POLITECNICO DI TORINO

TESI DI LAUREA MAGISTRALE

Business Intelligence nell'era dei Big Data

Candidato:
Giovanni MONTEMURRO
In collaborazione con:
Blue Reply unit DATA
Tutor aziendale:
Andrea BARDONE

Relatore:
Prof.ssa Tania CERQUITELLI
Correlatore:
Prof. Daniele APILETTI

Dipartimento di Ingegneria Gestionale e della Produzione

A.A 2019 - 2020

«*La formazione universitaria è stata per me un processo di crescita continua, che mi ha portato ad affrontare con pensiero libero e critico qualsiasi sfida. In nome dei valori di libertà e ingegno ho realizzato anche l'ultimo step di questo grande percorso.*»

Indice

Elenco delle figure

Elenco delle tabelle

I miei sentiti ringraziamenti vanno a :

- *I miei genitori Enza e Saverio e mia sorella Marilisa*
- *La nonna Maria*
- *Zia Mariella, Zio Mimmo e i cugini Checco,Doriana e Antonello*
- *Zia Rosaria, Zio Franco e i cugini Giovanni e Nicola*
- *La mia ragazza Silvia*
- *La prof.ssa Cerquitelli e il prof. Apiletti*
- *Il tutor aziendale Andrea Bardone*
- *I miei amici di Torino*
- *I miei amici di Matera*
- *I miei amici di Barcelona*
- *Tutti i professori che mi hanno messo alla prova ogni giorno*

Capitolo 1

Big Data: estrarre valore tramite la Business Intelligence

1.1 Introduzione

Il caso aziendale che accompagna il lavoro di tesi si propone di approfondire il tema della gestione dei dati finalizzata alla realizzazione di report e grafici in tempo reale attraverso strumenti sia commerciali che open-source, nell'ecosistema delle nuove tecnologie sviluppatesi all'interno del framework open-source Hadoop, uno strumento affidabile e scalabile, nato per effettuare operazioni di gestione dei calcoli su grandi quantità di dati.

I cosiddetti sistemi di Business Intelligence (BI) sono programmi, software e applicazioni progettati per aiutare a individuare, recuperare, analizzare e documentare i dati, consentendo alle organizzazioni di trasformare i dati in qualcosa di monetizzabile. All'interno di un ambiente in rapida evoluzione, la necessità di possedere informazioni "just-in-time" è fondamentale per il successo e per rimanere in competizione: le decisioni corrette richiedono il dominio su informazioni reali e complete, che i sistemi informativi transazionali non possono generare nel modo desiderato. Il termine Business Intelligence è stato introdotto dal Gruppo Gartner a metà degli anni '90 che lo ha definito come "un termine ombrello che include le applicazioni, le infrastrutture, gli strumenti e le pratiche migliori che permettono di accedere e di analizzare le informazioni per migliorare e ottimizzare le decisioni e le performance".

Con il fenomeno **Big Data** le modalità con cui gli strumenti di Business Intelligence hanno supportato gli utenti di Business delle imprese si sono evolute notevolmente. Il termine "rilevamento intelligente dei dati" è stato introdotto recentemente e si sta affermando come un potente fattore di differenziazione tra i vari settori. Per ottenere informazioni chiare, la maggior parte delle organizzazioni si sta concentrando sulla costruzione di modelli e sull'integrazione di dati per semplificarli e automatizzare le attività. Ciò ha aumentato notevolmente la domanda di *Data Scientist* nel mercato. Tuttavia, non tutte le aziende desiderano assumere figure professionali del Mondo informatico, ma alcune preferiscono acquistare licenze di prodotti, i cui

principali vendor saranno descritti nel Capitolo 3, che permettono di effettuare non solo operazioni di visualizzazione di dashboard interattive[1], ma anche di cosiddetta **agumented analytics** , attraverso cui gli utenti possono interagire con gli strumenti di analisi attraverso chatbot e interfacce conversazionali che utilizzano il linguaggio umano, affiancando la visualizzazione grafica dei dati. Il processo decisionale basato sui dati è facilitato più rapidamente e in modo più appropriato utilizzando queste tecniche. Le aziende che operano nel settore informatico, invece, possono, oltre ad acquisire licenze di prodotti commerciali o scegliere di utilizzare delle figure professionali in grado di raggiungere l'obiettivo attraverso la realizzazione di codice, decidere di indirizzarsi verso soluzioni open-source che assistono gli utenti nella realizzazione delle dashbboard, ma sono indirizzate a figure con un livello di specializzazione informatica discreta. In questo modo, le aziende informatiche possono effettuare operazioni di rilevamento intelligente dei dati senza alcun costo di licenza, risparmiando il tempo richiesto per la realizzazione del codice. Tuttavia va chiarito che gli strumenti open-source, che saranno descritti nella fase finale del progetto di tesi, non permettono di effettuare operazioni di agumented analytics, bensì permettono esclusivamente la realizzazione di dashboard interattive.

1.2 Un tentativo di definizione dei Big Data

Il termine Big Data è nato negli ultimi tempi in una forma così rapida e non ordinata, da non esistere una definizione comunemente riconosciuta, ma sono state assegnate diverse nel corso degli anni da parte di analisti, ricercatori esperti, aziende del settore.

La maggior parte di queste definizioni si concentra sulle caratteristiche e si sviluppa a partire dal lavoro di Laney, che nel presentare le sfide della gestione dei dati che le aziende hanno dovuto affrontare in risposta all'aumento del commercio elettronico nei primi anni 2000, ha introdotto un concetto che esprime il tridimensionale aumento di volume, velocità e varietà dei dati, invocando la necessità di nuove architetture tecnologiche che incidono sui portafogli applicativi e sulle decisioni di strategia aziendale [1]. Sebbene questo lavoro non menzionasse esplicitamente i Big Data, il modello, in seguito soprannominato "le 3 V", è stato associato al concetto di Big Data e usato come sua definizione da molti altri autori. Alle "3 V" è stata aggiunta una quarta "V" da parte di Dijcks nel Report realizzato per Oracle nel 2012, ossia quella di valore [2], che sta a indicare la monetizzabilità che deriva per le imprese e una quinta "V", quella di variabilità, che identifica i cambiamenti che ci possono essere all'interno dei dataset, da parte del "National Institute of Standards and Technology" (NIST). L'istituto asserisce la necessità per i Big Data di possedere una architettura "scalabile" per ottenere delle efficienti fasi di archiviazione, manipolazione e analisi dei dataset [3]. Il modello delle

[1] Una dashboard è una schermata che permette di monitorare in tempo reale l'andamento delle metriche aziendali.

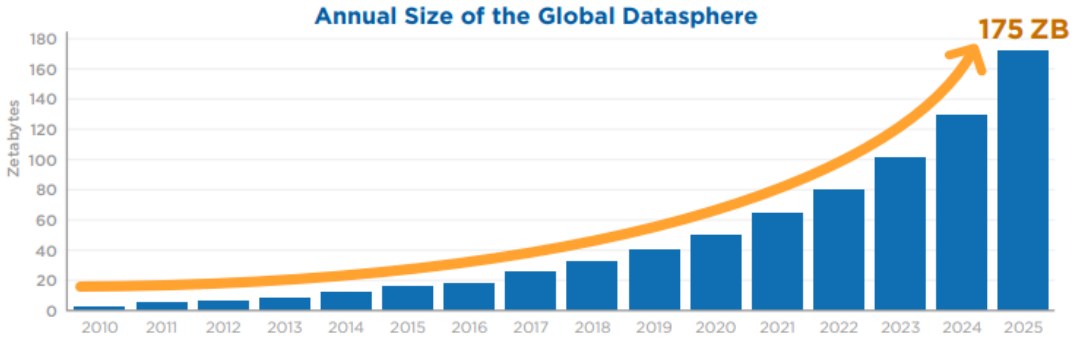

FIGURA 1.1: crescita della datasfera globale dal 2010 al 2025.
Fonte: IDC

"5V" sarà il punto di partenza del lavoro di tesi, ma appare opportuno evidenziare che ci sono molte altre definizioni e interpretazioni del termine in riferimento anche al cambiamento culturale originato dai Big Data. Daniel Gillick, ricercatore senior di Google, ha usato la definizione di "un cambiamento culturale in cui sempre più decisioni vengono prese dagli algoritmi." [4]. Un'altra definizione importante assegnata dal NIST, che ci permette di identificare le caratteristiche tecnologiche, è quella del *Big Data Paradigm*, che è descritto come "la distribuzione di sistemi di gestione dei dati all'interno di risorse associate orizzontalmente che permettono di ottenere la scalabilità necessaria per gestire dataset estensivi."

1.3 Volume, Varietà, Velocità e Variabilità

1.3.1 Volume

In un documento del 1965 Gordon Moore stimava che la densità dei transistor su un circuito integrato sarebbe raddoppiata ogni due anni [5]. Conosciuto come la legge di Moore, questo tasso di crescita è stato applicato a tutti gli aspetti dell'informatica e corrisponde ad un CAGR (Compound Annual Growth Rate) di circa il 41,4 %. In riferimento alla "datasfera globale", vale a dire il volume di dati generati globalmente ogni anno, il tasso di crescita è stato nel tempo anche superiore a quello della legge di Moore, anche se le stime per i prossimi anni indicano un valore che si attesta attorno al 27 %, come evidenziato nel report realizzato da IDC [6], secondo il quale l'incremento del volume dovrebbe passare da 33 a 175 Zetabyte dal 2018 al 2025. (Figura 1.1). Osservando le stime del CAGR diviso per settore (Figura 1.2) emerge che nel campo sanitario ("Helthcare") la crescita dovrebbe essere maggiore rispetto alla media, con un valore che si attesta al 36 %.

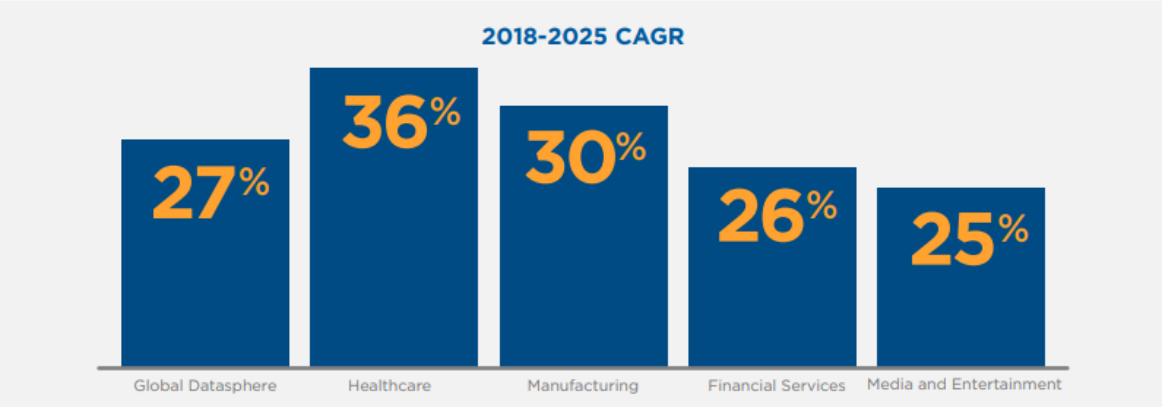

FIGURA 1.2: crescita della datasfera globale dal 2010 al 2025 per settore. Fonte: IDC

1.3.2 Varietà

Un incremento significativo del volume dei dati deriva dalla quantità di dati non strutturati, che non possono essere inseriti all'interno di tabelle Sql[2] e essere gestiti dal modello relazionale. Le raccolte dati sono state estese a nuove fonti, che non provengono necessariamente dalle operazioni interne dell'azienda, ma che derivano da Internet, come testi, pagine Web, immagini e video, senza tralasciarele informazioni che derivano dai dispositivi multi-connessi comunemente definiti come *Internet of Things*. La varietà rappresenta dunque il bisogno di analizzare i dati a partire da fonti e formati diversi.

1.3.3 Velocità

Grazie ad infrastrutture di rete che permettono lo scambio delle informazioni a velocità sempre più elevate è stato reso possibile implementare dei servizi in cui un client riceve da un server grosse quantità di dati in tempo reale: si tratta dei cosiddetti servizi di *streaming*. Spesso, nell'opinione comune, questi servizi sono considerati afferenti soltanto al campo della trasmissione dei segnali audio-visivi dal vivo o *on-demand*, ma in realtà il termine fa riferimento ad ogni flusso continuo di dati in tempo reale. La gestione di dati in *streaming* è molto comune nel settore bancario in cui gli Istituti che gestiscono i conti correnti e le varie carte elettroniche di pagamento, processano dati derivanti contemporaneamente da diversi dispositivi per monitorare le transazioni che avvengono in diverse località geografiche. La gestione dei dati in *streaming* avviene secondo modalità totalmente diverse rispetto a quelle dei dati *offline*, utilizzando un tipo di architettura in cui i dati non vengono archiviati, ma

[2]Sql (Structured Query Language) è un linguaggio per database basati sul modello relazionale , progettato per creare e modificare schemi di database, inserire, modificare e gestire dati memorizzati, interrogare i dati memorizzati, creare e gestire strumenti di controllo e accesso ai dati

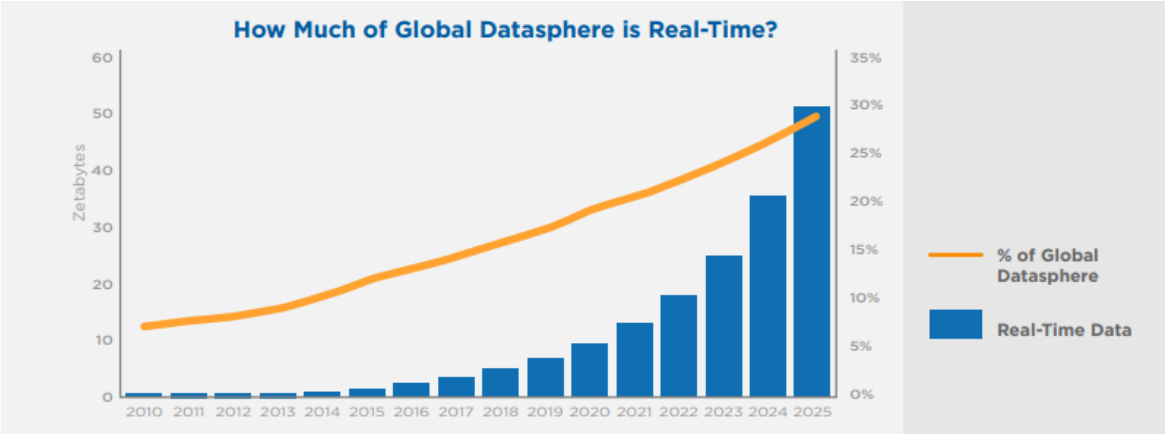

FIGURA 1.3: crescita dei dati prodotti in streaming nel Mondo dal 2010 al 2025. Fonte: IDC

vengono generalmente elaborati in memoria attraverso l'elaborazione distribuita, che può essere necessaria anche quando i set di dati sono relativamente piccoli, uno scenario spesso presente nell'*Internet of Things* (IoT).

Dalle stime effettuate dall'IDC, la crescita dei flussi di dati in *streaming* aumenterà notevolmente nei prossimi anni, arrivando a rappresentare circa un terzo della datasfera globale nel 2025 (Figura 1.3).

1.3.4 Variabilità

La variabilità rappresenta possibili cambiamenti nella velocità del flusso, nel formato, o nel volume, all'interno di un sotto-insieme di dati, che può portare alla necessità di ristrutturare completamente le architetture, gli algoritmi, le tecniche di integrazione e archiviazione. La variabilità dei volumi di dati implica la necessità di aumentare o ridurre le risorse per gestire in modo efficiente il carico di elaborazione: per ottenere una gestione automatizzata ed efficiente si utilizzano le funzionalità di ridimensionamento dinamico del *cloud computing* [7] , che consente maggiore efficienza rispetto al caso in cui il sistema sia progettato per poter gestire il massimo sforzo computazionale atteso, scenario che rischia di condurre ad un sotto-utilizzo del sistema.

Osservando i dati forniti dall'IDC, le stime sulla crescita della quantità dei dati che nei prossimi anni saranno archiviati in cloud pubblici rispetto ai server aziendali e ai consumatori finali, è stimata in forte aumento. (Figura 1.4) [8] .

1.4 Valore

Al contrario della prospettiva tecnologica, la prospettiva manageriale dei Big Data si concentra sulle domande legate al valore. Quindi, partendo dalla definizione della catena del valore ("value chain") si possono identificare i *driver* di creazione di vantaggio competitvo per le aziende che riescono a raccogliere, archiviare elaborare e analizzare grandi quantità di dati.

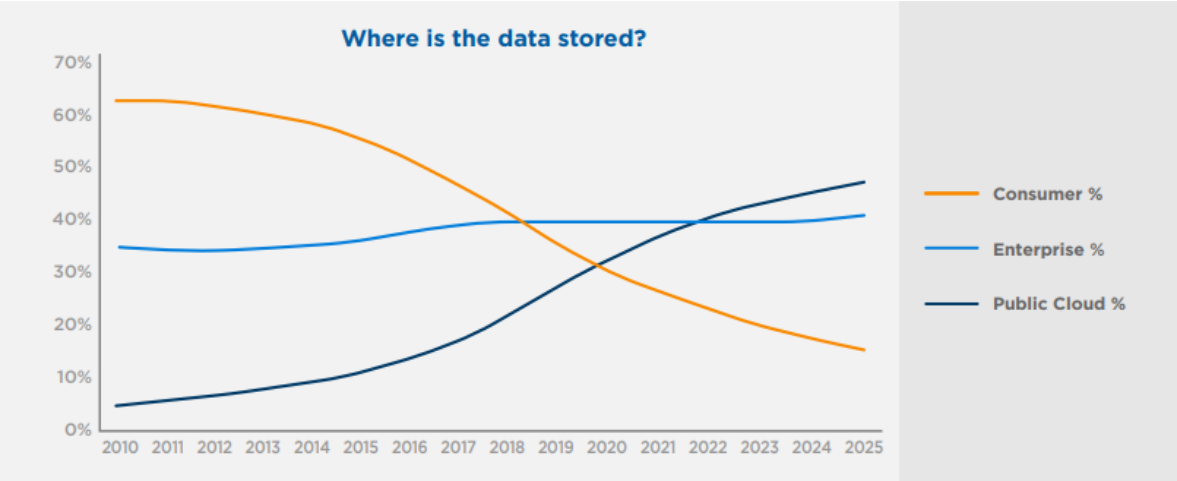

FIGURA 1.4: Strumenti di archiviazione dei dati dal 2010 al 2025. Fonte: IDC

1.4.1 Data Value Chain

La catena del valore inizia ovviamente con la generazione di dati, ossia la cattura di informazioni in un formato digitale. La seconda fase è la raccolta, che comprende la trasmissione e la validazione di più fonti di dati. Il passo successivo è l'analisi, che comporta la scoperta di *pattern* all'interno dei dati, mentre l'ultimo step riguarda lo scambio dei dati ad un utente finale, il quale può essere un cliente, un utente interno all'azienda, o un consulente esterno.

A differenza della maggior parte delle catene del valore, in questo caso i dati non vengono "consumati" dall'utente finale, ma possono essere riutilizzati e diventare in futuro parte di una tendenza storica: per questo si utilizza il termine scambio (Figura 1.5) [9] .

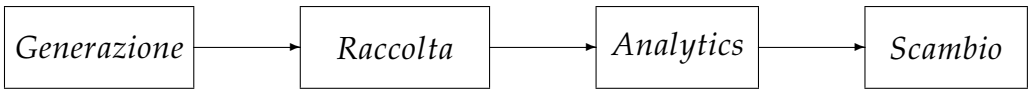

FIGURA 1.5: Data Value Chain

1.4.2 Generazione

La generazione può essere suddivisa in due sotto-fasi: acquisizione e consenso (Figura 1.6). Il primo step è acquisire i dati, la cui fonte può essere interna o esterna e può essere creata da un essere umano o da un dispositivo elettronico. Alcuni esempi sono i dati relativi alle transazioni economiche generati a seguito di pagamenti elettronici, i dati che derivano dalle operazioni degli utenti dei social-network, quelli che si generano dalle ricerche effettuate nei motori di ricerca, quelli che derivano dai check-in effettuati dai

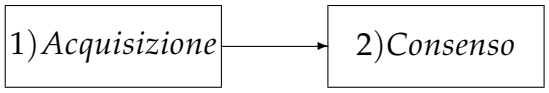

FIGURA 1.6: Generazione

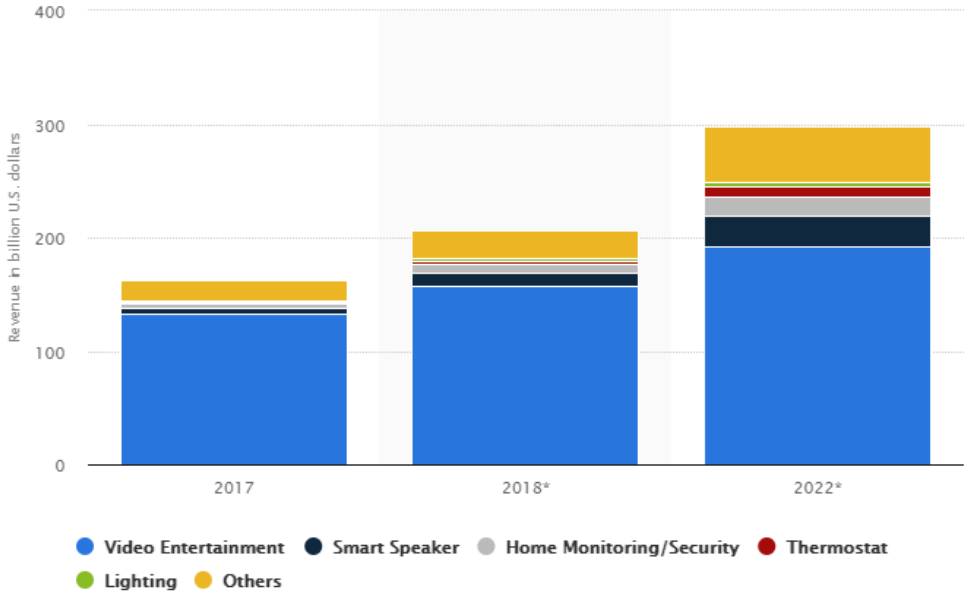

FIGURA 1.7: Mercato globale dei dispositivi Smart Home nel 2017, 2018 e 2022. Fonte: Statista

passeggeri dei veicoli aerei, quelli, in forte crescita, generati dai sensori multi-connessi che rilevano determinati indici, come ad esempio le stazioni meteo e dai dispositivi *Smart home* che consentono di controllare i servizi domestici di riscaldamento, aria condizionata, livello di umidità e che dovrebbero raggiungere un volume globale di vendite attorno ai 300 Milioni di dollari nel 2022 (Figura 1.7).

Per ottenere un vantaggio competitivo le imprese hanno bisogno che i dati generati siano di volume elevato, il più dettagliati possibile e difficilmente replicabili. Un volume di dati elevato permette infatti di effettuare delle analisi migliori rispetto a quelle che invece sono effettuate su un campione, mentre un livello di dettaglio più alto, permette alle aziende che gestiscono le campagne pubblicitarie di indirizzare al meglio le loro attività promozionali, grazie alle quali attrarre sempre più utenti, generando impatti positivi sotto forma di rendimenti di scala e di scopo, oltre a esternalità positive di rete.

Nel settore digitale è interessante osservare come i principali motori di ricerca e social networks, che sono i servizi che possiedono i più importanti effetti di rete, non offrano agli utenti un'opzione per pagare il servizio in cambio di non condividere i propri dati o ricevere pubblicità. Questo differisce dalle piattaforme di archiviazione dati o di *streaming* audo-visivo i cui *provider* devono generare un ricavo direttamente dalla vendita del servizio principale, mentre l'esternalità positiva derivante dai dati posseduti sugli

utenti, è sicuramente secondaria.

Le aziende che riescono ad ottenere il consenso alla cessione dei dati da parte di un pubblico numeroso di utenti, grazie alla fiducia e alla visibilità di cui godono, sono in grado di acquisire un vantaggio competitivo notevole per elaborare algoritmi di *machine-learning* in grado di effettuare campagne di *digital marketing* più efficaci e che portino a maggiori ritorni economici rispetto ai competitor. A differenza delle tradizionali attività commerciali, l'utente che concede i dati all'interno della navigazione internet, non ottiene alcun pagamento in cambio, in quanto il suo ritorno è costituito dal servizio stesso.

Nel settore industriale, infine, le aziende che producono macchinari, possono aggiungere alla propria offerta commerciale la presenza di dispositivi per la rilevazione di dati in tempo reale.

1.4.3 Raccolta

In questa fase i dati devono essere trasmessi dal suo punto di acquisizione al punto in cui sono archiviati utilizzando un' infrastruttura di rete, devono essere integrati insieme con altri dati provenienti da altre fonti e, infine, inseriti in un processo di convalida per garantirne la correttezza e l'integrità. Le due sotto-fasi all'interno di questo segmento della value chain sono pertanto trasmissione e convalida (Figura 1.8).

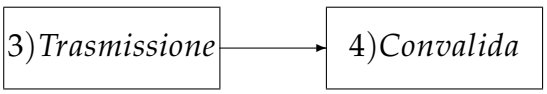

FIGURA 1.8: Raccolta

La capacità di connettere e trasmettere dati tra dispositivi e strumenti di archiviazione è una funzione essenziale all'interno della value chain e man mano che sono raccolti più dati a intervalli più frequenti, anche l'importanza di convalidare questi dati aumenta. Nella fase di convalida convergono tutte le operazioni che permettono di trasformare numerosi dati grezzi, in una collezione di dati che possono essere utilizzati per performare le successive operazioni di data mining e machine learning. All'interno di queste operazioni figurano innanzitutto la verifica della correttezza e dell'integrità dei dati, l'integrazione di dati provenienti da diverse fonti e eventualmente la definizione delle metriche e delle misure che sono interessanti ai fini dell'analisi, con lo scopo di aggregare i dati all'interno di ciò che prende il nome di data mart, ossia un sottoinsieme logico e fisico dei dati iniziali, che permetterà successivamente di ridurre i tempi necessari all'acquisizione dei dati da parte degli analisti.

In passato, le operazioni di salvataggio e archiviazione dei dati erano attività complesse, che richiedevano l'utilizzo di nastri magnetici e sistemi di archiviazione "fisici", con tutti i limiti connessi al limitato spazio di archiviazione, ai costi e alla complessità di gestione, mentre recentemente le tecnologie

basate sul *cloud computing* permettono una capacità di archiviazione pressoché illimitata su base "virtuale", senza la necessità di importanti investimenti iniziali, che rappresentavano in precedenza una barriera all'ingresso.

La crescita della potenza di elaborazione e della capacità di archiviazione degli strumenti a disposizione, correlati alla possibilità per le aziende di utilizzare con costi molto ridotti i servizi di cloud, che permettono di sfruttare risorse elevate, senza dover effettuare grossi investimenti, hanno portato all'abbattimento delle barriere all'ingresso all'interno della trasmissione dei dati.

1.4.4 Analytics

Il termine *analytics* sembrerebbe un sinonimo di analisi, ma in realtà, rappresenta un termine più ampio, che comprende anche la fase di processamento iniziale, utilizzata per trasformare i dati provenienti da database relazionali, data mart o, come si vedrà successivamente, filesystem distribuiti, in possibili informazioni, utilizzando algoritmi matematici per segmentare i dati e costruire collegamenti. Ciò prende il nome di *data mining*.

A partire dagli output del processamento è necessario sviluppare ipotesi intorno al contenuto dei dati e ha inizio la fase di pura analisi. (Figura 1.9).

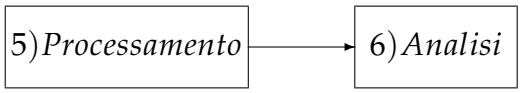

FIGURA 1.9: Analytics

Esistono molti modi in cui l'analisi può essere progettata e condotta, tra cui spiccano gli algoritmi di intelligenza artificiale e di un suo sottoinsieme, quello del *machine learning* [10]. L'analisi è comunemente considerata dagli economisti come il segmento della catena con il potenziale più elevato, poiché rappresenta lo strumento con le maggiori opportunità per le aziende di creare innovazione e proprietà intellettuale. La capacità di generare algoritmi di "autoapprendimento" utilizzando l'intelligenza artificiale è un mezzo per accelerare lo sviluppo di tali imprese, le quali attraverso l'analisi possono inoltre realizzare *business insight* e presentazioni molto più fruibili. Esistono sostanzialmente quattro categorie di operazioni che risiedono nel campo del *Big Data Analytics* [11]:

1. **Analisi descrittiva**. Consiste nel rispondere alla domanda: "cosa sta accadendo?". E' una fase preliminare che permette di identificare i dati storici.

2. **Analisi diagnostica**. Permette di rispondere alla domanda : "cosa è successo?". Essa serve a capire, in caso di un problema, quale ne sia stata la causa.

3. **Analisi predittiva**. Risponde alla domanda: "Cosa dovrebbe accadere in futuro?". Essa utilizza principalmente algoritmi di intelligenza artificiale e machine learning.

4. **Analisi prescrittiva**. Consente di rispondere al quesito :"Cosa bisognerebbe fare in futuro?". Partendo dall'analisi descrittiva, diagnostica e predittiva, si può definire una strategia di azione per il futuro.

Le aziende che intendono performare delle analisi che possano portare a creare vantaggio competitivo, devono investire sul talento delle risorse umane, alla stregua di ogni settore il cui obiettivo risiede nell'innovazione e nella proprietà intellettuale e devono poter accedere a dati rilevanti dai quali poter monetizzare. Nel Mondo dei Big Data questa peculiarità rappresenta la principale barriera all'ingresso per la generazione di un vantaggio competitivo: nell'ipotesi che due aziende diverse riescano a utilizzare algoritmi comparabili in termini di efficienza, sarà quella che possiede dati più rilevanti, che potrà trarne degli *insight* migliori.

Un altro elemento di creazione di valore è la capacità per le imprese di integrarsi orizzontalmente tra i vari segmenti del mercato in cui vengono generati dati: motori di ricerca, servizi in streaming, social network, permettono di migliorare la definizione del profilo degli utenti e conseguentemente, il ritorno economico derivante dalle possibili proposte commerciali che li sono rivolte. La capacità di unire integrazione orizzontale con quella verticale nelle varie fasi della value chain può portare chiaramente a degli squilibri nel mercato, in cui pochi gruppi di operatori riescono a monopolizzare il mercato, in quello che si potrebbe definire un "circolo vizioso" in quanto, se le imprese che possiedono dati eterogenei provenienti da varie fonti, riescono a ottenere risultati analitici migliori dei competitor e conseguentemente a venderli nell'utlimo stadio della value chain, il vantaggio competitivo risulta difficilmente erodibile da parte di eventuali nuovi entranti.

1.4.5 Scambio

L'ultimo step nella catena consiste nel commercializzare quanto ottenuto dalle operazioni di analisi attraverso il *packaging* e la vendita, o l'uso interno, di queste analisi (Figura 1.10). Il *packaging* consiste nell'impachettamento dei

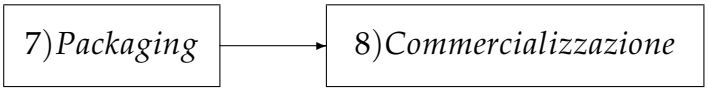

FIGURA 1.10: Scambio

dati in diverse modalità, come la visualizzazione grafica o la reportistica, al fine di essere fruibili per gli utenti di business, che devono prendere decisioni inerenti le attività di marketing da effettuare. La commercializzazione dei dati e delle analisi rappresenta il momento in cui il valore cumulativo derivante dalle fasi precedenti è maggiormente realizzato.

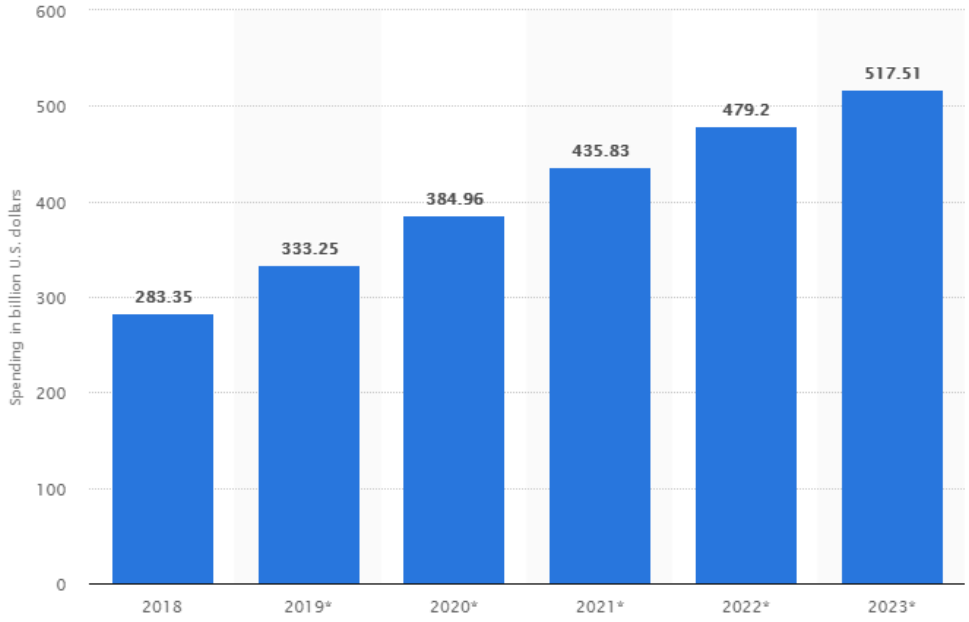

FIGURA 1.11: Crescita del mercato globale del digital advertising. Fonte: Statista

Quanto alle modalità di commercializzazione, esistono tre casistiche principali:

1. **Uso interno**. Ha come scopo l'ottimizzazione dei ricavi (come nel caso dell'utilizzo di strategie di prezzo dinamiche, tipiche delle compagnie di viaggio), la riduzione dei costi, la definizione delle strategie di marketing (ingresso in nuovi mercati, utilizzo di nuovi canali di distribuzione, creazione di nuovi prodotti etc.).

2. **Vendita a terzi dei dati**. Ha come scopo quello di fornire ad aziende terze informazioni sull'andamento dei mercati e dei suoi consumatori.

3. **Vendita a terzi di servizi**. Si tratta prevalentemente di fornire servizi di marketing ad aziende terze, aiutandole a incontrare gli utenti che, sulla base delle analisi, potrebbero essere il target ideale della loro attività commerciale. Per comprendere l'efficacia delle operazioni di advertising e la qualità delle analisi ottenute dai dati si utilizza l'indice del tasso di conversione, che identifica la percentuale di utenti *targetizzati* che hanno effettivamente visitato la pagina dell'azienda sponsor e effettuato degli acquisti o si sono iscritti a qualche servizio. Si tratta di un mercato in continua espansione che dovrebbe superare il valore di 500 milioni di dollari nel 2023, secondo le stime effettuate da Statista [12] (Figura 1.11), di cui i principali detentori di quote di mercato sono i colossi statunitensi Google e Facebook, seguiti a lunga distanza da Alibaba, Amazon e i restanti *player* [13] .

1.5 Business Intelligence: la sintesi dell'estrazione di valore.

Due concetti molto importanti sono quello di Business Intelligence **(**BI) e di Data warehouse **(**DW). Mentre il termine di Data warehouse è utilizzato per indicare senza ambiguità l'output delle operazioni di ***E**xtraction*, ***T**ransformation*, ***L**oading* effettuate sui dati, il termine Business Intelligence viene talvolta sovrapposto a quello di "**strumenti** di Business Intelligence", che nel linguaggio comune è riferito agli strumenti di visualizzazione grafica dei dati e di reportistica, dunque alla fase conclusiva dell'intero processo di trattamento dei dati. Tuttavia, seguendo le definizioni del termine maggiormente accreditate, come quella utilizzata da Gartner, che definisce i servizi di Business Intelligence come "offerte di progettazione, sviluppo e messa in atto di processi aziendali e di integrazione, supporto e gestione delle applicazioni e delle piattaforme tecnologiche. Queste includono applicazioni di business e infrastrutturali per le piattaforme BI, le attività di analytics e le infrastrutture di Data warehouse" [14] , si può definire un **servizio** di BI l'insieme complessivo delle operazioni che, all'interno della value chain va dalla convalida al *packaging*. Va detto che attualmente stanno acquisendo popolarità nella terminologia aziendale, i termini di Advanced Analytics (AA) e Business Analytics (BA), che possono essere definiti, stando alle precedenti assunzioni, come dei sotto-insieme dei servizi di BI.

Durante gli anni '60 i sistemi di supporto alle decisioni erano conosciuti come EDP (Electronic Data Processing), per poi lasciare spazio prima ai SIS (Strategic Information Systems), quindi ai sistemi DSS (Decison Support Systems). Il termine BI ha iniziato ad acquisire popolarità dalla fine degli anni '80 in relazione principalmente alle grandi aziende di retail che, spinte dal desiderio di innovazione organizzativa, desideravano utilizzare le informazioni sui clienti, sui prodotti e sui ricavi per scopi strategici.

La Figura 1.12 mostra uno schema molto semplificato di un'architettura di Business Intelligence.

1.5.1 I processi ETL

I processi ETL si suddividono in tre fasi :

Estrazione. I dati vengono estratti dalle diverse fonti costituite sia da sistemi interni, come i sistemi CRM o ERP, che esterni. Si parla di estrazione completa quando si attinge all'intera fonte per la prima volta e di estrazione incrementale se vengono aggiunti da una fonte solo i dati mancanti rispetto all'ultimo aggiornamento.

Trasformazione. Durante questa fase i dati vengono modificati dal formato iniziale per essere adattati al modello del DW. Il processo di trasformazione include attività come:

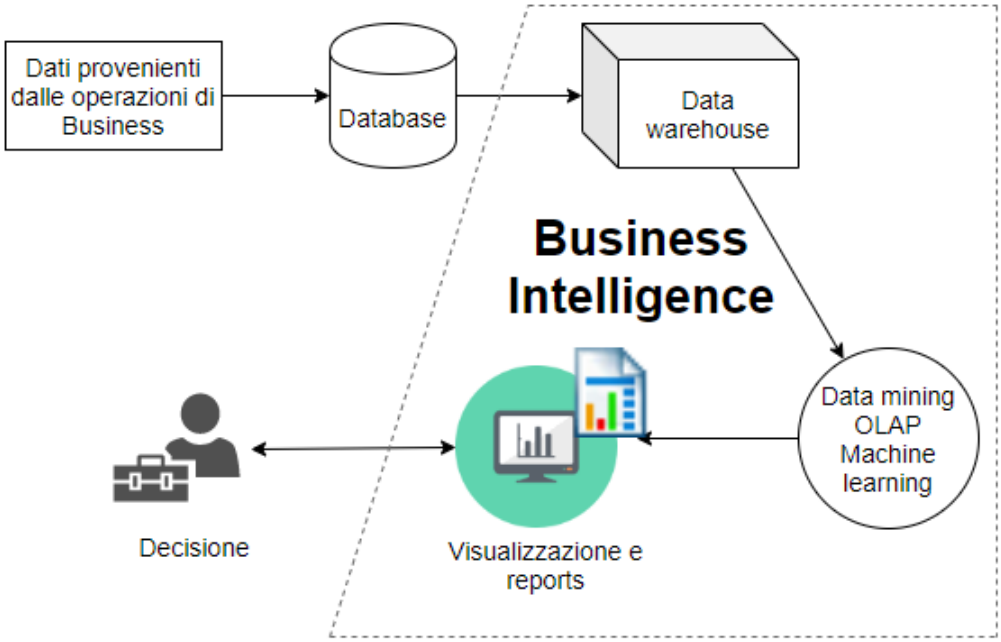

FIGURA 1.12: Architettura di Business Intelligence

Errore	Esempio
Formato sbagliato	Data = 01.16.2019
Violazione del vincolo di unicità	stesso UserID assegnato a due utenti diversi
Valori mancanti	Importo = NULL
Valore errato	Tipotitolo = "mutuo" non esiste
Violazione dei vincoli di dipendenza	lo zipCode di Matera non è 74100

TABELLA 1.1: Esempi di operazioni di pulizia dei dati

- Pulizia, che permette di correggere errori e incoerenze nei dati, che verranno convertiti in un formato standardizzato (Tabella 1.1);
- Integrazione di più fonti;
- Aggregazione.

Caricamento. I dati vengono inseriti nel Data warehouse finale che viene aggiornato con i nuovi dati memorizzati.

Il processo ETL di solito utilizza un database aggiuntivo per fornire un backup, che è chiamato "data staging area" (area di gestione temporanea dei dati) e serve a memorizzare i dati temporanei durante la fase di trasformazione.

1.6 Database e Data warehouse

Un vero e proprio spartiacque nella moderna gestione dei dati è derivato dalla nascita del linguaggio Sql (Structured Query Language) e del modello relazionale, sviluppato da Edgar F. Codd negli anni '70 . Una caratteristica dei modelli di dati relazionali è il loro uso di un linguaggio unificato durante la navigazione, la manipolazione e la definizione dei dati, anziché l'utilizzo di linguaggi separati per ogni attività.

Negli anni è stato portato avanti il modello relazionale all'interno di strumenti definiti database management system (**DMBS**), che andavano a formare insieme il **RDBMS** (Relational database management system).

Questo modello organizza i dati in una o più tabelle (o "relazioni") di colonne e righe, con una chiave univoca che identifica ciascuna riga. Le righe sono anche chiamate record o tuple, mentre le colonne sono anche chiamate attributi. In generale, ogni tabella / relazione rappresenta un "tipo di entità" (ad esempio cliente o prodotto), le righe rappresentano istanze di quel tipo di entità (come "Paolo" o "sedia") e le colonne rappresentano i valori attribuiti a quell'istanza (come l'indirizzo del cliente o il prezzo della sedia).

Negli anni le tecniche di gestione dei database relazionali hanno portato a dei processi di **normalizzazione** delle tabelle, con l'obiettivo di ridurre la ridondanza dei dati e migliorare l'integrità dei dati.

Il modello non relazionale, spesso identificato anche con il termine *No-Sql*, indica invece un distacco radicale rispetto al modello tradizionale e le sue versioni più comuni sono il modello documentale e il modello a grafo, generalmente più idonei per la memorizzazione di dati non strutturati e più veloci nelle operazioni di lettura dati.

Il modello relazionale si è rivelato molto adatto per l'elaborazione parallela, l'elaborazione client-server e le GUI (interfacce utente grafiche) e mediante un sistema di gestione del modello relazionale (RDBMS) è stato concesso a più utenti di accedere contemporaneamente allo stesso database. I RDBMS sono risultati efficaci per attività OLTP, ossia di tipo transazionale, quali le attività che deve gestire uno sportello bancario, ma nel momento in cui i flussi di dati generati sono diventati di dimensioni considerevoli, non si sono rivelati altrettanto efficaci nella gestione di attività di tipo OLAP (On Line Analitical Processing), ossia di selezione dei dati, attraverso delle ricerche all'interno dei database, finalizzate ad operazioni di analisi o più semplicemente alla restituzione di informazioni.

Le prime risposte tecnologiche sono pervenute attraverso la nascita dei Data warehouse, che possono essere descritti come dei contenitori centralizzati di informazioni, in cui i dati confluiscono a partire dagli stessi RDBMS e da altre fonti e a cui gli analisti aziendali e i decisori strategici possono accedere attraverso diversi tipi di client per effettuare le opportune analisi.

Il Data warehouse è una repository che contiene dati integrati da origini diverse, recuperati e raccolti periodicamente, memorizzati in archivi dimensionali o normalizzati. [15]. Si tratta di una collezione fisicamente separata dai sistemi operazionali che presenta le seguenti caratteristiche:

1. **Orientata ai soggetti** : significa che tutti i dati relativi agli stessi soggetti di business sono collegati.
2. **Integrata** : i dati sono raccolti da diversi applicativi.
3. **Non volatile** : il DW funziona in sola lettura e i dati non sono mai aggiornati o eliminati.
4. **Variabile con il tempo** : significa che vi è un aggiornamento dei dati ad intervalli predefiniti.

Esistono due approcci fondamentali nella realizzazione di un Data warehouse:

1. **Query-driven** : quando una query arriva al sistema intregrato, un mediatore genera delle sottoquery per i vari DBMS eterogenei, mette insieme i risultati e risponde alla query originale.
2. **Update-driven** : l'informazione è integrata in anticipo e dunque non c'è interferenza tra query al DW e query ai singoli database.

Esistono cinque architetture fondamentali (Figura 1.13) [16]:

1. **Data mart indipendenti** : significa che esistono diversi data mart non collegati tra loro, ciascuno dei quali facente riferimento a una specifica divisione aziendale, o una determinata area di business.
2. **Bus** : simile alla struttura precedente, presenta però un anello di congiunzione tra i data mart, costituito da un modello unico per tutti i data mart.
3. **Architettura hub e spoke** : in questo caso esiste un "livello riconciliato" in cui ci sono diversi data mart aggregati secondo determinate metriche.
4. **Architettura centralizzata** : in questo caso non esistono singoli data mart ma esclusivamente una repository centralizzata.
5. **Architettura confederata**: rappresenta un sistema integrato di diversi DW e data mart realizzato attraverso tecniche come querying distribuito, ontologie e interoparabilità dei metadata.

1.6.1 Le principali fonti di un Data warehouse.

Come anticipato, i DW utilizzano diverse fonti, che possono essere schematizzate nelle seguenti tipologie:

- **DBMS**

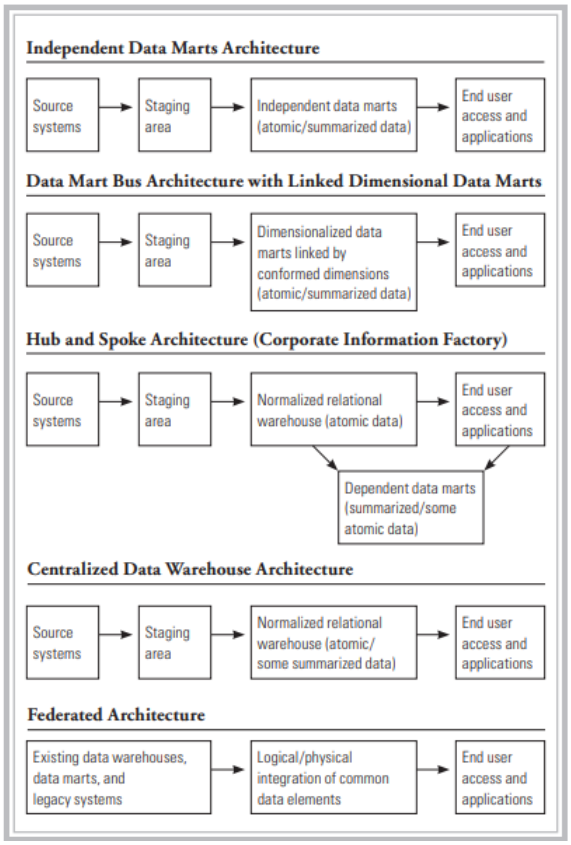

FIGURA 1.13: Le cinque architetture di Data warehouse.

- **Database semplice**: è una singola tabella generalmente archiviata e rappresentata come un semplice file testuale, come nel caso del Csv (Comma Separated Value), una tabella in formato testuale ASCII, un foglio di calcolo, il cui formato più comune è il XLSX utilizzato da Microsoft Excel, o un file Json (JavaScript Object Notation), che comprende due strutture: una collezione di coppie nome/valore e una lista ordinata di valori.

- **Altri Data warehouse**. Si tratta per l'appunto di DW con un'architettura confederata.

Il modello maggiormente utilizzato per le tecniche di aggregazione delle diverse fonti che compongono il DW è il cosiddetto **Dimensional Fact Model** (DFM), ideato da Golfarelli e Rizzi, che prevede la presenza di due entità fondamentali : i **fatti**, che rappresentano i calcoli aggregati, ossia le misure prese su una serie di tabelle sottoposte a un'operazione di *join* (intersezione)[3], come somme, medie, minimi, massimi, conteggi e le **dimensioni** che rappresentano le variabili per cui i calcoli sono effettuati.

Il modello dimensionale può essere rappresentato dallo schema "a stella" o quello a "fiocco di neve". Mentre il primo schema ha una tabella dei fatti al centro e tutte le tabelle dimensionali sono collegate direttamente al fatto

[3]Una clausola Sql di tipo join, mette insieme colonne di una o più tabelle, sulla base di uno specifico valore di una colonna che i record delle tabelle hanno in comune.

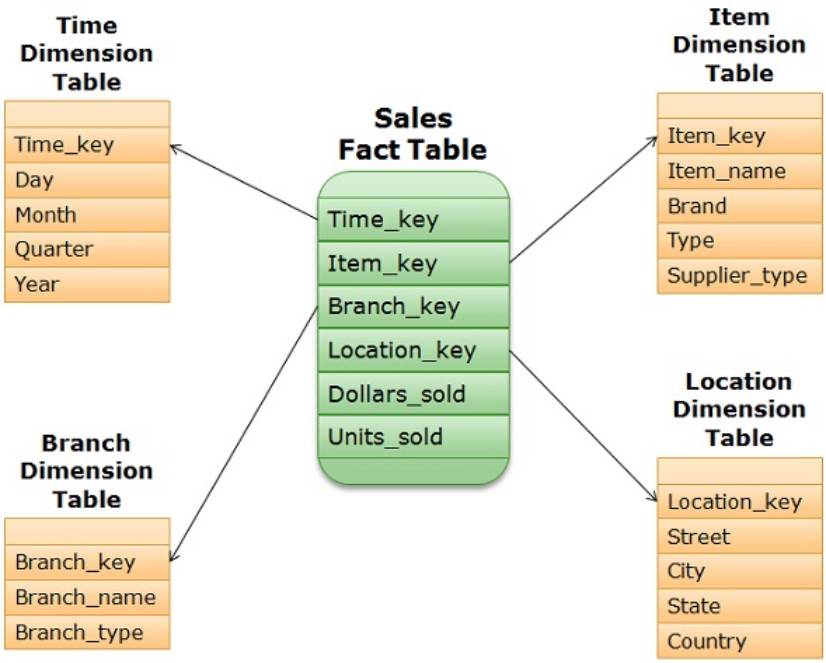

FIGURA 1.14: Dimensional Fact Model : schema a stella

centrale (Figura 1.14), nel modello a fiocco di neve, pur confermandosi la presenza della tabella dei fatti centrale, una tabella dimensionale potrebbe essere collegata anche ad un'altra tabella dimensionale, nel caso in cui ne rappresenti un suo attributo (Figura 1.15).

1.6.2 OLAP

I data mart vengono inseriti all'interno dei processi di OLAP. Questi sono degli strumenti che facilitano le operazioni di analisi che saranno effettuate sui data mart. Esistono tre tipologie di server OLAP:

1. **ROLAP** (Relational Online Analytical Processing) che forniscono i dati direttamente dal Data warehouse principale, costruendo una sua vista dinamica. In questo sistema i dati vengono fisicamente memorizzati in vettori e l'accesso è di tipo posizionale. Il sistema alloca una cella per ogni possibile combinazione dei valori delle dimensioni e l'accesso ad un fatto avviene in modo diretto, sulla base delle coordinate fornite.

2. **MOLAP** (Multidimensional Online Analytical Processing) che forniscono i dati a partire da database multidimensionali (MDDB), che concettualmente sono associati a dei cubi, le cui dimensioni sono delle dimensioni del modello, mentre i punti interni simboleggiano tutte le misure che si possono effettuare, ma dato che solitamente il modello è rappresentato da più di tre dimensioni, avrebbe più senso utilizzare il termine di ipercubo.

   In questo sistema il vantaggio è la velocità di accesso, mentre il limite è legato alla elevata complessità del cubo nel momento in cui aumentano le dimensioni e le loro gerarchie.

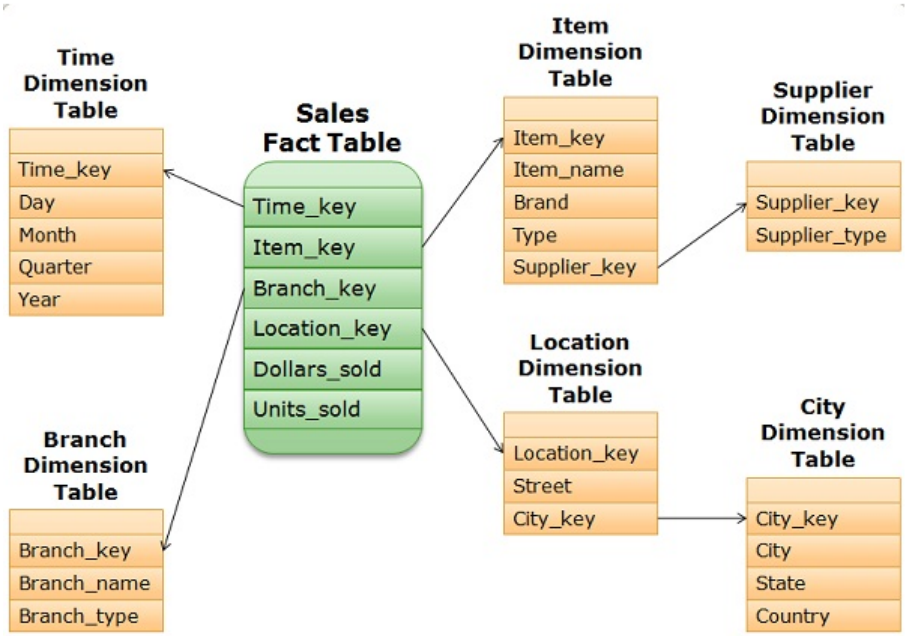

FIGURA 1.15: Dimensional Fact Model : schema a fiocco di neve

Nel sistema MOLAP viene generata dunque una vista statica dei dati.

3. **HOLAP** (Hybrid Online Analytical Processing), che utilizzano varie combinazioni delle tecnologie ROLAP e MOLAP.

1.6.3 MOLAP

Nei server MOLAP vengono utilizzati dei modelli simili al DFM, in cui le entità sono suddivise in fatti e dimensioni. Quest'ultime possono essere organizzate in una gerarchia, ossia un insieme di relazioni padre-figlio, in cui l'elemento padre riassume anche i suoi figli e può essere ulteriormente aggregato come figlio di un altro padre. Ad esempio, il genitore di "Novembre 2019" è il "quarto trimestre 2019", che è a sua volta figlio dell'anno "2019". Nella letteratura Data warehouse, ognuno dei cubi n-dimensionali è chiamato cuboide: ci sono cuboidi diversi a seconda delle dimensioni che vengono scelte e del livello di dettaglio di ogni dimensione.

I sistemi MOLAP permettono di attivare sui cuboidi le seguenti operazioni:

- **Pivoting** : consente agli analisti di ruotare il cubo nello spazio. Ad esempio, le città potrebbero essere organizzate in verticale e i prodotti in orizzontale durante la visualizzazione dei dati per un determinato trimestre.

  Il pivot potrebbe sostituire i prodotti con i periodi di tempo per permettere di visualizzare di un singolo prodotto nell'arco temporale.

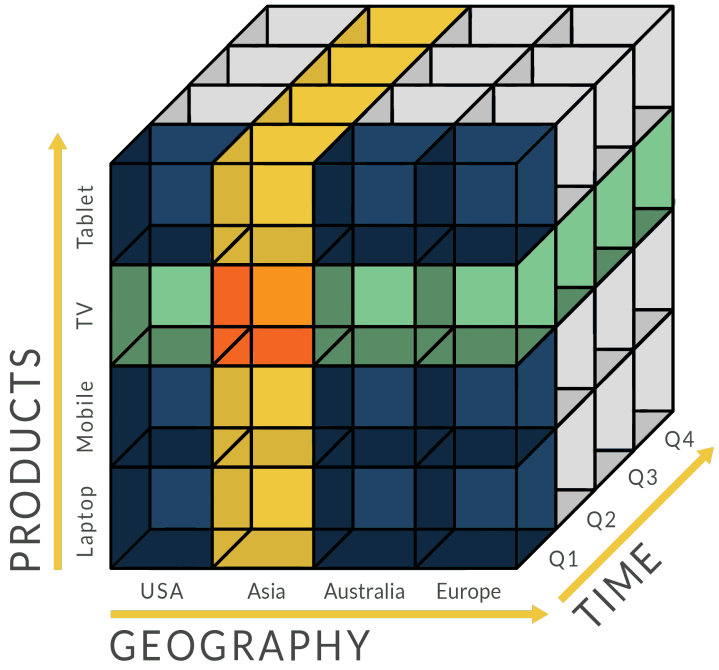

FIGURA 1.16: Esempio di operazione di Dicing in un cubo OLAP

- **Slicing** : è l'atto di scegliere un sottoinsieme rettangolare di un cubo selezionando un singolo valore per una delle sue dimensioni. Ad esempio si potrebbe scegliere di analizzare i dati di vendita di tutte le aree geografiche e di tutte le categorie di prodotti solo all'interno del primo trimestre, il che porterebbe a "tagliare" dal cubo i restanti trimestri.

- **Dicing** : l'operazione produce un sottocubo consentendo all'analista di scegliere valori specifici di più dimensioni. Ad esempio l'anlalista potrebbe voler scegliere di analizzare solamente la categoria prodotto "TV", nella zona geografica "Asia", nel primo trimestre (Figura 1.16).

- **Drill Down/Up** : permette all'utente di raggruppare le dimensioni secondo livelli più o meno alti di gerarchia. Corrisponde alla clausola "GROUP BY" di una query Sql.

- **Roll-up** : comporta il riepilogo dei dati lungo una dimensione, che può consistere in una funzione aggregata o nell'applicare una serie di formule come "profitto = vendite - spese".

  Le funzioni di aggregazione che possono essere determinate dalle celle sono note come funzioni di aggregazione scomponibili e consentono un calcolo efficiente. Ad esempio, è facile supportare conteggi, massimi, minimi e somme, poiché possono essere calcolati per ogni cella del cubo e quindi sommati, in quanto una somma complessiva (o un conteggio, un minimo o un massimo) è una somma di somme secondarie, ma è difficile supportare il calcolo della mediana, perché deve essere

calcolata separatamente per ogni vista: la mediana di un insieme non è la mediana delle mediane dei sottoinsiemi.

I server MOLAP nel momento in cui ricevono richieste dati, devono rispondere con il cuboide specifico e per fare ciò esistono due tipi di progettazione:

1. **Materializzazione totale** : tutti i cuboidi possibili sono già stati pre-calcolati e dunque il server deve soltanto restituire quello richiesto.

2. **Materializzazione parziale** : alcuni cuboidi sono già stati pre-calcolati, mentre altri vengono calcolati nel momento in cui il server non individua nessun cuboide pre-esistente.

La materializzazione totale sarebbe la più efficiente, ma richiede molta memoria: 2^n cuboidi per n dimensioni, se esistesse per ogni dimensione solo un livello di gerarchia, per questo di solito si preferisce pre-calcolare solo i cuboidi più utilizzati.

1.7 Data Mining

I dati conservati all'interno dei server OLAP sono pronti per essere utilizzati per le operazioni di analytics. Sebbene fortemente correlati tra loro, il termine machine learning è formalmente distinto dal termine data mining. Con esso si indica il processo computazionale di scoperta di pattern, che sono una rappresentazione sintetica di grandi dataset caratterizzati da attributi di **Validità**, **Comprensibilità** per l'utente e **Potenziale utilità** per l'estrazione di valore. Ad esempio, dire che "i clienti che acquistano formaggio e latte acquistano anche pane il 90 percento delle volte" è un pattern, utile per un negozio di alimentari. Allo stesso modo, poter dire che "le persone con una pressione maggiore di 160 hanno un alto rischio di morire per un malore al cuore" permette al campo della medicina di trattare tali pazienti con attenzione a eventuali disturbi cardiaci. All'interno di un progetto di data mining è richiesto un approccio strutturato in cui la scelta del migliore algoritmo rappresenta solo la fase conclusiva. Una delle proposte di approccio che può essere utilizzata, è la cosiddetta metodologia **CRISP-DM** [17] (Figura 1.17), che suddivide il lavoro in 6 fasi:

1. Capire il Business: comprendere gli obiettivi del progetto per soddisfare i bisogni dell'utente che devono essere tradotti in un problema di data mining.

2. Capire i dati : analizzare le caratteristiche principali dei dati e eventuali problemi di qualità.

3. Preparare i dati : selezionare record e attributi da utilizzare nel modello effettuando eventuali trasformazioni.

4. Costruire il modello : scegliere il modello migliore per il tipo analisi che si vuole effettuare.

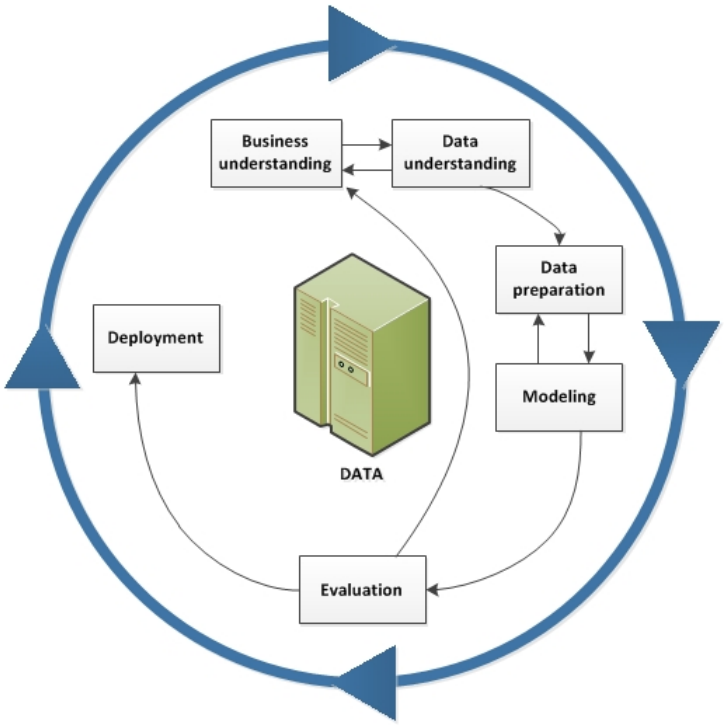

FIGURA 1.17: Modello CRISP-DM

5. Valutare il modello : testare il modello ottenuto dalle fasi precedenti per capire se risponde ai bisogni dell'utente.

6. Deployment: mettere il modello a disposizione degli utenti.

Esistono diverse categorie di pattern, tra cui le più importanti sono le **regole associative**, i **classificatori** e gli algoritmi di **clustering**.

1.7.1 Regole associative

Le regole di associazione sono un metodo di apprendimento automatico utilizzato per scoprire relazioni tra variabili.

Basandosi sul concetto di "regole forti", Rakesh Agrawal, Tomasz Imieliński e Arun Swami [18] hanno introdotto tali regole per scoprire regolarità tra prodotti in dati di transazioni su larga scala registrati dai sistemi di punti vendita (POS) nei supermercati.

Dati $I = \{i_1, i_2, \ldots, i_n\}$ e $D = \{i_1, i_2, \ldots, i_m\}$ dove $\boldsymbol{I}$ è un insieme di n elementi binari, mentre $\boldsymbol{D}$ è un database di m transazioni di cui ognuna ha un identificativo univoco e contiene un sottoinsieme degli elementi in $\boldsymbol{I}$ e dati due sottoinsiemi di $\boldsymbol{I}$, $\boldsymbol{X}$ e $\boldsymbol{Y}$, $\boldsymbol{X}$ si definisce associato ad $\boldsymbol{Y}$ con un determinato livello di ***Confidenza*** definito come:

$$\text{Confidenza}(X \Rightarrow Y) = \text{Supporto}(X \cup Y)/\text{Supporto}(X)$$

transaction ID	latte	uova	pane	acqua	carne
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

TABELLA 1.2: Esempio database con 5 transazioni e 5 elementi.

Per **Supporto**, considerando ***T*** come il sottoinsieme delle transazioni ***t*** in cui è presente ***X*** nel database, si intende:

$$\text{Supporto}(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

vale a dire il numero di volte che il set di elementi ***X*** ha valore 1 all'interno di ***T***, mentre la confidenza dell'associazione di X a Y rappresenta il numero di volte in cui l'unione tra i set di elementi ***X*** e ***Y*** ha valore 1, rispetto al Supporto di ***X***. Osservando la tabella 1.2, il Supporto del set $\{\text{uova}, \text{pane}\}$ è $1/5 = 0.2$, cioè la frazione di DB in cui entrambi gli elementi uova e pane hanno valore 1, mentre l'associazione $\{\text{uova}, \text{pane}\} \Rightarrow \{\text{latte}\}$ ha una confidenza di $0.2/0.2 = 1$, in quanto nell'unica transazione in cui "uova" e "pane" hanno valore 1, anche "latte" ha valore 1. Un altro importante indicatore è quello del Lift, che definisce il grado di dipendenza tra due set di elementi, ***X*** e ***Y*** ed è definito come segue:

$$\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X) \times \text{supp}(Y)}$$

Nel caso in cui il Lift valga 1, indicherebbe la totale indipendenza tra X e Y. Per valori > 1 esiste una dipendenza che cresce all'aumentare del valore, mentre per valori < 1 X e Y sono set di elementi sostituti, cioè la presenza di uno influenza negativamente la presenza dell'altro. Nel caso della regola vista in precedenza, il Lift ha valore 0.5. Le regole associative sono utilizzate prevalentemente per operazioni di

- **Marketing**: la relazione tra prodotti acquistati può portare a definire strategie di marketing, sia sul set di elementi *conseguenti*, per cercare di capire come incrementare la loro vendita, che sul set di elementi *antecedenti*, dai quali sarebbe possibile capire che impatto avrebbe smettere di venderli.

- **Disposizione delle merci**: le regole di associazione possono portare a strategie di disposizione delle merci finalizzate all'acquisto congiunto degli elementi associati.

- **Gestione dell'inventario**: le regole di associazione possono permettere alle società di capire in anticipo quali prodotti andrebbero utilizzati

sulla base di un determinato input. Esempio: azienda che ripara elettrodomestici vuole ridurre il numero di visite alle abitazioni dei clienti e per fare ciò cerca di associare al tipo di segnalazione, il tipo di pezzi necessario ad effettuare la riparazione.

1.7.2 Classificatori

Dato un set di dati costituito da coppie $\boldsymbol{x}$ e $\boldsymbol{y}$, dove x indica un elemento della popolazione del dataset e y la classe a cui appartiene, una regola di classificazione h(x) è una funzione che associa ad ogni elemento x una classe, o etichetta y:

$$\hat{y} = h(x).$$

Una classificazione binaria è tale che l'etichetta y può assumere solo due valori.

La classificazione utilizza una serie di modelli matematici e statistici per effettuare questa assegnazione, ma esiste un margine di errore, il quale definisce l'accuratezza del modello.

In una classificazione binaria, gli elementi che non sono classificati correttamente sono denominati falsi positivi e falsi negativi.

La regola di classificazione individua un modello che attribuisce a ciascun record uno specifico attributo di classe, in funzione del valore di alcuni attributi. L'obiettivo è quello di prevedere correttamente a quale classe assegnare ciascun record: per questo si utilizzano un dataset di training, che serve ad *educare* il modello ed uno di test che serve a capire l'accuratezza del modello che è stato realizato. La classificazione viene utilizzata per raggiungere i seguenti scopi :

1. **Direct marketing**: consiste nel classificare i consumatori finali sulla base delle loro caratteristiche (età, stipendio, provenienza geografica, sesso, etc.) per assegnarli o meno alla classe dei possibili interessati all'acquisto di un nuovo prodotto o servizio e quindi contattare solo coloro che entrano nella classe obiettivo, risparmiando sul costo della pubblicità.

2. **Individuazione di frodi** : consiste nel classificare le transazioni bancarie come fraudolente sulla base di alcune informazioni come la distanza del luogo geografico di acquisto rispetto a quello di residenze dell'utente, l'affinità del prodotto acquistato con quelli generalmente voluti dall'utente, l'orario in cui viene effettuato il pagamento etc.

3. **Individuazione dell'insoddisfazione del cliente**: classificando i clienti come soddisfatti o meno, le aziende possono cercare di prevenire la loro cessazione dal contratto attraverso proposte commerciali vantaggiose. Per prevedere ciò le aziende utilizzano i dati sull'uso del servizio, sul numero di operazioni effettuate, etc.

Esistono diverse tecniche di classificazione [19]:

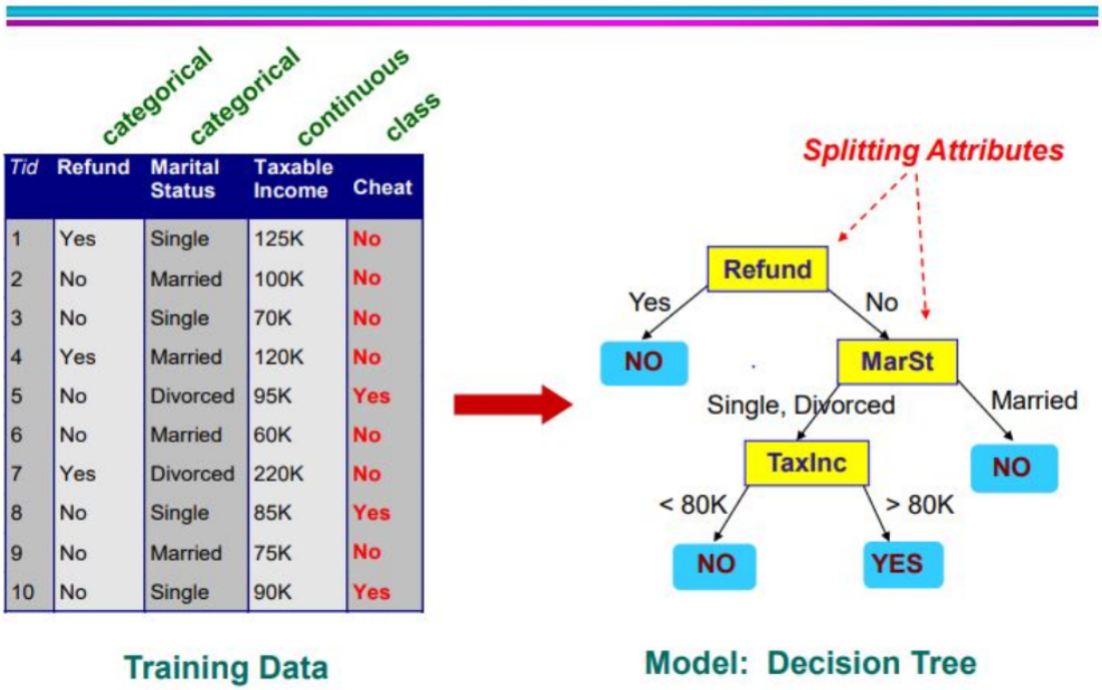

FIGURA 1.18: Esempio di albero delle decisioni

- **Alberi decisionali**
- **Classificatori rule-based**
- **Reti neurali**
- **Naïve Bayes**
- **Macchine a vettori di supporto**

1.7.3 Alberi decisionali

Un albero di decisione è un modello ad albero in cui ogni nodo rappresenta una variabile, mentre una ramificazione verso un nodo figlio (detto anche foglia) rappresenta un possibile valore per quella variabile e una foglia il valore predetto. Di conseguenza, ogni nodo interno risulta essere una macroclasse costituita dall'unione delle classi associate ai suoi nodi figli.

Dalla Figura 1.17 emerge un ritratto di quello che un albero delle decisioni rappresenta. Nel caso specifico esso cerca di definire se una persona tradisce il proprio coniuge sulla base di alcune variabili, mentre l'output finale rappresenta un valore binario della classe *cheat*, ossia la previsione stessa.

Nella definizione del modello dell'albero decisionale acquisiscono molta importanza due criteri: decidere come suddividere ciascun nodo padre nei successivi nodi figli e quando fermare l'algoritmo dall'effettuare nuove ramificazioni. Le variabili all'interno del dataset si distinguono principalmente in categoriche (o nominali), ordinali e continue. Le variabili categoriche e ordinali possono essere divise o per ciascuno dei valori che possono assumere,

oppure per gruppi di valori, mentre le variabili continue possono essere divise in modo binario, come nella Figura 1.17, dove si apre una ramificazione tra redditi inferiori o superiori a 80.000, ma anche per range di valori, come ad esempio gli intervalli [0,75.000], [75.000,150.000], [150.000, 225.000]. La scelta della strategia di ramificazione deve essere rivolta a premiare le soluzioni in cui il risultato preveda una distribuzione omogenea dei record all'interno dei vari rami; esiste un indice di purezza, il cosiddetto **GINI** che dà una panoramica dell'omogeneità delle ramificazioni. In un determinato nodo t dell'albero decisionale, l'indice è così calcolato, considerando le j classi esistenti in quel nodo:

$$GINI(t) = 1 - \sum_j [p_j]$$

dove p rappresenta la frequenza con cui la classe j compare nel nodo t. L'indice è compreso tra 0 e 1 ed è sintomo di una distribuzione dei record migliore se assume valori vicini allo 0, in quanto la ramificazione avrebbe creato dei nodi i cui record sono sbilanciati su una singola classe. Supponendo che in un nodo esistano 5 elementi di *Classe1* e 5 elementi di *Classe2* il valore dell'indice di Gini sarà : 1 - (0,5$\hat{(}$2) + 0,5$\hat{(}$2)) = 0,5. Nel caso nello stesso nodo esistano 9 elementi di *Classe1* e solo 1 di *Classe2*, l'indice di Gini è : 1 - ((9/10)$\hat{2}$ - (1/10)$\hat{2}$) = 0,18.

Un indice alternativo che fornisce indicazioni simili all'indice di GINI è l'*entropia*, che è calcolata in un generico nodo t come:

$$Entropia(t) = -\sum_j [p_j * log_2(p_j)]$$

e identifica una purezza maggiore del nodo al tendere del suo valore verso lo 0.

Spostando l'attenzione sulle tecniche da utilizzare per decidere quando interrompere le ramificazioni sui nodi, occorre che la profondità dell'albero, sia sufficiente per fornire abbastanza informazioni utili a performare un'analisi accurata e dunque ad evitare il cosiddetto fenomeno di *underfitting*, ma non così profondo da ricadere nel rischio di *overfitting*, cioè di fornire un numero troppo elevato di dettagli durante la fase di training, che comporti successivamente l'utilizzo da parte del modello di variabili non utili alla classificazione, che verrrebbero utilizzate durante la fase di test per determinare la classe giusta dei vari record del dataset: si parla di *overfitting dovuto alla presenza di rumore*, ma esiste anche un fenomeno in cui ci sono troppi dati solo di una sotto-classe della popolazione, il che porta il modello a stimare erroneamente quei record che occupano uno spazio non coperto dal campione stesso (overfitting dovuto a mancanza di dati).

Esistono diverse tecniche utilizzate per evitare l'*overfitting*, alcune utilizzate in anticipo (**pre-pruning**), altre successivamente (**post-pruning**). Tra le tecniche utilizzate in anticipo ci sono:

- Fermare l'algoritmo su quei nodi dove tutti i record appartengono alla stessa classe.

- Fermare l'algoritmo su quei nodi dove tutti i record hanno gli stessi valori su tutti gli attributi.
- Fermare l'algoritmo quando espandendo il nodo, non migliora l'indice di Gini o di Entropia.
- Fermare l'algoritmo quando la distribuzione dei record nelle classi è indipendente dagli attributi.

1.7.4 Classificatori rule-based

I classificatori "rule-based" sono dei classifficatori "se - quindi", che assegnano i record del database ad una determinata classe, sulla base di alcune regole o parametri. Nella Tabella 1.3 sono presenti alcuni esempi di come questo tipo di classificatori può funzionare. Sono emerse dagli esempi presentati le

nome	sangue	partorisce	vola	vive in acqua	specie
uomo	caldo	sì	no	no	mammifero
serpente	freddo	no	no	no	rettile
salmone	freddo	no	no	sì	pesce
balena	caldo	sì	no	sì	mammifero
rana	freddo	no	no	a volte	anfibio
pipistrello	caldo	sì	sì	no mammifero	
piccione	caldo	no	sì	no	uccello
squalo	freddo	sì	no	sì	pesce
tartaruga	freddp	no	no	a volte	rettile
pinguino	caldo	no	no	a volte	uccello
anguilla	freddo	no	no	sì	pesce
salamandra	freddo	no	no	a volte	anfibio
gufo	caldo	no	sì	no	uccello

TABELLA 1.3: Esempio di classificatori "rule-based".

seguenti 5 regole :
R1: (Partorisce = no) && (Vola = sì) = Uccello
R2: (Partorisce = no) && (Vive in acqua = sì) = Pesce
R3: (Partorisce = sì) && (Sangue = caldo) = Mammifero
R4: (Partorisce = no) && (Vola = no) = Rettile
R5: (Vive in acqua = a volte) = Anfibio
Un set di regole di classificazione si definisce **mutualmente esclusivo**, quando le regole sono indipendenti tra loro e ogni record fa riferimento ad almeno ad una regola ed **esaustivo** se prevede che ogni record possa essere classificato per ogni possibile combinazione degli attributi.

1.7.5 Classificazione Bayesiana

Un classificatore Bayesiano si basa sull'applicazione del teorema di Bayes, il quale definisce la probabilità condizionata. Considerando un insieme di

alternative

$$A_1, \ldots, A_n A_1, \ldots, A_n$$

che partizionano lo spazio degli eventi, si trova la seguente espressione per la probabilità condizionata di A rispetto all'evento E:

$$P(A_i|E) = \frac{P(E|A_i)P(A_i)}{P(E)} = \frac{P(E|A_i)P(A_i)}{\sum_{j=1}^{n} P(E|A_j)P(A_j)}$$

- P(A) è la probabilità a priori di A, che non tiene conto di nessuna informazione riguardo E.
- P(A | E) è la probabilità condizionata di A, noto E. Viene anche chiamata probabilità a posteriori, visto che dipende dallo specifico valore di E.
- P(E | A) è la probabilità condizionata di E, noto A.
- P(E) è la probabilità a priori di E.

Il classificatore Bayesiano sostituisce l'evento A con la classe C e l'evento E con il record $\boldsymbol{X}$. La classificazione avverrà dunque assegnando $\boldsymbol{X}$ alla classe per la quale la probabilità di appartenervi è maggiore.

Per definire P(C | X) si usano generalmente i seguenti approcci:

1. **Ipotesi ingenua** : si suppone che la probailità di P(C | X) sia $P(C|X) = \prod_i P(C|X_i)$ il che equivale a supporre l'indipendenza dei singoli attributi $X_1, X_2, \ldots, X_i, \ldots X_n$ tra di loro: un'ipotesi che spesso non è veritiera.
2. **Calcolare la probabilità di ciascun attributo** X_i: si utilizzano le distribuzioni di probabilità per ciascun attributo sulla base del suo valore per calcolare la probabilità che appartenga alla classe C all'interno del dataset.

1.7.6 Reti neurali

Le reti neurali artificiali sono reti elettroniche di neuroni basate sulla struttura neurale del cervello. Elaborano i record uno alla volta e apprendono confrontando la loro classificazione con la classificazione effettiva nota del record. Gli errori della classificazione iniziale del primo record vengono reimmessi nella rete e utilizzati per modificare l'algoritmo di rete per ulteriori iterazioni. Un neurone in una rete neurale artificiale è

- Un insieme di valori di input x_i e pesi associati w_i.
- Una funzione (f) che somma i pesi e mappa i risultati su un output (y).

Esistono degli strati intermedi in cui l'algorirmo opera, che sono definiti come livelli nascosti (Figura 1.18). Nella fase di addestramento, è nota la classe corretta per ogni record e ai nodi di output potrebbero essere assegnati valori

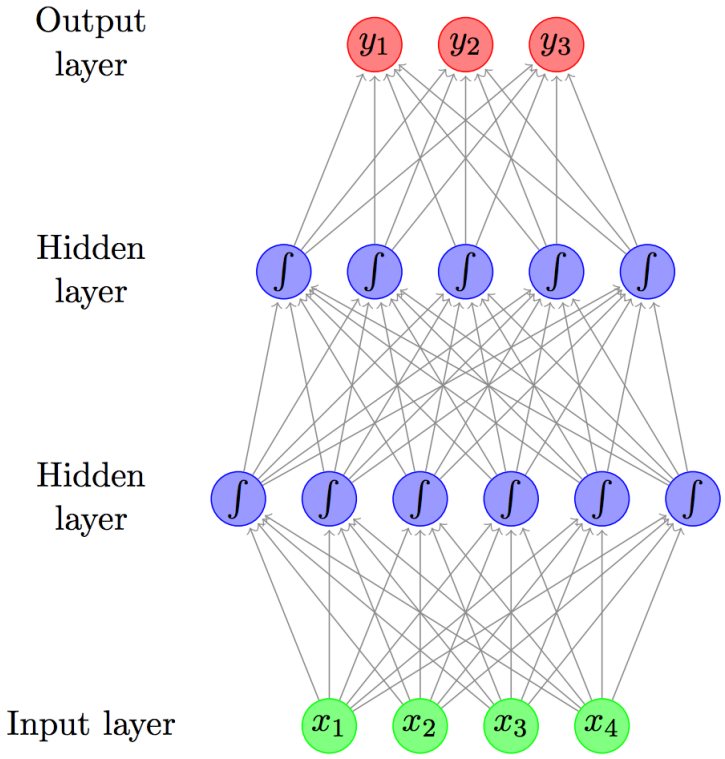

FIGURA 1.19: Esempio di rete neurale

1 per il nodo corrispondente alla classe corretta e 0 per gli altri ed è quindi possibile confrontare i valori calcolati della rete per i nodi di output con quelli corretti e calcolare un valore che esprime l' errore per ciascun nodo (la regola Delta). Questi termini di errore vengono quindi utilizzati per regolare i pesi nei livelli nascosti in modo che , durante la successiva iterazione, i valori di output siano più vicini ai valori corretti. I vantaggi delle reti neurali comprendono la loro elevata tolleranza ai dati rumorosi, nonché la loro capacità di classificare modelli su cui non sono stati addestrati. L'algoritmo di rete neurale più popolare è l'algoritmo di "back-propagation" proposto negli anni '80.

Non tutte le reti sono però in grado di apprendere: ciò si verifica quando i dati di input non contengono le informazioni specifiche da cui deriva l'output desiderato o se non ci sono abbastanza dati per consentire l'apprendimento completo. L'iterazione dell'algoritmo termina quando o è stata raggiunta la % di accuratezza (record classificati correttamenterecord totali) definita inizialmente dai progettisti, oppure la % di errore medio è sceso sotto una determinata soglia, o ancora è stato raggiunto il numero massimo di iterazioni possibili.

Il numero di livelli e il numero di elementi di elaborazione per livello sono decisioni importanti: non esiste una risposta quantificabile al layout della rete per una particolare applicazione, ma ci sono solo regole generali raccolte nel tempo e seguite dalla maggior parte dei ricercatori e ingegneri che applicano questa architettura ai loro problemi [20].

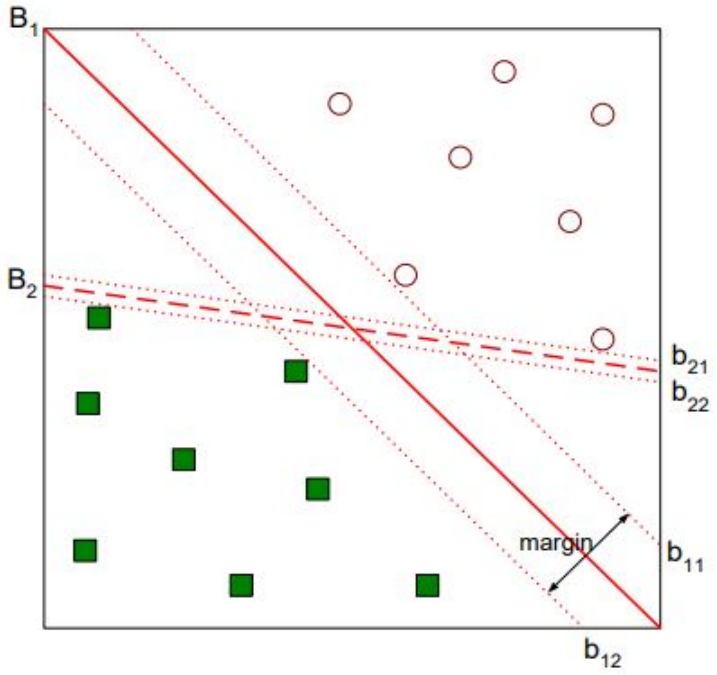

FIGURA 1.20: Esempio di Macchine a vettori di Supporto con due possibili strategie.

1.7.7 Macchine a vettori di supporto

Le macchine a vettori di supporto (SVM, *Support Vector Machines*) sono modelli che suddividono i record all'interno di due classi effettuando dunque una suddivisione binaria.

Data una serie di record di addestramento, ognuno appartenente ad una delle due classi, un algoritmo di addestramento SVM costruisce un modello che assegna nuovi esempi a una categoria o all'altra, rendendolo un classificatore lineare binario non probabilistico. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi delle categorie separate siano divisi da uno spazio vuoto il più ampio possibile. Nuovi esempi vengono quindi mappati nello stesso spazio e previsti per appartenere a una categoria in base al lato dello spazio su cui cadono.

Nella Figura 1.19 sono mostrate due possibili segmentazioni lineari dello spazio, che definiscono due classi distinte all'interno del dataset. Tra le due opzioni B1 e B2 la migliore risulta essere B1 in quanto definisce un margine più ampio tra le due classi. Non sempre però la segmentazione dello spazio può essere lineare: a volte si potrebbe avere necessità di far ricorso a funzioni più complesse. In tal caso le SVM possono eseguire in modo efficiente una classificazione non lineare usando quello che viene chiamato il trucco del kernel, mappando implicitamente i loro input in spazi ad alta dimensione.

1.7.8 Clustering

Dato un insieme di record e definita una misura di similarità tra i record sulla base dei valori dei loro attributi, il clustering consiste nel dividere i record in sottoinsiemi tali che i record appartenenti a uno stesso cluster sono più simili tra loro rispetto ai record presenti negli altri cluster. L'obiettivo del clustering è quello di minimizzare le distanze dei record all'interno dei cluster e massimizzare quella tra cluster.

Si tratta di algoritmi di apprendimento *non supervisionato*, in cui cioè non è prevista una fase di training del set di dati. Dati due record con n attributi, questi devono essere trasformati in due vettori numerici A e B con n dimensioni. Durante la clusterizzazione occorre definire che tipo di calcolo effettuare per valutare la similarità tra A e B. Ciò può essere misurato in diversi modi, di cui i più comuni sono la **distanza euclidea**, in caso di attributi con valori continui (o che vengono trasformati in valori continui) espressa come $\sqrt{\sum_{k=1}^{n} (A_k - B_k)^2}$, la **distanza coseno** che esprime la similarità con la formula $\frac{\sum_{k=1}^{n} A(k)B(k)}{\sqrt{\sum_{k=1}^{n} A(k)^2}\sqrt{\sum_{k=1}^{n} B(k)^2}}$, la **correlazione di Pearson** calcolata come rapporto tra la covarianza di A e B e il prodotto delle loro deviazioni standard, definita come $\rho_{AB} = \frac{\sigma_{AB}}{\sigma_A \sigma_B}$.

L'obiettivo principale del clustering è quello di **segmentare il mercato**: le aziende possono individuare clienti con caratteristiche simili legate alle scelte di consumo al fine di indirizzare su ciascun cluster delle specifiche operazioni di marketing: segmenti di mercato diversi possono avere preferenze e modelli comportamentali diversi.

Esistono due modelli fondamentali di clustering:

- **Partizionante**: quando ogni record appartiene ad un solo cluster.
- **Gerarchico**: quando ogni record può appartenere a un solo cluster o a più cluster in forma gerarchica.

Tra questi modelli si distinguono alcune tecniche fondamentali:

- **Well-Separated Cluster**: in ciascun cluster ogni punto risulta maggiormente vicino a ciascun punto interno al cluster rispetto che a qualsiasi punto situato al di fuori.
- **Center-based**: in questo caso in ogni cluster ciascun punto è più vicino al centro del suo cluster, definito **centroide** che al centro di qualsiasi altro cluster.
- **Cluster contigui**: esiste per ogni punto di ciascun cluster almeno un punto all'interno dello stesso cluster che sia più vicino rispetto a tutti i punti situati fuori dal cluster.
- **Density-based**: identifica cluster distintivi, in base all'idea che un cluster in uno spazio dati sia una regione contigua di alta densità di punti,

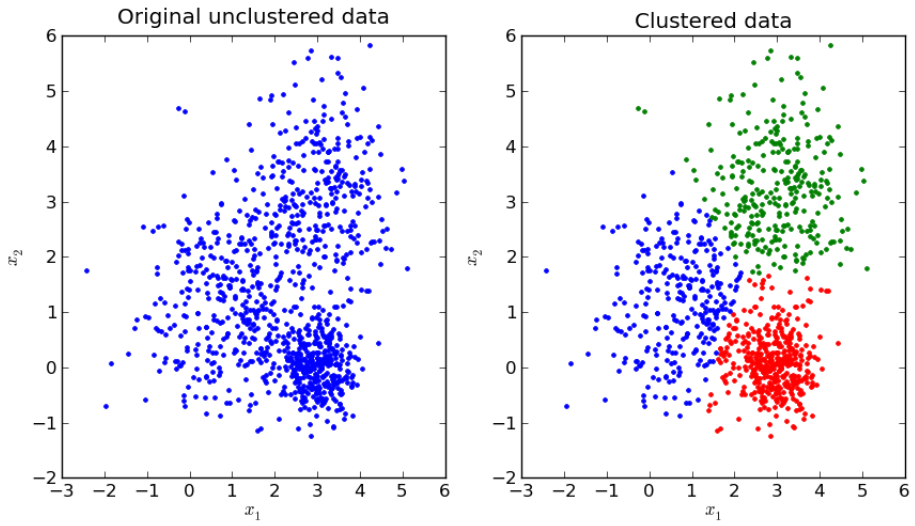

FIGURA 1.21: Esempio di esecuzione di K-means su un dataset

separata da altri cluster di questo tipo da regioni contigue di bassa densità di punti. I punti dati nelle regioni di separazione a bassa densità di punti sono generalmente considerati rumore o valori anomali.

Nelle sezioni successive verrano descritti i due algorirmi di clustering più utilizzati: **K-Means** e **DBSCAN**.

1.7.9 K-Means

L'algoritmo inventato da MacQueen nel 1967 [21] è una soluzione partizionante e *center-based* in cui viene stabilito a pripri il numero **K** di cluster da formare. L'idea principale è quella di definire K centroidi scelti in modo casuale, uno per ciascun cluster. Il passaggio successivo consiste nel prendere ciascun punto appartenente a un determinato dataset e associarlo al centroide più vicino. Dopo la prima iterazione bisogna ricalcolare i K centroidi, come baricentro dei cluster risultanti dal passaggio precedente. Dopodichè è necessario eseguire una nuova associazione tra gli stessi dati e il nuovo centroide più vicino e continuare finché i centroidi restano fermi, oppure fin quando la percentuale di punti che cambiano cluster in seguito ad una interazione si rivela molto ridotta.

Per applicare l'algorimo ad un dataset, questo deve essere trasformato in un insieme di vettori numerici n-dimensionali [22] attraverso una serie di operazioni :

- Gestire i valori mancanti: ci sono vari metodi disponibili per l'imputazione del valore mancante, ma bisogna fare attenzione a garantire che esso non distorca il calcolo della distanza nell'algoritmo. Solitamente si può utilizzare il valore 0.

- Gestire le variabili categoriche: esse possono essere trasformate in valori numerici univoci che sono mappati direttamente con quella variabile

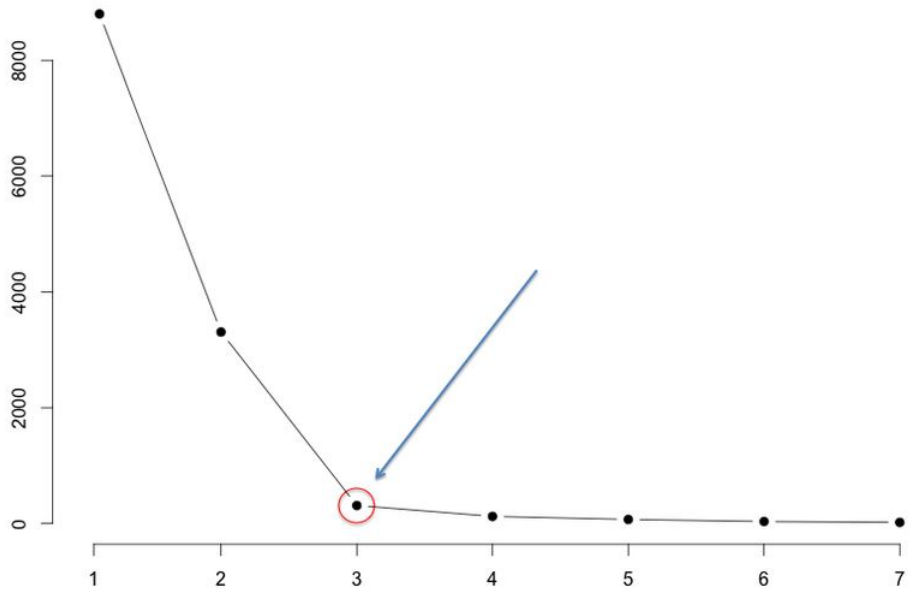

FIGURA 1.22: Esempio di esecuzione di K-means su un dataset

categorica, oppure possono esprimere il livello di grandezza del valore categorico, qualora questo sia di tipo ordinale, come nel caso dell'attributo "dimensione", che può assumere valori "piccola", "media", "grande" e possono essere mappati in 0, 1/2, 1.

- Normalizzare le dimensioni: Il calcolo della distanza in K-Means pesa equamente ogni dimensione e quindi bisogna fare attenzione a garantire che l'unità di misura non distorca il risultato. Il metodo più utilizzato consiste nel normalizzare singolarmente ciascun attributo.

Un altro aspetto importante è inerente alla scelta dei centroidi iniziali da cui partire, che vengono scelti in modo casuale, ma potrebbero portare, in alcuni casi, a dei risultati non buoni. Per questo occorre che diversi algoritmi vengano lanciati con diversi punti di partenza iniziali.

La misura di qualità maggiormente utilizzata per comprendere la bontà dell'algoriritmo è la misura SSE (Sum Squared Errors) che misura quanta distanza c'è all'interno di tutti cluster e il suo rispettivo centroide: : $SSE = \sum_{i=1}^{K} \sum_{X_j \in P_i} ||X_j - C_i||^2$, dove $C_i, \ldots, Ck$ sono i K cluster e X sono i record presenti all'interno dei vari cluster. Chiaramente l'obiettivo è minimizzare tale valore, che viene utilizzato anche per definire il numero di cluster "ottimale", attraverso il cosiddetto metodo **Elbow**, che consiste nello scegliere tra le diverse quantità di cluster, la quantità K per la quale la funzione di SSE ha una derivata seconda positiva, cioè la decrescità di SSE tra k e k-1 è stata inferiore rispetto a quella osservata tra k-1 e k-2 (Figura 1.21).

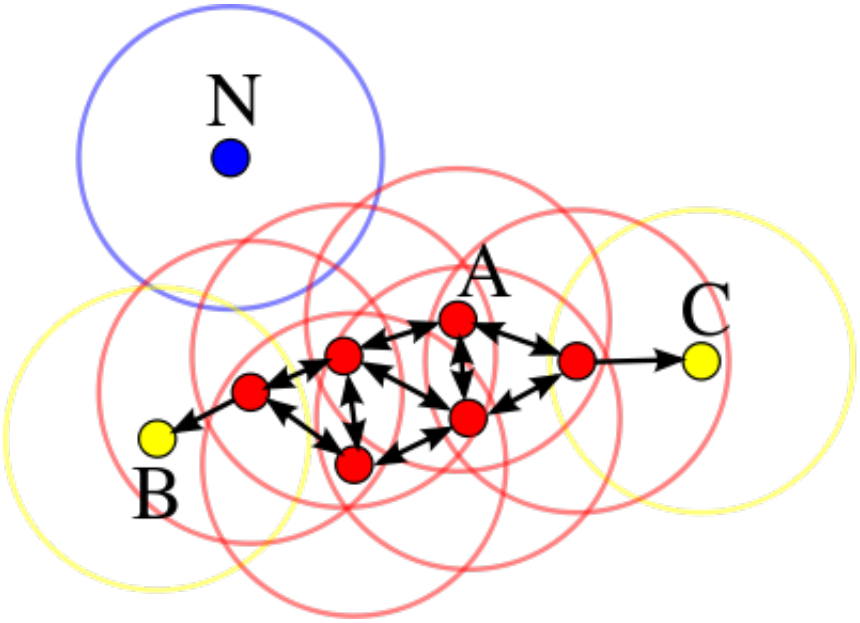

FIGURA 1.23: Esempio di esecuzione di DBSCAN

1.7.10 DBSCAN

DBSCAN è un algoritmo di clustering di dati proposto da Martin Ester, Hans-Peter Kriegel, Jörg Sander e Xiaowei Xu nel 1996 [23] è un algoritmo non parametrico di clustering basato sulla densità: dato un insieme di punti in un certo spazio, raggruppa in un cluster i punti che hanno numerosi vicini nello spazio, contrassegnando come punti anomali quelli che si trovano in regioni a bassa densità.

Dato un insieme di punti nello spazio e definito ϵ come il parametro che specifica la distanza massima per considerare due punti vicini, l'algoritmo classifica i punti dello spazio in punti centrali, punti raggiungibili e valori anomali, nel seguente modo:

- Un punto p è un punto *centrale* se c'è una quantità minima *minPts* di punti che sono distanti al più ϵ da esso.
- Un punto q è *direttamente* raggiungibile da p se la sua distanza da p è minore o uguale a ϵ.
- Un punto q è *densamente* raggiungibile da p se esiste almeno un percorso $p_1, \ldots, p_b$ con $p1 = p$ e $p_n = q$ dove ogni $p_i + 1$ è *direttamente* raggiungibile da p_i.
- Un punto q non raggiungibile da alcun punto centrale è definito un valore di *rumore*.

Se p è un punto centrale, allora forma un cluster insieme a tutti i punti che sono raggiungibili da esso. Ogni cluster contiene almeno un punto centrale; i punti non centrali possono far parte di un cluster, ma formano il suo "bordo", poiché non possono essere utilizzati per raggiungere altri punti.

Nella Figura 1.22, dove è stato impostato come *MinPts* il valore 4, i punti *centrali* sono quelli rossi, perché nel raggio ϵ sono raggiungibili da almeno altri 4 punti, mentre quelli gialli sono *di bordo* perché sono raggiungibili da un punto *centrale* senza essere tali. Infine, i punti blu sono dei punti *rumore*.

I vantaggi di DBSCAN rispetto a K-means sono legati al fatto che non ha bisogno di settare il numero di cluster arbitrariamente, la resistenza agli outliers, in quanto riesce più agevolmente a separarli avendo una nozione di rumore, ma risulta più lento nell'implementazione e richiede una attenta scelta dei parametri di partenza.

1.8 Report, dashboard e decisioni

Le operazioni illustrate nelle sezioni precedenti hanno l'obiettivo di produrre dei risultati che possano essere facilmente analizzati dagli utenti di Business: analisti e manager che devono intervenire e prendere delle decisioni strategiche sulla base degli obiettivi di crescita, di quelli che erano stati definiti come **KPI**(Key Performance Indicator), delle eventuali inefficienze che l'analisi del dato ha fornito in determinati segmenti di mercato. Per questo vengono realizzati report e dashboard interattive che riassumono le informazioni in un formato semplice da interpretare e possibilmente anche visivamente gradevole per poter essere utilizzato all'interno di presentazioni e meeting aziendali, in cui la qualità visiva ha un peso molto importante per il successo. Di quì è nato un mercato sempre più *affollato* di prodotti che permettono di realizzare i report e le dashboard a figure professionali prive di spiccate competenze informatiche, che saranno analizzate nel Capitolo 3 e che in concomitanza con le soluzioni tecnologiche derivanti dalle soluzioni Big Data, analizzate nel Capitolo 2, pongono le imprese nella condizione di dover attentamente valutare in che modo, il risultato delle operazioni sui dati, possa essere al meglio valorizzato.

Capitolo 2

Big Data technology: tecniche di processamento distributo e parallelo

2.1 La fondazione Apache: prodotti streaming e batch per i Big Data

Le aziende che lavorano con i Big Data operano con due tecniche principali: lo **streaming**, che consiste nell'elaborazione e analisi di dati in tempo reale e il **batch** che consiste invece nel lavorare con una serie di dati raggruppati in un intervallo di tempo specifico ed è spesso associato al termine "finestra di dati". Entrambi i modelli sono importanti e ognuno può essere utilizzato per affrontare diversi casi d'uso.

La fondazione no-profit Apache, nata nel 1999, fornisce soluzioni open-source per il Mondo tecnologico ed è portatrice dei principali progetti innovativi nel Mondo dei Big Data. Tra questi, il primo ad essere portato avanti è stato **Hadoop** e il suo ecosistema, ossia una serie di strumenti per l'elaborazione di dati sia in batch, che in streaming.

Si tratta di una quantità davvero elevata di prodotti, di cui all'interno di questo capitolo vengono analizzati, oltre ad Hadoop, anche **Spark**, **Kafka**, **Hive** e **Hbase**, in quanto saranno utilizzati nel capitolo successivo per il caso di studio, oltre agli strumenti ad oggi più popolari per i diversi modelli di database NoSql.

2.2 Apache Hadoop

La tecnologia dei Big Data consiste in un gran numero di componenti software open-source della fondazione Apache, ideali per lavorare in un ambiente distribuito, tra cui spicca sicuramente Apache Hadoop, un framework scritto in Java, nato nel 2008 dai laboratori di Yahoo!.

I sistemi attuali devono adattarsi alle dimensioni della crescente quantità di dati da analizzare, del numero di utenti da servire, della complessità dei problemi.

Di solito vengono utilizzati due approcci per affrontare problemi di scalabilità:

1. Scalabilità verticale: consiste nel rafforzare i singoli server che contengono i dati, utilizzando soluzioni tecnologiche di qualità e costo maggiore.

2. Scalabilità orizzontale: consiste nell'affiancare delle macchine di economiche in parallelo a quelle già esistenti.

Hadoop utilizza la scalabilità orizzontale attraverso un'architettura distribuita in parallelo, chiamata **HDFS** (Hadoop Distributed File System), progettata per essere distribuita all'interno di un cluster i cui server possono essere situati fisicamente in un datacenter locale o in un' infrastruttura cloud. Esistono due distribuzioni di Hadoop principalmente utilizzate a livello enterprise:

- **Cloudera Data Platform**: è la piattaforma nata nel 2019 dalla fusione di HortonWorks e Cloudera, che dovrebbe superare il 65% del market-share unendo gli utenti delle due vecchie distribuzioni.

- **MapR**: utilizza un file system chiamato MapR file system, che eredita le API[1] di HDFS e utilizza C++ anziché Java come linguaggio, il che permette di velocizzare le operazioni di scrittura sul disco.

2.2.1 Hadoop Distributed File System

HDFS è un file system distribuito su diversi nodi, progettato per essere eseguito su hardware di fascia "commodity", di costo molto inferiore rispetto ai classici RDBMS e per permettere ai sistemi di operare in regime di resistenza al guasto e completa portabilità su qualsiasi piattaforma.

HDFS ha un'architettura master-slave, in cui il master è definito **NameNode**, un nodo principale che gestisce lo spazio dei nomi (*namespace*) del file system e regola l'accesso ai file da parte dei client, mentre esistono diversi **DataNode**, generalmente uno per nodo nel cluster, che gestiscono l'archiviazione collegata ai nodi su cui vengono eseguiti.

I problemi che l'utilizzo di HDFS cerca di risolvere sono [24] :

- **Guasti nell'hardware**: avere dei guasti nell'hardware è una casistica molto comune, per questo avere diversi nodi che lavorano in parallelo e permettono al sistema di funzionare anche in caso di guasto, è una soluzione che permette al sistema di non bloccarsi.

- **Throughput**: HDFS è disegnato per performare al meglio con una modalità di processamento di tipo *batch*, ossia di processi che possono essere eseguiti senza l'interazione con l'utente, il cui obiettivo principale non è quello di ridurre il tempo di accesso ai dati, bensì di massimizzare la velocità di processamento su grandi dataset.

- **Spostare le computazioni vicino ai dati**: un calcolo richiesto da un'applicazione è molto più efficiente se viene eseguito vicino ai dati su cui

[1] Con Application Programming Interface (API) si indicano le librerie software di un linguaggio di programmazione.

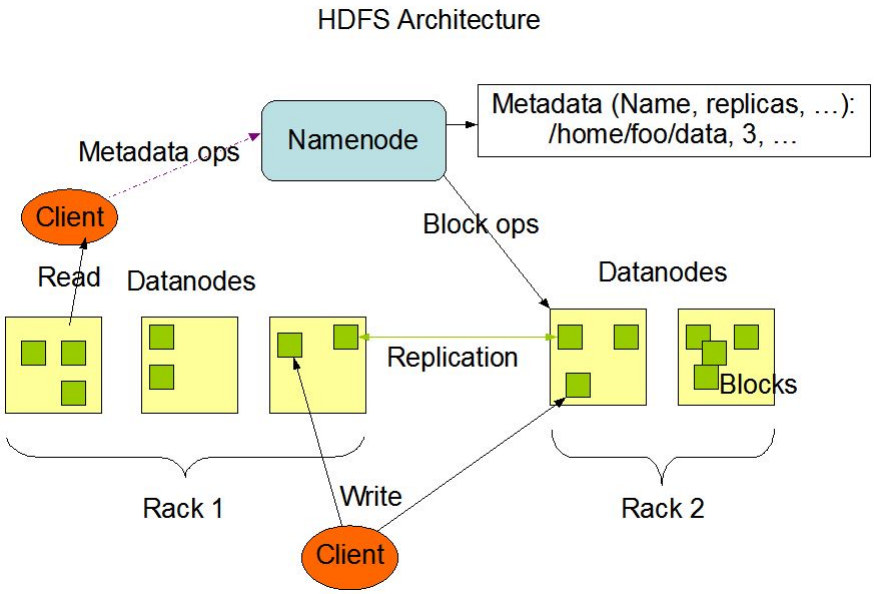

FIGURA 2.1: Architettura HDFS. Fonte:hadoop.apache.org

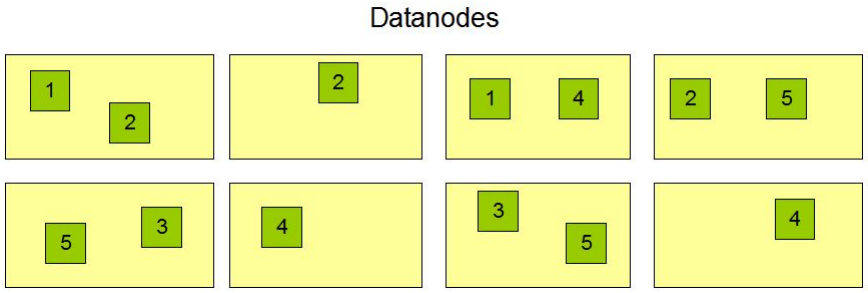

FIGURA 2.2: La replicazione dei blocchi nei DataNode

opera, soprattutto quando le dimensioni del dataset sono enormi, in quanto permette di ridurre la congestione della rete e aumenta la velocità complessiva del sistema. HDFS fornisce interfacce per le applicazioni per avvicinarsi ai dati.

- **Gestione dei dati non strutturati**: il file system permette di archiviare file di qualsiasi formato, andando oltre la gestione di tabelle Sql.

Lo strumento con cui il NameNode controlla il file system prende il nome di *namespace*: una tecnica gerarchica di gestione dei file. L'utente e le applicazioni possono creare cartelle e memorizzare i file al loro interno, spostarli tra cartelle, rimuoverli, con il NameNode che memorizza ogni modifica.

Internamente, un file viene suddiviso in uno o più blocchi, generalmente di dimensione di 128 MB, che vengono archiviati e replicati in un set di DataNode, con un fattore di replica che può essere specificato al momento della creazione del file e modificato in un secondo momento.

Il posizionamento delle repliche sui nodi rappresenta un fattore critico per l'affidabilità e la performance dell'HDFS. Nel caso più comune il fattore di replica utilizzato per ciascun blocco del file è 3 e la policy prevede di

posizionare una replica sulla macchina locale del DataNode sul quale viene salvato il file, una replica su un nodo posizionato in remoto su un server distinto e l'ultima su un terzo nodo posizionato nello stesso server in remoto. In questo modo le operazioni di lettura si interfacciano solo con 2 server anziché 3 e la banda di rete utilizzata, che costituisce il collo di bottiglia principale per le applicazioni, viene ridotta.

Il NameNode tiene traccia di ogni operazione all'interno dell' "EditLog" e memorizza il namespace in un file chiamato "FsImage", oltre a monitorare lo stato di attività di tutti i DataNode, attraverso quello che viene definito un "battito cardiaco", cioè un messaggio con cui il DataNode comunica di essere in funzione. Qualora questo messaggio non dovesse arrivare da parte di uno specifico DataNode, questo verrà considerato fuori servizio, le repliche memorizzate su di esso non disponibili e non verranno memorizzate ulteriori repliche.

Il problema principale di HDFS è costituito dalla presenza di un unico punto di guasto, il NameNode, che costringe ad interventi manuali in caso di malfunzionamenti: per questo esiste un *Secondary Namenode*, il quale non si comporta come un NameNode, ma memorizza EditLog e FsImage a intervalli regolari, in modo che in caso di guasto del NameNode, i metadata possono essere presi dall'ultima copia memorizzata dal Secondary NameNode, il che comunque non esclude che rispetto all'ultimo aggiornamento non ci siano state delle modifiche che vengono perse.

2.2.2 MapReduce

MapReduce è il framework posizionato sopra HDFS per l'esecuzione dei calcoli. L'applicazione in grado di essere eseguita sui dati archiviati in HDFS viene definita come Job e prevede che i blocchi dei dati in input siano processati in parallelo dalle operazioni di Map, che resituiscono l'elenco delle coppie chiave-valore su cui vengono eseguite le operazioni di Reduce, ossia le operazioni di aggregazione (Figura 2.3). Tipicamente i nodi su cui vengono eseguiti i calcoli sono gli stessi in cui i dati sono stati archiviati e inoltre MapReduce utilizza le stesse risorse di HDFS, lavorando direttamente sul disco.

Quando un'applicazione viene eseguita sul cluster Hadoop, essa viene presa in carico da YARN (Yet Another Resource Negotiatior), che prevede la presenza di 3 entità :

1. **ResourceManager** che coordina l'allocazione delle risorse computazionali tra i vari nodi del cluster.

2. **NodeManager** che lancia e monitora l'esecuzione del Job sul nodo.

3. **Application Master** che coordina le operazioni di Map e Reduce, negozia con il ResourceManager le risorse necessarie ed è allocato per ogni client che richiede l'esecuzione di un Job.

Ogni volta che viene eseguita un'applicazione su YARN, il client contatta il ResourceManager, che individua il NodeManager idoneo per lanciare il Job

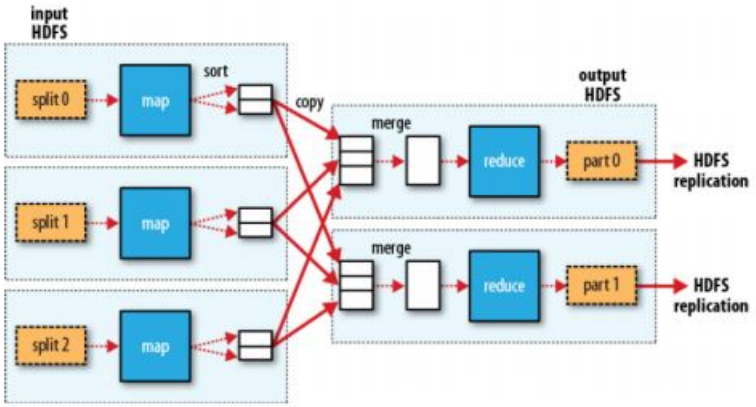

FIGURA 2.3: Il flusso di una operazione di MapReduce

e gli assegna l'ApplicationMaster, che eseguirà le funzioni che sono state implementate all'interno dell'applicazione, scritta generalmente in Java o Scala, la quale prevede l'esecuzione del Job attraverso il metodo *submit()* invocabile dalle API di Hadoop.

Un caso d'uso comune utilizzato per comprendere le funzionalità di MapReduce è quello di un' applicazione che, dati dei file di testo, individua tutte le parole presenti e calcola il numero di volte in cui ciascuna parola compare.

Supponendo che ci siano due file di input, di cui il primo contiene il testo:

```
Ciao Mondo Arrivederci Mondo
```

e il secondo contiene il testo:

```
Ciao Hadoop Addio Hadoop
```

le operazioni di Map individuano tutte le diverse parole all'interno di ciascun file, quelle di Reduce invece calcolano il numero di ricorrenze.

L'output delle operazioni di Map sono due liste composte da coppie chiave-valore, in cui la chiave è la parola e il valore il numero di volte che la parola stessa comapre:

```
< Ciao, 1>
< Mondo, 1>
< Arrivederci, 1>
< Mondo, 1>
```

```
< Ciao, 1>
< Hadoop, 1>
< Addio, 1>
< Hadoop, 1>
```

Successivamente viene effettuata una prima aggregazione all'interno della singola lista, per mettere insieme le parole che compaiono più di una volta, attraverso il cosiddetto "Combiner", che rappresenta un vero e proprio

Reducer che opera sul singolo nodo. Oltre all'aggregazione è anche prevista una fase di ordinamento delle coppie definito sul valore delle chiavi. Supponendo di ordinare alfabeticamente le chiavi, l'output finale sarà:

```
< Arrivederci, 1>
< Ciao, 1>
< Mondo, 2>
```

```
< Addio, 1>
< Ciao, 1>
< Hadoop, 2>
```

Infine saranno applicate le operazioni di Reduce a entrambe le liste di coppie chiave-valore, ottenendo il seguente output finale:

```
< Arrivederci, 1>
< Addio, 1>
< Ciao, 2>
< Hadoop, 2>
< Mondo, 2>
```

2.3 Apache Spark

Apache Spark è un framework per il calcolo distribuito e parallelo, sviluppato inizialmente dall'Università di California e successivamente donato alla fondazione Apache. Spark è stato sviluppato in risposta ai limiti di MapReduce di lavorare esclusivamente sul disco, sviluppando i cosiddetti **RDD** (Resilient Distributed Dataset), delle strutture dati immutabili, essendo disponibili in sola lettura, che sfruttando la memoria RAM riescono a ottenere dei tempi di esecuzione più rapidi dei calcoli, dell'ordine di grandezza compreso tra le 10 e le 100 volte.

Successivamente agli RDD, nelle versioni 2x di Spark sono stati introdotti l'utilizzo dei **DataFrame** e dei **DataSet**. Mentre i Dataset sono delle collezioni distribuite fortemente *tipizzate* di oggetti JVM (Java Virtual Machine) definiti da una classe scritta in Java o in Scala, i DataFrame sono delle collezioni distribuite *non tipizzate* in cui ciascun record è visto come un generico oggetto JVM e per questo le API dei DataFrame sono disponibili anche su Python e R (oltre che Java e Scala), che essendo linguaggi di scripting, non compilativi, non supportano i DataSet poiché il loro *Schema* deve essere definito in fase di compilazione [25] .

Apache Spark per funzionare necessita di un cluster e un sistema di archiviazione distribuito. Per la gestione dei cluster, Spark supporta autonomamente Hadoop YARN, Apache Mesos o Kubernetes. Per l'archiviazione distribuita, Spark può interfacciarsi con un'ampia varietà di soluzioni, tra cui HDFS, ma anche Alluxio, MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon S3, Kudu, Lustre file system, Hbase. Spark supporta anche una modalità locale distribuita, solitamente utilizzata solo a scopo di

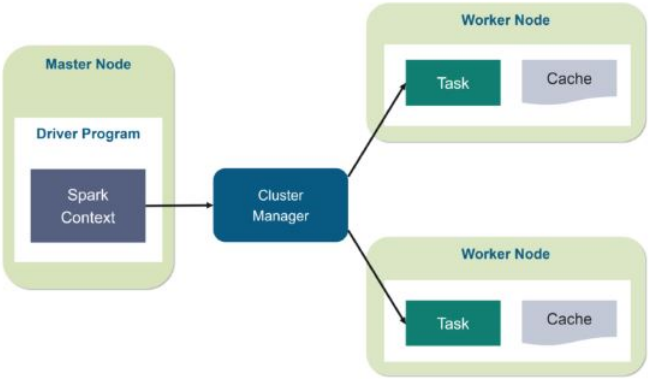

FIGURA 2.4: Architettura Spark.

sviluppo o test, in cui non è richiesta la memoria distribuita e al suo posto è possibile utilizzare il file system locale; in tale scenario, Spark viene eseguito su una singola macchina con un esecutore per core della CPU.

Spark permette di eseguire applicazioni Big Data in diversi casi d'uso, estendendo l'utilizzo di applicazioni di tipo batch a casi d'uso in streaming.

L'architettura di Spark prevede la presenza di un nodo master, il cosiddetto **MasterNode**, in cui è posizionato il *Driver*, ossia ciò che lancia l'esecuzione del programma, che può essere un'applicazione oppure un comando utilizzando la linea di comando di Spark e di alcuni nodi slave, definiti **WorkerNode**, i quali sono responsabili di eseguire le operazioni dei Job. All'interno del MasterNode è presente il cosiddetto **SparkContext**, una porta che contiene tutte le principali funzionalità di Spark, il quale invia le istruzioni al cluster manager, responsabile di spezzare il Job in diverse operazioni che vengono gestite da più WorkerNode. Il WorkerNode, dopo aver eseguito le operazioni restituisce a SparkContext il risultato in formato di RDD, DataFrame o DataSet (Figura 2.4).

2.3.1 Spark core

Spark si basa su un componente di base, lo Spark Core che viene sfruttato da tutti i componenti di analisi dei dati di alto livello. Questa soluzione fornisce una soluzione più uniforme ed efficiente rispetto a MapReduce, dove sono disponibili molti strumenti di analisi non integrati tra loro.

Spark Core fornisce processamento distribuito, la pianificazione e le funzionalità di IO di base, esposte attraverso un'interfaccia di programmazione dell'applicazione (per Java, Python, Scala e R) centrata sull'astrazione RDD, su cui sono stati costruite le API DataFrame e DataSet. Questa interfaccia rispecchia un modello di programmazione funzionale: un programma "driver" richiama operazioni parallele come *map*, *filter* o *reduce* su un RDD, un DataFrame o un DataSet, passando una funzione a Spark, che quindi pianifica l'esecuzione della funzione in parallelo sul cluster.

Nella Figura 2.5 sono illustrati i componenti di alto livello di Spark :

- **SparkSQL** è un modulo Spark per l'elaborazione strutturata di dati. Esistono diversi modi per interagire con SparkSQL, inclusi Sql e le API

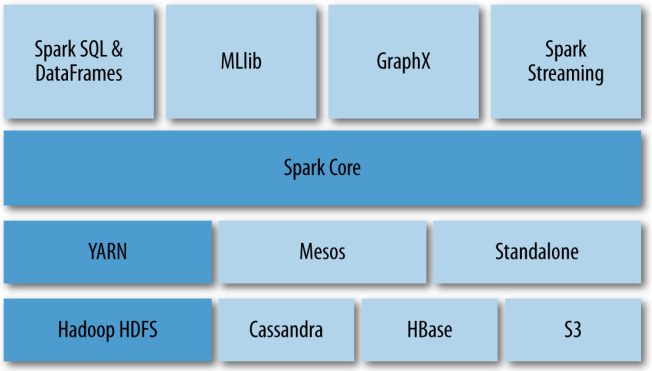

FIGURA 2.5: L'ecosistema Spark.

FIGURA 2.6: Spark Streaming

DataSet/DataFrame. Un uso di SparkSQL è l'esecuzione di query Sql per leggere i dati da Hive. Altre fonti dati a cui SparkSQL può collegarsi sono Parquet, Json, Csv, file orc, file testuali e tutte le fonti che hanno un driver Jdbc [26] .

- **Spark streaming** è un'estensione dell'API Spark che consente l'elaborazione in streaming scalabile, ad alta velocità e tollerante agli errori dei flussi di dati in tempo reale. I dati possono essere ingestionati da molte fonti come socket Kafka, Flume, Kinesis o TCP e possono essere elaborati utilizzando algoritmi complessi espressi con funzioni di alto livello come *map*, *reduce* e *join*. Infine, i dati elaborati possono essere inviati a file system, database e altri strumenti. Spark Streaming legge dei dati in input in streaming e li divide in batch, utilizzando i cosiddetti **DStream**, delle sequenze di RDD (Figura 2.6). [27]

- **MLlib** fornisce oltre ad algoritmi di Machine Learning come classificazione, regressione, clustering, anche strumenti per operazioni di ETL e di calcolo statistico, sfruttando le potenzialità del motore Spark di distribuzione e parallelismo dei calcoli. Ciò rende MLlib uno strumento idoneo a sfruttare le potenzialità dei Big Data all'interno di un'unica piattaforma integrata.

- **GraphX** è un framework che permette di sfruttare le proprietà dei grafi per operazioni di ricerca operativa, calcolo combinatorio e algoritmi ricorsivi. I grafi rappresentano un'estensione degli RDD e per questo sono immutabili, resistenti al guasto e distribuiti [28] . Ciascun grafo sarà formato da un elenco di vertici e delle connessioni tra vertici

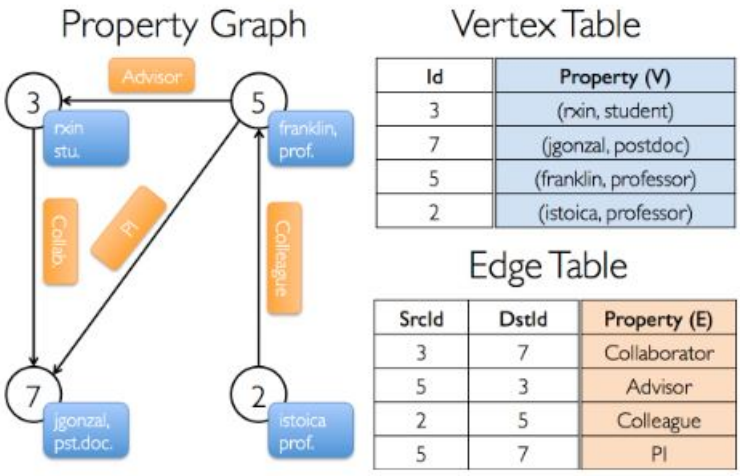

FIGURA 2.7: Esempio di grafo generato da GraphX

che possono possedere delle proprietà specifiche che rappresentano la relazione.

2.4 Apache Hive

Apache Hive, sviluppato inizialmente da Facebook, è un Data warehouse basato su Apache Hadoop per fornire un'interfaccia simile a Sql per la manipolazione dei dati.

Apache Hive supporta l'analisi di grandi dataset archiviati in HDFS e altri file system compatibili come Amazon S3 e Alluxio, attraverso un linguaggio molto simile ad Sql, chiamato HiveQL, che viene convertito da Hive in Job di tipo MapReduce, Tez o Spark. Per accellerare le query, Hive fornisce la possibilità di indicizzare e di creare delle partizioni sulle colonne delle tabelle per raggruppare i record con lo stesso valore per un determinato attributo. Come impostazione predefinita, Hive archivia i metadati in un database Apache Derby incorporato, ma è possibile utilizzare altri database come MySQL.

I primi 4 formati supportati da Hive sono stati file testuali, file sequenziali (costituiti da coppie binarie chiave / valore, ampiamente utilizzati in MapReduce come formati di input / output [29] , file ORC (Optimized Row Columnar) e RCFile.

Hive permette di lavorare sia con tabelle esterne, che rappresentano dei puntatori al file archiviato sul file system, che con tabelle interne, ossia tabelle che sono archiviate all'interno di una cartella riservata ad Hive. Esso garantisce una serie di proprietà di tipo *ACID* (atomicity, consistency, isolation, durability) [2]su operazioni di inserimento (insert), modifica (update) e rimozione (delete) delle tabelle.

[2]**Atomicity** significa garantire che tutte le transazioni abbiano successo o che altrimenti nessuna va a buon fine.

Consistency significa garantire che tutti i dati siano coerenti. Tutti i dati saranno validi in base a tutte le regole definite, inclusi eventuali vincoli, cascate e trigger applicati sul database.

I suoi componenti principali sono [29] :

- **HCatalog** è uno strumento di gestione di tabelle e archiviazione per Hadoop che consente agli utenti con diversi strumenti di elaborazione dei dati, inclusi Pig e MapReduce, di leggere e scrivere i dati sulla griglia più facilmente.
- **WebHCat** fornisce i servizi per eseguire Job Hadoop MapReduce e permette di eseguire operazioni sui metadati di Hive utilizzando un'interfaccia HTTP.

Hive è pensato per elaborazioni di tipo batch, in quanto le query che riceve vengono tradotte in Job e ciò comporta dei tempi di latenza elevati per poter pensare di gestire operazioni che richiedono dei tempi di risposta molto rapidi, come nel caso di operazioni di visualizzazione dati.

2.5 Apache HBase

HBase è un database open-source non relazionale, modellato sul prodotto di Google Big Table, eseguito su Hadoop HDFS oppure su Alluxio, fornendo prestazioni molto elevate di lettura di dati "sparsi" all'interno di grandi volumi, come ad esempio quando si vogliono indentificare soltanto 50 elementi all'interno di 2 miliardi di record. Il modello dati di HBase è composto dai seguenti elementi:

- **Tabella**, composta da diverse righe, viene dichiarata nella fase di definizione dello schema.
- **Riga**, composta da una chiave e una o più colonne. Le righe sono ordinate in base alla chiave, per questo il suo design è molto importante, in quanto l'obiettivo è quello di archiviare i dati in modo che le righe correlate siano tra loro vicine.
- **Colonna**, consiste in una famiglia (Column Family) e un identificatore (Column Qualifier)
- **Famiglia della colonna**. Ogni riga all'interno di una tabella ha le stesse famiglie, inoltre ogni elemento di una famiglia ha lo stesso prefisso, come ad esempio *corso:storia* e *corso:matematica*, entrambi membri della famigia corso.
- **Qualificatore della colonna**, che rappresenta un indice. Riprendendo l'esempio precedente sia *storia*, che *matematica*, sono degli indici, dunque dei qualificatori di *corso*.

Isolation significa assicurarsi che tutte nessuna transazione sia influenzata da qualsiasi altra transazione.

Durability significa che, una volta effettuata, una transazione rimarrà nel sistema, anche in caso di arresto anomalo del sistema.

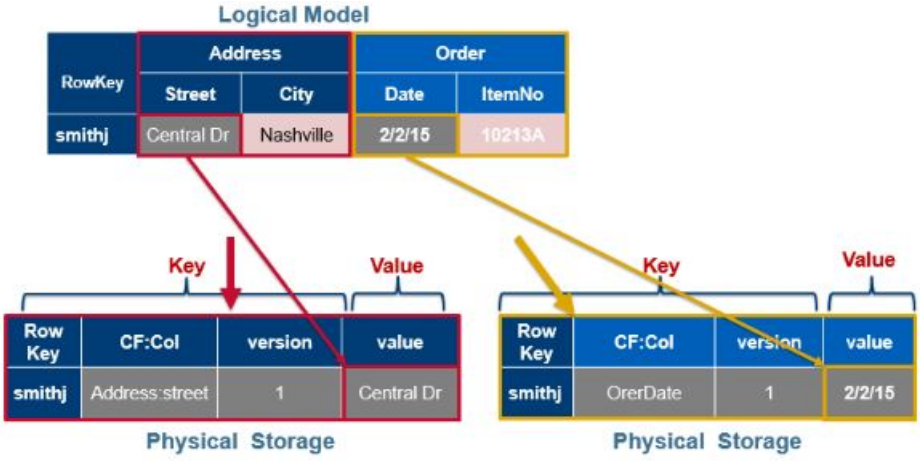

FIGURA 2.8: Modello logico e fisico di Hbase.

- **Cella**, è una combinazione di riga, famiglia della colonna e identificatore della colonna, a cui sono associati un *valore* e un *timestamp* che rappresenta la versione del valore.
- **Timestamp**, associato ad ogni cella, è generalmente calcolato nel momento in cui il dato viene archiviato, ma può essere modificato.

Le tabelle sono divise in sequenze di righe, per intervallo di chiavi, chiamate regioni, che vengono assegnate a determinati DataNode nel cluster, denominati **RegionServer**, all'interno dei quali le famiglie delle colonne sono memorizzate in file separati.

C'è quindi una distinzione tra modello logico, che segue le regole descritte di una struttura tabulare e modello fisico, nel quale invece ciascuna famiglia è memorizzata separatamente (Figura 2.8).

Le funzionalità di HBase possono fungere da input e output per i lavori MapReduce/Spark eseguiti in Hadoop e sono accessibili tramite l'API Java e le API gateway REST [3], Avro [4] o Thrift [5], ma a differenza di Hive, HBase può essere direttamente interrogato per applicazioni che richiedono dei tempi di risposta immediati per la fornitura dei risultati delle query.

L'interrogazione di HBase non utilizza il linguaggio Sql, tuttavia il progetto Apache Phoenix fornisce un livello Sql per HBase e un driver JDBC che può essere integrato con varie applicazioni di analisi e business intelligence.

2.6 Apache Kafka

Apache Kafka è una piattaforma distribuita di streaming [30], generalmente utilizzata per due casi d'uso:

[3] Le API RESTful sono basate sull'architettura REST e utilizzano il protocollo HTTP per recuperare, creare, cancellare e aggiornare le risorse di un sistema.

[4] Avro è un framework di chiamate e procedure di serializzazione dei dati orientato alle righe sviluppato nell'ambito del progetto Hadoop.

[5] Apache Thrift 'e un protocollo di comunicazione binario

1. Costruire pipeline di dati in streaming che lavorano prendendo i dati, in modo affidabile, tra sistemi o applicazioni.

2. Costruire applicazioni che lavorano con dati in streaming e attivano determinate funzioni sulla base dei dati stessi.

Kafka è eseguito come cluster su uno o più server e memorizza i flussi in streaming all'interno di categorie, chiamate *topic*, in cui vengono posizionati i vari record, composti da una chiave, un valore e un timestamp.

Kafka ha 4 API fondamentali:

1. **Producer**, che consente ad un'applicazione di pubblicare i record all'interno di uno o più topic.

2. **Consumer**, che consente a un'applicazione di iscriversi a uno o più topic ed elaborare il flusso di record da essi prodotti.

3. **Streams**, che consente ad un'applicazione di agire come un processore, prendendo un flusso in input da uno o più topic e producendo un flusso di output.

4. **Connector**, che consente di creare e gestire producer o consumer che collegano topic ad applicazioni o sistemi di dati esistenti: ad esempio, un connettore a un database relazionale potrebbe acquisire tutte le modifiche effettuate su una tabella.

Per ogni topic, il cluster Kafka memorizza un log partizionato, in cui ogni partizione rappresenta una squenza immutevole di record, ciascuno dei quali viene assegnato a un codice numerico identificativo univoco. Il cluster Kafka mantiene tutti i record pubblicati in modo durevole, a prescindere che siano stati consumati o meno e li rende disponibili per le applicazioni successivamente al cosiddetto periodo di retenzione.

Le partizioni del log sono distribuite nei server del cluster e replicate in diversi server per gestire la resistenza al guasto. Ogni partizione ha un server che agisce da *leader*, che gestisce tutte le richieste di lettura e scrittura, mentre gli altri agiscono da *follower*, replicando passivamente il leader; ogni server agisce da leader per alcune delle partizioni e da follower per altre, garantendo così il bilanciamento all'interno del cluster.

I Consumer vengono divisi all'interno di diversi gruppi, come mostrato nella Figura 2.9, all'interno della quale esistono 2 gruppi di Consumer, **A**, che comprende 2 Consumer e **B**, che ne comprende 4, mentre nei 2 Server del Cluster vengono posizionate 4 partizioni.

Le garanzie fornite da Kafka sono le seguenti:

- I messaggi inviati da un produttore a una particolare partizione di un topic verranno aggiunti nell'ordine in cui vengono inviati. Cioè, se un record R1 viene inviato dallo stesso produttore di un record R2 e R1 viene inviato per primo, R1 avrà un identificativo univoco inferiore rispetto a R2 e apparirà prima nel registro.

- Il Consumer vede prima i messaggi che sono stati inviati prima.

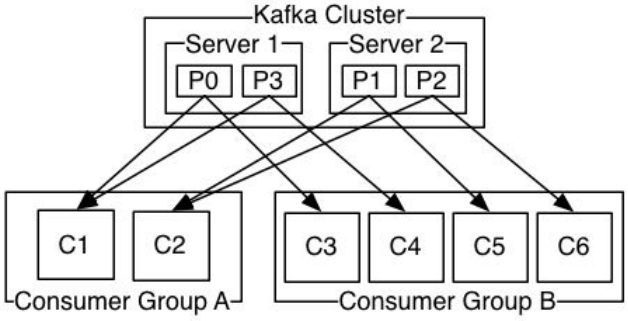

FIGURA 2.9: Gruppi di Consumer in Kafka

- Per un topic con fattore di replica **N**, il cluster Kafka tollera fino a **N-1** guasti senza perdere alcun record.

I sistemi di messaggistica tradizionale si distinguono in due modelli: le code e i modelli *publish-subscribe*. Mentre in una coda, un gruppo di Consumer legge da un server e ogni record finisce su uno specifico Consumer, nel modello "publish-subscribe" il record viene inviato a tutti i Consumer. Il punto di forza della coda è che consente di dividere il processamento dei dati in numerose istanze di Consumer, il che permette di ottenere scalabilità, ma non permette di leggere i dati più di una volta, in quanto una volta che vengono consumati, essi non possono più essere attinti. Il modello publish-subscribe, invece, permette di attingere ai dati più volte, ma perde la scalabilità, in quanto il messaggio viene inviato a tutti i Consumer.

Il concetto di gruppo di Consumer in Kafka generalizza questi due concetti. Come per una coda, il gruppo di Consumer consente di dividere l'elaborazione su una raccolta di processi (i membri del gruppo) e come nel modello publish-subscribe, Kafka consente di trasmettere messaggi a più gruppi di consumatori.

Kafka ha anche garanzie di ordinamento più forti rispetto a un sistema di messaggistica tradizionale. Una coda tradizionale conserva i record in ordine sul server e se più consumatori consumano dalla coda, il server distribuisce i record nell'ordine in cui sono memorizzati. Tuttavia, sebbene il server distribuisca i record in ordine, i record vengono consegnati in modo asincrono ai Consumer. Ciò significa effettivamente che l'ordinamento dei record viene perso in presenza di Consumer paralleli. I sistemi di messaggistica spesso aggirano il problema con la nozione di "Consumer esclusivo", che consente a un solo processo di consumarsi da una coda, ma ovviamente ciò significa negare il parallelismo nell'elaborazione.

Kafka consente ai Producer di attendere il "riconoscimento", una procedura che permette di non considerare la scrittura completa finché non è completamente replicata e garantita. Ciò permette di considerare Kafka anche come un file system distribuito idoneo al processamento dei dati di log, provenienti principalmente da applicazioni appartenenti al dominio della cosiddetta *Internet of Things*.

Un altro caso d'uso di Kafka, è quello del processamento dei flussi di dati in tempo reale: ad esempio, un'applicazione di vendita al dettaglio potrebbe includere flussi di input di vendite e spedizioni e generare un flusso di riordini e adeguamenti dei prezzi calcolati da questi dati.

2.7 Database NoSql

Dagli anni '80 fino al 2011 circa, il mercato dei database era concentrato quasi tutto nelle mani di Oracle, che con più del 50 % di market share, otteneva dei risultati superiori al doppio rispetto al principale competitor, IBM, al termine del 2011 [31] . Il primo passo verso la rottura di questo monopolio è pervenuto attraverso la nascita di prodotti open-source sul campo relazionale come MySQL e PostgreSQL e con l'avanzata prorompente del database di Microsoft.

Il vero momento di rottura è arrivato però con la nascita degli strumenti , che hanno aperto una nuova partita nel mercato dei database, che come si può osservare nella Figura 2.10 stanno acquisendo una popolarità rilevante, specialmente grazie a strumenti come MongoDB, ElasticSearch, Redis e Cassandra, i quali appartengono alla top 10 dei database più popolari, ranking che viene calcolato dal sito **db-engines.com** con un metodo che tiene conto del numero di menzioni del prodotto su Google, Bing e Yandex, la frequenza di ricerca del prodotto, preso da Google Trends, la frequenza delle domande tecniche sul prodotto effettuate su StackOverflow e DBA Stack Excange, il numero di offerte di lavoro in cui il sistema è menzionato, il numero di profili professionali LinkedIn e Upwork in cui il prodotto è citato e infine il numero di Tweet su Twitter in cui compare il prodotto [32].

I database sono degli strumenti che non utilizzano il modello relazionale e dato che inizialmente non erano interrogabili attraverso il linguaggio Sql (peculiarità che è stata ampiamente rilassata negli anni) , hanno assunto questa denominazione.

I vantaggi dei database rispetto ai modelli relazionali sono la scalabilità orizzontale su grandi set di dati, la presenza di schemi meno rigidi e la facilità di gestione e manutenzione, oltre che chiaramente l'economicità che, come visto per HDFS, deriva dall'utilizzo di hardware di basso prezzo. Gli svantaggi sono legati all'assenza dello stesso livello di supporto che viene garantito su strumenti Sql e la difficile integrazione con strumenti di Analytics.

Il Teorema CAP, formulato da Eric Brewer nel 1998, afferma che se i dati di un sistema sono distribuiti tra i nodi di una rete, solo due delle successive tre caratteristiche sono garantibili nello stesso istante:

- Consistency: una richiesta di lettura restituisce il dato proveniente dall'ultima operazione di scrittura dello stesso: tutti i client hanno la stessa vista del dato.
- Availability: l'accesso ai dati è sempre garantito: ogni client può sempre leggere e scrivere

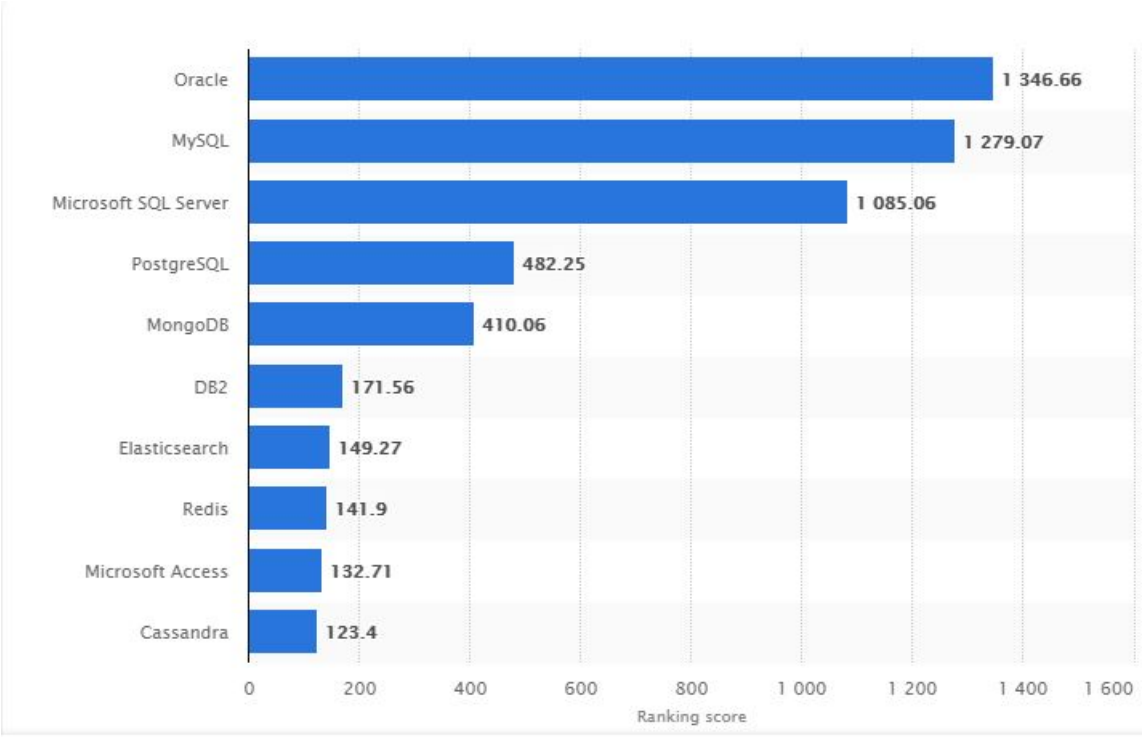

FIGURA 2.10: Popolarità dei database nel 2019 a livello globale.
Fonte: Statista

- Partition tolerance: il sistema continua a funzionare nonostante il partizionamento fisico della rete

I database sono orientati alla gestione di dataset di elevato volume e variabilità , per questo sono spesso utilizzati all'interno di sistemi distribuiti o su piattaforme cloud. Questi sistemi, avendo distribuito i dati su più nodi della rete, richiedono necessariamente la partition tolerance del teorema CAP. Ne consegue che nell'acceso ai dati distribuiti non siano soddisfacibili contemporaneamente le proprietà di availability e consistency [33].

Non va però confuso il significato di consistenza del Teorema CAP con quello delle caratteristiche ACID delle transazioni di un database relazionale, in quanto il teorema CAP parla di consistenza in termini di lettura del valore più aggiornato, mentre nelle transazioni la consistenza fa riferimento al rispetto dei vincoli di integrità, prima e dopo l'esecuzione di una transazione.

Rinunciando all'accesso alla versione più aggiornata di un dato remoto, è possibile realizzare meccanismi di accesso, che non rinunciano completamente alla consistenza CAP dei dati tra i sistemi, ma contano su una particolare forma di consistenza, di tipo asintotica, a regime, che prende il nome di **eventual consistency**.

Questo meccanismo, permette di realizzare il modello transazionale noto con l'acronimo BASE (Basically Available, Soft state, Eventually consistent). L'eventual consistency quindi non è una garanzia di consistenza ma solo una caratteristica del sistema di integrazione, non promette una istantanea e complessiva consistenza dei dati distribuiti ma solo una tendenza delle basi dati ad allinearsi, senza però poter garantire l'istante in cui ciò accadrà.

Key	Document
1001	{ "CustomerID": 99, "OrderItems": [{ "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }
1002	{ "CustomerID": 220, "OrderItems": [{ "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }

FIGURA 2.11: Esempio di database document store.

Le tipologie più comuni di database sono le seguenti:

- **Column-oriented**.Utilizza tabelle, righe e colonne, ma a differenza di un database relazionale, i nomi e il formato delle colonne possono variare da riga a riga nella stessa tabella. Può essere interpretato come un archivio chiave-valore bidimensionale. Hbase e Cassandra sono i due database column-oriented più popolari [32] .

- **Document Store**. Molti database di questa categoria utilizzano formati come Xml, Json, Bson e i dati sono accessibili attraverso API RESTful che utilizzano il protocollo Apache Thrift. Ogni documento, rappresenta una riga del database e può avere strutture completamente differenti rispetto agli altri. I prodotti più popolari in questo campo sono ElasticSearch e MongoDB [32].

- **Key-Value Store**. L'approccio key-value utilizza un array associativo (noto anche come mappa o dizionario) come modello di dati. In questo modello, i dati sono rappresentati come una raccolta di coppie chiave-valore, in modo che ogni possibile chiave appaia al massimo una volta nella raccolta. Il modello può essere esteso a un modello ordinato in modo discreto che mantiene le chiavi in ordine lessicografico.

  Esistono varie implementazioni hardware: alcuni database mantengono i dati in memoria RAM, mentre altri utilizzano unità a stato solido (SSD) o dischi rotanti (HDD).

  I database con il più alto ranking di popolarità tra quelli che utilizzano questo modello è Redis [32] .

- **Graph** Questo tipo di database è progettato per dati le cui relazioni sono ben rappresentabili come un grafo costituito da elementi collegati tra loro con un numero finito di relazioni. Il database a grafo più popolare è Neo4j [32] .

2.7.1 Apache Cassandra

Apache Cassandra è un DBMS open-source, disegnato per scalare orizzontalmente su diverse macchine di basso prezzo, disposte in parallelo, assicurando resistenza al guasto. Il database è nato dagli sviluppatori di Facebook nel 2009, divenendo inizialmente un progetto nella fase incubatrice di Apache e nel 2017 un progetto "Top-level" [34] .

Le caratteristiche principali di Cassandra sono :

- **Sistema distribuito**. Cassandra è eseguito su un cluster in cui ciascun nodo è identico agli altri: non esistono dunque nodi master e slave.
- **Possibilità di replica**. Così come accade all'interno di HDFS, anche Cassandra permette di scegliere una strategia di replica delle partizioni, in modo da garantire resistenza al guasto.
- **Sistema AP**. Cassandra è in genere classificato come un sistema AP, il che significa che la disponibilità (availability) e la tolleranza (partition tollerance) sono generalmente considerate più importanti della Consisency.
- **Integrazione con Hadoop**. Cassandra è integrabile con HDFS e Hive.
- **Linguaggio di interrogazione simile ad Sql**. Cassandra ha un linguaggio, chiamato CQL, che è simile ad Sql e permette ad utenti che hanno familiarità con il noto linguaggio di interrogazione di dati di operare con facilità.

Il modello dati di Cassandra prevede che, come visto per HBase, i dati vengano salvati separatamente per ciascuna **famiglia colonnare**, la quale rappresenta praticamente l'equivalente di un database relazionale, in quanto ciascuna famiglia contiene delle righe, identificate univocamente da una chiave e delle colonne, che possono variare per ciascuna riga, un valore e un timestamp. Tuttavia non è possibile eseguire clausole di Join o query annidate. L'insieme delle famiglie prende il nome di **keyspace**, l'equivalente di un database nel modello relazionale.

2.7.2 MongoDB

MongoDB , scritto in C++ sotto le licenze di GNU Affero General Public[6] e della fondazione Apache , è un DBMS open source distributo orizzontalmente non relazionale utilizzato per applicazioni Web che utilizzano grandi

[6] La GNU Affero General Public License è una licenza di software open-source pubblicata dalla Free Software Foundation.

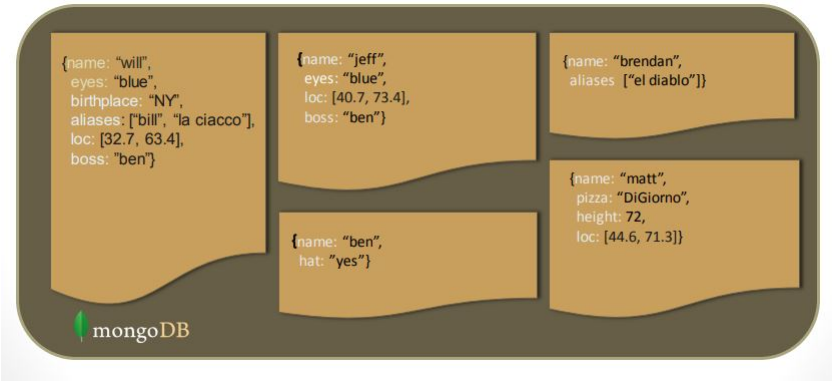

FIGURA 2.12: Esempio di collezione di MongoDB.

quantità di dati, orientato ai documenti che vengono archiviati in formato Json con schema dinamico, definito BSON. La struttura di MongoDB è gerarchica, un database contiene una serie di collezioni, all'interno delle quali sono presenti i documenti, ciascuno avente degli attributi.

Appartiene alla classe di prodotti **CP**, che garantiscono Consistency e Partition tollerance, a scapito dell'Availability.

Le caratteristiche principali di Mongo sono:

- **Query ad-hoc**. Mongo supporta la ricerca di parole chiavi all'interno dei documenti (*field search*), le *range-query*, ossia delle operazioni che recuperano tutti i record in cui un valore è compreso tra un limite superiore e inferiore e la ricerca di espressioni regolari. Le query possono restituire campi specifici di documenti e includere anche funzioni JavaScript definite dall'utente.

- **Indicizzazione**. Vengono utilizzati indici primari e secondari.

- **Bilanciamento del carico**. MongoDB scala orizzontalmente usando lo sharding, una forma di partizionamento orizzontale: l'utente sceglie una chiave di shard, che determina come verranno distribuiti i dati in una raccolta. I dati vengono suddivisi in intervalli (in base alla chiave della partizione) e distribuiti su più partizioni.

- **Replicabilità**. Le partizioni vengono replicate sulle diverse macchine con un'architettura master-slave, in cui esistono delle repliche primarie e delle repliche secondarie.

- **Aggregazione**. MongoDB offre tre modi per eseguire l'aggregazione: la pipeline di aggregazione, la funzione di map-reduce e metodi di aggregazione monouso.

- **Non ci sono Join**. Le operazioni di Join non sono eseguibili.

- **Schema-free**. Il modello dati non deve necessariamente avere uno schema definito: in questo modo ogni documento all'interno della collezione può avere attributi diversi (Figura 2.12).

2.7.3 ElasticSearch

Elasticsearch, sviluppato in Java, è un motore di ricerca basato sulla libreria Lucene[7] che gestisce documenti Json privi di schema. Alcune parti del software sono concesse con varie licenze open source (principalmente la licenza Apache), mentre altre parti rientrano nella licenza proprietaria di Elastic Search. Secondo la classifica di DB-Engines, Elasticsearch è il motore di ricerca aziendale più popolare [32].

Elasticsearch può essere utilizzato per cercare tutti i tipi di documenti, fornisce ricerca scalabile quasi in tempo reale e supporta la multi-tenancy[8].

I dati correlati vengono spesso archiviati nello stesso indice, costituito da uno o più frammenti primari (*shard*) e zero o più partizioni di replica. Una volta creato un indice, il numero di frammenti primari non può essere modificato.

Elasticsearch è sviluppato insieme a un motore di raccolta e analisi dei dati chiamato Logstash, una piattaforma di analisi e visualizzazione chiamata Kibana e Beats, una raccolta di spedizione di dati leggeri. I quattro prodotti sono progettati per essere utilizzati come soluzione integrata, denominata "Elastic Stack" [36] .

Ogni funzione di ElasticSearch è esposta attraverso una API REST:

- **Index API**: utilizzata per indicizzare i documenti.
- **Get API**: utilizzata per richiedere un documento.
- **Search API**: serce a cercare qualcosa all'interno dei documenti.
- **Put Mapping API**: utilizzata per definire il mapping dei documenti.

Elasticsearch ha una proprio linguaggio specifico per effettuare query in formato Json, chiamato DSL, che permette di effettuare interrogazioni complesse attraverso una singola query (Figura 2.13).

2.7.4 Redis

Redis (Remote Dictionary Server) è un DBMS che utilizza strutture chiave-valore distribuite in memoria, supportando diversi tipi di strutture di dati astratte.

Redis può essere considerato allo stesso tempo un archivio e una cache, utilizzando un design in cui i dati vengono sempre modificati e letti dalla memoria del computer principale, ma anche memorizzati su disco per ricostruire i dati in memoria quando viene riavviato il sistema.

Le sue caratteristiche principali sono:

[7] Apache Lucene è una libreria di software per motori di ricerca gratuita e open source.

[8] Il termine "multi-tenancy" si riferisce a un'architettura software in cui una singola istanza di software viene eseguita su un server ed è fruita da diverse organizzazioni che, ciascuna con le sue peculiarità, costituiscono concettualmente uno specifico **tenant**.

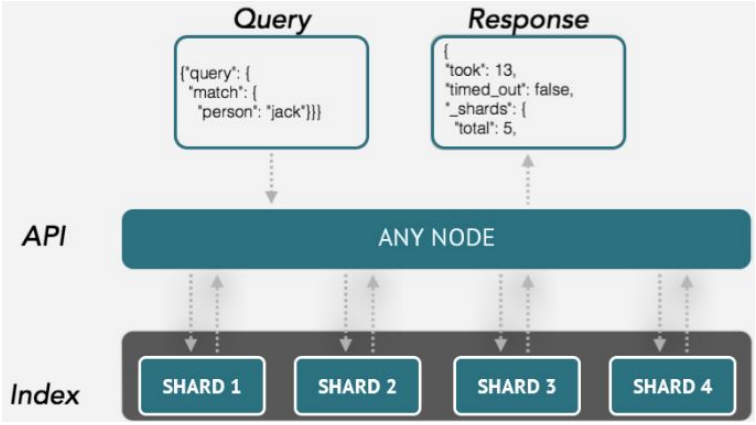

FIGURA 2.13: Esempio di query ElasticSearch.

- **Persistenza**. La persistenza in Redis può essere ottenuta attraverso due metodi. Il primo è definito *snapshot* e consiste nel trasferimento dei dati in modo asincrono dalla memoria al disco a intervalli regolari. Il secondo metodo consiste nel *journaling*, uno strumento che prevede l'aggiunta di un record per ogni modifica che viene effettuata, ad un file di sola aggiunta (AOF) in un processo in background.
- **Replicazione**. I dati di qualsiasi server Redis possono replicarsi su qualsiasi numero di repliche. L'architettura è quella master-slave in cui la replica master è considerata primaria e quelle slave secondaria (Figura 2.14).
- **Performance elevate**. La natura in memoria di Redis gli consente di performare meglio rispetto ai database che scrivono ogni modifica sul disco prima di considerare una transazione conclusa.
- **Clustering**. Un cluster Redis può scalare fino a 1.000 nodi.

Redis supporta diversi tipi di strutture dati [38]:

- Stringhe binarie.
- Liste, cioè delle collezioni di stringhe ordinate.
- Set, cioè delle collezioni di stringhe univoche non ordinate.
- Set ordinati (*Sorted set*), sono dei Set in cui stringa è associata a un valore numerico chiamato score.
- Hash, sono mappe composte da campi associati a valori. Sia il campo che il valore sono stringhe
- Array di bit.
- HyperLogLogs, una struttura di dati probabilistici utilizzata per stimare la cardinalità di un insieme.
- Streams, ossia delle raccolte dati che forniscono una struttura astratta per la gestione dei log.

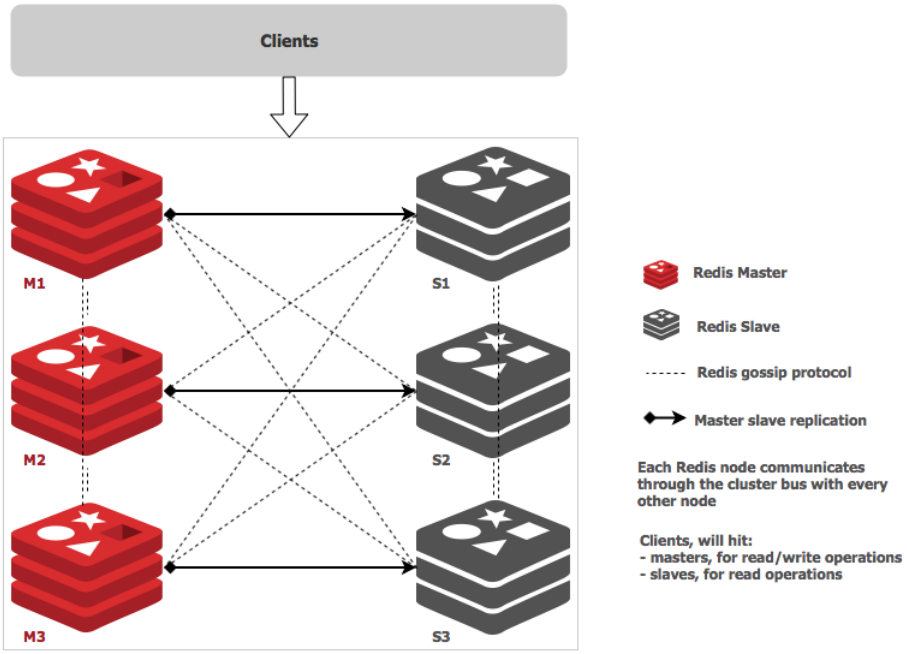

FIGURA 2.14: Esempio di architettura Redis.

2.7.5 Neoj4

Neo4j è un database open-source, scritto in Java e Scala, che utilizza il modello a grafo e supporta le proprietà ACID dei database transazionali. Il suo sviluppo iniziale è cominciato nel 2003, mentre nel 2007 è stata pubblicata la sua prima versione disponibile al pubblico. Neo4j ha sia una edizione Community che una Enterprise. Alcune delle sue caratteristiche fondamentali sono:

- **Cypher**, un linguaggio di query dichiarativo simile a Sql, ma ottimizzato per i grafi.
- **Scalabilità**, attraverso una rappresentazione efficiente dei nodi e delle relazioni, su hardware di prezzo moderato.
- **Flessibilità**, resa possibile da grafi il cui schema può essere adattato ai cambiamenti dei bisogni.
- **Driver** presenti in diversi linguaggi per poter interrogare Neo4j, tra cui Java, Javascript, .NET, Python.

Neo4j offre un servizio di indicizzazione arbitrario ed uno basato sul timestamp, che permette di lavorare solo sui nodi che appartengono a un intervallo di tempo specificato dall'utente.

Neo4j non supporta però il partizionamento dei dati su diverse macchine, quindi ciascun grafo deve essere memorizzato in una sola macchina, ma è possibile eseguire più istanze dello stesso database contemporaneamente su diverse macchine, utilizzando Zookeeper[9].

[9]ZooKeeper è un progetto open-source di licenza Apache, che permette di gestire la sincronizzazione di grandi sistemi distribuiti

Capitolo 3

Studio di soluzioni commerciali di BI e progettazione di un'architettura di BI open-source

3.1 Introduzione

L'obiettivo aziendale principale al quale ho lavorato consisteva nell'individuazione di soluzioni tecnologiche open-source per l'implementazione di un'architettura di Business Intelligence partendo da dati archiviati con i nuovi strumenti Big Data, sperimentando due casi d'uso:

1. Visualizzazione di grafici realizzati su dati archiviati in Hdfs e Hive.
2. Visualizzazione di grafici su dati provenienti da un flusso in streaming gestito da Kafka.

Entrambi i casi d'uso avevano come focus quello di ridurre al minimo la latenza temporale che intercorre tra il momento in cui l'utente vuole visualizzare i dati e il momento in cui ottiene questa visualizzazione.

L'altro obiettivo al quale ho lavorato consisteva invece nell'analisi delle soluzioni *commerciali*, cioè in tutti quelli strumenti proprietari che diverse società vendono alle aziende che desiderano ispezionare intuitivamente i propri dati, fornendo una serie di funzionalità aggiuntive che permettono agli utenti di Business di ispezionare in maniera sempre più approfondita le informazioni presenti all'interno dei dati.

Nel corso degli ultimi anni, i principali vendor di strumenti di Business Intelligence stanno spostando il focus dalla semplice visualizzazione, che permette di effettuare un match con i KPI definiti dal management aziendale, verso un approccio di **agumented analytics**, degli strumenti di analisi in grado di interagire con l'essere umano nel suo linguaggio e di individuare autonomamente i dati più significativi senza la mediazione degli analisti. Invece delle dashboard, gli utenti possono interagire con gli strumenti di analisi attraverso chatbot e interfacce conversazionali in grado di comunicare con un linguaggio umano e di affiancare la presentazione grafica dei dati suggerendo soluzioni. Questo approccio moderno non è però presente, come si vedrà successivamente, nelle soluzioni gratuite che ho testato lavorando al caso di Business, in quanto si tratta di funzionalità di elevato valore.

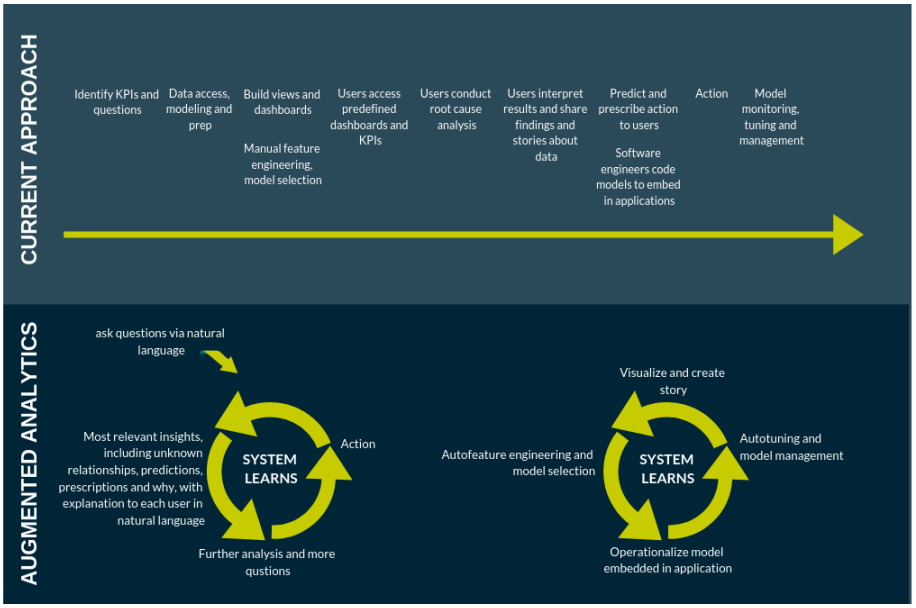

FIGURA 3.1: Dalla visualizzazione di dashboard all'agumented analytics

3.2 I principali strumenti proprietari di Business Intelligence

La società spagnola Gartner realizza ogni anno un report con il quale, attraverso test e sondaggi effettuati alle imprese, valuta la competitività degli strumenti commerciali di Business Intelligence, realizzando il cosiddetto "Magic Quadrant" [39] . Questo tool utilizza due metriche di giudizio : l'abilità del venditore di sviluppare le varie funzionalità e la completezza di visione del prodotto. Attraverso queste metriche viene realizzato un quadrante in cui le varie soluzioni sono classificate all'interno di 4 aree:

1. **Niche players**. Si tratta dei prodotti che non hanno completezza di visione nè capacità di implementazione delle varie funzionalità.

2. **Challengers**. Si tratta delle soluzioni che difettano in visione, ma sono considerate buone dal punto di vista delle funzionalità.

3. **Visionares**. Sono le soluzioni innovative che hanno dei limiti nell'esecuzione delle funzioni.

4. **Leaders**. A questa categoria appartengono i prodotti innovativi, che riescono a fornire anche funzionalità di alto livello e offrono un'elevata user experience. Quì spiccano Tableau e Microsoft, seguiti da Qlik e Thoughtspot.

Dal Magic Quadrant del 2019 emerge un chiaro vantaggio di Tableau e PowerBI rispetto ai competitor. Di questi due prodotti ho potuto testare le funzionalità all'interno delle loro versioni gratuite "Power BI desktop" e "Tableau Public". Inoltre ho analizzato il report di Gartner, per estrarre informazioni

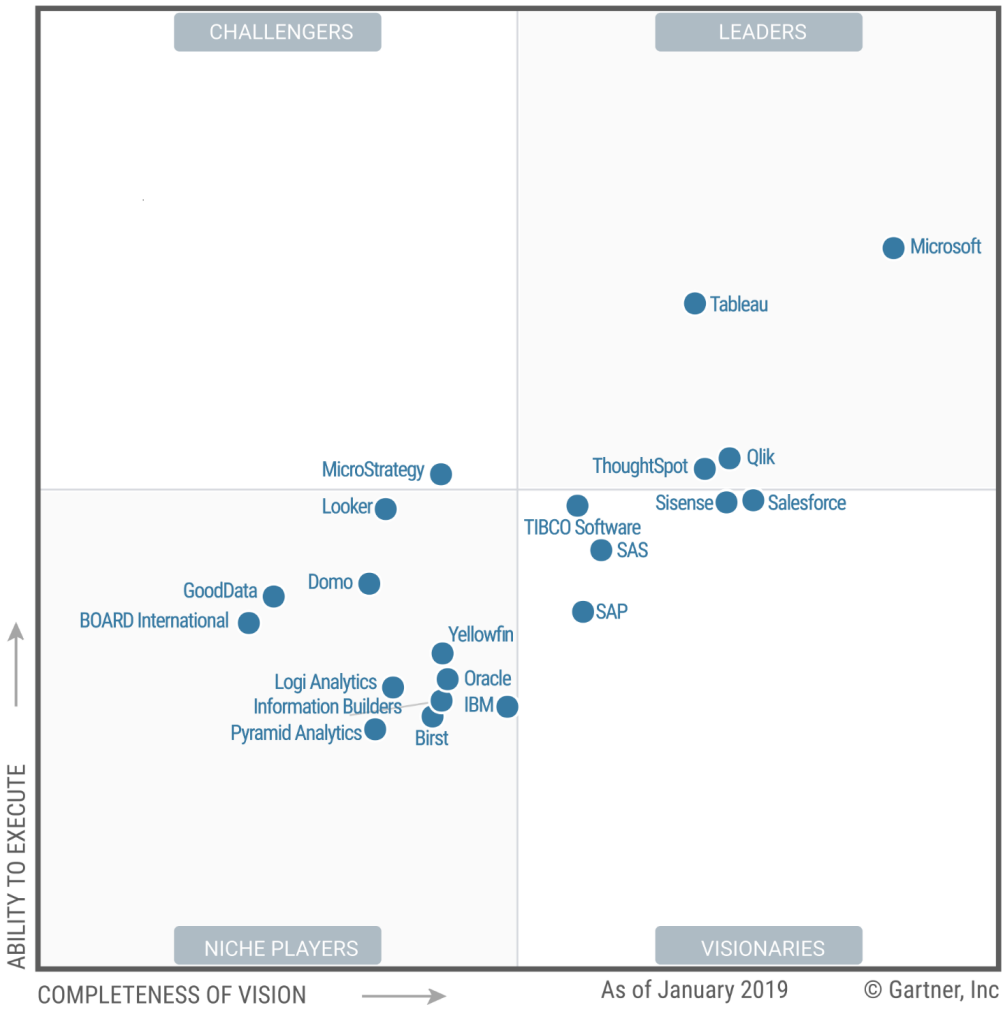

FIGURA 3.2: Magic Quadrant Gartner 2019

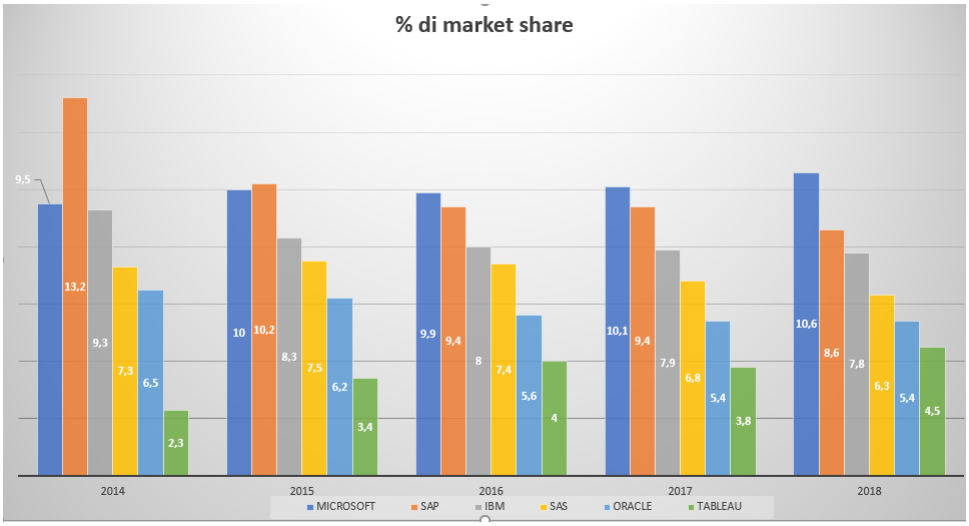

FIGURA 3.3: Market share strumenti BI 2014- 2019

relative ai prodotti con maggiore presenza sul mercato, vale a dire IBM e Oracle, che ho potuto testare in quanto l'azienda con cui ho collaborato nel progetto è partner IBM mentre Oracle fornisce gratuitamente una piattaforma Desktop, SAP e SAS, che invece non ho testato direttamente perché non hanno una versione gratuita e le cui indicazioni sono integralmente estratte dal report di Gartner e dai loro siti ufficiali. Si tratta dei prodotti che hanno maggiore presenza sul mercato e che l'azienda pensava di suggerire ai propri clienti in virtù del brand forte che li caratterizza.

3.2.1 Tableau

Tableau, di proprietà dell'azienda statuitense Tableau Software, offre un'esperienza di esplorazione intuitiva, interattiva e visiva che consente agli utenti aziendali di accedere, preparare, analizzare e presentare i risultati dei propri dati senza competenze tecniche o di programmazione. L'offerta di Tableau viene distribuita principalmente in locale, come applicazione desktop autonoma o integrata con un server per la condivisione di contenuti, mentre Tableau Online è l'offerta SaaS[1] basata su cloud. L'azienda ha inoltre acquisito Empirical Systems nel 2018 per ampliare le sue capacità di analytics. La compagnia è stata riconosciuta come leader nel Magic Quadrant per otto anni consecutivi tra il 2012 e il 2020 raddoppiando la sua quota di mercato dal 2014.
Punti di forza :

1. **Facile esplorazione visiva e manipolazione dei dati**. Tableau consente agli utenti di inserire rapidamente dati da una vasta gamma di fonti, fonderli e visualizzare i risultati utilizzando le migliori pratiche

[1]Saas: Software as a service. Rappresenta un modello di distribuzione di un software applicativo in cui un produttore di software sviluppa e gestisce un'applicazione Web che mette a disposizione dei propri clienti via Internet, spesso utilizzando il cloud computing.

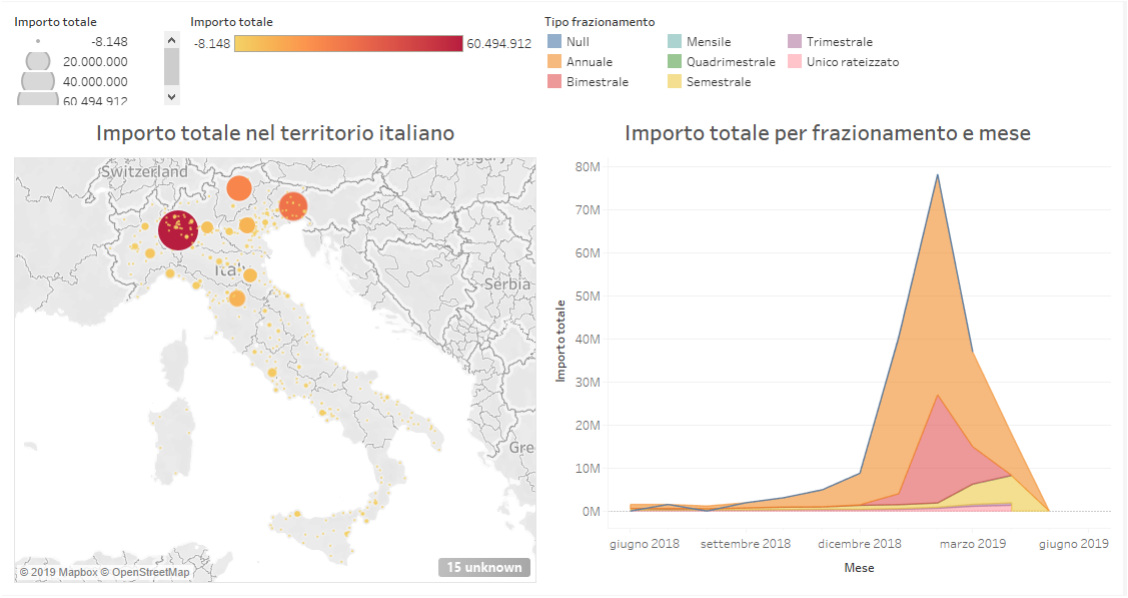

FIGURA 3.4: Esempio di dashboard Tableau Public

nella percezione visiva. I dati possono essere manipolati durante la visualizzazione con un alto grado di facilità d'uso.

2. **Clienti come fan**. I clienti hanno un atteggiamento da fan nei confronti di Tableau, come dimostra il record di 17.000 utenti che hanno partecipato alla sua conferenza annuale del 2018. I clienti che hanno risposto al sondaggio di Gartner hanno collocato Tableau nel primo terzo per esperienza del cliente e hanno assegnato punteggi elevati nel permettere il raggiungimento degli obiettivi aziendali. Tableau organizza numerosi meeting di gruppo e tutorial online che portano ad una maggiore fidelizzazione dei clienti.

3. **Momentum**. Tableau ha aumentato le sue entrate totali a poco più di 800 milioni di dollari durante il terzo trimestre del 2018, una crescita a due cifre rispetto al 2017 e continua ad espandersi all'interno della sua base.

4. **Completezza di connettività**. Tableau permette di connettersi ad un numero vastisstimo di fonti, compresi i prodotti dell'ecosistema Big Data.

Punti di debolezza:

1. **Prezzo elevato** La sottoscrizione di una licenza Tableau per un singolo utente ha un costo di 70 $ al mese, superiore rispetto a quello di altri prodotti.

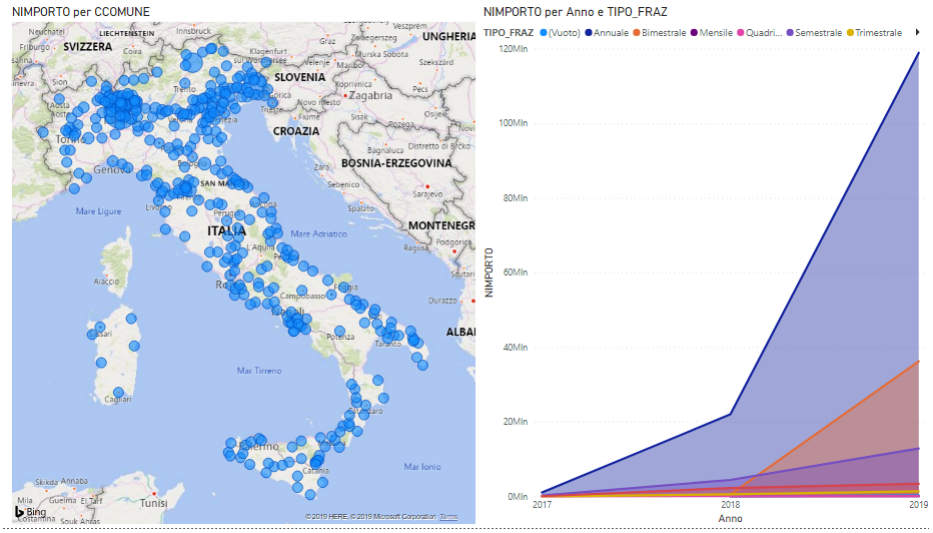

FIGURA 3.5: Esempio di dashboard PowerBI

3.2.2 Power BI

Microsoft offre preparazione dei dati, rilevamento di dati visivi, dashboard interattive e analisi tramite un unico strumento: Power BI. È disponibile come opzione SaaS in esecuzione nel cloud di Azure o come opzione locale in Power BI Report Server. Microsoft ha una roadmap di prodotti completa e visionaria volta a globalizzare e democratizzare Power BI per tutti i casi d'uso. Attualmente si tratta del prodotto con il più elevato market share globale.
Punti di forza :

1. **Prezzi bassi**. Uno dei vantaggi del prodotto di Microsoft è il costo nettamente competitivo: solo 10 $ al mese per la versione Premium.

2. **Facilità d'uso**. Microsoft ha realizzato un prodotto che risulta molto familiare per gli utenti, anche per la vicinanza con Microsoft Excel, un prodotto utilizzato dagli utenti di tutto il Mondo.

Punti di debolezza:

1. **L'opzione cloud è vincolata**. Microsoft non dà ai propri clienti che scelgono di installare Power BI sul cloud la possibilità di scegliere soluzioni alternative ai cloud Azure di loro proprietà.

3.2.3 IBM Cognos

IBM Cognos è una piattaforma di Business Intelligence di IBM. La suite si compone in diverse soluzioni per la creazione di report e analisi e per il monitoraggio di KPI e altri indicatori aziendali. Cognos può attingere al potenziale di calcolo offerto dallo strumento di intelligenza artificiale IBM Watson. Cognos si colloca al limite tra prodotti visionari e *niche players* all'interno del quadrante. **Punti di forza**:

FIGURA 3.6: Esempio di dashboard IBM Cognos 11.

1. **Integrazione dei diversi servizi**. Il desiderio delle organizzazioni di modernizzare i loro portafogli BI tradizionali li ha spesso portati a utilizzare più strumenti di BI per scopi distinti. IBM Cognos Analytics 11.1 è una delle poche offerte che include reporting aziendale esplorazione visiva e agumented analytics in un'unica piattaforma.

2. **Visione**. Alla fine del 2014, IBM è diventato uno dei primi fornitori a rilasciare funzionalità di agumented analytics. L'ultima versione include anche un'interfaccia di assistente di intelligenza artificiale e generazione di linguaggio naturale nativo (NLG).

Punti di debolezza:

1. **Prodotto immaturo**. Sebbene il nuovo prodotto Cognos Analytics sia migliorato rispetto alla versione precedente, gran parte di esso è stato riprogettato, il che ha comportato incoerenze nelle funzionalità. Vi sono inoltre lacune nelle capacità di esplorazione visiva e formattazione. Il nuovo prodotto mostra una scalabilità dei dati limitata e un flusso di lavoro ingombrante per supportare più tabelle.

2. **Prezzo elevato**. La sottoscrizione di una licenza premium Cognos per un singolo utente ha un costo di 70 $ al mese.

3. **Fonti dati compatibili limitate**. IBM Cognos supporta una quantità inferiore di fonti rispetto ai competitor.

3.2.4 SAP

Il prodotto principale di SAP, azienda che, dopo essere stata leader del mercato, è stata superata da Microsoft, è SAP Analytics Cloud, utilizzato principalmente da aziende che possiedono licenze su alcune applicazioni SAP. Il

prodotto è posizionato nel Magic Quadrant tra i visionari, dunque è considerato una soluzione innovativa, ma con dei limiti nella capacità di realizzare le funzionalità.
Punti di forza:

1. **Agumented analytics**. SAP Analytics Cloud è considerato come uno dei migliori prodotti per le funzionalità di agumented analytics.

2. **Differenziazione**. La SAP Digital Boardroom è un altro elemento di differenziazione, particolarmente attraente per i dirigenti perché supporta analisi e simulazioni degli eventi futuri.

3. **Contenuto analitico preconfezionato**. Attingendo all'approccio aziendale che ha avuto inizio con SAP Business Warehouse, SAP offre una libreria di contenuti predefiniti disponibile online. Il contenuto copre una vasta gamma di settori e funzioni di business.

4. **Prezzi contenuti**. Il costo di una licenza di SAP Analytics Cloud per Business Intelligence si aggira attorno ai 25$ al mese, una fascia di prezzo medio-bassa.

Punti di debolezza:

1. **Limiti funzionali**. Sebbene la facilità d'uso sia migliorata, SAP Analytics Cloud è ancora indietro in una serie di aree, in particolare sulla scalabilità al crescere delle dimensioni dei dataset.

2. **Solo cloud**. Una minoranza significativa delle organizzazioni è restia all'utilizzo dei cloud. L'offerta locale alternativa di SAP, SAP Lumira, sarà supportata fino al 2024 e per questo i clienti SAP che richiedono un'implementazione completamente locale di strumenti di Business Intelligence, stanno valutando di cambiare venditore.

3.2.5 SAS

SAS offre Visual Analytics, un'offerta che combina reportistica, preparazione dei dati esplorazione visiva e dashboard in un unico prodotto. Nel 2018, SAS ha rilasciato funzionalità di agumented analytics e NLG, il che lo colloca nel Magic Quadrant, tra i prodotti innovativi, tuttavia i limiti relativi alla realizzazione delle funzionalità, lo conducono tra le soluzioni visionarie, ma non leader.
Punti di forza:

1. **Scalabilità e differenziazione**. SAS offre governance e scalabilità con un'architettura aperta che include Hdfs e Direct Network File System (DNFS) e un proprio motore in memoria per velocizzare le prestazioni. Inoltre si differenzia dai competitor con il supporto per le piattaforme di streaming.

2. **Roadmap innovativa**. SAS è uno dei pochi fornitori del Magic Quadrant a supportare nativamente l'analisi testuale ed è stata inserita anche l'integrazione vocale con gli assistenti digitali personali (come Amazon Alexa).

Punti di debolezza:

1. **Difficoltà d'uso**. La facilità d'uso rimane uno dei criteri di acquisto più importanti e SAS Visual Analytics si posiziona nel terzo inferiore dei venditori del Magic Quadrant in questa tipologia di valutazione.

2. **Problemi di migrazione**. SAS Viya, il nuovo motore di SAS Analytics, ha portato da un lato alla creazione di un ambiente più aperto, ma dall'altro ha portato diverse problematiche legate alla migrazione delle funzionalità dalla vecchia versione.

3. **Prezzo elevato**. SAS Visual Analytics ha un prezzo di partenza di circa 100$ mensili, anche se la licenza può essere utilizzata da 2 utenti permanenti.

3.2.6 Oracle

Oracle offre una vasta gamma di funzionalità di analisi, sia in Oracle Cloud, con Oracle Analytics Cloud (OAC), che in locale. OAC comprende servizi di preparazione e rilevamento dei dati e dashboard interattive; inoltre Oracle Data Visualization Desktop è disponibile in versione gratuita. Tra i prodotti con il più grande market share, Oracle è quello che ha ottenuto il punteggio più basso all'interno del Magic Quadrant.
Punti di forza:

1. **Funzionalità**. Oracle supporta diverse funzioni tra cui NLG, *insight* automatici, agumented analytics e ha anche un'app Mobile chiamata "Day by day".

2. **Visione**. Oracle sta facendo investimenti significativi nell'agumented analytics e nella realtà virtuale, che dovrebbe portare a migliorare l'accuratezza e la pertinenza delle sue capacità di analisi e inferenza dei dati.

3. **Data center globali**. I clienti che scelgono Oracle solitamente sfruttano le sue funzionalità cloud: Oracle offre data center in cloud in quasi tutte le regioni del Mondo. Inoltre, i clienti apprezzano la possibilità di integrazione con le applicazioni Oracle.

Punti di debolezza:

1. **Integrazione tra diverse fonti**. OAC non consente l'integrazione di diverse tabelle in un'unica connessione dati.

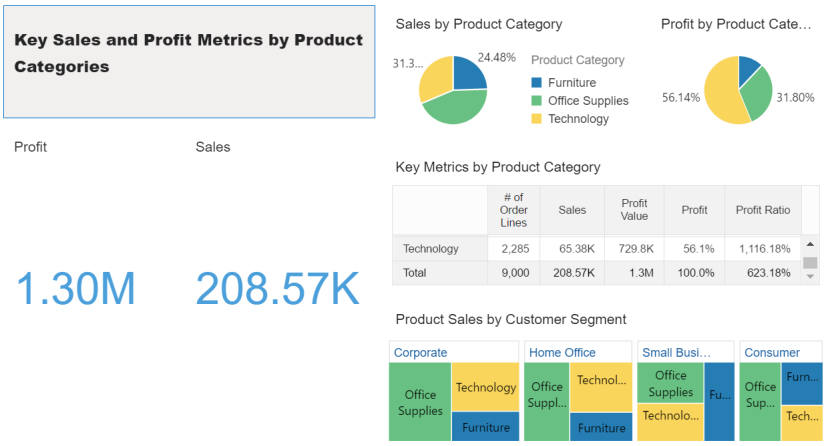

FIGURA 3.7: Esempio di dashboard Oracle Analytics Desktop

2. **Numero di utenze minime da acquistare elevato**. Il costo di OAC per utenza mensile è esiguo, essendo di soli 20 dollari. Tuttavia Oracle vende pacchetti minimi di 20 utenze, il che porta il prodotto ad essere idoneo esclusivamente per organizzazioni che utilizzano in modo massivo operazioni di analisi dati.

3.3 Magic quadrant 2020

All'inizio del mese di Febbraio 2020, la società spagnola Gartner ha pubblicato il Magic Quadrant aggiornato.

Nel nuovo report i risultati sono molto simili rispetto a quelli evidenziati nell'anno precedente.

- PowerBI e Tableau vengono posizonati come leader, in una posizione di netto vantaggio rispetto agli altri competitor.
- Ancora una volta QLik e ThoughtSpot sono i principali competitor dei due prodotti più apprezzati.
- Oracle ha ricevuto un miglioramento del giudizio complessivo, che mi sento di condividere sulla base della mia esperienza con Oracle Visual Analytics Desktop
- IBM, Sas e Sap mantengono una posizione analoga a quella precedente.

3.4 Consigli al management aziendale

Ho potuto testare personalmente, tra i prodotti descritti, le versioni gratuite di Tableau, PowerBI, Oracle Data Visualization e la versione enterprise di prova di IBM Cognos 11.1. A mio avviso, PowerBI è il prodotto con il miglior rapporto qualità-prezzo, in quanto abbina ad un'elevata facilità d'uso e un buon bagaglio di funzionalità, un costo davvero ridotto.

FIGURA 3.8: Magic Quadrant 2020 Gartner

Tableau, è sicuramente un'alternativa valida, grazie alla sua capacità di permettere agli utenti di creare dashboard in pochissimi minuti, di trasformare i dati in modo rapido e di integrare una vastissima quantità di fonti.

IBM Cognos mi è parso invece un prodotto di uso non intuitivo che pur avendo grande potenziale, in quanto le funzionalità di agumented analytics risultano molto potenti e innovative, grazie alle risorse fornite da IBM Watson, non riesce a fornire all'utente di Business una navigazione fluida e presenta alcuni bug visualizzativi poco gradevoli.

Oracle sembra una soluzione adeguata per sfruttare le funzionalità in Cloud, accompagnando una buona esperienza di navigazione. Si tratta di un prodotto che ha maggiori potenzialità rispetto a quelle che emergono dal Magic Quadrant 2019 e il cui utilizzo potrebbe essere approfondito nel caso debba essere utilizzato da clienti che necessitano di numerose utenze.

Non ho potuto testare i prodotti di Sap e Sas, perché non hanno versioni gratuite, ma sicuramente il secondo andrebbe escluso dalle scelte, considerato il costo molto maggiore rispetto ai competitor.

3.5 I principali prodotti open-source per dashboard

Dopo aver analizzato le soluzioni proprietarie che i principali vendor offrono sul mercato, ho indirizzato le operazioni sulle soluzioni open-source: l'obiettivo è quello di proporre ai un'architettura di Business Intelligence che funzionasse senza l'acquisto di licenze, permettendo al contempo all'utente finale di costruire le dashboard senza solide competenze informatiche. Tra i prodotti esplorati ho scelto di concentrarmi principalmente su due soluzioni: Apache Superset e Metabase.

3.5.1 Apache Superset

Apache Superset è un prodotto di Business Intelligence inizialmente sviluppato dai laboratori di AirBnB "engineering data science" per creare dashboard interattive via web. Il progetto è partito nel 2015 da un hackathon assumendo il nome di Caravel, poi di Panoramix e infine di Superset. L'interesse che si è creato intorno al progetto ha fatto sì che venisse ospitato nello spazio GitHub di Apache Fondation e che le ultime versioni siano state rilasciate sotto la licenza ufficiale di Apache.

Superset è scritto in Python e utilizza il framework Flask per la gestione della parte web, mentre la parte che genera i grafici interattivi utilizza NVD3, una libreria Javascript costruita su D3.js. L'installazione di Superset avviene in pochi passaggi su una macchina dove si ha già installato Python3 e alcune dipendenze di sistema. Una volta installato, occorre definire un utente di amministrazione attraverso cui inizializzare alcune configurazioni e eventualmente, caricare dei dati di esempio. Le istruzioni sono presenti per i sistemi operativi più diffusi (distribuzioni GNU/Linux, Windows e MacOSX). A questo punto è possibile connettersi a uno dei database supportati, tra cui nella guida ufficiale di Superset compaiono: Amazon Athena, Amazon

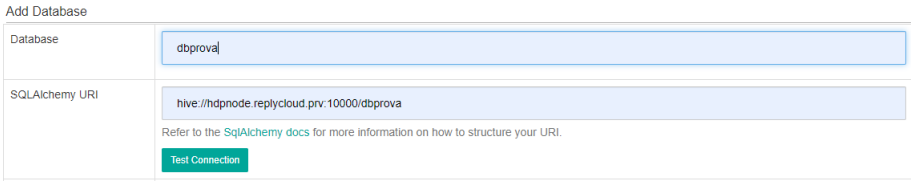

FIGURA 3.9: Esempio di connessione ad Apache Hive tramite Superset.

Redshift, Apache Drill, Apache Druid, Apache Hive, Apache Impala, Apache Kylin, Apache Pinot, BigQuery, ClickHouse, Google Sheets, IBM Db2, MySQL, Oracle, PostgreSQL, Presto, Snowflake, SQLite, SQL Server, Teradata, Vertica.
Questi database sono interrogabili attraverso un kit di strumenti che prende il nome di SQLAlchemy [41], installando un driver Python che si collega ad uno strumento denominato *dialect*, il quale permette al client di Superset di interrogare il database in linguaggio SQL.

3.5.2 Le fasi di creazione di una dashboard con Superset

Dopo aver installato Superset e creato un utente, il servizio può essere lanciato da linea di comando con la seguente istruzione:

```
superset run -p 8081 -h 0.0.0.0 --with-threads --reload --debugger
```

In questo modo il servizio è esposto sull'indirizzo della macchina su cui è stato lanciato il comando, alla porta 8081. L'utente può dunque accedere tramite un qualsiasi browser all'applicazione web e scegliere se visualizzare grafici o dashboard già caricati, oppure crearne di nuovi, a partire da database ai quali il servizio è già collegato, oppure aggiungendone di nuovi. Una funzionalità molto utile è quella di poter caricare un Csv, che Superset trasforma in una tabella Sql all'interno del database SQLIte3 "main" integrato. Qualora l'utente voglia connettersi ad un nuovo database in remoto, dovrà inserire la stringa di connessione e il nome del *data source* assicurandosi che il tipo scelto sia compatibile con i *dialect* già installati. Ad esempio, volendo stabilire una connessione con Apache Hive, occorre installare il pacchetto *pyhive* sulla macchina su cui è stato posizionato Superset, con il comando *pip install pyhive* e successivamente inserire, nella sezione di Superset apposita, la url di connessione nel seguente formato: *hive://host:port/database*

Dopo aver stabilito la connessione l'utente dovrà scegliere su tabella realizzare i grafici. La tabella sarà caricata da Superset all'interno di un dataframe esposto dalla libreria Pandas, in cui ciascuna colonna potrà essere essenzialmente di 3 formati: Stringa, Numero o Data. I dati devono essere conformi alla formattazione gestita da Pandas. Inoltre, numerosi grafici richiedono la presenza dell'informazione legata alla data, in formato Timestamp, per poter essere realizzati. Occorre fare attenzione alla formattazione con cui la colonna del Timestamp viene caricata su Superset.

Di seguito alcuni esempi errori che scaturiscono da problemi di formattazione su Superset:

- I numeri non devono devono avere segni di punteggiatura come separatori, ad esclusione del punto, che separa la parte intera da quella decimale. Esempio : Il numero **"millequattrocentrocinquanta virgola dieci"** deve essere caricato come **"1450.10"** altrimenti sarà parsificato come NULL.
- Ogni data deve essere nel formato Timestamp secondo la notazione "anno-mese-giorno ora:minuti:secondi". Esempio: la data **"26 Ottobre 2019"** deve essere inserita come **"2019-10-26 00:00:00"**

L'utente dopo aver scelto la tabella, può realizzare il grafico, scegliendo tra i 48 disponibili ed entrando nell'editor in cui può scegliere le variabili necessarie a formulare la query. Nella Figura 3.10 si nota come sia possibile per un utente che non conosce la sintassi SQL, inserire i parametri che generano la query e dopo aver lanciato il comando *run*, ricevere in risposta, se i parametri sono corretti, il grafico richiesto. Se l'azione è andata a buon fine l'utente può salvare il grafico e inserirlo all'interno di una dashboard, dove verrà posizionato al fondo rispetto ai grafici già inseriti. Sempre dalla Figura 3.10 sono riassunte le variabili necessarie per la generazione di un grafico:

- **Tabella**;
- **Tipo di visualizzazione**;
- **Intervallo temporale**. Rappresenta l'intervallo di tempo per il quale si vogliono osservare i dati e corrisponde ad una clausola WHERE nella query SQL;
- **Dimensione / serie**. Rappresenta la varabile da analizzare, che verrà inserita nella query all'interno della clausola SELECT e GROUP BY;
- **Misura**: rappresenta il calcolo che si intende effettuare, può essere una somma (SUM), una media (AVG), un minimo (MIN), un massimo (MAX), un conteggio (COUNT), rispetto alla dimensione.
- **Caratteristiche del grafico**: si riferisce a tutte le caratteristiche grafiche che sono personalizzabili durante la creazione del grafico, come la leggenda, i colori, gli assi etc.

Un'importante funzione è quella di poter scaricare i risultati mostrati nel grafico in formato Json o Csv senza limiti di numero di righe del risultato. Accedendo alla dashboard si possono modificare le dimensioni dei grafici, personalizzare il Css[2], inserire titoli, cambiare la posizione dei grafici nella dashboard.

[2]Il Css è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML, ad esempio i siti web e relative pagine web.

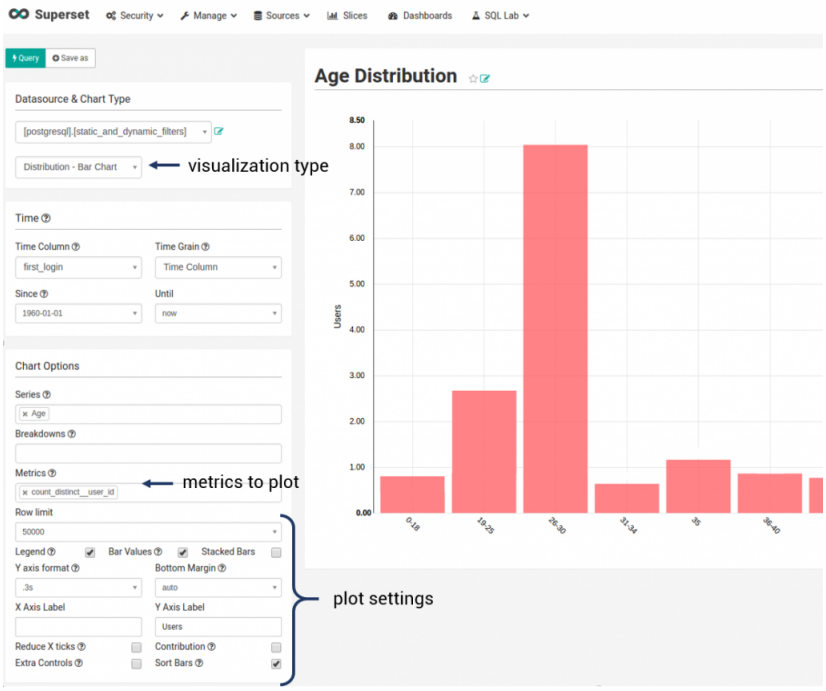

FIGURA 3.10: Esempio di creazione grafico su Superset.

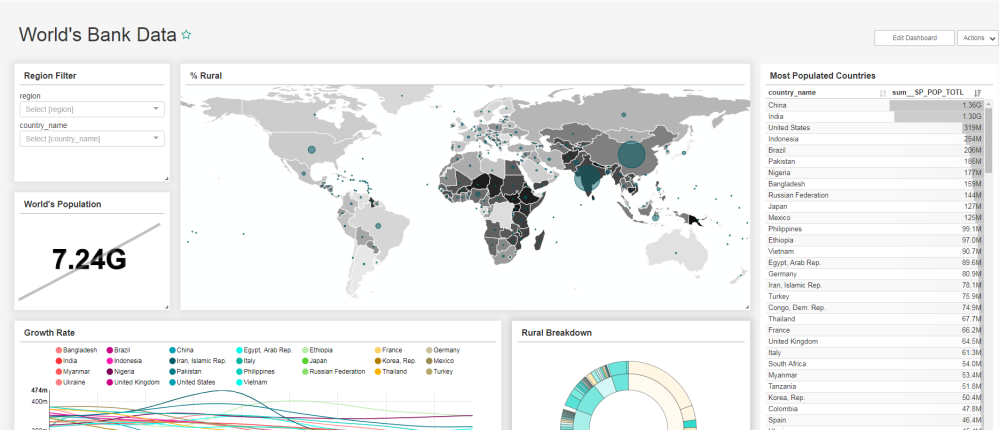

FIGURA 3.11: Esempio di dashboard realizzata su Superset.

3.5.3 Metabase

Metabase è l'alternativa più credibile a Superset tra i prodotti open-source. Scritto in Java, risulta molto semplice da installare su una macchina in cui è presente Java8 e uno tra OpenJDK e Oracle JRE. Dopo aver scaricato Metabase, il semplice comando *"java -jar metabase.jar"* lo espone sulla porta 3000 della macchina su cui viene lanciato. Una volta collegatosi al servizio web, l'utente può registrare le proprie credenziali e cominciare la navigazione. Metabase incorpora al suo interno una serie di Driver per la connessione alle diverse fonti e ha un numero limitato di database a cui collegarsi e non supporta l'opzione di permettere all'utente di caricare un Csv. L'elenco dei database supportati è: Postgres, MySQL, Druid, SQL Server, Redshift, MongoDB, BigQuery, SQLite, H2, Oracle, Vertica, Presto, Snowflake, SparkSQL.

3.5.4 Le fasi di generazione di una dashboard su Metabase

Anche Metabase permette all'utente di creare un nuovo grafico senza dover scrivere una query SQL, semplicemente andando a selezionare tramite interfaccia grafica le variabili desiderate. Chiaramente, per poter effettuare un'analisi, bisogna innanzitutto stabilire una connessione con il database. A differenza di Superset, che prevede che il grafico sia creato accedendo alla sezione "crea nuovo grafico", in Metabase invece, bisogna entrare nella sezione di esplorazione della tabella per poter realizzare il grafico e cliccare nella sezione "Visualizzazione" esposta in basso, dalla quale si apre un menù che permette di scegliere la tipologia di grafico, tra i 14 disponibili.

Dopo aver scelto il grafico l'utente dovrà inserire le stesse variabili che erano state descritte nel caso di Superset: intervallo temporale, dimensioni, misure e caratteristiche del grafico, queste ultime personalizzabili in modo molto più ampio rispetto a Superset. Facendo, ad esempio, riferimento alla realizzazione di un grafico a torta, mentre in Superset si può soltanto scegliere la scala di colori che saranno automaticamente mostrati nel grafico finale, Metabase permette di selezionare il colore per ciascuna fetta della torta. C'è però un limite sul numero massimo di righe che il risultato della query può avere, cioè 2000, dopodichè i dati sono troncati dalla visualizzazione. Un valore non così modesto ma che rappresenta comunque un limite rispetto a Superset. Anche Metabase permette di scaricare il risultato del grafico mostrato in formato Csv,Json e anche Xlsx. Una volta realizzato il grafico, può essere aggiunto a una Dashboard.

Le funzionalità di Metabase che lo differenziano dal principale competitor sono la possibilità di esplorare le tabelle attraverso dei grafici che vengono automaticamente mostrati all'utente al fine di conferire una panoramica globale del dataset, senza alcuna operazione manuale e la possibilità di eseguire delle join tra più tabelle da interfaccia grafica.

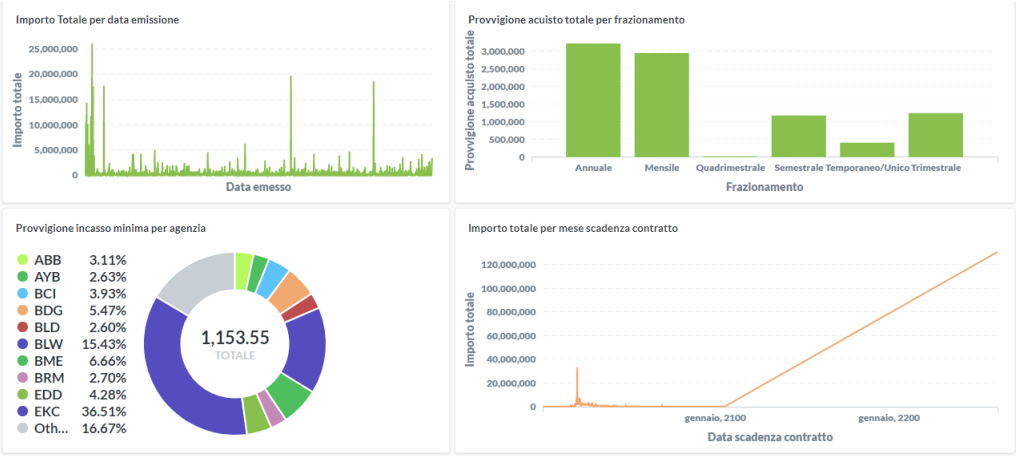

FIGURA 3.12: Esempio di dashboard su Metabase.

3.5.5 Confronto tra Metabase e Superset, test con Hive, scelta finale.

Nella tabella presente nella Figura 3.13 sono analizzati in modo schematico le differenze tra Metabase e Superset. L'analisi comparativa mostra elementi di forza e debolezza per entrambe le soluzioni: se da un lato Metabase è più semplice da installare e ha delle funzionalità aggiuntive, dall'altro Superset si mostra come un prodotto più aperto, in quanto la connessione ai database non è limitata ai driver installati al suo interno, bensì delegata al kit SQLAlchemy, nel quale vengono di frequente aggiunte nuove fonti dati da parte della community: ad esempio esiste un dialect che permette la connessione con Apache Kylin. Tale prodotto è stato molto importante per la definizione della soluzione finale, in quanto permette di realizzare dei Data Warehouse a partire da tabelle Hive o code in streaming Kafka. I Data Warehouse permettono all'utente di interrogare i dati con latenze temporali minime, dato che i calcoli vengono effettuati nella fase di creazione del Data Warehouse e non per ogni volta che l'utente richiede la visualizzazione del grafico.

Durante i test da me effettuati sia tramite Superset che Metabase, di tentativo di generazione di grafici connettendomi direttamente ad una tabella posizionata su Hive (caso batch), i tempi di latenza sono stati molto elevati, passando dai 15/20 secondi richiesti per una tabella con 8 record, ad oltre 1 minuto per una tabella con 250 record. Latenze che per un utente di Business appaiono non tollerabili e che risulterebbero chiaramente superiori su dataset con alcune migliaia di record.

Tali evidenze mi hanno portato ad accantonare l'idea di poter proporre ai clienti Superset o Metabase sulla base delle funzionalità esposte e cercare di testare se un dataset posizonato su Hive, attraverso l'utilizzo di Kylin, potesse essere trasformato in un Data Warehouse in grado di consentire la realizzazione di grafici e dashboard con latenze temporali ridotte. Dato che Metabase non può essere collegato ai Data Warehouse esposti da Kylin, l'unico prodotto che avrei potuto utilizzare è stato Superset.

FUNZIONALITA	SUPERSET	METABASE
Connettività con i principali database SQL	Sì	Sì
Connettività con Kylin	Sì	No
Connettività con database NOSQL	Possibile per i DB che hanno un dialect SQLAlchemy	MongoDB
Esportabilità risultato query	Json e CSV No limite righe	JSON, CSV e XLSX Limite di 1.000.000 righe
Difficoltà installazione	Medio/alta	Bassa
Numero di grafici supportati	48	14
Join tra tabelle	No	Sì
Esplorazione automatica delle tabelle	No	Sì
Qualità visiva dei grafici	Elevata	Media
Importazione CSV da interfaccia utente	Abilitata	Non abilitata

FIGURA 3.13: Superset vs Metabase.

Prodotto	**Numero record**	**Latenza [s]**
Superset	8	16.12
Metabase	8	16.79
Superset	250	67.88
Metabase	250	68.56

TABELLA 3.1: Latenza media di connessione ad Hive tramite Superset e Metabase

3.6 Apache Kylin: OLAP su Hadoop

Apache Kylin è un prodotto open-source utilizzato come motore di analisi distributo per effettuare operazioni di OLAP su Hadoop, riproponendo le tecnologie di progettazione di cubi multi dimensionali e di pre-calcolo dei risultati aggregati, nell'era Big Data, al fine di ridurre la latenza temporale per i client che vogliono effettuare operazioni (visualizzazione grafica, data mining e machine learning) sui Big Data. Kylin costruisce cubi multidimensionali (MOLAP), a partire da tabelle Hive (ROLAP), seguendo le specifiche dei metadati. Gran parte degli strumenti che permettono di interrogare tabelle archiviate su Hadoop, sono strumenti ROLAP, che performano meglio su grandi quantità di dati, ma che richiedono dei tempi di accesso incompatibili con la bassa latenza richiesta dagli utilizzatori di strumenti di Business Intelligence. Kylin memorizza i cubi MOLAP su HBase e quando riceve una query, la indirizza ad HBase e fornisce una risposta quasi immediata, se il risultato richiesto fa parte del cubo, mentre indirizza la query ad Hive in caso contrario. Il progetto, nato nel 2013, negli uffici di Ebay a Shanghai, ha avuto la sua prima versione disponibile nell'Ottobre del 2014, quando è stato pubblicato su github.com con il nome di "KylinOLAP" [42]. Successivamente il prodotto è entrato nell'icubatore dell'Apache Software Foundation, nel Dicembre 2015 è divenuto un "Top Level Project" e sempre nello stesso anno sono stati aggiunti i supporti per PowerBI, Excel e Tableau. Nel 2016, con la versione 1.5.2 è stato aggiunto il supporto ufficiale di MapReduce, mentre nell'anno successivo è stata inserita la compatibilità con Spark per eseguire i calcoli. Tra il 2018 e il 2019 sono state aggiunte altre funzionalità importanti come l'utilizzo del più efficiente SparkSQL e la possibilità di connettersi a topic Kafka .

Tra le altre funzionalità c'è la possibilità di utilizzare il driver JDBC da parte dei vari client per accedere ai dati e l'esistenza di un dialect SQLAlchemy che lo rende compatibile con Superset.

3.6.1 Background di Kylin

La sfida affrontata da eBay dal 2013 era legata alla grande quanità di dati che erano generati giornalmente, contemporaneamente all'esigenza degli utenti delle aree analitiche e di business di ottenere con basse latenze degli output esplorativi attraverso strumenti come Tableau, Excel e PowerBI. Di quì, Ebay ha pensato a quali fossero le caratteristiche di un prodotto che potesse rispondere al bisogno descritto:

- Latenza delle query molto bassa su enormi quantità di dati.
- Compatibilità con gli standard ANSI di SQL.
- Gestione della concorrenza per diversi utenti.
- Compatibilità con Hadoop.

Dato che nel 2013 non esisteva alcuna soluzione che soddisfacesse i punti elencati, sono stati avviati i lavori per la progettazione della piattaforma.

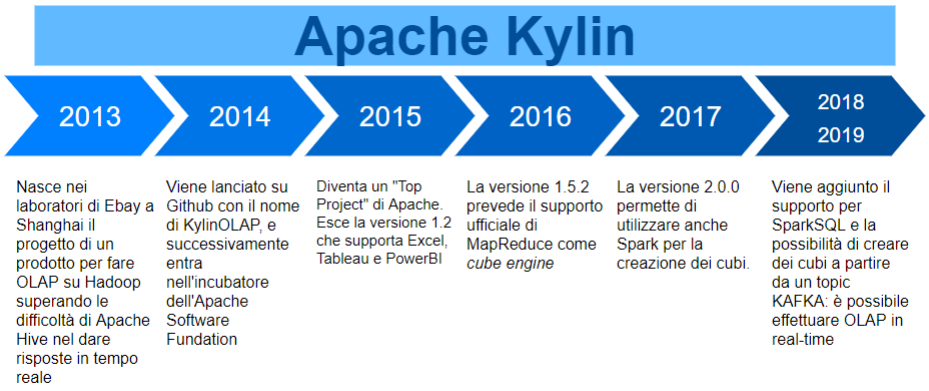

FIGURA 3.14: Apache Kylin dal 2013 al 2019

3.6.2 La spina dorsale di Kylin

Kylin utilizza le tecniche di creazione di cubi multi-dimesnionali analizzate nel Capitolo 1 effettuando delle operazioni di "pre-calcolo", le quali quando i dati assumono dimensioni molto elevate possono portare a latenze non gestibili, anche con un hardware potente. Tuttavia, con il vantaggio della potenza di calcolo distribuita di Hadoop, Kylin porta i processi di calcolo a sfruttare numerosi nodi, il che consente di eseguire calcoli in parallelo, riducendo in modo significativo il tempo di elaborazione.

3.6.3 Dal modello relazionale a quello chiave-valore

Supponendo di avere una tabella memorizzata su Hive, che contiene i dati delle operazioni gestitite da tutti i possessori di un conto bancario di uno specifico Istituto, qualora gli utenti di Business della banca intendano monitorare "In quali province italiane c'è stato un flusso di transazioni superiore a 10 Milioni di Euro nel 2019?", l'interrogazione diretta di Hive fornirà un risultato con latenze molto elevate, come si può immaginare dai test esposti nella Tabella 3.1.

Attraverso Kylin, viene fornita una soluzione a questo problema, attraverso i seguenti step:

1. Collegamento al database
2. Realizzazione di un Dimensional Fact Model
3. Costruzione del cubo tramite MapReduce/Spark ed esposizione dei risultati su HBase
4. Permette, tramite Apache Phoenix, di richiedere i dati presenti sul cubo tramite query Sql da parte di numerosi client.

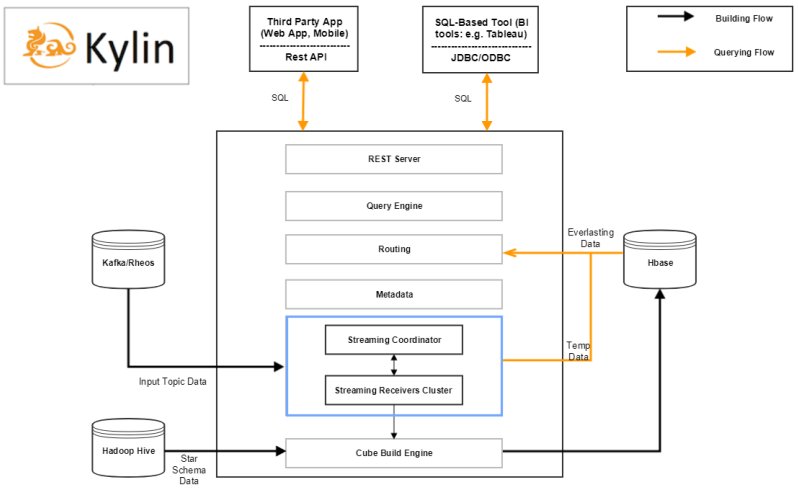

FIGURA 3.15: Architettura di Apache Kylin.

3.6.4 Architettura

L'immagine 3.12 mostra l'architettura completa di Apache Kylin. Il rettangolo blù rappresenta l'insieme dei componenti streaming aggiunti nell'architettura di Kylin dalla versione 3.0.0. Gli elemeti fondamentali di Kylin sono:

- **Metadata Manager**. Kylin è un'applicazione basata sui metadati e questo è il componente chiave che li gestisce.
- **Job Engine**. Si tratta del motore progettato per gestire tutti i Job.
- **Storage Engine**. Questo motore gestisce l'archiviazione, in particolare dei cuboidi, che vengono memorizzati come coppie chiave-valore.
- **Server Rest**. Il server REST è un punto di ingresso per le applicazioni web, che possono inviare query, ottenere risultati, attivare processi di creazione del cubo, ottenere metadati etc.
- **ODBC/JDBC Driver**. Permette ai client di interrogare i cubi realizzati da Kylin tramite Driver Jdbc/Odbc.
- **Query Engine**. Questo motore riceve e indirizza le query dei client verso i cubi esposti su HBase.

3.6.5 Creazione di cubi a partire da code in streaming

Come anticipato, Kylin può essere utilizzato sia per la realizzazione di Data Warehouse a partire da tabelle memorizzate su Hive, che a partire da code generate da un Producer dell'applicativo di streaming distribuito Kafka.

Le fasi che portano un topic (una coda) di Kafka a essere trasformato in un cubo e che richiedono l'intervento di un **Coordinator** e di più **Receiver**, sono le seguenti:

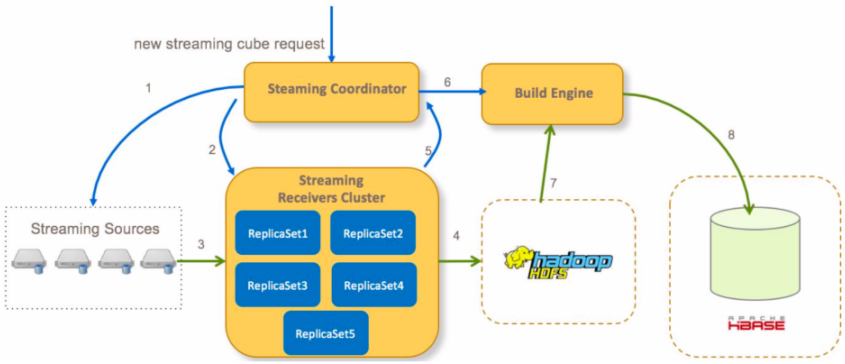

FIGURA 3.16: Kylin: streaming e batch

1. Il Coordinator richiede la sorgente di streaming per tutte le partizioni del cubo.
2. Il Coordinator decide a quali Receiver assegnare la lettura dei dati.
3. I Receiver indicizzano gli eventi in streaming.
4. I Receiver copiano i segmenti da file locali in file remoti Hdfs.
5. Il Coordinator invia un job per la creazione del cubo al "Build Engine" dopo che tutti i Receiver hanno inviato i propri segmenti.
6. Il Build Engine costruisce il cubo a partire dai file Hdfs.
7. Il Build Engine archivia i dati del cubo su Hbase, quindi il Coordinator chiede ai Receiver di rimuovere i relativi dati locali.

3.6.6 Creazione di cubi: gli step realizzati da Kylin

Kylin decompone un'attività di creazione del cubo in più passaggi e quindi li esegue in sequenza. Questi passaggi includono operazioni Hive, job MapReduce/Spark e altri tipi di Job.

1. **Creazione di una tabella intermedia Hive**. Kylin estrae i dati dalle tabelle Hive di origine e li inserisce in un'unica tabella intermedia. Se il cubo è partizionato, Kylin aggiungerà una condizione temporale in modo che vengano recuperati solo i dati nell'intervallo.
2. **Ridistribuzione della tabella intermedia**. Dopo il passaggio precedente, Hive genera i file di dati nella cartella HDFS: mentre alcuni file sono grandi, alcuni sono piccoli o addirittura vuoti. La distribuzione sbilanciata dei file porterebbe allo sbilanciamento anche dei successivi Job: alcuni finirebbero rapidamente, mentre altri sarebbero molto lenti. Per bilanciarli, Kylin aggiunge questo passaggio per "ridistribuire" i dati.

3. **Estrazione dei valori distinti dalle dimensioni** . In questo passaggio Kylin esegue un MapReduce per recuperare tutti i valori distinti che possono avere le varie dimensioni. Questo passaggio inoltre raccoglie le statistiche del cubo utilizzando i contatori HyperLogLog per stimare il conteggio delle righe di ciascun cuboide.

4. **Costruzione del dizionario delle dimensioni**. Con i valori distinti recuperati nel passaggio precedente, Kylin costruirà dizionari in memoria.

5. **Salvataggio delle statistiche sui Cuboidi e creazione della tabella HBase**.

6. **Costruzione del cuboide base**. In questo step viene costruito il cuboide di base dalla tabella intermedia.

7. **Costruzione del cubo**. Questo passaggio utilizza un algoritmo per creare tutti i cuboidi a partire dal cuboide base, utilizzando un Job Spark o MapReduce, in base alla scelta dell'utente.

8. **Conversione del cuboide in un HFile**. Questo passaggio avvia un Job MR per convertire i cuboidi nel formato HFile di HBase. Kylin calcola il numero delle regioni HBase utilizzando le statistiche del cubo, che di default è di 1 regione ogni 5 GB.

9. **Caricamento del HFile su HBase**. Questo passaggio utilizza l'API HBase per caricare l'HFile sui server HBase.

10. **Aggiornamento delle informazioni sul cuboide**. Dopo aver caricato i dati in HBase, Kylin contrassegna questo segmento del cubo come completato nei metadati.

11. **Pulizia**. Viene eliminata la tabella intermedia da Hive.

3.7 Soluzione proposta

Dopo aver fornito le evidenze descitte fino al paragrafo precedente, mi è stato consegnato dal referente aziendale un file Csv proveniente dai pagamenti di polizze sulla vita di una famosa compagnia assicurativa italiana e l'accesso ad un server dotato del sistema operativo CentOs 7.0, concepito per fornire una piattaforma di classe enterprise per utilizzare GNU/Linux. Nel server ho installato l'architettura Docker, una tecnologia di containerizzazione che consente la creazione e l'utilizzo dei container Linux, in cui effettuare il deployment di determinati servizi a partire da un'immagine. Ciò semplifica la condivisione di un'applicazione o di un insieme di servizi, con tutte le loro dipendenze, nei vari ambienti. Dopo aver installato l'architettura Docker ho scaricato un container, lanciando il comando "*docker pull apachekylin/apache-kylin-standalone:3.0.0-alpha2*" e dopo aver terminato il download, ho lanciato il comando

```
-m 8G \
-p 7071:7070 \
-p 8089:8088 \
-p 50071:50070 \
-p 8033:8032 \
-p 8043:8042 \
-p 60011:60010 \
apachekylin/apache-kylin-standalone:3.0.0-alpha2}
```

per attivare i seguenti servizi: Java, Hadoop 2.7.0, Kafka 2.11, Spark 2.3.1, HBase 1.1.2, Hive 1.2.1, Kylin-3.0.0-alpha2.

Gli step successivi sono statii:

1. Modificare il Csv in modo che sia compatibile sia con Apache Hive che con Pandas, attraverso uno script Python, caricarlo su Hdfs esporlo come tabella esterna su Hive, creare il cubo con Kylin e interrogarlo da Superset per realizzare dashboard.

2. Realizzare un'applicazione Java che contiene un Producer Kafka che invia dei messaggi in streaming archiviati all'interno di un topic e utilizzare Kylin per costruire un cubo da archiviare su HBase, che può essere interrogato da Superset per realizzare una dashboard.

3.8 Caso d'uso batch

Il Csv di esempio contiene 91.941 record, con 62 colonne. Per realizzare la dashboard, mi è stato consigliato dal referente aziendale di utilizzare solo alcune delle colonne come dimensioni sulle quali effettuare i calcoli, cioè:

- **Agenzia**. Può avere diversi valori corrispondenti al nome dell'agenzia di riferimento. La cardinalità dell'attributo è 775.

- **Ramo**. In questo caso esistono solo 4 possibili valori: 19, 37, 38, 39.

- **Tipo_Titolo**. Le transazioni possono far riferimento a 4 tipologie di titoli: Nuova Polizza, Quietanza di Rinnovo, Appendice, Quietanza di Frazionamento.

- **Frazionamento**. I pagamenti delle polizze hanno diverse scadenze: Mensile, Temporaneo/Unico, Trimestrale, Semestrale, Quadrimestrale, Annuale.

- **Data_Inserimento**. Fa riferimento alla data di inserimento del pagamento nel sistema della compagnia assicurativa.

- **Data_Scadenza_Contratto**. Chiaramente indica la data in cui scade la polizza.

- **IdContraente**. Indica l'identificativo univoco dell'individuo che ha contratto la polizza.

- **Importo**. Rappresenta l'importo della transazione a cui il record fa riferimento.
- **Provvigione_acquisto**. Indica la provvigione sulla conclusione di un nuovo contratto.
- **Provvigione_incasso**. E' l'incasso dell provvigione su un contratto già in essere.
- **Sovrapprovvigione_acquisto**.
- **Sovrapprovvigione_incasso**.
- **Effetto quota**. Indica la data di inizio di validità della transazione effettuata.
- **Data emissione**. Corrisponde generalmente a data inserimento.
- **Importo provvigioni**. Rappresenta l'importo delle provvigioni, senza specificare se si tratti di acquisto o incasso.

Considerando che tra i campi scelti, quelli numerici e quelli temporali devono essere compatibili con gli standard di HiveQL e la libreria Pandas utilizzata da Superset, ho realizzato uno script Python, il cui codice è descritto nell'Appendice A.

Dopo aver eseguito lo script , il Csv è pronto per essere messo sul cluster Hdfs presente all'interno del container e per essere poi posizionato una tabella esterna Hive. I comandi utilizzati sono descritti nell'Appendice B.

La tabella posizionata su Hive può essere utilizzata da Kylin per la costruzione del cubo. Con la connessione da browser web alla porta 7071 dell'indirizzo IP della macchina su cui è stato installato il container, è possibile accedere a Kylin. Una volta eseguito l'accesso, all'interno della pagina *MODEL* è possibile sincronizzare una nuova tabella Hive, inserendo il nome della tabella. Si può procedere alla costruzione del modello che fungerà da base del cubo, cioè le dimensioni e le misure. Va specificato che in questo caso la tabella utilizzata contiene sia le misure che le dimensioni, ma nella teoria del Dimensional Fact Model, andrebbe utilizzata una tabella dei fatti per le misure e tabelle diverse per le varie dimensioni, operando delle operazioni di join.

L'utente può decidere nel MODEL quali sono le vaiariabili per cui effettuare le misure. Ho scelto gli attributi **Agenzia**, **Ramo**, **Frazionamento**, **Tipo_titolo**, **IdContraente**, **Data_inserimento**, **Effetto_quota** e **Data_Scadenza_Contratto** come dimensioni. Come misure ho selezionato somma, massimo e minimo applicati ai campi **Importo**, **Importo_Provvigioni**, **Provvigioni_incasso**, **Provvigioni_acquisto**, **Sovrapprovvigioni_acquisto** e **Sovrapprovvigioni_incasso**.

Dopo aver completato il modello si può procedere alla creazione del cubo, selezionando MapReduce o Spark come **Cube Engine**. Per i benefici che Spark ha nella velocità di processamento, ho scelto di utilizzare Spark e ho lanciato l'esecuzione del cubo. A distanza di 9.48 minuti, il cubo è stato completato.

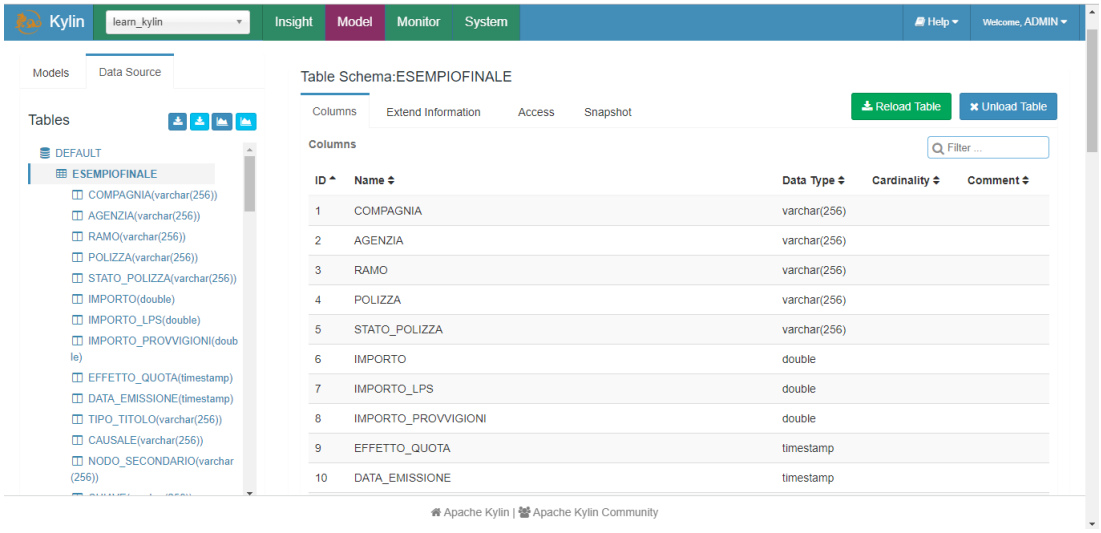

FIGURA 3.17: Sincronizzazione tabella su Hive

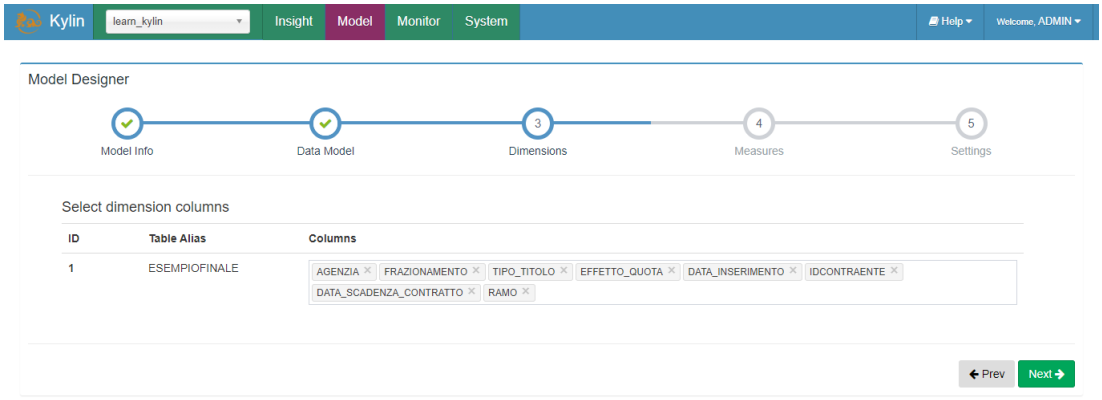

FIGURA 3.18: Dimensioni del modello della tabella Hive su Kylin

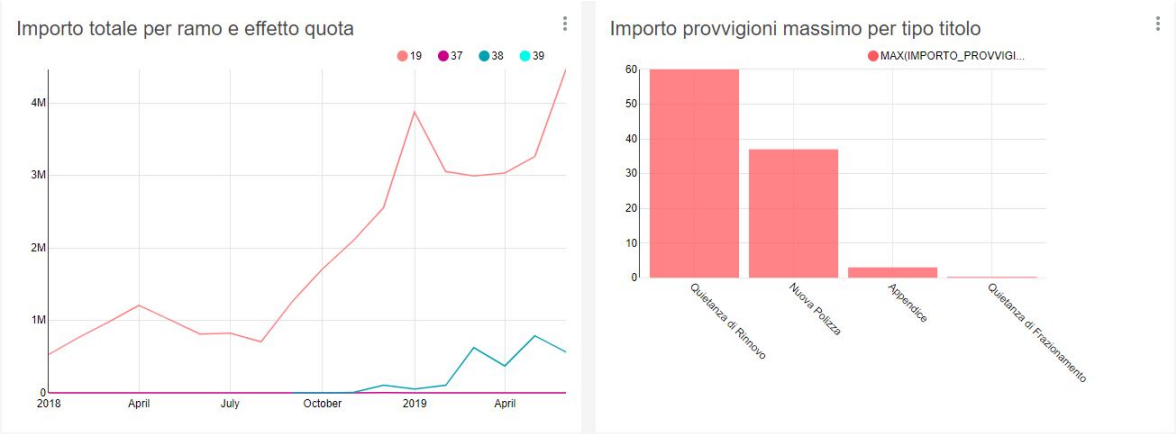

FIGURA 3.19: Dashboard finale batch (parte 1)

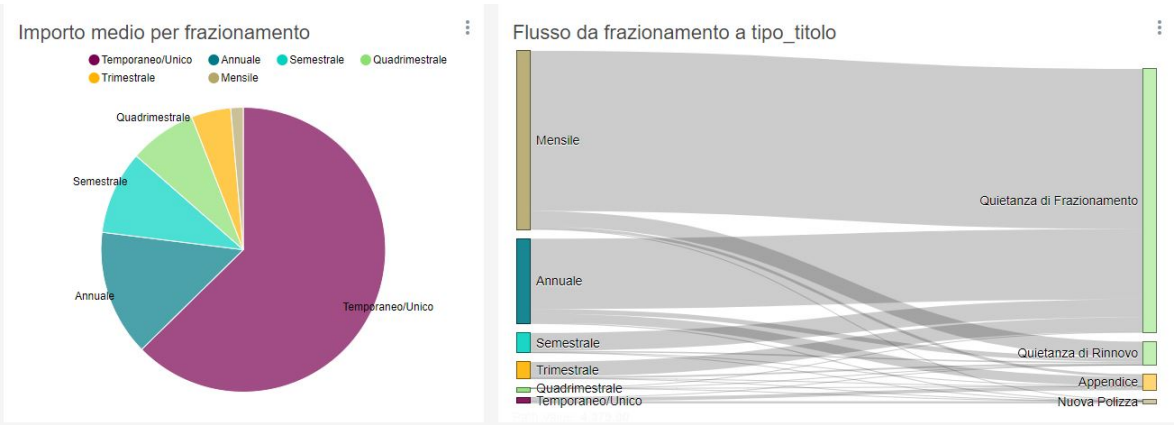

FIGURA 3.20: Dashboard finale batch (parte 2)

3.8.1 Dashboard finale e confronto Kylin vs SQLite

Collegandomi al cubo, ho definito la tabella "DEFAULT.ESEMPIOFINALE" all'interno di Superset e ho scelto di realizzare quattro grafici per comporre la dashboard:

- Un **line chart** che mostra il valore Totale dell'importo delle transazioni suddivisi per Ramo e Data_emissione a partire dal 2018.
- Un **bar chart** in cui viene visualizzato il valore massimo di Importo_Provvigioni suddiviso per Tipo_Titolo.
- Un **Sankey diagram** che mostra il flusso dal Frazionamento al Tipo_Titolo.
- Un **Pie Chart** che visualizza l'Importo medio per Frazionamento.

La nota poitiva dei test eseguiti è legata ai tempi di risposta delle query, che risultano sempre inferiori a 1 secondo: l'obiettivo è stato raggiunto! L'obiettivo successivo è mostrare che, l'architettura progettata permetta di ottenere visualizzazioni in modo più rapido rispetto al caricamento del Csv direttamente sul database SQLite integrato in Superset. Chiaramente il dataset di esempio essendo di dimensioni ridotte, potrebbe mostrare solo lievemente

Test	record	Dimensione
1	91.941	86.919 KB
2	183.882	174.329 KB
3	495.705	435.820 KB
4	919.410	871.639 KB

TABELLA 3.2: Test Kylin vs SQLite: numero record e dimensioni dei Csv

Test	Tempo [s] Kylin	Tempo[s] SQLite
1	0.27	0.82
2	0.31	1.26
3	0.26	2.57
4	0.34	4.84

TABELLA 3.3: Secondi medi di attesa Kylin vs SQLite

gli effetti positivi sui tempi di risposta delle query effettuate. Per questo, ho deciso di effettuare 4 test, duplicando la tabella originale per 2, per 4 e per 10, attraverso il comando Python:

```
df = pd.concat([df]*n, ignore_index=True)
```

in cui n è il numero di volte che la tabella viene duplicata. Il numero di record e la dimensione dei Csv utilizzati nel test sono descritti nella Tabella 3.1.

I test sono stati eseguiti contemporaneamente utilizzando due computer portatili identici e ciascuno dei test è stato replicato 3 volte su ciascuno dei 4 grafici realizzati precedentemente. In totale, dunque i risultati esposti nella Tabella 3.3, sui secondi medi di risposta, fanno riferimento alla media ottenuta da 12 osservazioni. Dalla tabella emerge come la necessità di ottenere risposte quasi immediate per l'esplorazione dei dati sia stata ottenuta con l'architettura sviluppata esclusivamente con prodotti open-source, mentre volendo utilizzare un database tradizionale, i tempi di risposta passano da meno di 1 secondo con una tabella di circa 86 MB, fino a quasi 5 secondi con dimensioni vicine a 1GB. Per questioni legate al ridotto spazio di archiviazione che avevo a disposizione sul server, non ho potuto sperimentare test con tabelle di dimensioni molto più elevate, ma appare chiara la tendenza del tempo di risposta ad aumentare al crescere della dimensione della tabella. Ciò porta a ritenere l'utilizzo di Kylin una soluzione assolutamente idonea per coprire casi d'uso di processamento batch di dataset molto estesi, posizionati sul file system Hadoop, anche se chiaramente, le operazioni necessarie alla generazione del cubo, richiedono una serie di risorse da utilizzare e un tempo di attesa prima di poter interrogare il cubo.

La Tabella 3.4 rappresenta il tempo in minuti necessario per la generazione dei cubi , realizzati sempre ex-novo, in quanto prima di creare ogni cubo

Cubo	Tempo [min]
1	9.48
2	10.12
3	10.43
4	11.37

TABELLA 3.4: Minuti di attesa per la realizzazione dei cubi nel caso offline

è stato cancellato quello vecchio.

Il risultato fa notare da un lato l'impossibilità di visualizzare le dashboard in tempo completamente reale rispetto al momento in cui i dati sono generati, perché deve intercorrere un periodo di alcuni minuti per rendere il cubo pronto, dall'altro evidenzia come il tempo di attesa per record scenda al crescere della dimensione delle tabelle.

3.9 Caso d'uso in Streaming

Per questo caso d'uso ho supposto di dover processare dati sparsi su tutto il territorio italiano, provenienti da pagamenti digitali gestiti da una grande banca nazionale su acquisti effettuati all'interno di supermercati. Per questo ho realizzato in Java un'applicazione che attraverso la API Producer di Kafka invia dei messaggi in streaming, all'interno di multi-thread[3] della durata di 1000 millisecondi, (dunque 1 secondo) in cui vengono inviati su un topic chiamato "kafkaTesi" 100 messaggi che contengono i seguenti campi:

- Tipo prodotto (Salumeria, Bevanda analcolica, Bevanda alcolica, Panetteria, Prodotto, Surgelato, Frutta o verdura, Prodotto fresco).
- Fascia età (0-15, 15-25, 25-40, 40-60, 60+).
- Percentuale sconto.
- Quantità.
- Provincia, scelta tra le Province il cui capoluogo è anche capoluogo di Regione.
- Genere del cliente (uomo o donna).
- Timestamp (lo stesso del momento in cui il Producer invia il messaggio sul topic Kafka).

La fascia d'età è estratta con le seguenti probabiltà: 0-15 : 5% , 15-25: 25% , 25-40: 25% , 40-60: 30% , 60+ : 20% .

[3]Un **thread** è la suddivisione di un processo in sottoprocessi che vengono eseguiti in modo concorrenziale.

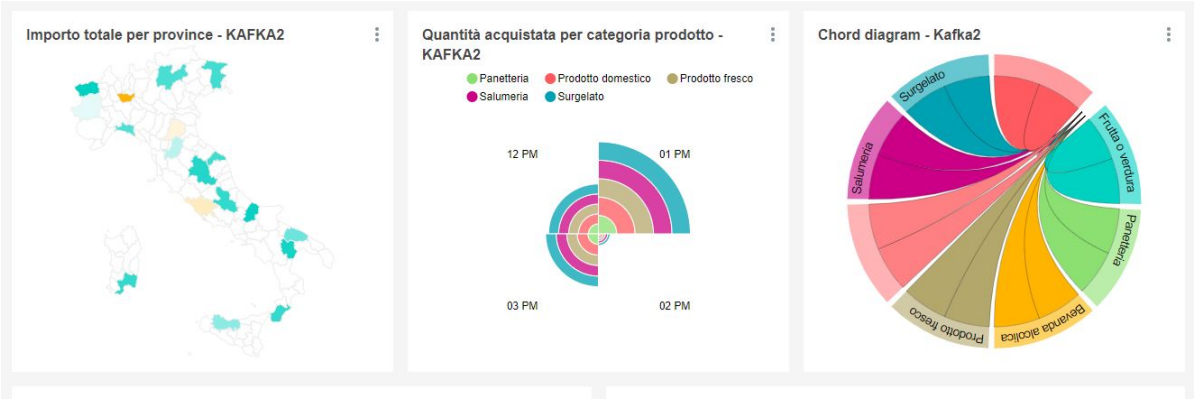

FIGURA 3.21: Dashboard finale in streaming (parte 1)

La Provincia viene invece scelta con una probabilità pari (previa arrotondamento) alla frazione di PIL che la Regione di riferimento possiede rispetto al PIL nazionale. L'importo del singolo prodotto viene definito da un valore Random compreso tra 1 e 20, moltiplicato per un coefficiente che deriva dall'ora in cui il messaggio viene inviato, che è di 3 dalle 7 alle 10, di 10 dalle 10 alle 17. Lo sconto è un valore Random compreso tra 0 e 0.5. La quantità è un valore completamente randomico compreso tra 0 e 10 e il sesso è scelto in modo totalmente randomico tra uomo e donna.

L'applicazione, il cui codice Java è descritto nell'Appendice C, viene eseguita da una classe Main, chiamata **ProducerBoostrap** che riceve da linea di comando il parametro che indica il numero di messaggi totali che il Producer deve inviare prima di fermarsi.

Ho deciso di inviare 4 flussi di streaming, il primo con 800000 record, il secondo con 400000, il terzo con 200000 e l'ultimo con 100000. Dopo ogni flusso, ho creato un cubo su Kylin incrementale, senza dunque cancellare il vecchio cubo, utilizzando come dimensioni Provincia, TipoProdotto, Fascia età, TimeStamp, Genere e come misure soltanto la somma dell'Importo (già scontato) e della Quantità.

Dopo aver creato i 4 cubi, ho realizzato i seguenti grafici:

1. Una mappa dell'Italia che rappresenta per ogni Provincia l'importo totale.

2. Un grafico circolare che mostra la quantità acquistata per TipoProdotto e fascia oraria.

3. Un chord diagram che mostra la quantità acquistata per TipoProdotto.

4. Una tabella partizionata che mostra l'importo totale per TipoProdotto.

5. Un sankey diagram che mostra i flussi di acquisti sulla base del genere utente, verso il TipoProdotto .

6. Un filter box che permette all'utente di selezionare uno o più valori di TipoProdotto che sono utilizzati per la costruzione degli altri grafici.

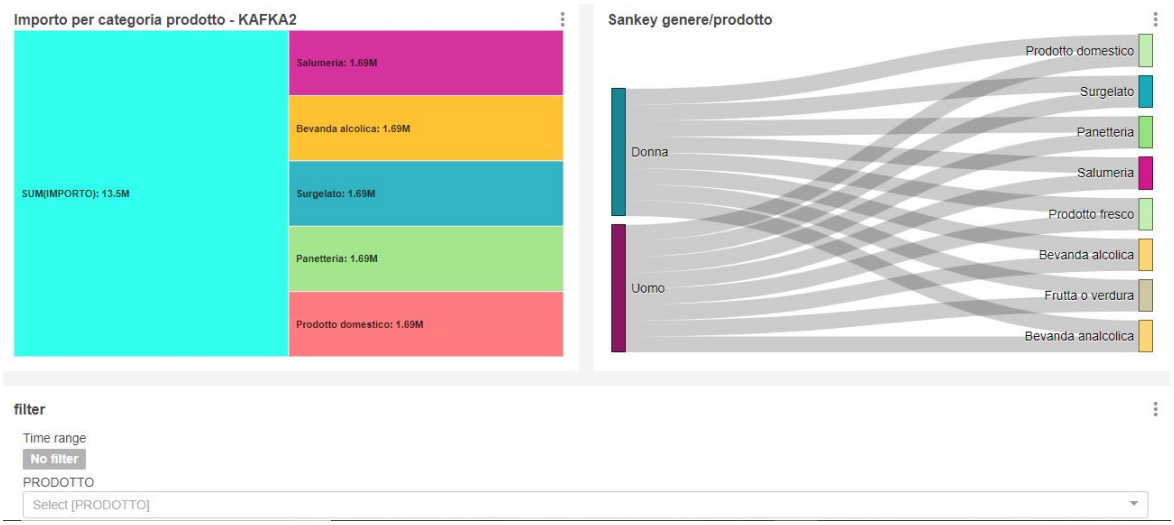

FIGURA 3.22: Dashboard finale in streaming (parte 2)

Cubo	Tempo [min]
1	6.32
2	5.29
3	4.55
4	4.37

TABELLA 3.5: Minuti di attesa per la realizzazione dei cubi nel caso streaming

Il tempo di attesa per la visualizzazione dei grafici, cioè per l'esecuzione della query da parte di Superset al cubo e il caricamento dell'immagine, è rimasto molto contenuto, compreso tra 0.25 e 0.4 secondi. Inoltre, il tempo necessario alla costruzione dei cubi si è ridotto notevolmente rispetto al caso in batch, principalmente per due motivi, ossia da un lato il numero inferiore di calcoli che sono stati eseguiti, in quanto nel cubo sono stati selezionati solamente 3 misure, cioè Somma per Importo, Somma per Quantità e Conteggio, dall'altro la maggiore leggerezza del formato Json delle code di streaming. Nella Tabella 3.5 sono illustrati i tempi di attesa necessari alla creazione dei cubi. Da essa emerge come anche in questo caso il tempo di attesa per record per la generazione del cubo, decresce all'aumentare della dimensione dei dati.

3.10 Conclusioni

Le attività portate avanti in collaborazione con l'azienda **Blue reply** mi hanno permesso di effettuare una navigazione completa all'interno delle dinamiche riguardanti i Big Data, la Business Intelligence e la commistione delle due tematiche finalizzata all'estrazione di valore mediante visualizzazione di dashboard, per le soluzioni open-source e agumented analytics, per le soluzioni

"proprietarie".

La scelta di effettuare un focus sulle soluzioni con maggiore presenza sul mercato è derivata dal requisito aziendale di proporre soluzioni con forte appeal sul cliente. Tra queste, PowerBI e Tableau rappresentano le soluzioni da me preferite per funzionalità, facilità d'uso e possibilità di acquisire una sola licenza per singolo utente, senza dover acquistare pacchetti di licenze (come per Oracle e Sas).

Gli sforzi maggiori del caso di Business sono però stati effettuati nel ricercare di incatenare diversi strumenti open-source che permettessero di disporre di un'architettura completa e funzionante per la visualizzazione di dashboard, nelle quali il caricamento delle immagini avvenisse con brevi latenze temporali. Dopo aver testato e analizzato diverse soluzioni, sono riuscito a realizzare l'obiettivo iniziale attraverso principalmente l'integrazione di Kylin e Superset, con altri strumenti Big Data (Hdfs, Hadoop, Spark, MapReduce, Kafka, HBase).

Al termine del progetto di tesi l'azienda potrà utilizzare le analisi per allargare su diversi casi d'uso la vendita di servizi di Business Intelligence e data visualization in parallelo all'adozione delle nuove tecniche di processamento distribuito e scalabile che costituiscono, di fatto, la traduzione tecnologica della necessità di gestire il fenomeno sociale dell'avvento dirompente dei Big Data.

Appendice A

Script Python trasformazione dati

A.1 script.py

```
import pandas as pd
from datetime import datetime
df = pd.read_csv(r"csvEsempio.csv", delimiter = ";")
def date_convert(date_to_convert):
    try:
        result = pd.to_datetime(date_to_convert,
        format='%d/%m/%Y')
        .strftime('%Y-%m-%d %H:%M:%S')
        return result
    except ValueError:
        return pd.Timestamp.max
def parseNumber(value):
    try:
        number = float(str(value).replace('.','')
        .replace(',', '.'))
        return number
    except ValueError:
        return float(0.0)
df.DATA_INSERIMENTO=df.DATA_INSERIMENTO
.apply(date_convert)
df.DATA_SCADENZA_CONTRATTO=df.DATA_SCADENZA_CONTRATTO
.apply(date_convert)
df.EFFETTO_QUOTA=df.EFFETTO_QUOTA
.apply(date_convert)
df.IMPORTO=df.IMPORTO
.apply(parseNumber)
df.PROVVIGIONE_INCASSO=df.PROVVIGIONE_INCASSO
.apply(parseNumber)
df.PROVVIGIONE_ACQUISTO=df.PROVVIGIONE_ACQUISTO
.apply(parseNumber)
df.SOVRAPPROVVIGIONI_ACQUISTO=
df.SOVRAPPROVVIGIONI_ACQUISTO
.apply(parseNumber)
df.SOVRAPPROVVIGIONI_INCASSO=
df.SOVRAPPROVVIGIONI_INCASSO
.apply(parseNumber)
```

```
df.IMPORTO_PROVVIGIONI=
df.IMPORTOPROVVIGIONI
.apply(parseNumber)
export_csv = df.to_csv (r'
esempioFinale.csv', index = None,
header=True,sep=";")
```

Appendice B

Comandi Hive

Command Line

```
[root@acer giovanni]# docker cp esempioFinale.csv
condescending_bassi:/
[root@acer giovanni]# docker exec -it
condescending_bassi bash
[root@427566eb5425 admin]# hive
Logging initialized using configuration in
jar:file:/home/admin/apache-hive-1.2.1-bin/
lib/hive-common-1.2.1.jar!/hive-log4j.properties
```

Command Line

```
hive> CREATE TABLE esempioFinale
(COMPAGNIA STRING,AGENZIA STRING,RAMO STRING,
POLIZZA STRING,STATO_POLIZZA STRING,IMPORTO FLOAT,
IMPORTO_LPS STRING, IMPORTO_PROVVIGIONI FLOAT,
EFFETTO_QUOTA TIMESTAMP,DATA_EMISSIONE STRING,
TIPO_TITOLO STRING,CAUSALE STRING,
NODO_SECONDARIO STRING,CHIAVE STRING,RID STRING,
PRESENZA_VINCOLO STRING,DATA_SCADENZA_VINCOLO STRING,
ENTE_VINCOLATORIO STRING,FRAZIONAMENTO STRING,
CODICE_DESCRIZIONE_PRODOTTO STRING,ITER_CONTENZIOSO STRING,
TIPO_COASSI STRING,QUOTA INT,NOME_COMPAGNIA_DELEGATARIA STRING,
DATA_INSERIMENTO TIMESTAMP,CODICE_SIRE STRING,
CODICE_OPERATIVO STRING, CODICE_COMPAGNIA_DELEGATARIA STRING,
NUMERO_POLIZZA_COMPAGNIA_DELEGATARIA STRING,
CODICE_ANNULLAMENTO_POLIZZA STRING,
DESCRIZIONE_ANNULLAMENTO_POLIZZA STRING,
EFFETTO_ANNULLAMENTO_POLIZZA STRING,
CODICE_INTERMEDIARIO_1 STRING,
CODICE_INTERMEDIARIO_2 STRING,
CODICE_INTERMEDIARIO_3 STRING,IMPONIBILE STRING,
NETTO_DIRITTI STRING,PROVVIGIONE_ACQUISTO FLOAT,
SOVRAPPROVVIGIONI_ACQUISTO FLOAT,
PROVVIGIONE_INCASSO FLOAT,
SOVRAPPROVVIGIONI_INCASSO FLOAT,
COPERTURA_AMMINISTRATIVA STRING,
DATA_SCADENZA_CONTRATTO TIMESTAMP,
PROVVIGIONE_RICORRENTE_PRECONTATO STRING,
PULIZIA_DI_BILANCIO STRING,
DA_DATA_SPEDIZIONE_PRIMA_MAIL_RICHIESTA_INFORMAZIONI STRING,
DA_DATA_SPEDIZIONE_MAIL_SOLLECITO STRING,
DA_ESITO_FLAG_ITER_DA STRING,DA_DATA_ESITO STRING,
DA_NOTE STRING,CHIAVE_ESTERNA STRING,
IDCONTRAENTE STRING,RECORD_RDNANG_ARRICCHITO STRING,
LIBRO_MATRICOLA STRING,
DATA_ESIGIBILITA STRING,DATA_EMESSO STRING,
CODICE_AGGREGAZIONE STRING,NUMERO_POLIZZA_UNQ STRING,
NUMERO_POLIZZA_MADRE STRING,
TIPOLOGIA_INCASSO STRING,PROPOSTA STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\;'
STORED AS TEXTFILE
LOCATION '/esempioFinale'
TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.539 seconds
hive> quit;
[root@427566eb5425 admin]# hadoop fs -put
esempioFinale.csv /esempioFinale/
```

Appendice C

Applicazione Java Producer Kafka

C.1 KafkaProducer.class

```
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;import java.util.Hashtable;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import com.fasterxml.jackson.databind.ObjectMapper;
public class Producer implements Runnable {
private static final LOGGER=Logger.getLogger(Producer.class.getName());
private static final String topic = "kafkaTesi";
private KafkaProducer<String, String> producer = null;
private final String KAFKA_CLUSTER_ENV_VAR_NAME="KAFKA_CLUSTER";
private int numTransazione;
private ProvinciaRandom province = new ProvinciaRandom();
private EtaRandom fasciaeta = new EtaRandom();
private static final ObjectMapper mapper = new ObjectMapper();
private List<String> genders = new ArrayList<>();
private List<String> prodotti = new ArrayList<>();
private int oracoeff;
private int numMessaggiMax;
public Producer(int numMessaggi) {
Properties props = new Properties();
String defaultClusterValue = "localhost:9092";
String kafkaCluster = System.getProperty(KAFKA_CLUSTER_ENV_VAR_NAME,
    defaultClusterValue);
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, kafkaCluster);
props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
    "org.apache.kafka.common.serialization.StringSerializer");
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
    "org.apache.kafka.common.serialization.StringSerializer");
```

```
props.put(ProducerConfig.ACKS_CONFIG, "0");
props.put("retries", 0);
props.put("batch.size", 16384);
props.put("linger.ms", 1);
props.put("buffer.memory", 33554432);
this.producer = new KafkaProducer<>(props);
this.numMessaggiMax=numMessaggi;
province.addEntry("IT-AQ", 2.0,"LAquila",45.4,07.2);
province.addEntry("IT-PZ", 1.0,"Potenza",40.4,16.4);
province.addEntry("IT-NA",9.0,"Napoli",40.2,14.1);
province.addEntry("IT-RC",2.0,"Reggio",38.1,15.4);
province.addEntry("IT-BO", 10.0,"Bologna",44.30,11.21);
province.addEntry("IT-UD", 2.5,"Udine",46.04,13.14);
province.addEntry("IT-RM",11.0,"Roma",41.6,12.3);
province.addEntry("IT-GE",3.0,"Genova",44.2,08.6);
province.addEntry("IT-MI",17.5,"Milano",45.3,09.1);
province.addEntry("IT-AN", 2.5,"Ancona",43.4,13.3);
province.addEntry("IT-CB",0.5,"Campobasso",41.3,14.4);
province.addEntry("IT-TO",8.0,"Torino",45,07.4);
province.addEntry("IT-BA",4.0,"Bari",41.1,16.5);
province.addEntry("IT-CA",2.0,"Cagliari",39.1,09.1);
province.addEntry("IT-PA",5.0,"Palermo",38,13.2);
province.addEntry("IT-FI", 6.5, "Firenze",43.46,11.15);
province.addEntry("IT-TN",2.75,"Trento",46,11.1);
province.addEntry("IT-PG",1.5,"Perugia",43.1,12.2);
province.addEntry("IT-AO",0.25,"Aosta",45.4,07.2);
province.addEntry("IT-VE",9.0,"Venezia",45.3,12.2);
@SuppressWarnings ("deprecation")
int ora = new Date( ).getHours( );
if ( ora >= 7 && ora <= 10 )
oracoeff = 3 ;
elseif( ora > 10 && ora <= 17)
oracoeff = 1 0 ;
numTransazione = 0 ;
genders.add("Uomo") ;
genders.add("Donna") ;
prodotti.add("Salumeria") ;
prodotti.add ("Bevanda analcolica " ) ;
prodotti.add( "Bevanda alcolica" );
prodotti.add("Panetteria") ;
prodotti.add( " Prodotto domestico " ) ;
prodotti.add(" Surgelato " ) ;
prodotti.add( " Frutta o verdura " ) ;
prodotti.add( " Prodotto fresco" ) ;
fasciaeta.addEntry( " 0 -15 " , 5.0 ) ;
fasciaeta.addEntry( " 15 -25 " , 2 0.0 ) ;
fasciaeta.addEntry( "25-40" , 25.0 ) ;
fasciaeta.addEntry( "40-60" , 30.0 ) ;
fasciaeta.addEntry(" 60+" ,2 0 . 0 ) ;
}
```

```
@Override
public void run ( ) {
try {
produce( ) ;
} catch ( Exception e ) {
LOGGER. log ( Level . SEVERE, e . getMessage ( ) , e ) ;
}
}
private void produce ( ) throws Exception {
Map<String , Object > record = new HashMap< >( ) ;
try {
Random rnd = new Random( ) ;
boolean alive = true ;
while ( alive ) {
for ( int i = 1 ; i <= 100 ; i ++) {
double sconto = rnd.nextDouble( )0 . 5 ;
int quantita = rnd.nextInt( 1 0 ) ;
record.put( " timestamp " , (new Date( ).getTime( ) ) ) ;
record.put( " prodotto " , prodotti.get(
rnd.nextInt(prodotti.size( )))) ;
record.put("quantita" , rnd.nextInt(10)) ;
record.put( " sconto " ,Math.round((rnd.nextDouble( ) 0.5 ) 100.0 ) /
    100.0 ) ;
record.put( " importo " ,Math.round((rnd.nextDouble( ) oracoeff
(1-sconto )quantita ) 20.0 )/20.0 ) ;
Provincia p = province.getRandom ( ) ;
record.put( "provincia " , p.ge tProvincia( ));
record.put( " ISO2 " ,p.getISO2 ( ) ) ;
record.put(" genereUtente " , genders.get(rnd.nextInt(2 ))) ;
record.put("etaCliente " , fasciaeta.getRandom ( ) ) ;
numTransazione++;
ProducerRecord<String , String > data =
new ProducerRecord <>( topic , System . currentTimeMillis( )
+ " " , mapper.writeValueAsString( record ) ) ;
producer.send( data ) ;
LOGGER.info( "Messaggio "+numTransazione+" inviato " ) ;
if( numTransazione >= numMessaggiMax ) {
alive = false ; } }
Thread.sleep(1000) ;
}
}
catch ( Exception e ) {
LOGGER. log ( Level . SEVERE, " Thread interrotto " ) ;
} finally {
producer.close( ) ;
LOGGER.log ( Level . INFO, " Producer chiuso " ) ;
}
}
}
```

C.2 ProducerBootstrap.class

```
import java.util.logging.Logger;

public class ProducerBootstrap {

    private static final Logger LOGGER =
        Logger.getLogger(ProducerBootstrap.class.getName());

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

        new Thread(new Producer(Integer.parseInt(args[0]))).start();
        LOGGER.info("Kafka producer triggered");

    }
}
```

C.3 Provincia.class

```
public class Provincia {
   private String ISO2;
   private String provincia;
   private double accumulatedWeight;
   private double latitude;
   private double longitude;
   public Provincia(String iSO2, double accumulatedWeight, String
       provincia, double latitude, double longitude) {

      this.ISO2 = iSO2;
      this.provincia = provincia;
      this.accumulatedWeight = accumulatedWeight;
      this.latitude=latitude;
      this.longitude=longitude;
   }
   public String getISO2() {
      return ISO2;
   }
   public String getProvincia() {
      return provincia;
   }
   public double getAccumulatedWeight() {
      return accumulatedWeight;
   }
   public double getLatitude() {
      return latitude;
   }
   public double getLongitude() {
```

```
      return longitude;
   }


}
```

C.4 ProvinciaRandom.class

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class ProvinciaRandom {


    private List<Provincia> province = new ArrayList<>();
    private double accumulatedWeight;
    private Random rand = new Random();


    public void addEntry(String object, double weight,String
        provincia, double latitude, double longitude) {
        accumulatedWeight += weight;
        Provincia p = new
            Provincia(object,accumulatedWeight,provincia,latitude,longitude);
        province.add(p);
    }

    public Provincia getRandom() {
        double r = rand.nextDouble() * accumulatedWeight;

        for (Provincia p : province) {
            if (p.getAccumulatedWeight() >= r) {
                return p;
            }
        }
        return null;
    }
}
```

C.5 EtaRandom.class

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
```

```
public class EtaRandom {
    private Random rand = new Random();

    private double accumulatedWeight;
    private List<String> fasce = new ArrayList<>();
       public void addEntry(String object, double weight) {
           accumulatedWeight += weight;
           fasce.add(object);
       }

       public String getRandom() {
           double r = rand.nextDouble() * accumulatedWeight;

           for (String p : fasce) {
               if (accumulatedWeight >= r) {
                   return p;
               }
           }
           return null;

      public int getSize() {

         return fasce.size();
      }

}
```

Appendice D

Bibliografia e sitografia

1 Laney, D. (2001), "3D Data Management:Controlling Data Volume,Velocity and Variety", META Group Research Note,Volume 6

2 Dijcks, J.(2012), "Big Data for the Enterprise, Oracle report"

3 NIST Special Publication 1500-1r1(2018), "Big Data Interoperability Framework: Volume 1, Definitions.", Versione 2

4 https://www.kdnuggets.com/2014/09/what-is-big-data-answers-thought-leaders.html, consultato il 29/10/2019

5 Moore, G. E. (1965),"Cramming more components onto integrated circuits", Electronics, Volume 38(8)

6 Reinsel, D. Gantz, J. Rydning, J. (2018), "The Digitization of the World From Edge to Core", IDC White Paper

7 Xiao, Z.,Song, W.,Chen, Q. (2013), "Dynamic Resource Allocation using Virtual Machines for Cloud Computing Environment", IEEE TRANSACTION ON PARALLEL AND DISTRIBUTED SYSTEMS, Volume 24(6)

8 Hashem, I. A. T. Yaqoob, A. Anuar, N. B. Mokhtar, S. Gani, A. Khan, S. U. (2015), "The rise of “big data” on cloud computing: Review and open research issues", Information Systems Volume 47, pp 102-103

9 GSMA (2018), "The Data Value Chain"

10 Shalev-Shwartz, S. Ben-David, S. (2014)"Understanding Machine Learning: From Theory to Algorithms", Cambridge University Press

11 Riahi, Y. (2018), "Big Data and Big Data Analytics: Concepts, Types and Technologies", Université Internationale de Rabat

12 "https://www.statista.com/statistics/237974/online-advertising-spending-worldwide/", consultato il 30 Ottobre 2019

13 "https://www.emarketer.com/content/global-digital-ad-spending-2019", consultato il 30 Ottobre 2019

14 "https://www.gartner.com/en/information-technology/glossary/business-intelligence-bi-services", consultato il 10 Novembre 2019

15 Rainardi, V. (2008), "Building a Data Warehouse"

16 Ariyachandra, T. Watson, H. J. "Which Data Warehouse Architecture Is Most Successful?",BUSINESS INTELLIGENCE JOURNAL, Volume 11(1)

17 "http://www.crisp-dm.org/", consultato il 10/12/2019

18 Agrawal, R. Imieliński, T. Swami, A. (1993). "Mining association rules between sets of items in large databases". Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93. p. 207"

19 Han, Kamber (2006),"Data mining; Concepts and Techniques"

20 "https://www.solver.com/xlminer/help/neural-networks-classification-intro", consultato il 10/12/2019

21 J. MacQueen, B. (1967), "Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1:281-297

22 "https://www.edupristine.com/blog/k-means-algorithm", consultato il 10/12/2019

23 Ester, M. Kriegel, H. Sander, J. Xu, X. (1996), "A density-based algorithm for discovering clusters in large spatial databases with noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)", AAAI Press. pp. 226–231

24 "https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html", consultato il 13/12/2019

25 "https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html, consultato il 17/12/2019

26 "https://spark.apache.org/docs/latest/sql-data-sources.html", consultato il 21/12/2019

27 "https://spark.apache.org/docs/latest/streaming-programming-guide.html", consultato il 21/12/2019

28 "https://spark.apache.org/docs/latest/graphx-programming-guide.html", consultato il 22/12/2019

29 "https://cwiki.apache.org/confluence/display/HADOOP2/SequenceFile", consultato il 21/12/2019

30 "https://kafka.apache.org/intro", consultato il 27/12/2019

31 "https://seekingalpha.com/article/277145-oracles-database-business-munches-ibms-market-share", consultato il 29/12/2019

32 "https://db-engines.com/en/ranking", consultato il 29/12/2019

33 "http://www.cs.unibo.it/ difelice/dbsi/2017/slides/pdf/20.pdf", consultato il 29/12/2019

34 "https://en.wikipedia.org/wiki/Apache_Cassandra", consultato il 29/12/2019

35 Durant, K. ,"Introduction to NoSQL and MongoDB", Lesson 20 CS 3200, Northeastern University

36 "https://techcrunch.com/2016/10/26/elastic-finally-brings-order-to-its-product-line-with-elastic-stack/", consultato il 29/12/2019

37 "https://en.wikipedia.org/wiki/Redis", consultato il 30/12/2019

38 "https://redis.io/topics/data-types-intro", consultato il 30/12/2019

39 "https://www.gartner.com/doc/reprints?id=1-65P04FGct=190125st=sb", consultato il 15/11/2019

40 "https://superset.incubator.apache.org/installation.htmlpython-virtualenv", consultato il 20/11/2019

41 "https://www.sqlalchemy.org/", consultato il 20/11/2019

42 "https://tech.ebayinc.com/engineering/announcing-kylin-extreme-olap-engine-for-big-data/", consultato il 21/11/2019