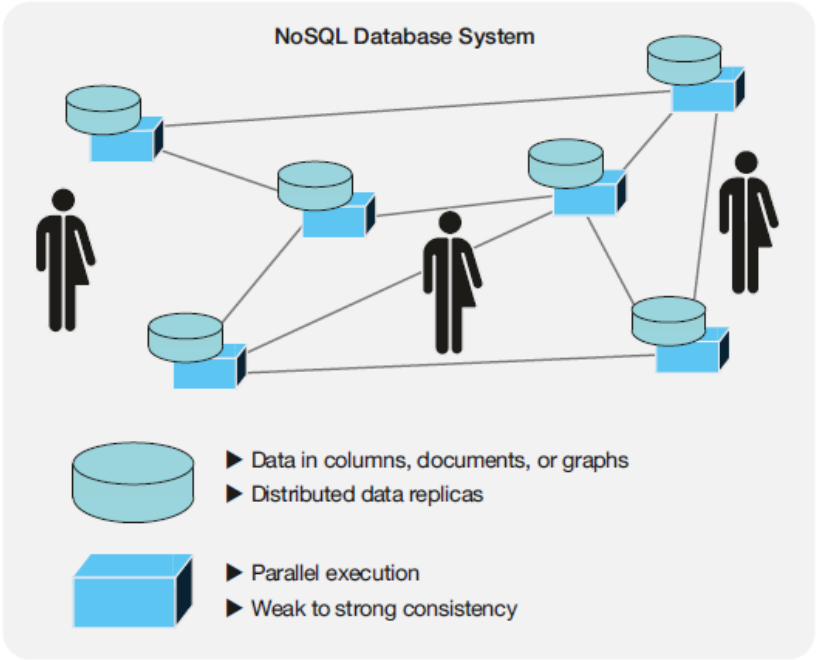
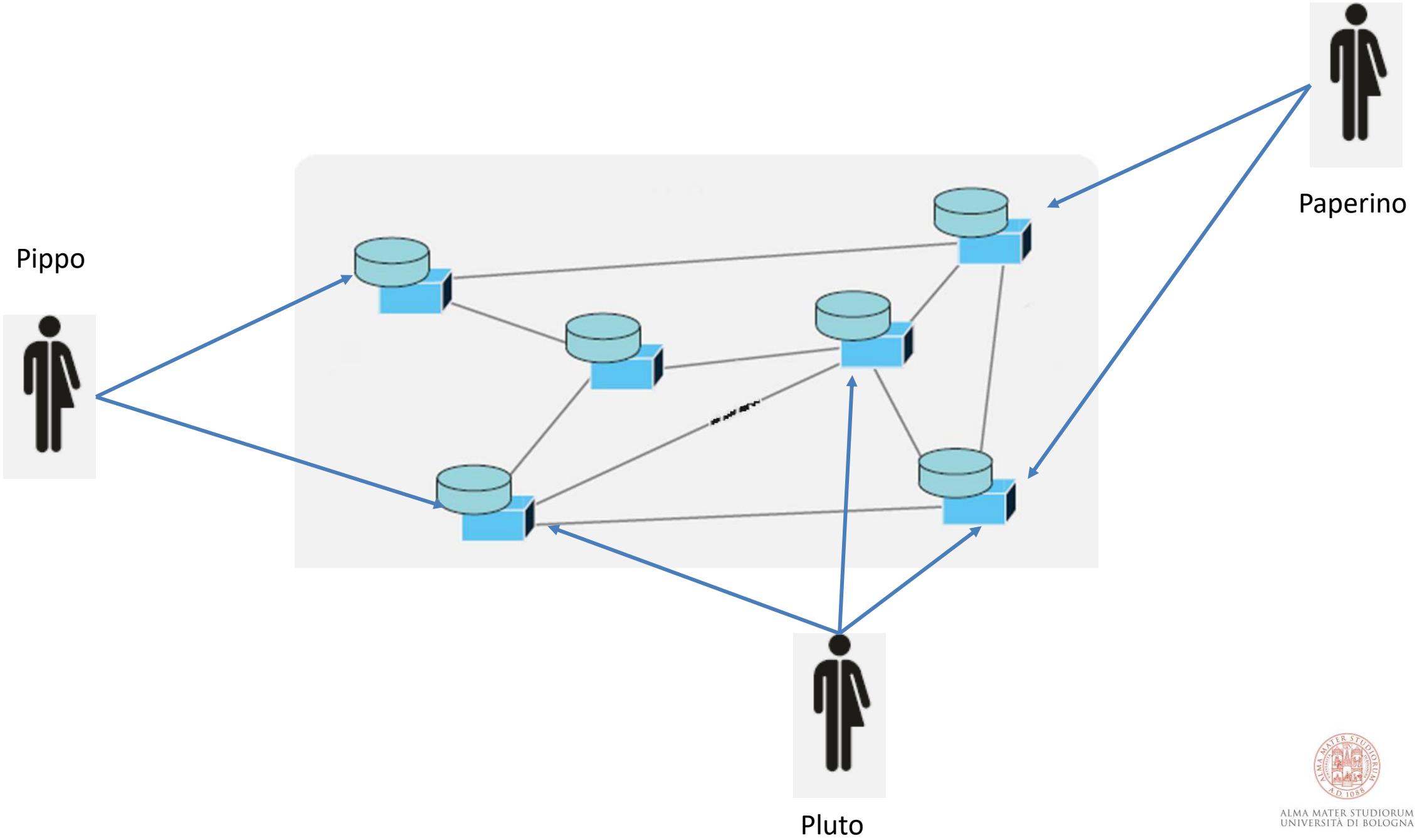
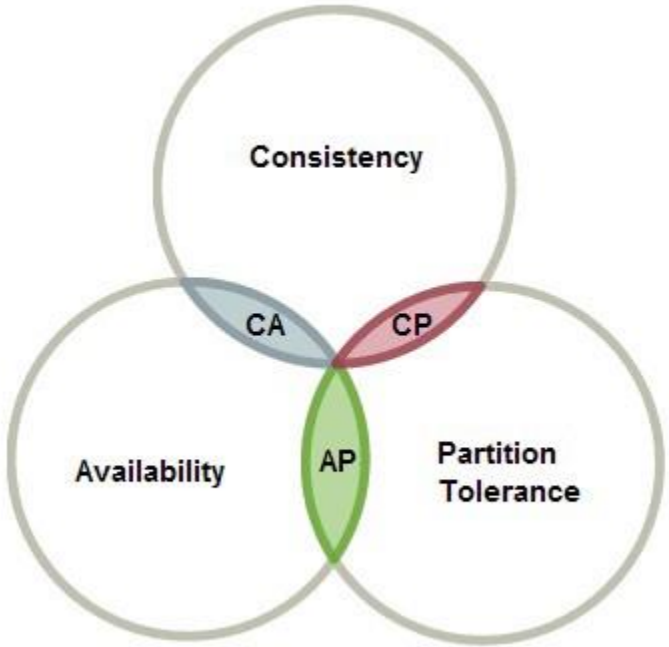


Fig. 1.10 Basic structure of a NoSQL database management system



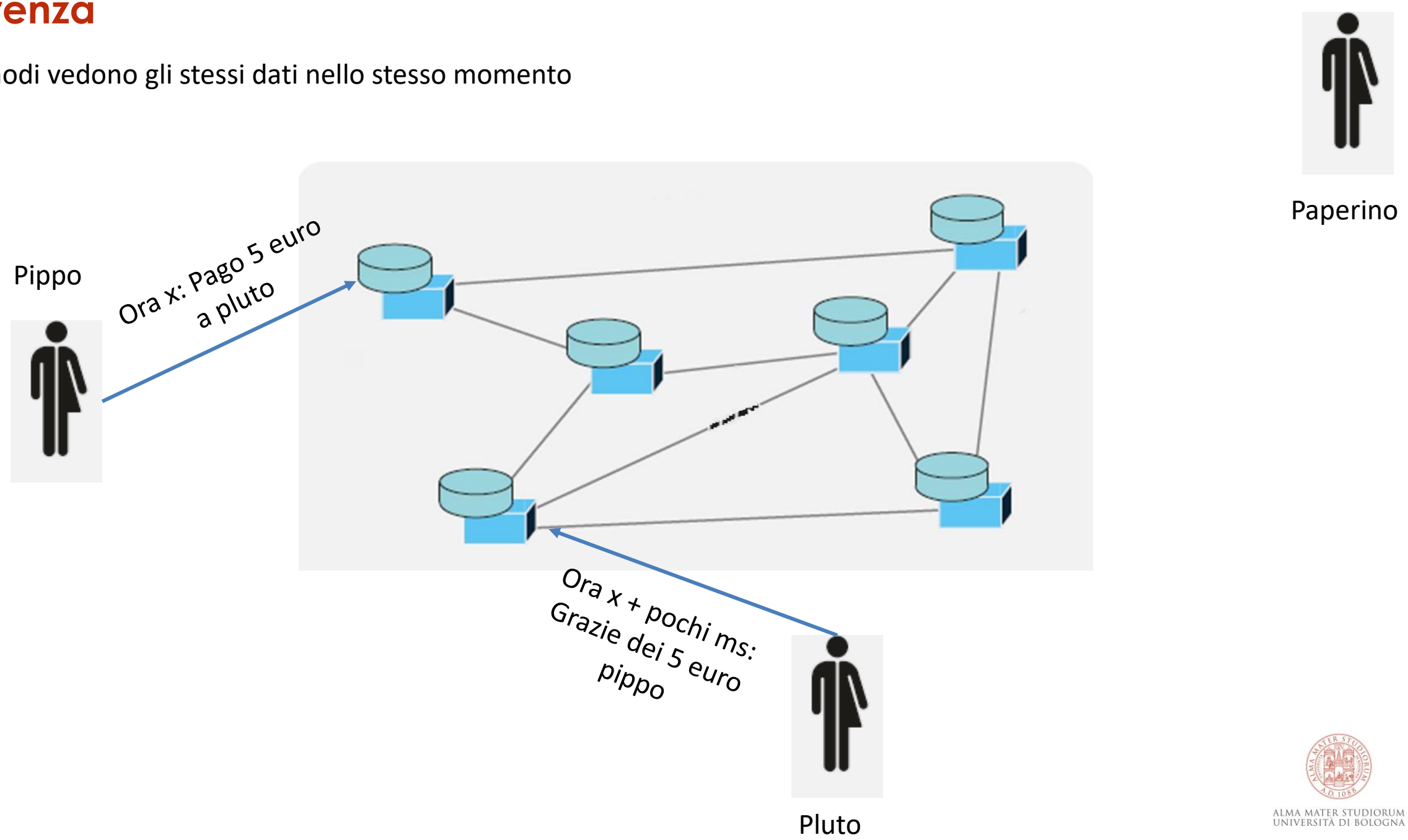
Il teorema CAP

- Il teorema CAP afferma che è impossibile per un **sistema distribuito** fornire simultaneamente tutte e tre le seguenti garanzie:
    - **Coerenza**: tutti i nodi vedano gli stessi dati nello stesso momento;
    - **Disponibilità:** la garanzia che ogni richiesta riceva una risposta su ciò che è riuscito o fallito;
    - **Tolleranza di partizione:** il sistema continua a funzionare nonostante arbitrarie perdite di messaggi o malfunzionamenti;
- Secondo il teorema, un sistema distribuito è in grado di soddisfare al massimo due di queste garanzie allo stesso tempo, ma non tutte e tre!



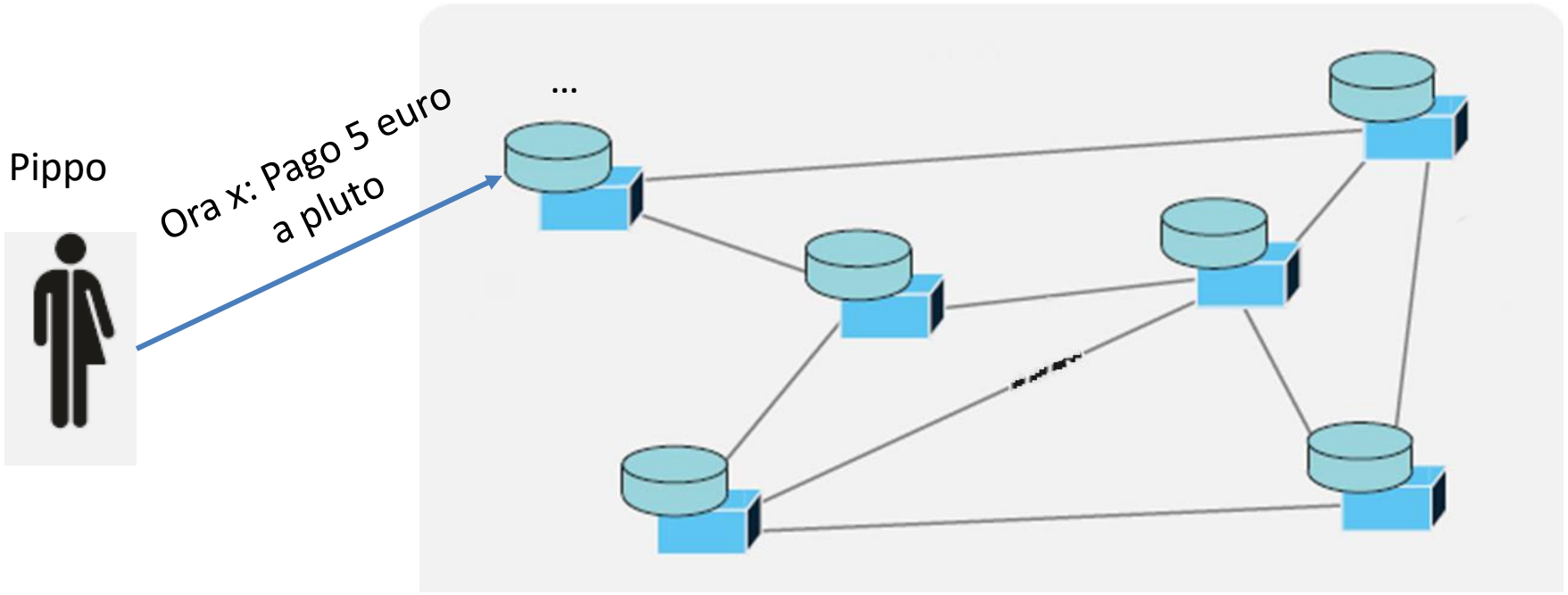
Coerenza

Tutti i nodi vedono gli stessi dati nello stesso momento



Disponibilità

La garanzia che ogni richiesta riceva una risposta su ciò che è riuscito o fallito



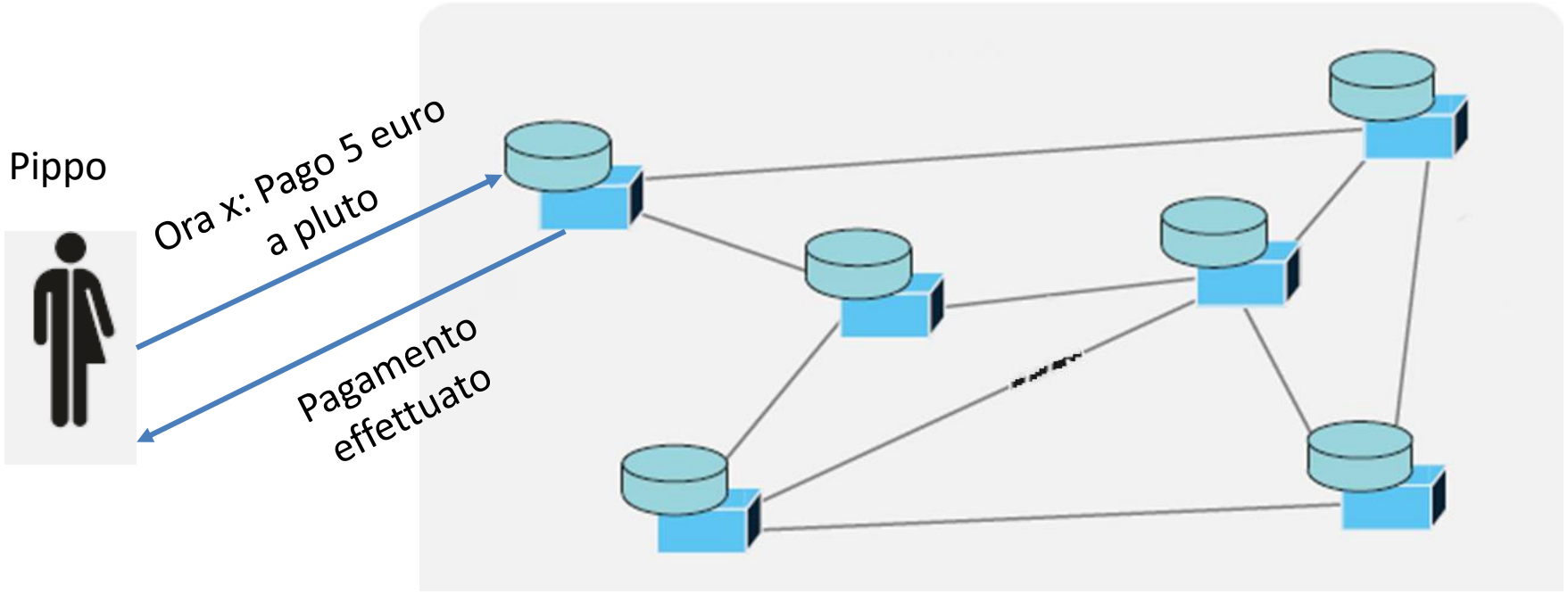
Paperino

Pluto

Disponibilità

La garanzia che ogni richiesta riceva una risposta su ciò che è riuscito o fallito

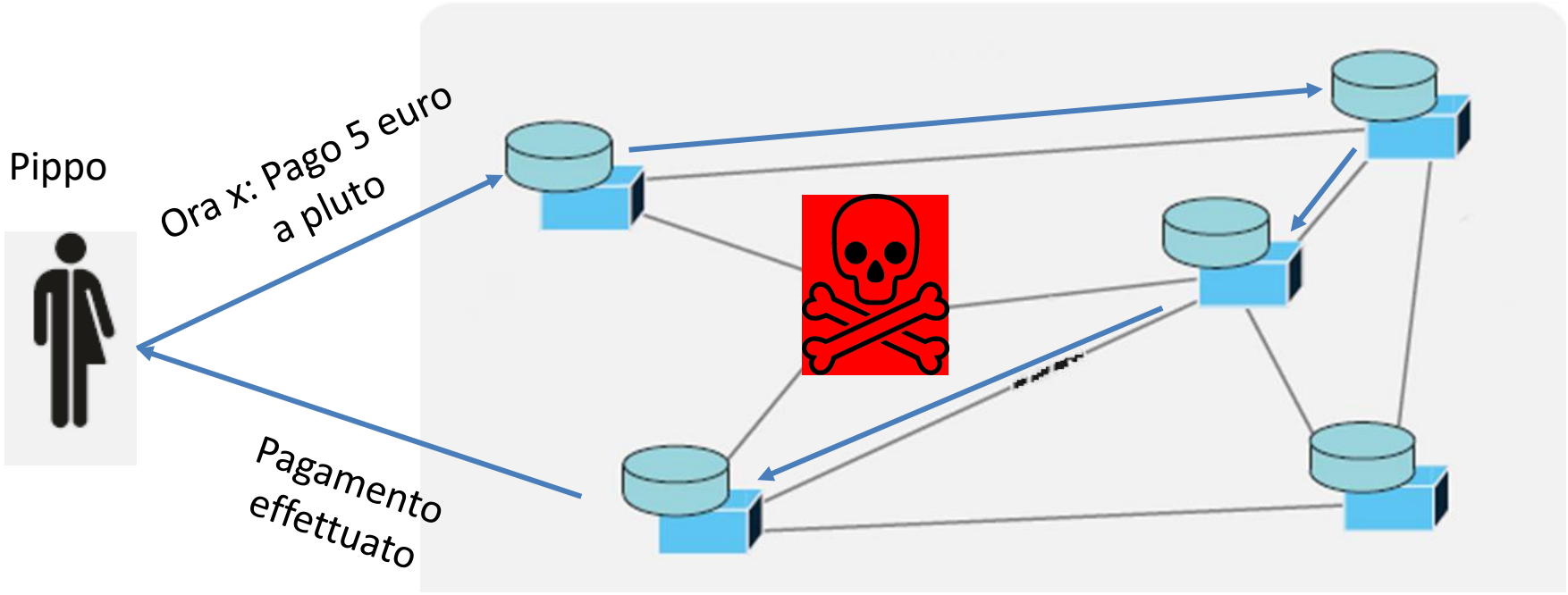
Paperino



Pluto

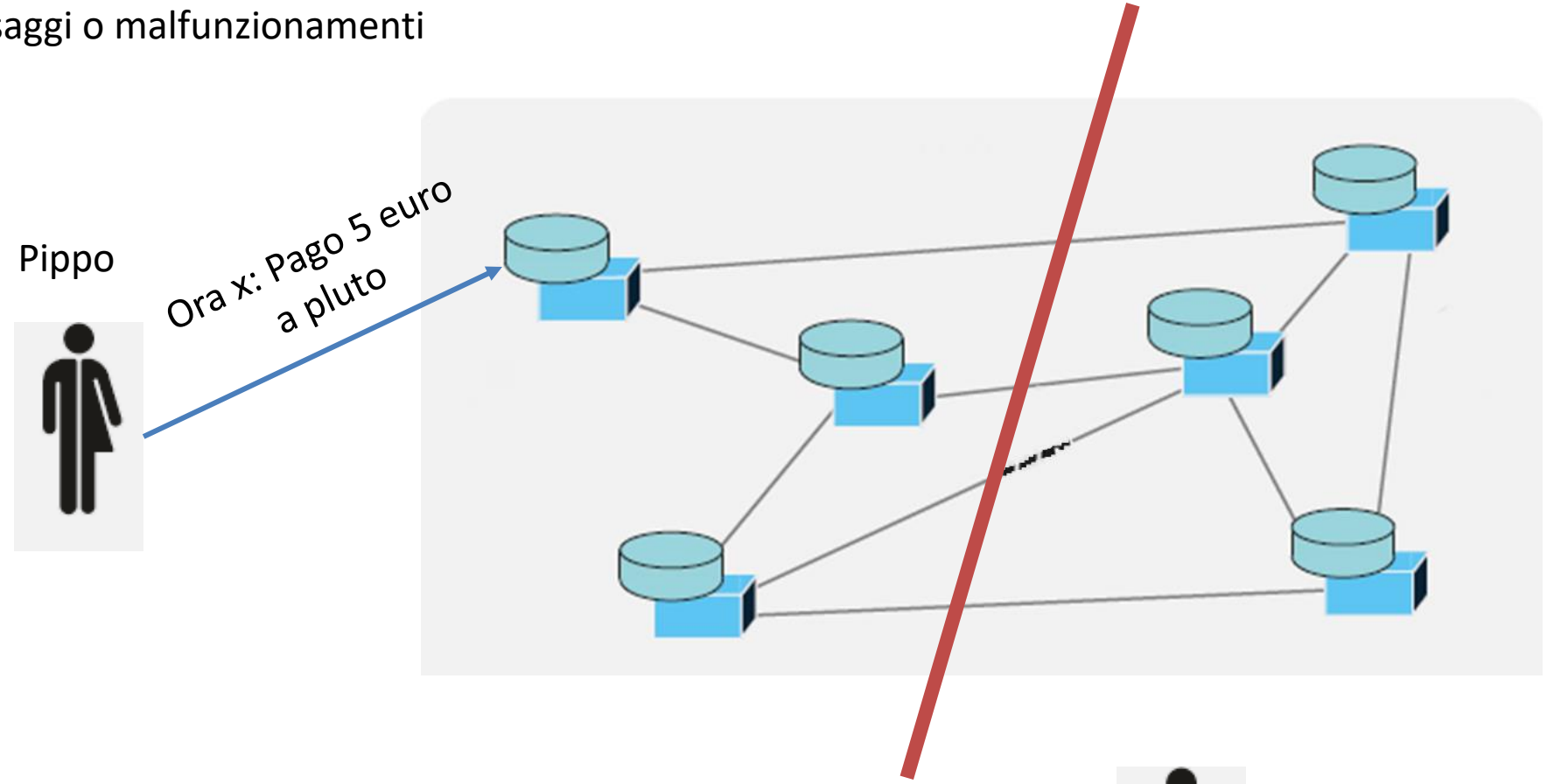
Tolleranza alla partizione

Il sistema continua a funzionare nonostante arbitrarie perdite di messaggi o malfunzionamenti



Tolleranza alla partizione

Il sistema continua a funzionare nonostante arbitrarie perdite di messaggi o malfunzionamenti



Dati strutturati, semi-strutturati e non strutturati

Dati strutturati: sono i dati tipicamente conservati in database relazionali, organizzati secondo schemi e tabelle rigide;

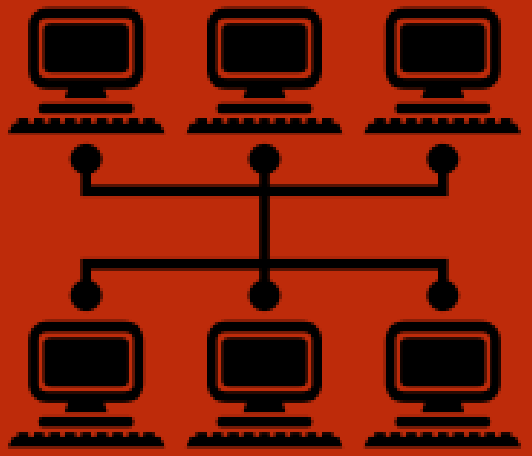
Database relazionali

Dati non strutturati: sono i dati conservati senza alcuno schema. Un esempio possono essere i file contenenti testi a carattere narrativo prodotti per mezzo di uno dei più diffusi software di editing testuale o un file multimediale.

Database nosql

Dati semi-strutturati: nei dati semi strutturati s'incontrano alcune delle caratteristiche dei dati strutturati e alcune delle caratteristiche dei non strutturati. Un esempio esplicativo di questa tipologia di organizzazione di informazioni è il file compilato con sintassi XML. Nonostante non vi siano limiti strutturali all'inserimento dei dati, le informazioni vengono, comunque, organizzate secondo logiche strutturate e interoperabili.

Database relazionali e non



Verso il calcolo distribuito

Lorenzo Stacchio

Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Calcolo monoprocessore

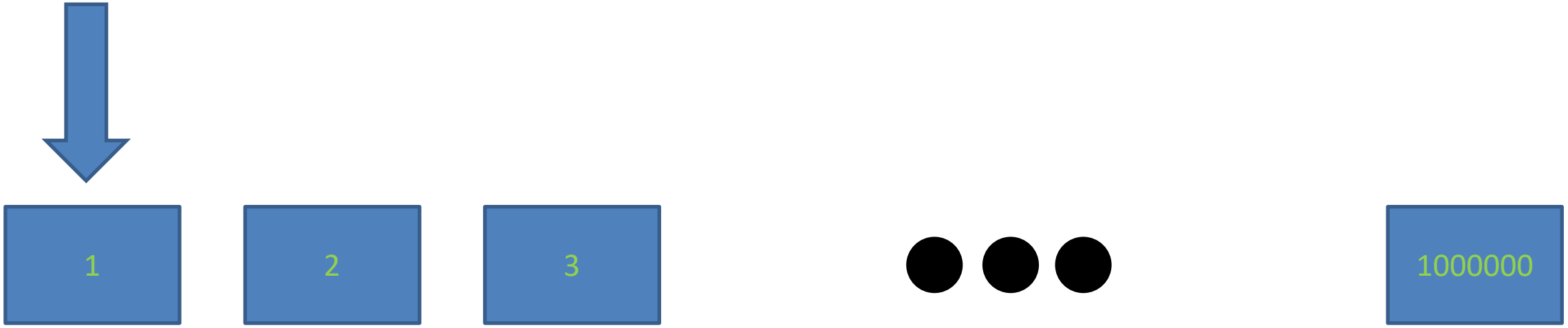
- **Tutta la programmazione fatta in python nella scorsa lezione** era basata sul concetto di sequenzialità o di esecuzione mono-processore;

- In pratica, tutte le istruzioni sono sempre state eseguite una per una sequenzialmente!

- Rifletitamo ora sulla gravità di questa tipologia di computazione per gestire I big data:

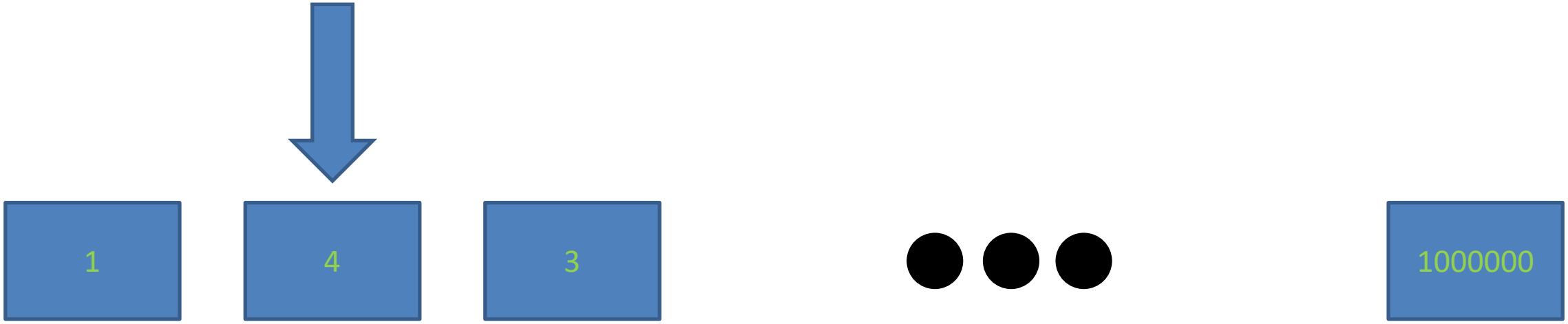
    - Ipotizziamo di avere una tabella di 1000000 righe;
    - Ipotizziamo di dover fare delle operazioni su tutte le righe sequenzialmente come abbiamo sempre fatto!
    - Quanto tempo ci mettiamo?
    - Perchè questa cosa non si adatta ai big data?

Calcolo monoprocessore: esempio quadrato di una lista



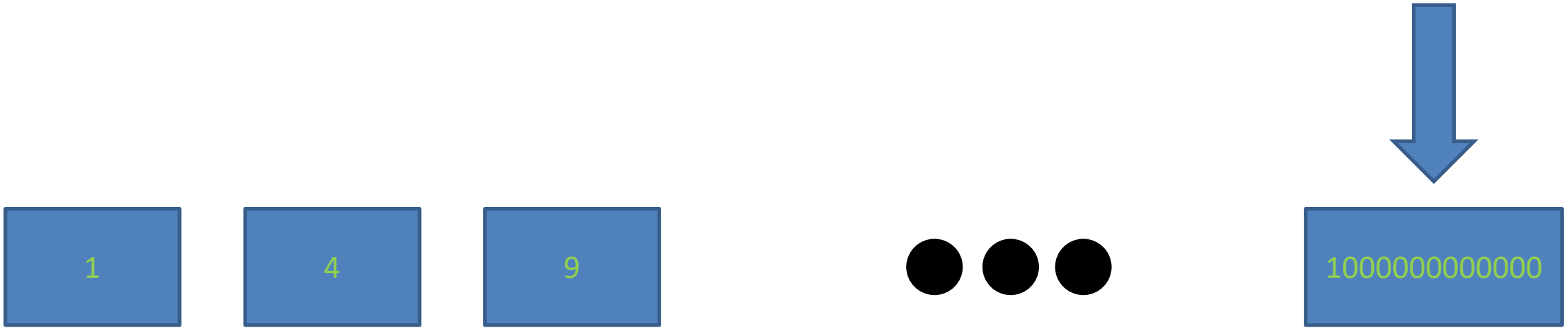
Tempo 1

Calcolo monoprocessore: esempio quadrato di una lista



Tempo 2

Calcolo monoprocessore: esempio quadrato di una lista

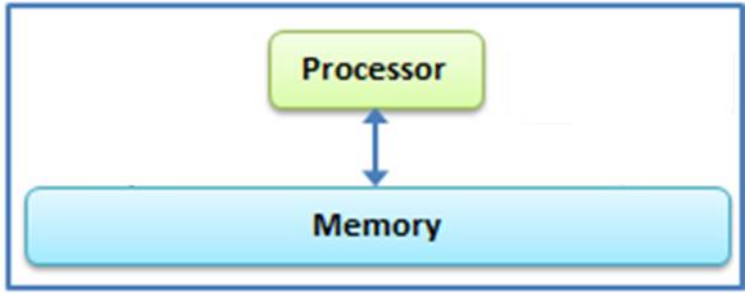


Tempo 1000000

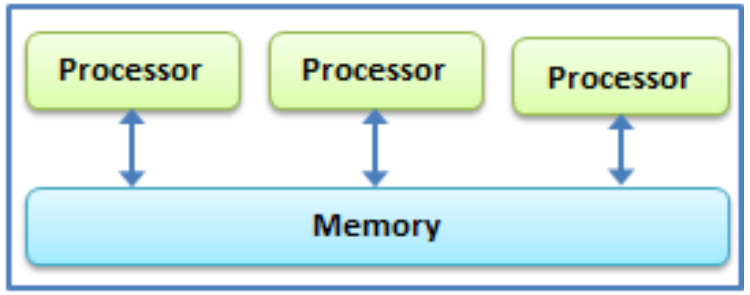
Abbiamo effettuato la stessa operazione in 1000000 tempi!

Architetture mono e multi-processore

- **Architettura monoprocessore**: una singola unità di elaborazione è responsabile per gestire tutti i processi di un elaboratore;
- **Architettura multi-processore**: un sistema dove molte unità di elaborazione **concorrono** all'esecuzione:
  - Ci permette di effettuare molto più lavoro in un'unità di tempo;
  - Maggiore affidabilità (tolleranza ai guasti);
- Più processori si dice **concorrano** all'esecuzione perché pur essendo multi-processore la memoria RAM rimane sempre una ed è condivisa tra tutti (**calcolo parallelo**)!
- I vostri computer non sono multi-processore ma multi-core (i vostri processori hanno n core fisici che si comportano come n processori diversi);



Architettura mono-processore

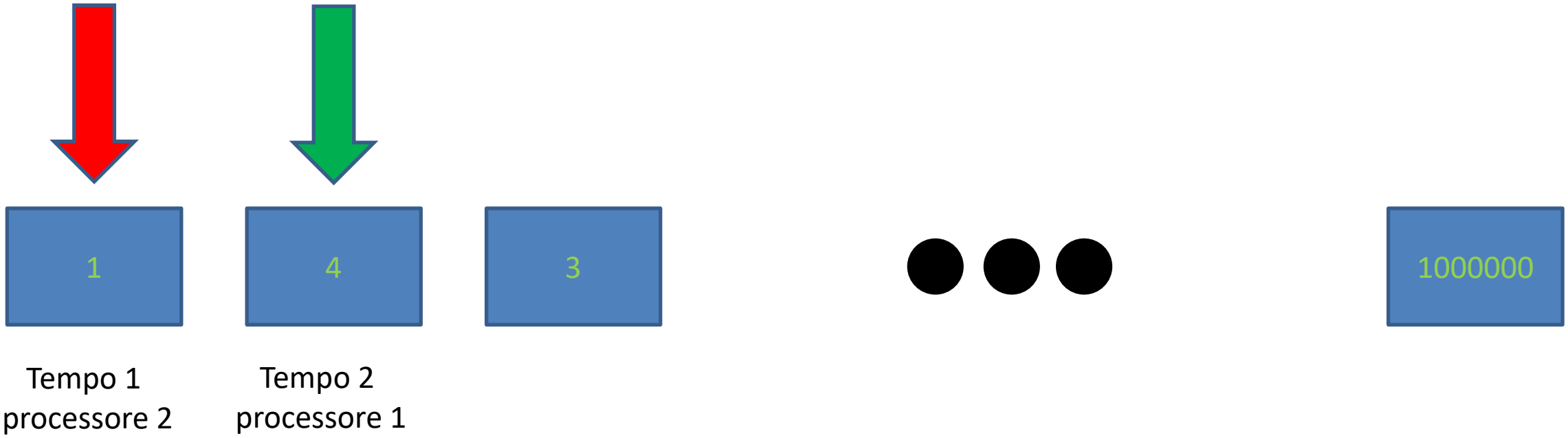


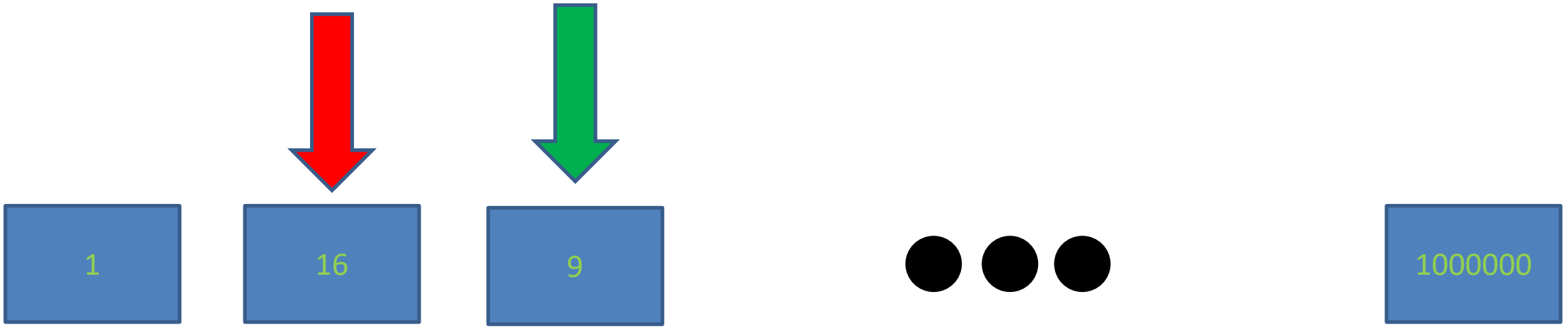
Architettura multi-processore

[Approfondimento: Calcolo concorrente vs parallelo]

Calcolo parallelo: a volte è pericoloso per i dati

- **Supponiamo di avere due processori che contemporaneamente modificano la lista (memoria condivisa)**;
- E' responsabilità del programmatore non fare danni!
- Ad esempio, supponiamo che il primo processore stia modificano l'intera lista, mentre il secondo processore modifichi solo la prima metà… cosa succederebbe?

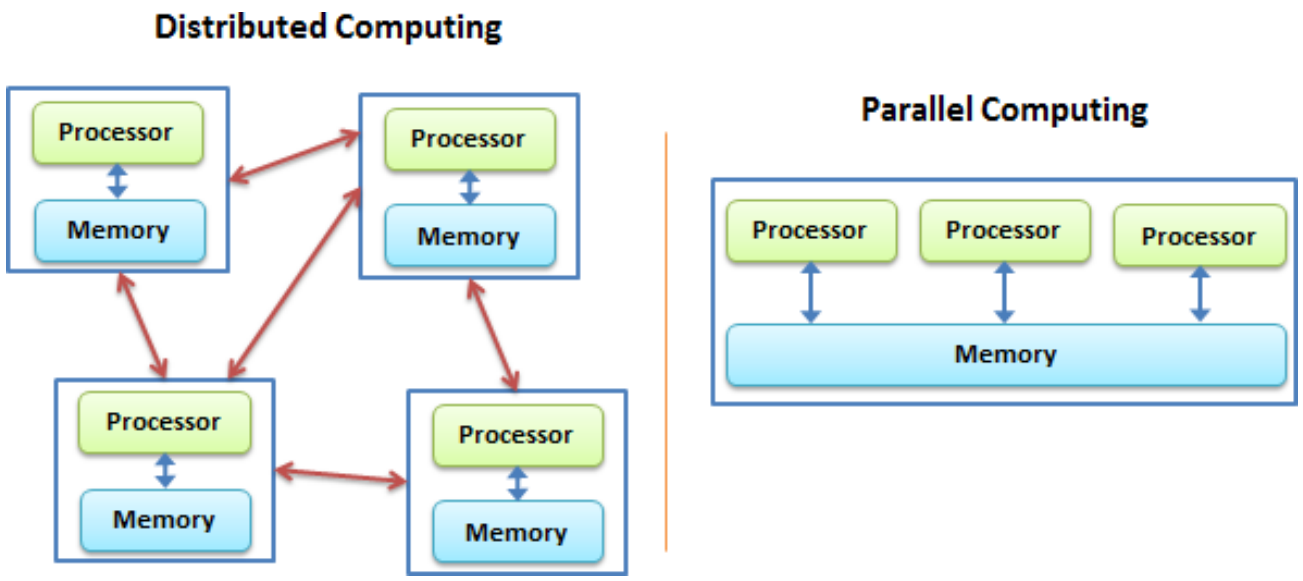




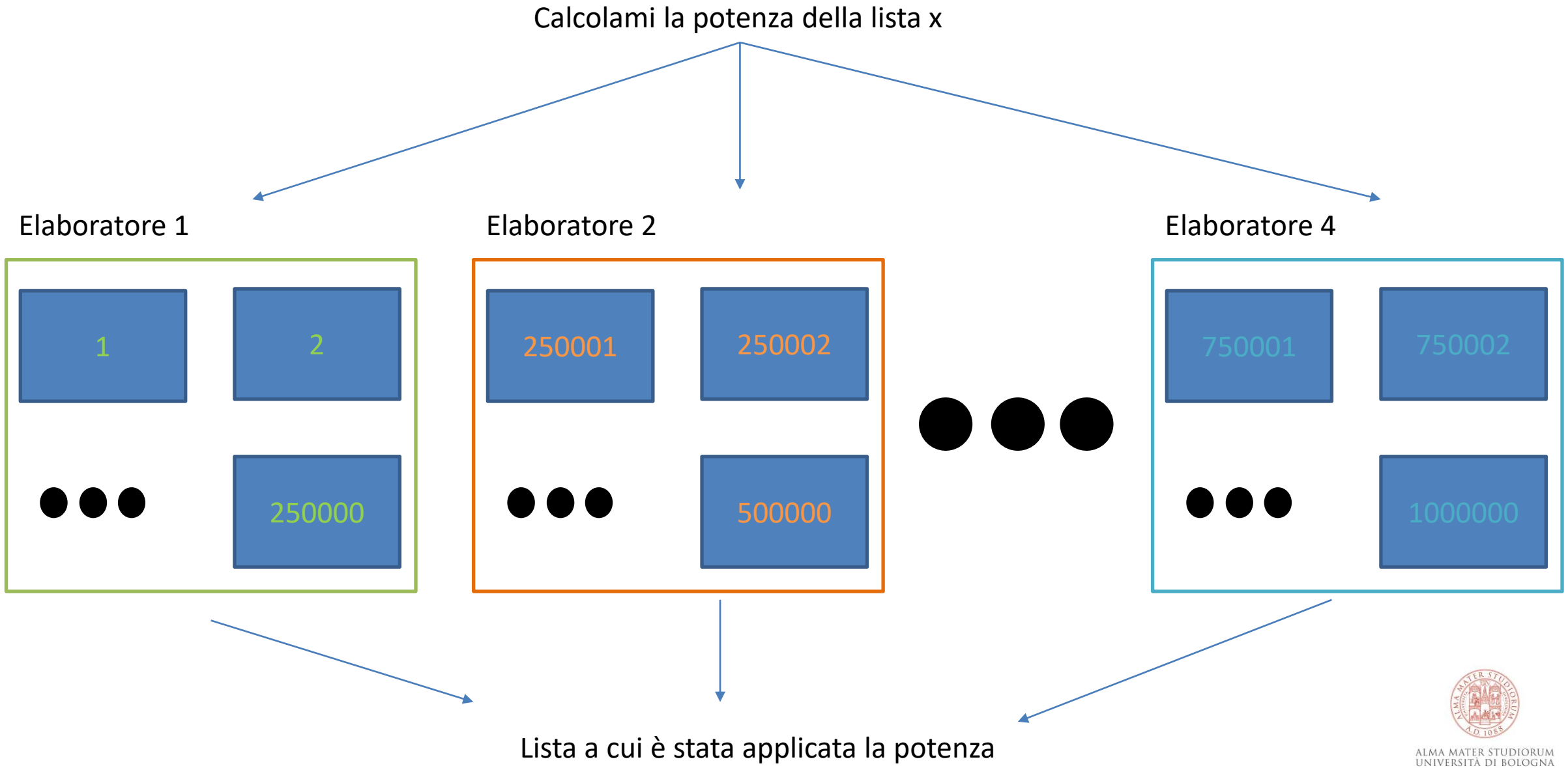
- Questa situazione è molto pericolosa per noi… come risolvere?
  - **Troviamo un programmatore più bravo** che si ricordi di **partizionare correttamente i dati**;
  - Oppure, ci muoviamo verso la computazione distribuita!

Calcolo distribuito

- **Il calcolo distribuito** separa **i processori** già collegati tra loro da collegamenti di comunicazione.
- Mentre i modelli di elaborazione parallela assumono memoria condivisa, i sistemi distribuiti si basano fondamentalmente sul passaggio dei messaggi da un elaboratore all'altro.
- Il calcolo distribuito si effettua ovviamente su sistemi distribuiti, ovvero **sistemi formati da molti elaboratori** posizionati in maniera geograficamente distante che **comunicano e coordinano azioni** in modo da apparire all'utente finale **come un unico sistema**.



Calcolo distribuito: potenza di una lista



Vantaggi del calcolo distribuito sul parallelo

- **Affidabilità,** elevata tolleranza agli errori: un arresto anomalo del sistema su un server non influisce sugli altri server;
- **Scalabilità:** è possibile aggiungere più macchine secondo necessità;
- **Alte prestazioni:** fornire prestazioni più elevate e migliori prestazioni in termini di costi rispetto ai normali sistemi centralizzati;
- Si adatta perfettamente per gestire dei database NOSQL e alla gestione dei Big Data!
- Ad esempio, pensate alla semplicità con la quale possiamo distribuire un database key-value e la semplicità con la quale possiamo distribuire il calcolo sulle varie macchine che contengono molte istanze di questi dati!

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

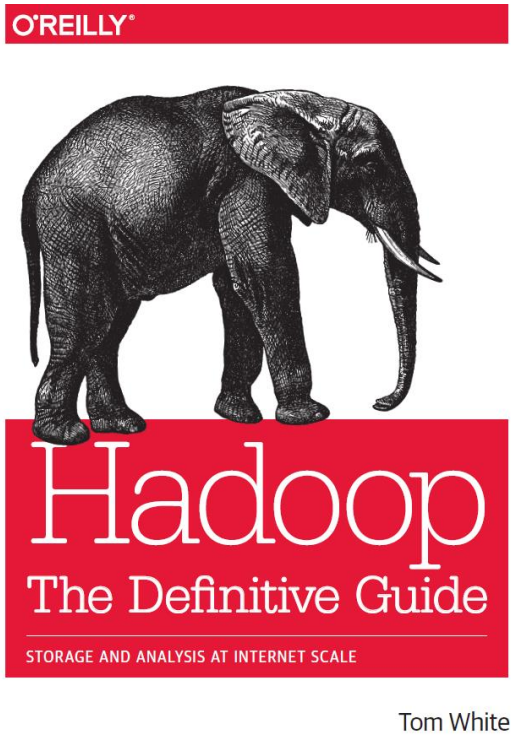
Hadoop e HDFS

Lorenzo Stacchio

Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

Materiale di approfondimento



- Purtroppo il materiale di approfondimento per i big data è inglese;

Apache Hadoop

- Apache Hadoop è un framework **che consente l'elaborazione distribuita di grandi insiemi di dati tra cluster di computer che utilizzano semplici modelli di programmazione;**

- **È progettato per passare da singoli server a migliaia di macchine, ognuna delle quali offre elaborazione e archiviazione locali.**

- Piuttosto che fare affidamento sull'hardware per fornire alta disponibilità e affidabilità**, la libreria stessa è progettata per rilevare e gestire i guasti a livello di applicazione,** in modo da fornire un servizio ad alta disponibilità su parte superiore di un cluster di computer (ognuno dei quali può essere soggetto a fallimenti).

- Sviluppato originariamente da Yahoo! e ispirato da una celeberrima pubblicazione di Google nel 2004 *[2]*;
- **Obiettivo**: archiviazione ed elaborazione di set di dati su larga scala;
- Infrastruttura: cluster di hardware di base
- Moduli forniti da Hadoop:
  - fornisce molte librerie per facilitare la vita del programmatore;
  - **HDFS**, un file system distribuito;
  - **MapReduce**, un modello di programmazione per l'elaborazione di dati su larga scala
  - **YARN**, una piattaforma di gestione delle risorse
- Utilizzato in produzione da Google, Facebook, Yahoo! e molti altri!

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters." (2004).

Storia di Apache Hadoop

- **2003**: Google pubblica la sua architettura di **cluster** e il file system distribuito (**GFS**)

- **2004**: Google pubblica il suo modello di programmazione MapReduce utilizzato su **GFS**
  - Scritto in C++
  - API closed-source, Python e Java disponibili solo per i programmatori di Google

- **2006**: Apache e Yahoo! Rilasciano Hadoop e HDFS:
  - Implementazioni basate su **Java** di Google MapReduce e GF;
  - **Hadoop è stato originariamente scritto in Java**, perché è stato utilizzato per "risolvere" i problemi in Nutch, anch'esso scritto in Java.
  - Tema caldo e dibattuto ma personalmente credo che **altri** linguaggi come **C o C++** sarebbero stati una scelta migliore;

- **2008**: diventa un progetto Apache indipendente

- **2012**: viene rilasciato Hadoop 1.0 … Hadoop 3.0 viene rilasciato nel 2017

- **Oggi**: utilizzato come piattaforma di archiviazione e analisi per tutti gli usi per i big data
  - Supporto e distribuzioni da IBM, Microsoft, Oracle, Cloudera…
  - Comunità attiva di sviluppatori;
  - La ricerca e sviluppo continua attivamente.

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters." (2004).

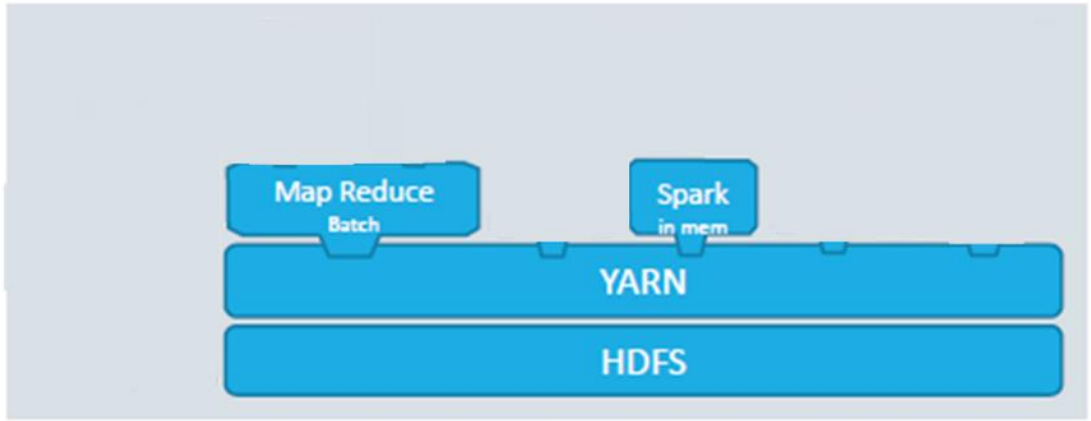
Hadoop ecosystem

- **Hadoop è in realtà** un ecosistema di tecnologie;

- Ai fini del corso esploreremo quattro di tutte le tecnologie messe a disposizione da Hadoop:
  - **HDFS** (Hadoop distributed file system): Un file system distribuito che fornisce accesso ad alta velocità ai dati e replica degli stessi;

  - **YARN** (yet another resource negotiatior): un framework per la pianificazione del lavoro e la gestione delle risorse del cluster;

  - **Map Reduce:**
    - A **YARN-based** system for parallel processing of large data sets
    - The powerhouse behind most of today's big data processing and is also used in other environments and NoSQL databases;

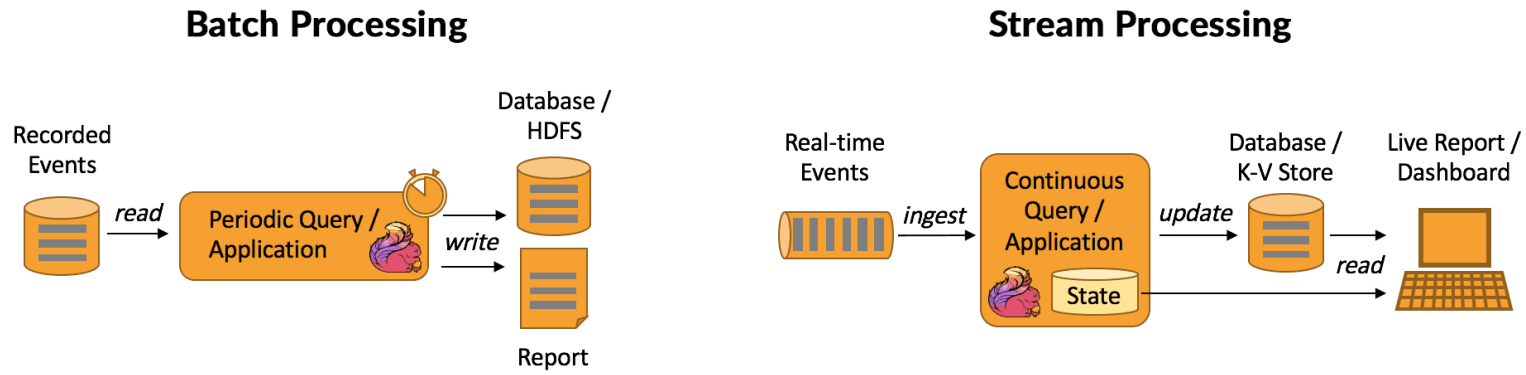


HDFS

- **HDFS** è un **filesystem** progettato per l'archiviazione e l'accesso a file molto grandi eseguendosi su un sistema **distribuito** (cluster) di hardware:
  - Un file in HDFS ha dimensioni di gigabyte o terabyte. Ci sono cluster Hadoop che oggi memorizzano petabyte di dati;
  - Le applicazioni eseguite su HDFS necessitano **dell'accesso in tempo reale ai dati che gli servono**.
  - HDFS è progettato **più per l'elaborazione in batch che per l'uso interattivo (scrittura)**.
  - Le applicazioni HDFS devono quindi seguire il modello «scrivi una volta leggi molte» per i file**. Un file una volta creato, scritto e chiuso non deve essere più modificato**. Questa ipotesi semplifica i problemi di coerenza dei dati e consente l'accesso ai dati ad alta velocità!
  - Il guasto dell'hardware è la norma piuttosto che l'eccezione. Pertanto, il rilevamento dei guasti e il loro ripristino rapido e automatico è un obiettivo architettonico fondamentale di HDFS

Documentazione hadoop

Batch vs stream analysis



- **L'elaborazione batch** viene spesso utilizzata quando si tratta di grandi quantità di dati e/o quando i sistemi non sono in grado di fornire dati streaming;
- L'elaborazione batch funziona bene **in situazioni in cui non sono necessari risultati di analisi in tempo reale** e quando è più importante elaborare grandi volumi di informazioni piuttosto che ottenere risultati di analisi in tempi rapidi (**es**. ordini dei clienti, quanti clienti hanno comprato i prodotti di venditore x nel mese di giugno);
- **L'elaborazione streaming** è fondamentale **se si desiderano risultati di analisi in tempo reale**. Creando flussi di dati, puoi inserire i dati negli strumenti di analisi non appena vengono generati e ottenere risultati di analisi quasi istantanei;
- L'elaborazione streaming è utile per attività come il **rilevamento delle frodi** ed interrompere transazioni fraudolente prima che vengano completate.
- L'analisi batch fornisce tuttavia capacità di analisi a tutto tondo, mentre l'analisi streaming ha molte limitazioni sul tipo di analisi!

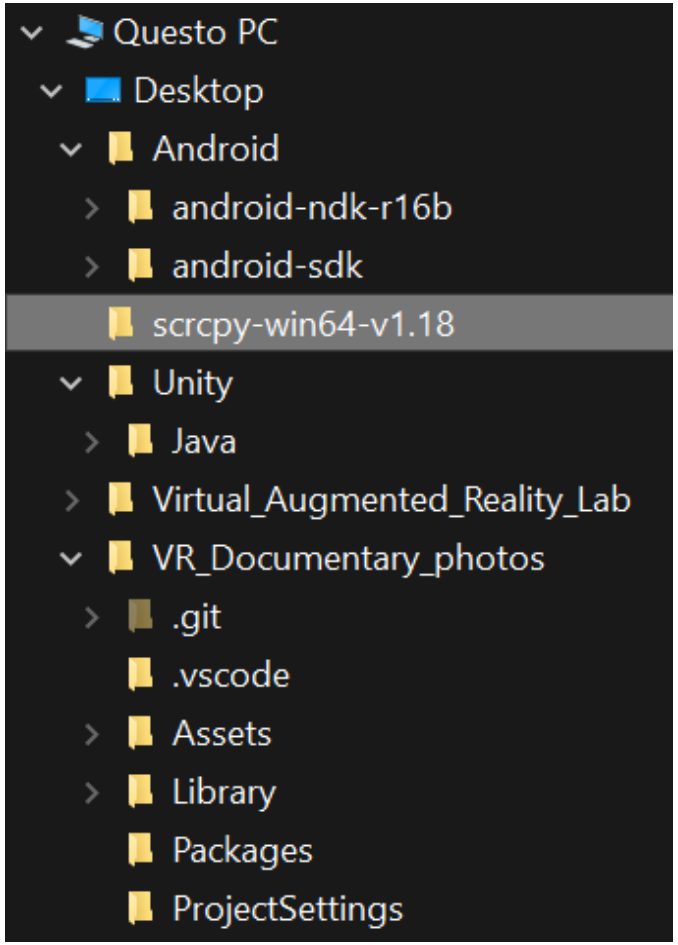
File system

- Un **file system** (FS), indica informalmente un meccanismo con il quale i file sono posizionati e organizzati su **dispositivi elettronici** utilizzati per l'archiviazione dei dati (es. unità di memoria di massa) o su dispositivi remoti utilizzando protocolli di rete;

- Esistono diversi tipi di file system, tra cui **FAT** (File Allocation Table), **HPFS** (High Performance File System) e **NTFS** (NT File System);

- Ad esempio, Windows adotta due file system: FAT ed NTFS per gestire differenti operazioni cogliendo i vantaggi sia dell'uno che dell'altro;

- Non parleremo in dettaglio dei file system e del loro funzionamento ma capiremo perché HDFS è una ottimo file system distribuito!

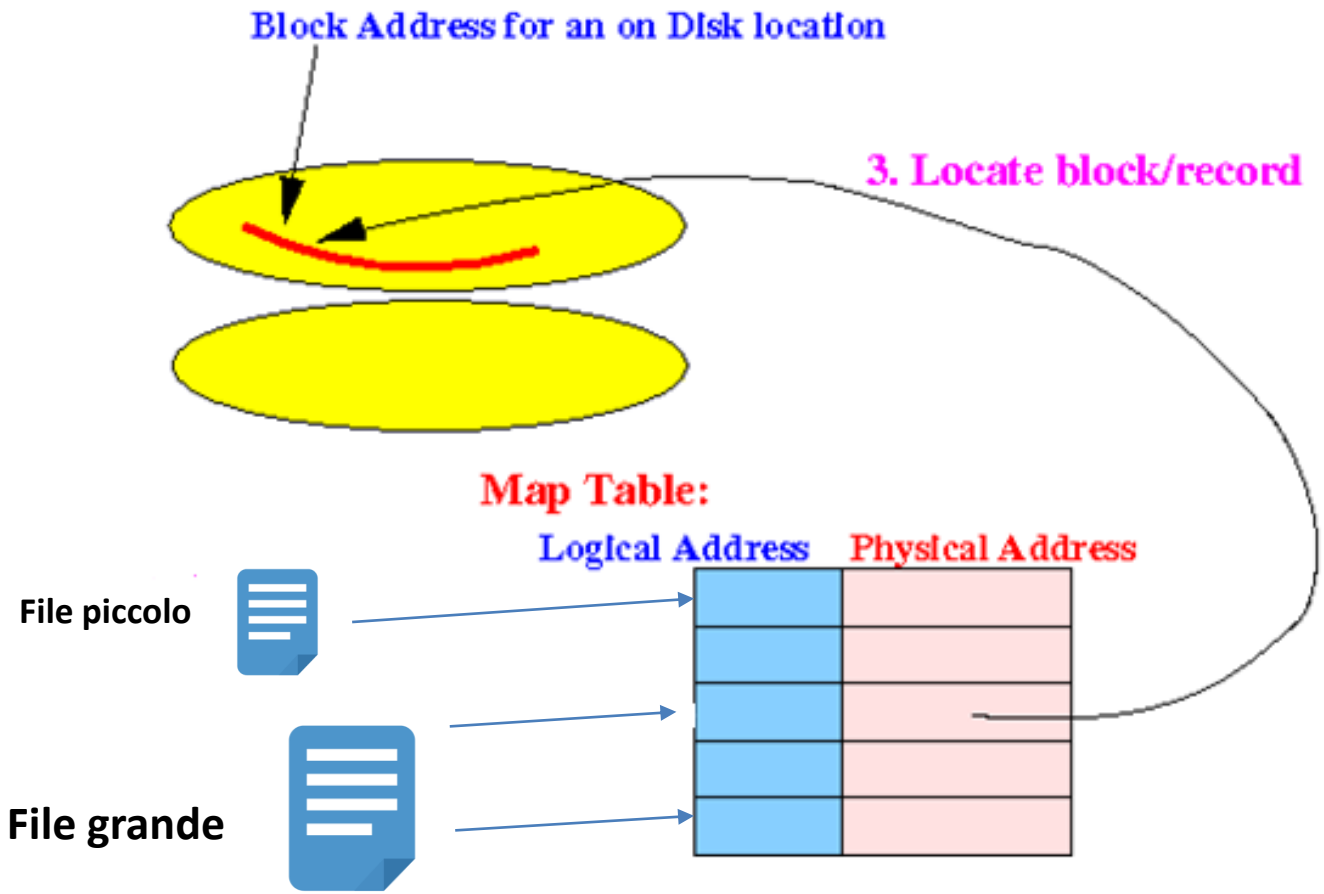


Un FS consente di salvare e accedere a file tramite una struttura astratta

Approfondimento: Vantaggi e svantaggi dei file system citati

HDFS: blocchi

- **Come tutti i file system**, HDFS divide **logicamente il dispositivo di archiviazione in quelli che vengono definiti blocchi;**



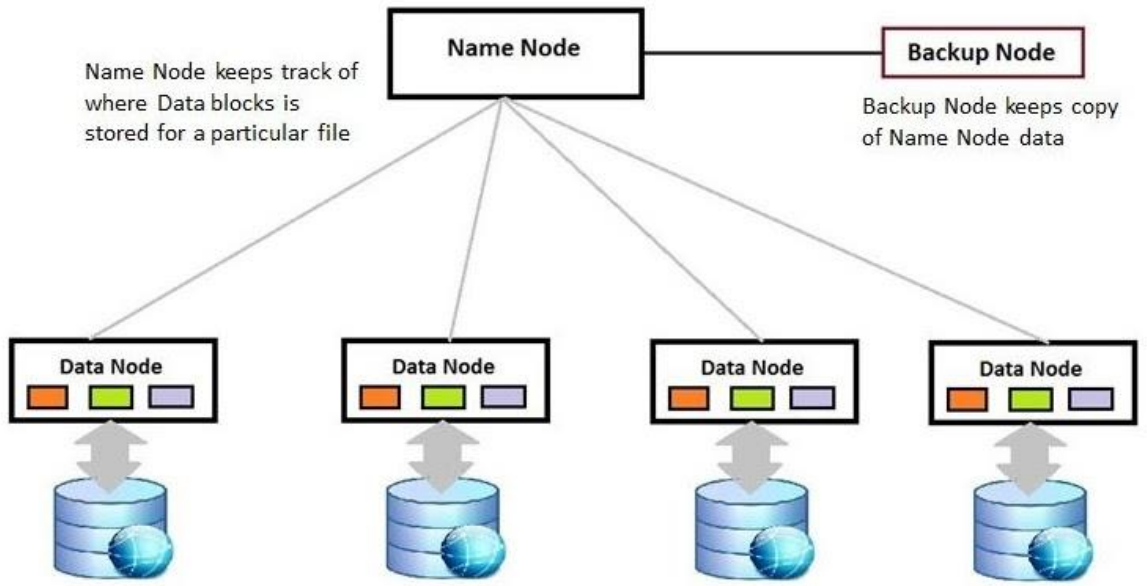
- Diversamente dai normali file system, che dividono la memoria in blocchi di dimensione molto piccola (in genere pochi kilobyte), HDFS imposta dei blocchi dai 64 megabyte fino ad 1 gigabyte;

- Questo è fatto per gestire dati **molto più grandi del normale** in maniera efficiente;

- Perché dividere la memoria in **blocchi**?

    - Avendo a che fare con dati **molto grandi** alcuni file potrebbero essere talmente grandi da non poter essere salvati su un unico dispositivo di memoria;
    - Dividendo la memoria in blocchi è più facile gestire file che si trovano suddivisi in più dispositivi di memoria diversi anche geograficamente distanti!

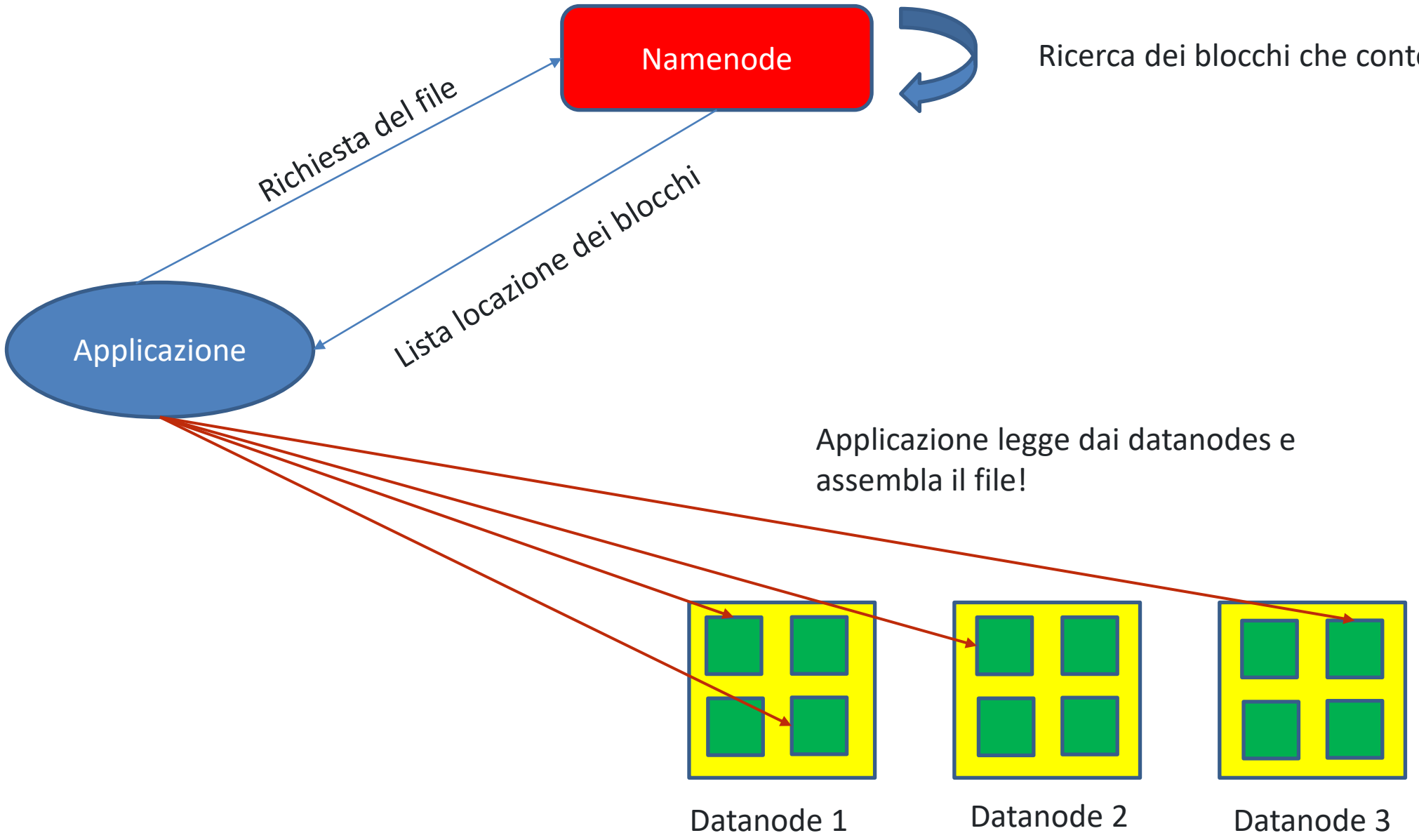
HDFS: namenodes e datanodes



- In HDFS, ci sono due entità principali che controllano la vita del file system:
  - Il **namenode** (master) mantiene **persistentemente** la struttura del filesystem e tutti i metadati di file e cartelle e conosce la posizione **dei blocchi appartenenti a ciascun file** (block pool);
  - **I datanode** (slaves) memorizzano e forniscono i blocchi di memoria. Inoltre, riferiscono periodicamente al namenode l'elenco dei blocchi che stanno memorizzando, **mostrando quindi di essere ancora attivi**;

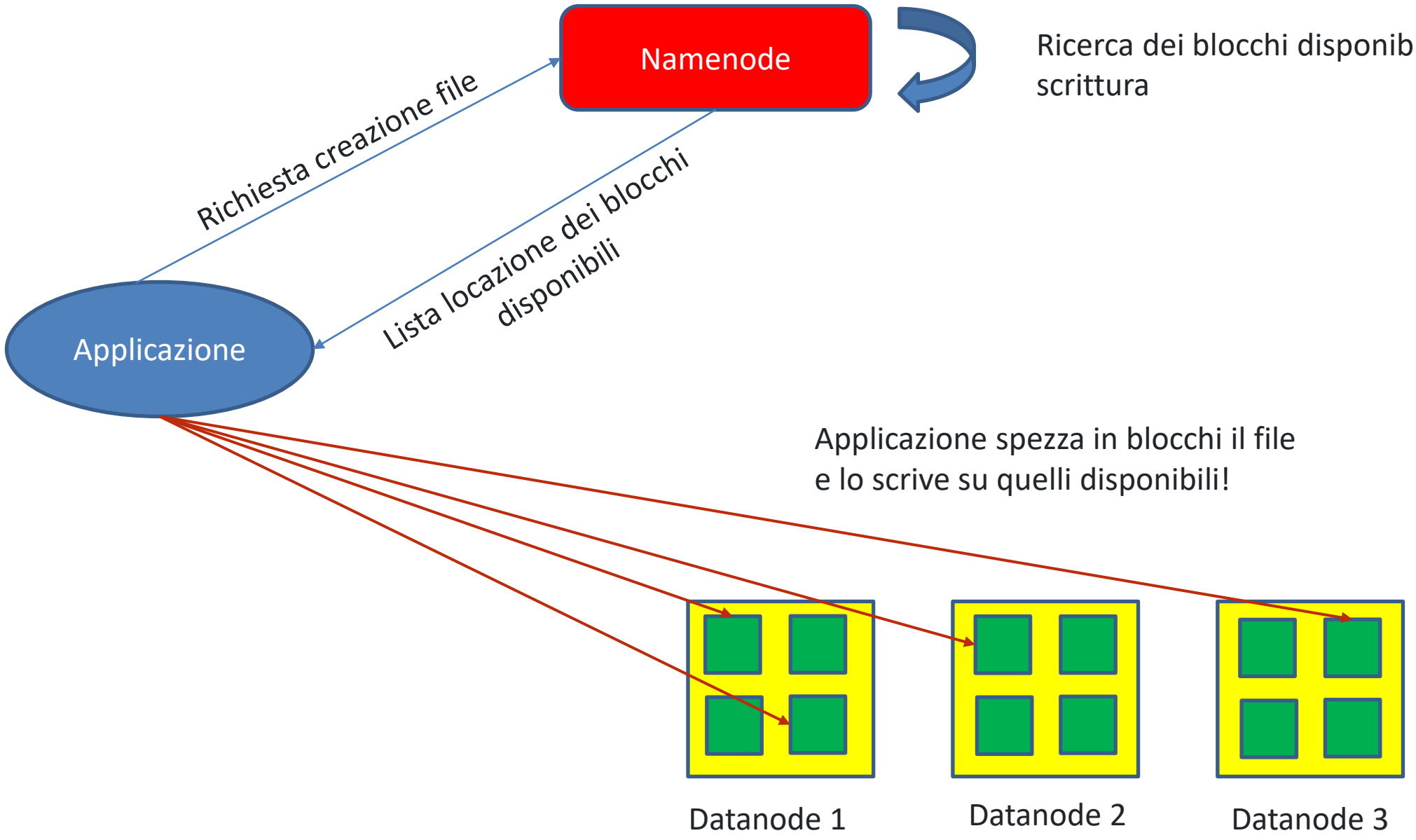
HDFS: lettura di un file

- L'applicazione che desidera leggere un file (o una parte di esso) deve prima contattare il **namenode** per determinare dove sono archiviati i dati effettivi (cioè quali **datanode**);
- Il client contatta quindi i datanode che contengono il file per recuperare i blocchi in essi memorizzati e portartli sul file system della singola macchina che li sta contattando;
- Caratteristica fondamentale del design che rivoluzionato il mondo dei big data: i dati non vengono mai spostati attraverso sul **namenode,** tutto il trasferimento dei dati avviene direttamente tra applicazione datanode!
- Si risparmia moltissimo tempo e non si fanno spostamenti di dati inutili!



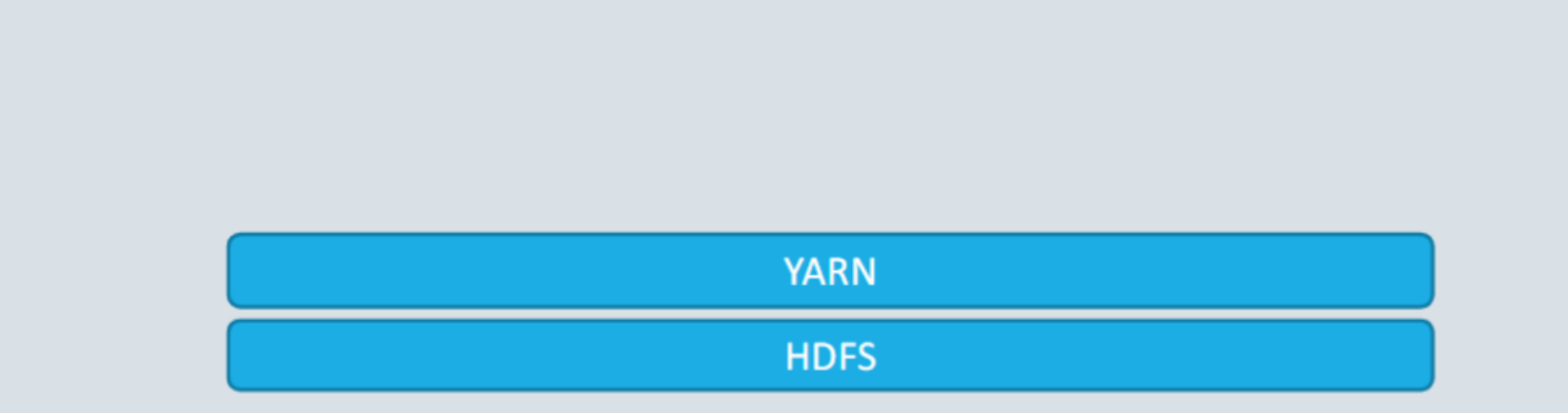
HDFS: scrittura di un file

- L'applicazione che desidera scrivere un file chiede al **namenode** di fornirgli una serie di blocchi su diversi **datanode**;
- Il namenode cerca dei blocchi disponibili sui diversi datanode e, se li trova, registra il nuovo file segnandosi i blocchi in cui esso è stato salvato!
- Il namenode quindi invia la lista dei blocchi all'applicazione che è quindi in grado di scrivere sui diversi datanode!



YARN

- YARN è il gestore delle risorse per i cluster Hadoop;
- Introdotta in Hadoop 2 per migliorare l'implementazione del paradigma **MapReduce**, ma abbastanza flessibile per supportare anche altri paradigmi di calcolo distribuito;
- Tipicamente, le applicazioni moderne contattano YARN che poi si interfaccia con HDFS per effettuare le varie operazioni!
- Permette di gestire moltissime operazioni diverse da parte di moltissimi agenti diversi!



HADOOP FILE FORMATS

- File in formato standard:
  - Dati testuali, **CSV**;
  - **XML, json;**
  - **Dati binari** (es. immagini)
- **File specifici di hadoop;**
- **Formato file personalizzato;**

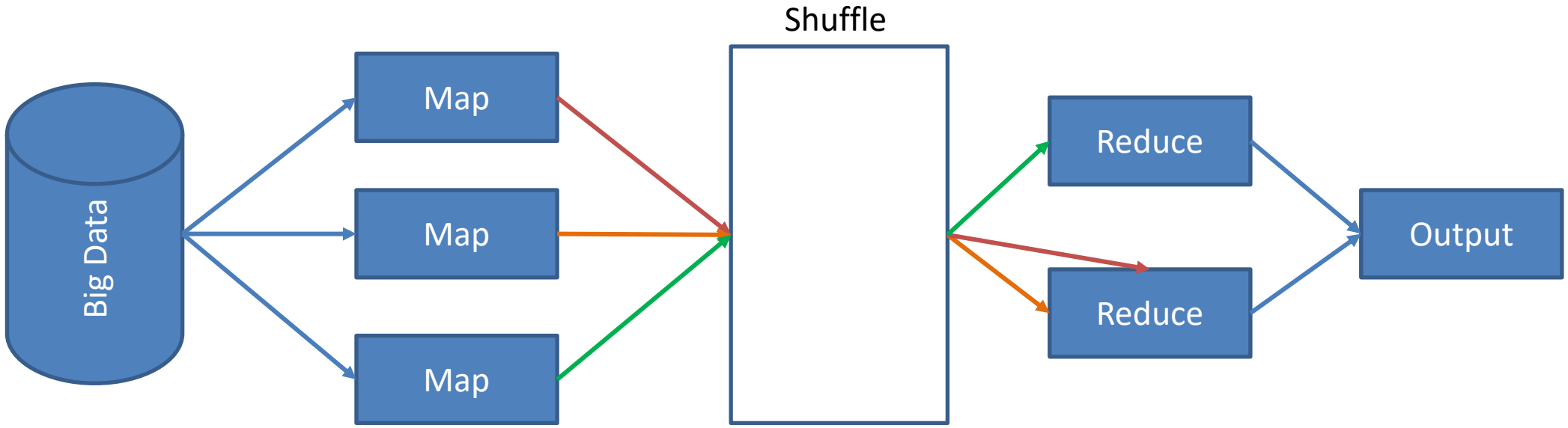
Map reduce

Lorenzo Stacchio

Studente di dottorato in Computer Science

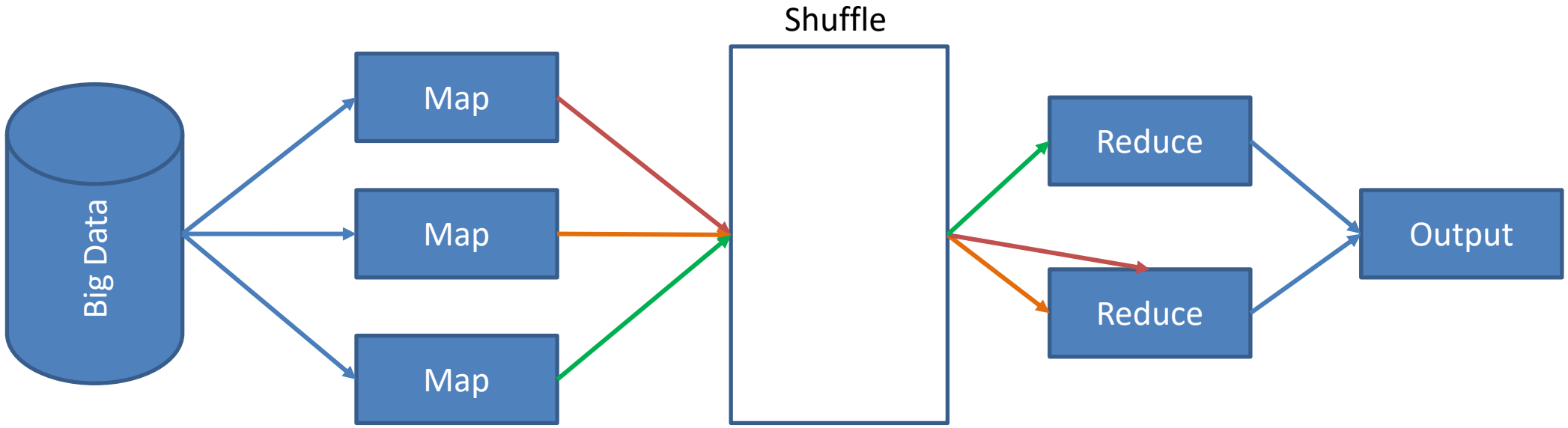
Dipartimento di Scienze per la Qualità della Vita

Map reduce



- MapReduce è un modello di programmazione per l'elaborazione dei dati, **semplice ma potente**;
- Il framework Hadoop può ovviamente eseguire programmi di tipo MapReduce scritti in varie linguaggi (Python, Java);
- Prima di parlare di cosa sia il map-reduce, bisogna capire il perché si usa: i programmi MapReduce sono intrinsecamente **paralleli e distribuibili** quindi permettono l'analisi dei dati su larga scala nelle mani di chiunque abbia abbastanza risorse.

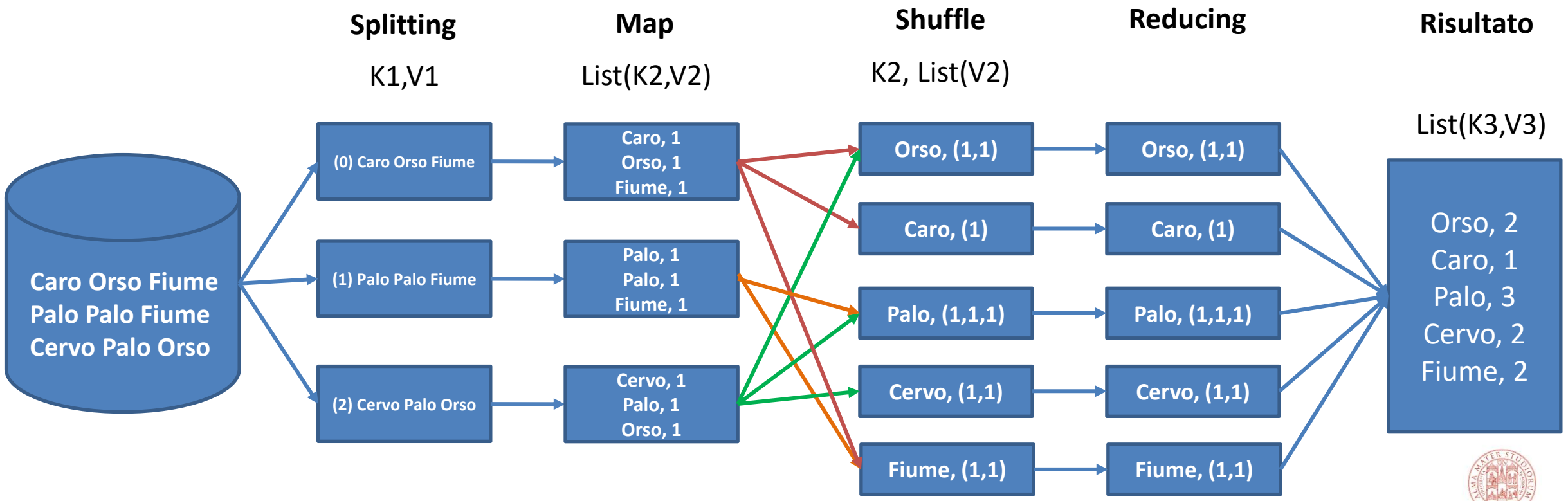
Map reduce: steps



- Come avrete intuito il paradigma map reduce è composto in genere da 3 step:
  - Lo step di mapping implementa **una funzione che agisca su coppie chiave-valore** rappresentanti i dati e li trasforma in nuove coppie **chiave-valore**;
  - Lo step di shuffle, ri-organizza i dati sui vari processi del sistema;
  - Lo step di **reduce** prende a sua volta coppie chiave-valore e restituisce l'output finale che sarà sempre un insieme di coppie chiave-valore **aggregando tipicamente i valori**!

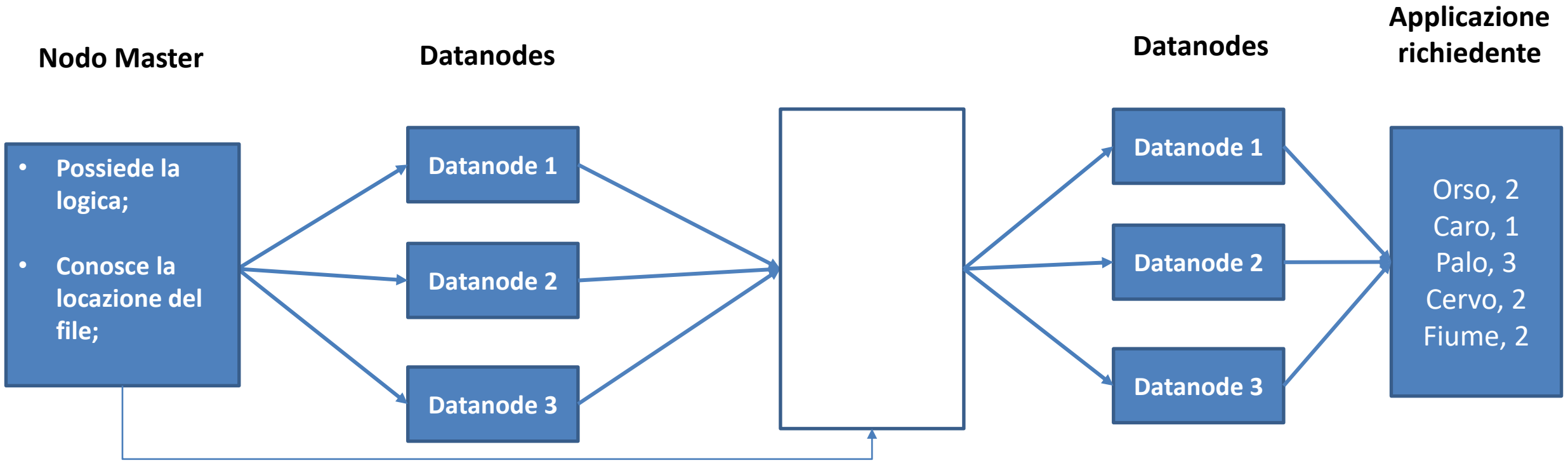
Map reduce: conteggio parole

- Cerchiamo di trovare l'intuizione al funzionamento del map-reduce con l'esempio del conteggio delle parole all'interno di un testo (anche molto grande);
- Prendiamo come testo di riferimento questo testo di parole a caso;



Map reduce: cos'è successo da un punto di vista architetturale?

- Map reduce ha sfruttato al massimo **l'hardware distribuito a sua disposizione**, chiaramente appoggiandosi su YARN e HDFS!



Il master indica come effettuare lo shuffle!

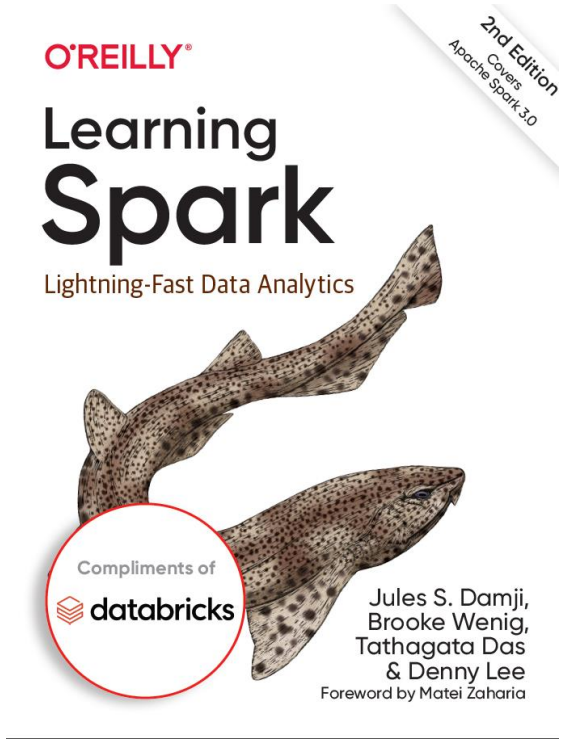
Spark: un framework per la programmazione big data

Lorenzo Stacchio

Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

Materiale di approfondimento



- Purtroppo il materiale di approfondimento per i big data è inglese;

Apach spark è popolare!

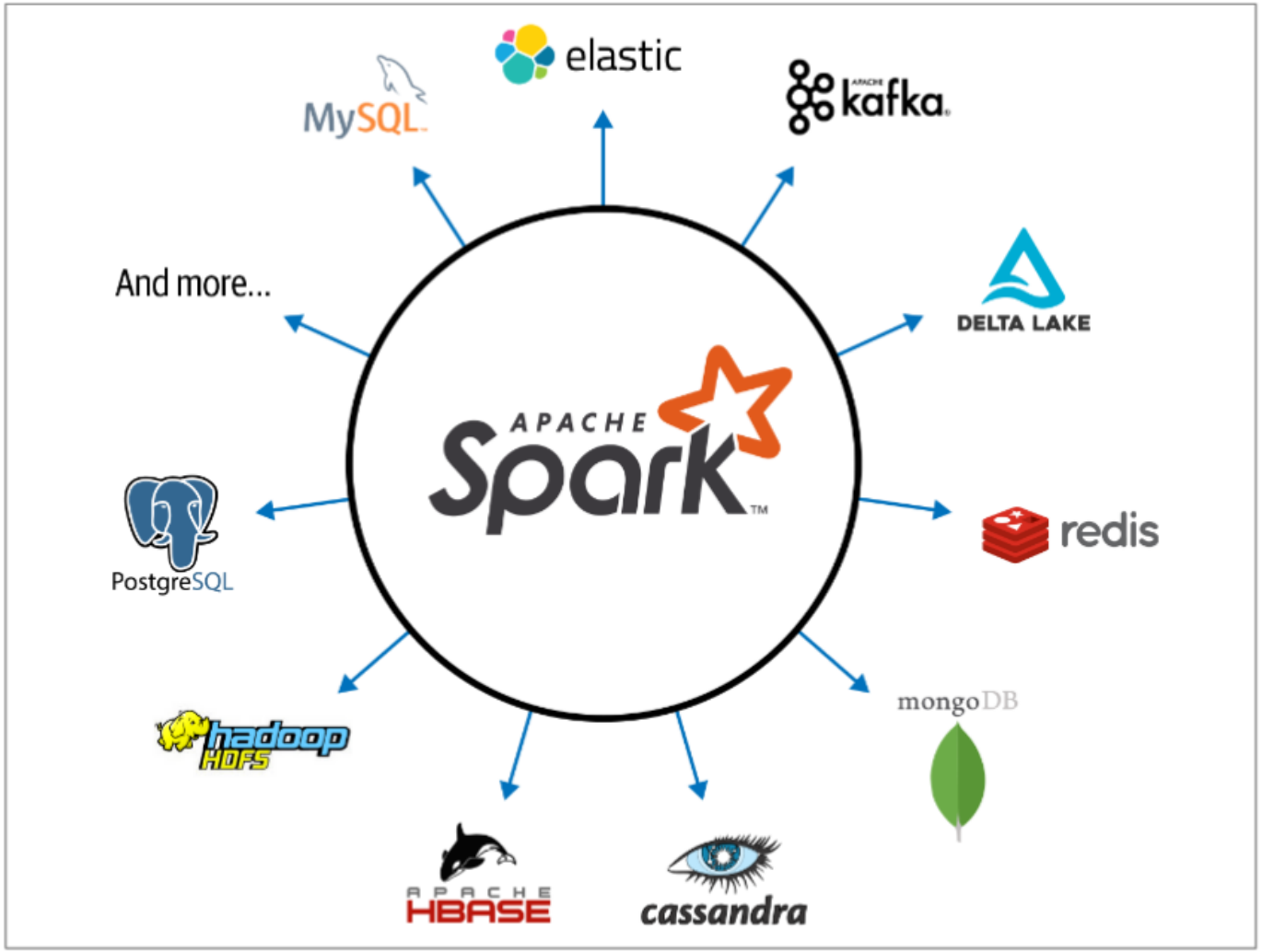


Figure 1-2. Apache Spark's ecosystem of connectors

Limite di Hadoop Map-reduce

- Sebbene Apache Hadoop abbia ottenuto un'ampia diffusione ispirando una vasta comunità di sviluppatori, il framework MapReduce su HDFS presenta alcune carenze;
- Tra queste c'è la modalità con cui Hadoop gestisce MapReduce sequenziali;
- In sostanza, se dobbiamo fare più MapReduce concatenati, ad ogni singola operazione Mapreduce c'è bisogno di salvare i dati sulla memoria fissa;
- Questo richiede un quantitativo molto più grande di tempo!

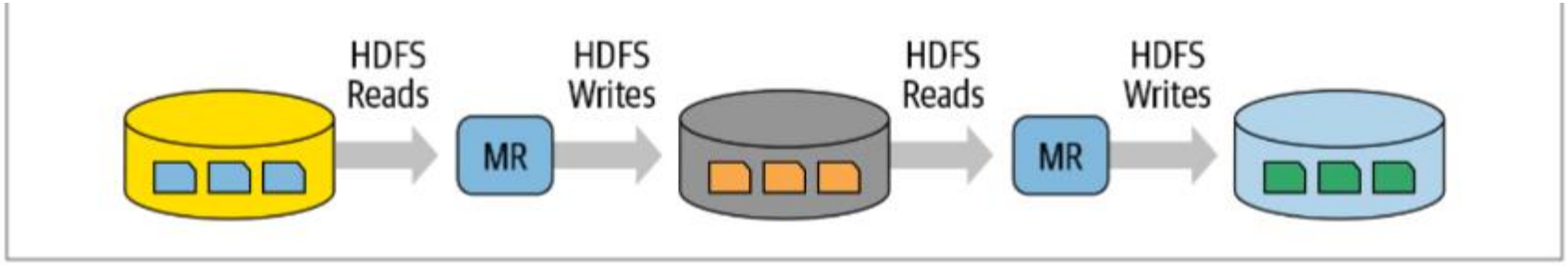


Figure 1-1. Intermittent iteration of reads and writes between map and reduce computations

Storia di apache spark

- I primi papers pubblicati su Spark hanno dimostrato che era da 10 a 20 volte più veloce di Hadoop MapReduce per determinati lavori.
- Oggi è di molti ordini di grandezza più veloce.
- Quello che ha dato a Spark questo slancio è stato prendere idee in prestito da Hadoop Map-Reduce e migliorarle;
- Principalmente, ha elevato all'ennesima potenza il parallelismo supportando l'archiviazione dei processi intermedi **non sul disco ma in RAM riducendo inoltre i calcoli da effettuare!**
- **Come è stato possibile effettuare miglioramento?** Grazie ai DAG (Directed Acyclic Graph), che sostanzialmente riesce a generare tutte le diverse fasi di calcolo a priori!
- In questo modo ottimizziamo il piano di esecuzione (es. per ridurre al minimo lo spostamento dei dati);

https://data-flair.training/blogs/dag-in-apache-spark/

Apache spark + python = pyspark!

Vediamo cosa possiamo fare con PySpark!

Come trarre valore dai dati?

Lorenzo Stacchio

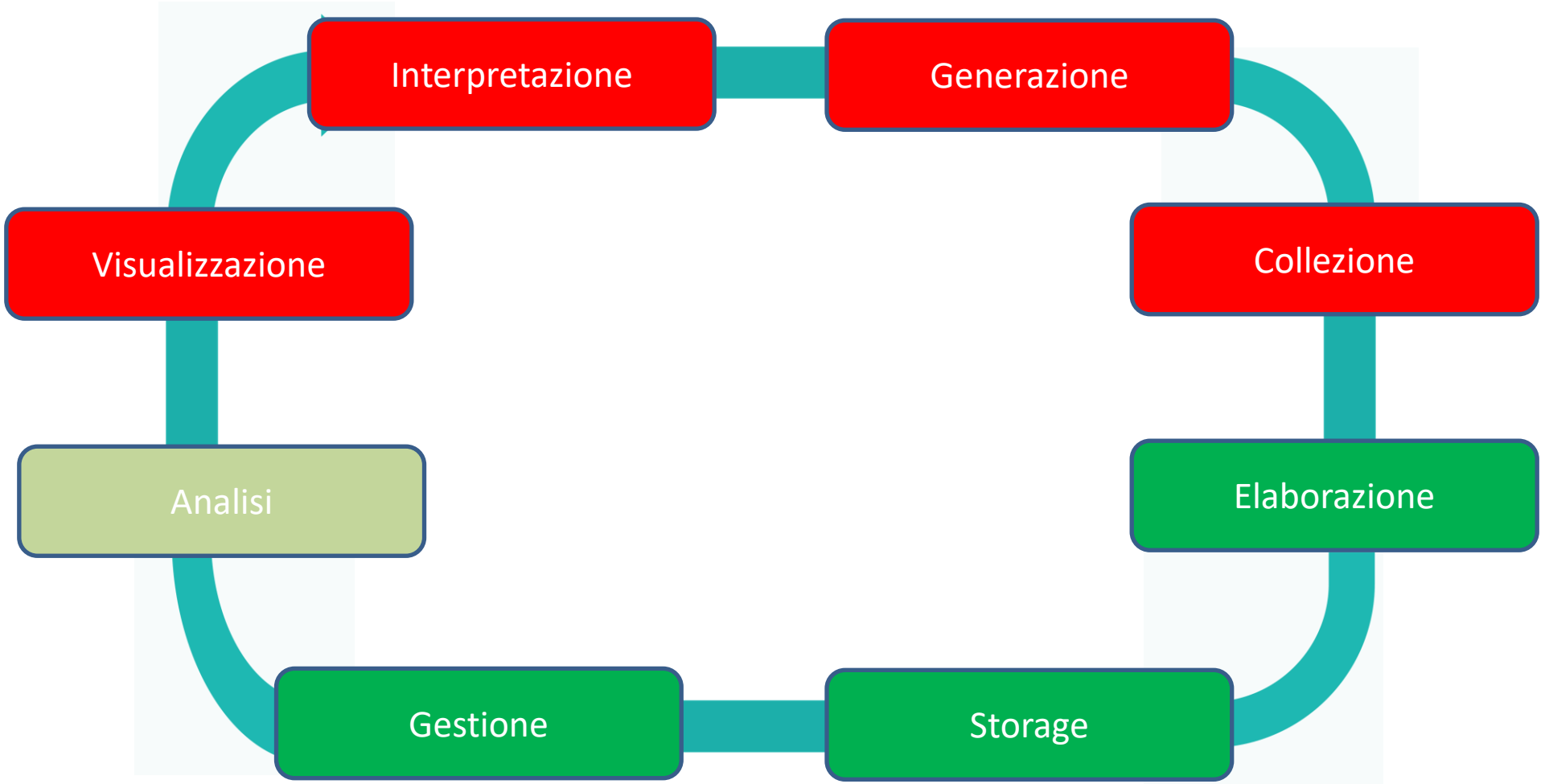
Studente di dottorato in Computer Science

Dipartimento di Scienze per la Qualità della Vita

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Lo scopriremo nelle prossime lezioni!

Cosa abbiamo studiato finora?



Cosa studieremo da ora in poi?