UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Sviluppo di componenti software per la visualizzazione e l'analisi multidimensionale di dati

RELATORE

Prof. Deufemia Vincenzo

Dott.ssa Pipino Claudia

CANDIDATO

Daniele Russo

Matricola: 0512109800

Anno Accademico 2022-2023

Several people have told me that my inability to suffer fools gladly is one of my main weaknesses

- Dijkstra

Abstract

Oggi giorno veniamo investiti da una grande quantità di dati proveniente da più fonti (Big Data e IoT) e in diversi formati. È possibile elaborare e processare tali dati provando a trarre del valore aggiunto da essi? Ciò è possibile con l'utilizzo di determinate tecnologie che si occupano della pulizia, normalizzazione e gestione dei dati. Tali tecnologie possono essere applicate anche al contesto sanitario, per favorire la ricerca e migliorare i processi decisionali. Quest'ultimi potrebbero ricavare delle preziose informazioni dall'analisi dei dati sanitari, attraverso i quali comprendere meglio l'ambiente in cui si verificano tali processi. Per dimostrare ciò, in questo lavoro di tesi, sono stati sviluppati diversi casi d'uso, grazie alla realizzazione di un middleware che semplifica tale processo. Con questo middleware è stato possibile effettuare un'analisi multidimensionale su dati sanitari e verificare alcune dinamiche note in letteratura medica. L'importanza di questo progetto è nella replicazione della stessa infrastruttura, consentendo a chiunque di utilizzarla per effettuare analisi multidimensionale di dati.

Keyword: Data Warehouse e OLAP; Big Data, Apache Superset; ETL; Data Integration; Data Ingestion; Data Mining; Docker; Containers;

Ringraziamenti

Mi è doveroso dedicare questo spazio della mia tesi a tutte le persone che mi hanno supportato nel mio percorso di crescita universitaria. Partendo da I.T.Svil che mi ha permesso di sviluppare tale lavoro, affidandomi come tutor la Dott.ssa Pipino Claudia che è stata fondamentale in tutto il mio percorso di tirocinio e di tesi. Un sentito ringraziamento va al mio relatore Deufemia Vincenzo che mi ha seguito, con disponibilità e gentilezza, in ogni step della realizzazione dell'elaborato, fin dalla scelta dell'argomento. Non posso non ringraziare le persone che hanno avuto più influenza nel mio percorso educativo: la mia famiglia, i miei nonni e gli zii; con particolare attenzione a mio padre Liberato, il quale è sempre stato al mio fianco; a mio fratello Francesco che mi ha supportato seppure la distanza e a mia nonna Mariarosaria, che mi ha seguito e cresciuto da quando ero un ragazzo. Un grazie ai miei amici di sempre, Giovanni, Carlo e Mattia che hanno alleggerito i momenti più pesanti e mi hanno spronato a dare sempre di più accompagnandomi in questo percorso. Infine, non per importanza, un ringraziamento alla mia mamma che anche se non più qui, è stata una guida, senza la quale io oggi non sarei ciò che sono, e a Valeria con la quale ho ritrovato una parte di me persa da tempo.

Indice

Elenco delle Figure **iii**

Elenco degli Acronimi **v**

Introduzione **1**

1 Analisi dei Big Data e Cloud Computing **3**

1.1 Big Data e Data Mining 3

1.1.1 Data Ingestion 7

1.1.2 Data Processing 9

1.1.3 Data Warehouse 9

1.2 Cloud Computing 11

1.2.1 Definizione di Cloud e Cloud Computing 11

1.2.2 Cloud Computing e Big Data 12

1.3 OLAP 12

1.4 Sistemi per l'analisi dei dati 14

1.4.1 Architettura Big Data 14

1.4.1.1 Componenti di un'architettura Big Data 14

1.4.1.2 Tipi di architettura 16

1.4.2 Fasi dell'analisi dei dati 18

1.4.2.1 Acquisizione dei Dati 18

1.4.2.2 Elaborazione dei Dati 20
1.4.2.3 Archiviazione dei Dati 21

2 Caso di Studio e Architettura Proposta **23**
2.1 Dominio Applicativo 23
2.1.1 Data Ingestion 24
2.1.2 Data Processing 31
2.2 Architettura Proposta 32
2.2.1 ETL 33
2.2.2 OLAP 35
2.2.2.1 Apache kylin 35
2.2.2.2 Cube.js 37
2.2.3 Superset 39

3 Casi d'uso e Risultati **41**
3.1 Risultati e Discussioni 43
3.1.1 Primo caso d'uso 43
3.1.2 Secondo caso d'uso 43
3.1.3 Terzo caso d'uso 44

Conclusioni **47**

Bibliografia **49**

Elenco delle figure

1 Layer soluzione 2

1.1 Caratteristiche dei Big Data 4
1.2 Fasi dell'estrapolazione di conoscenza tramite Data Mining 6
1.3 Data Ingestion steps 8
1.4 Data Processing 9
1.5 Data Werehouse workflow 10
1.6 Operazioni sui Cubi OLAP 13
1.7 Architettura per i Big Data 14
1.8 Architettura Lambda 16
1.9 Architettura Kappa 18
1.10 Batch Processing vs Real Time Processing 20
1.11 Differenze schema a snowflake e stella 22

2.1 Architettura della soluzione 32
2.2 Schema ERD 34
2.3 Struttura softwere 35
2.4 Struttura cubi 38
2.5 Dashboard Superset 40

3.1 Andamento dei ricoveri tra 2017 e 2019 43

3.2 Andamento dei ricoveri suddivisi per provenienza tra 2017 e 2019 . . 43
3.3 Incidenza "coronaropatia" 44
3.4 Incidenza "insufficienze renali" 44
3.5 Analisi enzimi cardiaci nei pazienti affetti da coronaropatia 45
3.6 Comparazione analisi tra pazienti con enzimi cardiaci nella norma e non 45
3.7 Comparazione analisi tra pazienti con insufficienza renale e non . . . 46

Elenco degli Acronimi

Acronimo	Significato
DB	Data Base
DBMS	Database management system
RDBMS	Relational Database management system
CVS	Comma-separated values
OLAP	On-Line Analytical Processing
ETL	Extract, transform, load
IT	Information technology
IoT	Internet of Things
DWH	Data Warehouse
SQL	Structured Query Language
NoSQL	No Structured Query Language
ERD	Entity-Relationship Diagrams
JPA	Java Persistence API
SaaS	Software as a service
DFS	Distributed File System

Introduzione

Nel contesto contemporaneo, la società si trova immersa in un flusso incessante di dati provenienti da molteplici fonti, in diversi formati e connotati da una complessità crescente, ciò dunque ci porta nello scenario dei Big Data. Questo scenario ci pone un interrogativo centrale: è possibile sfruttare appieno questa vasta mole di informazioni per generare valore aggiunto e promuovere il progresso socio-economico? La risposta affermativa a questa domanda trova fondamento nell'adozione e nell'implementazione di tecnologie all'avanguardia dedicate al trattamento, alla normalizzazione e alla gestione dei dati. Tali tecnologie, oltre a rivoluzionare settori chiave dell'industria e del commercio, come è noto, possono trovare una concreta applicazione anche nel comparto sanitario. La loro integrazione potrebbe rappresentare un balzo in avanti nel campo della ricerca medica e dell'assistenza sanitaria, consentendo non solo di accrescere la qualità dell'assistenza fornita, ma anche di ottimizzare i processi decisionali e di governance. Questo ambiente, alimentato dai dati, può fungere da catalizzatore per l'identificazione di nuove opportunità di ricerca, per il miglioramento dei servizi sanitari e per supportare l'innovazione continua in questo settore. Pertanto, la corretta integrazione di tali tecnologie rappresenta non solo una sfida, ma anche un'opportunità imperdibile per plasmare il futuro della sanità su basi solide e orientate alla conoscenza. Questo è parte di ciò che ho appreso durante il mio tirocinio, svolto presso l'azienda I.T.Svil, dove mi è stata presentata l'opportunità di effettuare un lavoro di analisi di dati in ambito sanitario.

Contestualmente a ciò, mi è stata prospettata la sfida di tale lavoro, vista la grande mole di dati e informazioni con la quale bisognava lavorare. Questo, d'altra parte, ha portato allo sviluppo di un architettura facilmente replicabile e scalabile, per rendere più accessibile l'analisi dei dati in questo settore. Possiamo vedere in Figura 1, una vista semplificata di tale architettura, nella quale vediamo come il flusso dei dati viene ripartito tra i vari moduli. Possiamo vedere come partendo da un csv, riusciamo ad estrarre i dati con l'ETL, permettendoci di creare i cubi con Cube.js e di mostrare le analisi di tali dati attraverso Apache Superset. Ho deciso di intraprendere tale percorso per l'importanza di tale lavoro: analizzare i dati, specialmente in ambito sanitario, permetterebbe di risparmiare molte risorse economiche nella fase di prevenzione ma anche di allocare le rimanenti risorse in modo efficiente e ottimale. Per permettere una massima comprensione del mio lavoro, ho deciso di strutturare la seguente tesi in tre capitoli:

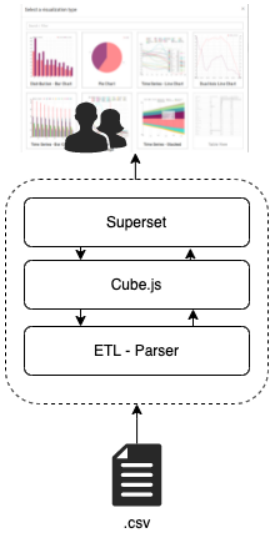

Figura 1: Layer soluzione

- **Analisi dei Big Data e Cloud Computing:** questo capitolo è uno sguardo ai concetti che stanno alla base delle tecnologie che ho utilizzato, facendo particolare riferimento ai processi standard reperiti in letteratura, sui quali ho basato il mio lavoro.

- **Caso di Studio e Architettura Proposta:** in questo capitolo viene analizzato il dominio applicativo e come l'architettura è stata definita e implementata.

- **Casi d'uso e Risultati:** in questo capitolo vengono analizzati e sviluppati tutti i casi d'uso, analizzando i risultati ottenuti.

CAPITOLO 1

Analisi dei Big Data e Cloud Computing

1.1 Big Data e Data Mining

Oggi giorno ci troviamo difronte ad una produzione massiva di dati. Basti pensare alle più grandi piattaforme presenti sul mercato, come ad esempio Facebook, che nel 2014 aveva archiviato 300 petabyte di dati e ne riceveva ogni giorno 400 terabyte. Ad oggi, la big company è passata a 4 petabyte al giorno [1], con un conseguente incremento di dieci volte in dieci anni. Stoccare, analizzare e trarre valore aggiunto da questi dati è particolarmente importate ma difficile con le attuali tecnologie: per tale scopo nasce il concetto di Big Data. Il termine "Big Data" descrive set di dati enormi che sono troppo grandi per essere gestiti o archiviati utilizzando i metodi standard, come i sistemi di gestione di database relazionali (RDBMS). Sebbene non esista una dimensione fissa per definire i Big Data, in genere si considerano tali quando i dati diventano troppo voluminosi per essere gestiti o elaborati da un singolo computer. I Big Data hanno caratteristiche specifiche che li differenziano dai set di dati tradizionali, come mostrato in Figura 1.1.

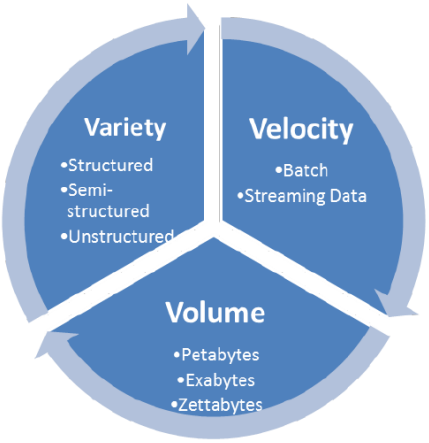

Figura 1.1: Caratteristiche dei Big Data

I ricercatori hanno riassunto tre aspetti importanti dei Big Data, ossia Volume, Velocità e Varietà, conosciuti anche come le 3V. Queste V, rappresentano le peculiarità che questi dati hanno, restituendoci la complessità con la quale dobbiamo interagire. Perciò ogni V può essere vista come una sfida che deve essere affrontata nell'ottica dell'utilizzo dei Big Data.

La sfida posta dal **volume** dei dati è la più evidente. Non esiste un accordo su quale dimensione utilizzare come base per quantificare i Big Data. Tale quantificazione dipende da vari fattori, come la complessità della struttura dei dati ed i requisiti delle applicazioni target. La complessità della struttura dei dati è un fattore importante, infatti, mentre un set di dati relazionali di diversi petabyte non può essere chiamato Big Data, poiché può essere facilmente gestito dagli odierni DBMS, al contrario invece, un set di dati grafici di diversi terabyte è comunemente considerato un Big Data, poiché l'elaborazione dei grafici è molto impegnativa per le nostre tecnologie. In secondo luogo, anche i requisiti delle applicazioni target dovrebbero essere considerati un fattore, in quanto in determinati ambiti tempi di risposta di diverse ore potrebbero non essere considerati problematici, mentre in altri ambiti potremmo avere la necessità di tempi di risposta molto brevi: questo introduce la sfida successiva [2].

La sfida della **velocità** deriva dalla necessità di gestire la velocità con cui vengono creati nuovi dati o aggiornati quelli esistenti. La velocità comporta sfide per ogni stack di una piattaforma per la gestione dei dati, sia per il livello di archiviazione che per quello di elaborazione delle query, in quanto devono essere estremamente

veloci e scalabili. La tecnologia dello streaming dei dati è studiata da diversi anni per gestire l'alta velocità. Tuttavia, la capacità dei sistemi di streaming esistenti è ancora limitata, soprattutto quando si fa fronte al crescente volume di dati in entrata degli odierni sistemi. Nelle applicazioni del mondo reale, i dati spesso non provengono da un'unica fonte. Le implementazioni dei Big Data richiedono la gestione di dati provenienti da varie fonti, in cui i dati possono avere formati e modelli diversi [2].

Ciò fa emergere la sfida della **varietà** dei dati. La varietà dei dati fornisce maggiori informazioni per risolvere problemi o per fornire un servizio migliore. La domanda è come catturare i diversi tipi di dati in modo da consentire di correlare i loro significati. In genere, i dati possono essere classificati in tre tipi generali: dati strutturati, dati semistrutturati e dati non strutturati. Esistono tecnologie sofisticate per gestire ciascuno di questi tipi di dati. Tuttavia, una perfetta integrazione di queste tecnologie rimane una sfida [2].

Sebbene le 3V sopra menzionate disegnano un quadro generale della sfida dei Big Data, ci sono altri problemi che potremmo incontrare quando si ha a che fare con i Big Data. Ad esempio, recentemente sono state proposte diverse altre alternative per la quarta V, tra cui Variabilità, Valore e Virtuale, che si riferiscono ad altri aspetti della gestione dei dati. Inoltre, le sfide variano a seconda degli scenari applicativi. [2]

L'analisi dei Big Data offre vantaggi significativi in vari settori, tra cui le aziende private e la pubblica amministrazione. Difatti, tale analisi, aiuta le aziende a diventare più competitive, consentendo loro di valutare rischi e opportunità di mercato, comprendere meglio i bisogni dei clienti e ottimizzare le attività per ridurre i costi.

Effettuate queste considerazioni, possiamo far rientrare i dati sanitari nel contesto dei BigData? Certo, sappiamo che "solo negli Stati Uniti, l'utilizzo del data mining nell'informatica sanitaria potrebbe far risparmiare al settore sanitario fino a 450 miliardi di dollari ogni anno. Questo perché il campo dell'informatica sanitaria genera una quantità ampia e crescente di dati. Nel 2011, le organizzazioni sanitarie avevano generato oltre 150 exabyte di dati (un exabyte equivale a 1000 petabyte). Questi dati devono essere vagliati e analizzati in modo efficiente per poter essere utili al sistema sanitario"[3]. Di conseguenza anche il settore pubblico, in particolare la sanità, può beneficiare dell'analisi dei Big Data per migliorare l'analisi dei costi, la prevenzione e altri aspetti. Come possiamo ricavare delle informazioni da questa

mole di dati? Ciò è possibile grazie al Data Mining.

Data Mining

Il Data Mining rappresenta una tecnica cruciale per affrontare il crescente flusso di dati che caratterizza il nostro mondo contemporaneo. In un'epoca in cui la disponibilità di dati è abbondante, il "core" risiede nell'estrarre le informazioni significative da queste enormi masse di dati. Tuttavia, i dati spesso si presentano in modo eterogeneo, ridondante e non strutturato, richiedendo quindi un'attenta preparazione prima di poter essere analizzati in modo efficace. Il Data Mining si è evoluto come risposta a questa sfida, integrando principi provenienti dalla statistica e dall'Intelligenza Artificiale per esplorare e analizzare automaticamente le informazioni al fine di individuare pattern utili per vari scopi. Le fasi fondamentali del processo di Data Mining includono la selezione dei dati, il pre-processamento, la trasformazione, l'estrazione e la valutazione dei dati. Tali fasi sono riportate graficamente in Figura 1.2.

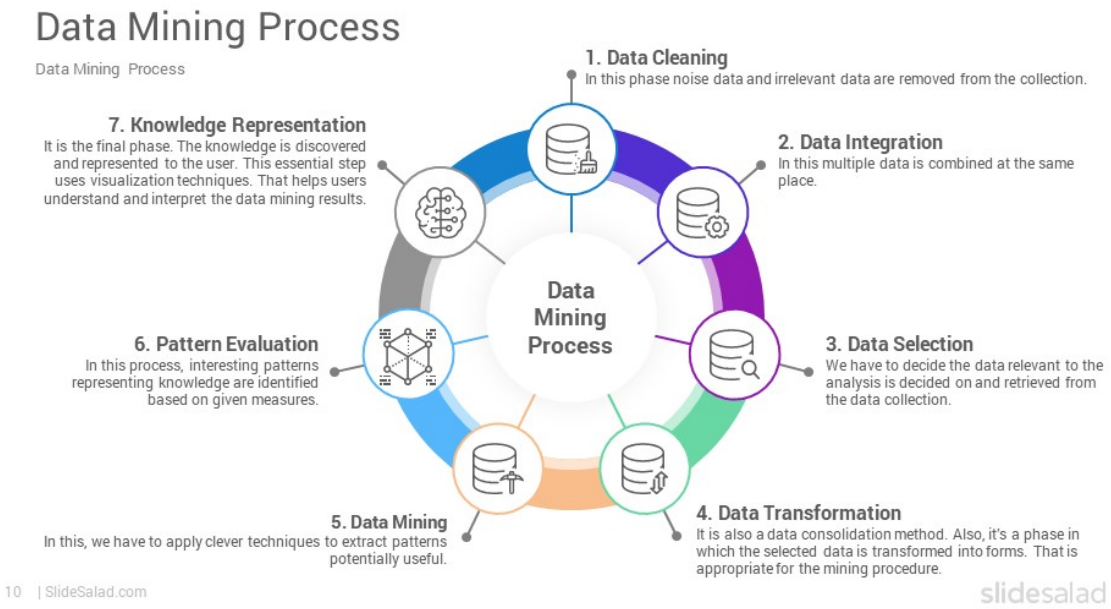

Figura 1.2: Fasi dell'estrapolazione di conoscenza tramite Data Mining

Nella selezione dei dati, è fondamentale scegliere il set di dati appropriato, rimuovendo le informazioni non pertinenti per ridurre il tempo di analisi. Il pre-processamento dei dati implica la pulizia dei dati da rumore e inconsistenze, preparandoli per le fasi successive. La trasformazione dei dati mira a organizzare e

consolidare i dati in formati adatti all'analisi, riducendo la varietà dei dati senza comprometterne la qualità.

Successivamente, l'estrazione dei dati utilizza algoritmi specifici per individuare pattern e informazioni rilevanti nel set di dati. Infine, la fase di valutazione produce la documentazione per interpretare i risultati del processo di Data Mining.

Attraverso il Data Mining, è possibile comprendere in profondità i dati, individuare modelli comportamentali e formulare previsioni che altrimenti rimarrebbero sconosciute. La corretta selezione e correlazione delle informazioni può portare a una comprensione più approfondita dei fenomeni, consentendo di trarre vantaggio da una vasta quantità di dati disponibili.
Esistono diversi passi che devono essere effettuati sui dati, questi processi possono essere denominate come **fasi per la gestione dei Big Data**. Tra i processi chiave per la gestione dei Big Data vi sono: Data Ingestion, Data Processing e Data Warehouse. La **Data Ingestion** riguarda l'acquisizione e l'importazione dei dati grezzi da varie fonti, mentre il **Data Processing** coinvolge la trasformazione e l'analisi dei dati per estrarre informazioni significative. Infine, il **Data Warehouse** fornisce un sistema di archiviazione specializzato per gestire grandi volumi di dati, facilitando l'accesso e l'analisi. Attraverso questi processi, le organizzazioni possono sfruttare il potenziale dei Big Data per ottenere "insight" preziosi e prendere decisioni informate.

1.1.1 Data Ingestion

La Data Ingestion rappresenta il primo passo cruciale nel processo di gestione dei dati, coinvolgendo l'acquisizione, l'aggregazione e la preparazione dei dati grezzi provenienti da una varietà di fonti. Queste fonti possono includere database transazionali, file di log, dispositivi IoT, piattaforme di social media e molto altro ancora. Durante questa fase, i dati vengono raccolti e trasferiti in un sistema di archiviazione centralizzato, come un Data Lake o un Data Warehouse, dove possono essere ulteriormente elaborati e analizzati. Le attività di Data Ingestion comprendono la connessione e l'accesso ai diversi tipi di sorgenti di dati, l'estrazione dei dati grezzi, la trasformazione in un formato standardizzato e l'inserimento nei repository di destinazione. È fondamentale garantire l'affidabilità e la sicurezza dei dati durante

l'intero processo di Data Ingestion. Possiamo vedere graficamente in Figura 1.3 una rappresentazione semplificata dell'intero processo.

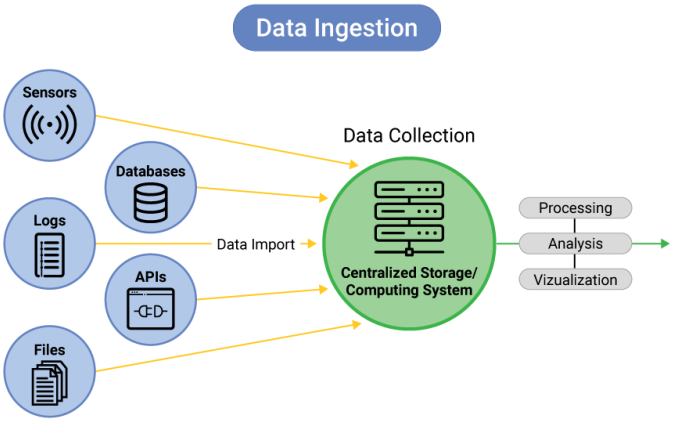

Figura 1.3: Data Ingestion steps

1.1.2 Data Processing

Il Data Processing, è una fase critica nel ciclo di vita dei dati, dove i dati grezzi vengono trasformati ed elaborati in una forma più significativa e utile per l'analisi e la generazione di "insight". Durante questa fase, i dati possono essere sottoposti a una serie di operazioni, tra cui l'operazione di pulizia dei dati per rimuovere errori e inconsistenze, l'integrazione di dati da diverse fonti, la trasformazione dei dati in un formato comune e l'arricchimento dei dati con metadati aggiuntivi. Questo processo è spesso eseguito utilizzando strumenti e tecnologie come Apache Spark, Hadoop MapReduce e Python con la libreria *pandas*. L'obiettivo principale del Data Processing è preparare i dati in modo che possano essere utilizzati per l'analisi e la generazione di report significativi. Possiamo vedere una rappresentazione astratta del concetto di Data Processing in Figura 1.4.

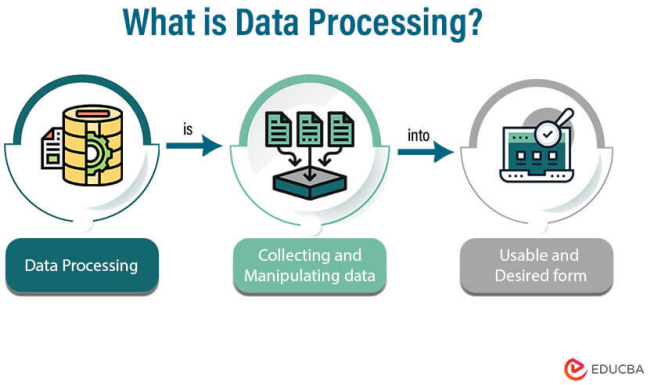

Figura 1.4: Data Processing

1.1.3 Data Warehouse

Un Data Warehouse è un sistema specializzato, progettato per archiviare grandi quantità di dati provenienti da varie fonti in un'unica posizione centralizzata, ottimizzata per l'analisi aziendale e la generazione di report. Questi sistemi integrano dati da fonti operative, archivi storici, applicazioni esterne e altri sistemi dati, organizzandoli in una struttura progettata per supportare interrogazioni complesse e analisi ad hoc. I Data Warehouse utilizzano spesso modelli dimensionali, come il modello a stella o il modello a fiocco di neve, per organizzare i dati in tabelle di "dimensions" e "facts", facilitando l'analisi OLAP (Online Analytical Processing). Le tecnologie popolari per

la creazione e la gestione di Data Warehouse includono Microsoft SQL Server, Oracle Database e Snowflake. In Figura 1.5 possiamo vedere come i vari flussi interagiscono con il Data Werehouse.

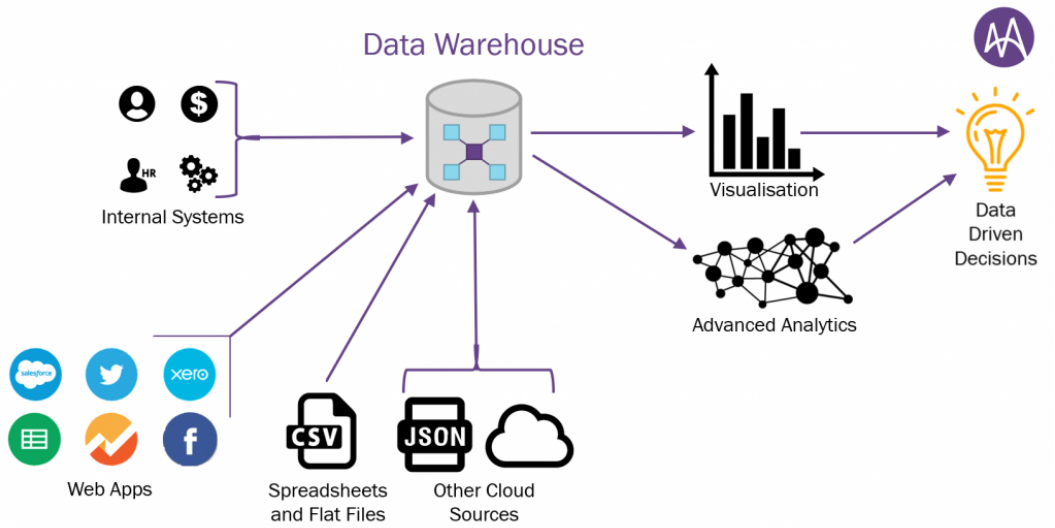

Figura 1.5: Data Werehouse workflow

1.2 Cloud Computing

In questa sezione analizziamo il concetto di Cloud Computing e come questo sia correlato ai Big Data.

1.2.1 Definizione di Cloud e Cloud Computing

Nel contesto dei servizi e delle piattaforme On Demand, va fatta una distinzione tra il termine "Cloud" e "Cloud Computing", in quanto indicano tecnologie diverse. Per tale motivo approfondiamo tali concetti di seguito.

Definizione Cloud

"Un Cloud è un pool di risorse informatiche virtualizzate. Un Cloud:

- Ospita una varietà di carichi di lavoro diversi, inclusi lavori back-end in stile batch e applicazioni rivolte all'utente.
- Consente la distribuzione e la scalabilità dei carichi di lavoro rapidamente attraverso il provisioning rapido di macchine virtuali o macchine fisiche.
- Supporta modelli di programmazione ridondanti, con ripristino automatico e altamente scalabili che consentono la ripartizione dei carichi di lavoro e la protezione da guasti hardware/software attraverso la ridondanza.
- Monitora l'utilizzo delle risorse in tempo reale per consentire il bilanciamento delle allocazioni quando necessario."[4]

Definizione Cloud Computing

Il termine "Cloud Computing", noto anche come "computazione in Cloud", è usato per descrivere sia una piattaforma che un tipo di applicazione. Una piattaforma di Cloud Computing effettua il provisioning, la configurazione, la riconfigurazione e il deprovisioning dinamico dei server in base alle necessità. Invece, per applicazioni Cloud intendiamo software accessibili tramite Internet. Queste applicazioni Cloud utilizzano grandi Data Center e potenti server per offrire servizi Web. Chiunque disponga di una connessione Internet adeguata e di un browser può accedere a un'applicazione Cloud. Questa flessibilità consente agli utenti di accedere a risorse

hardware e software in base alle loro esigenze, senza dover fare investimenti iniziali in costose infrastrutture IT e senza la necessità di gestirle in proprio.[4].

1.2.2 Cloud Computing e Big Data

Nel contesto dei Big Data, il Cloud Computing svolge un ruolo essenziale. I Big Data si riferiscono a insiemi di dati così enormi e complessi che richiedono strumenti specializzati per la loro elaborazione. Il Cloud Computing, in quanto piattaforma che fornisce pool di risorse dinamici, virtualizzazione e grande disponibilità di potenza si lega perfettamente alle necessità richieste per gestire queste enormi quantità di dati, consentendo alle aziende di trarre valore da essi, senza il bisogno di dover possedere fisicamente questa infrastruttura, in quanto questi servizi sono offerti On Demand. Uno degli obiettivi delle aziende è quello di ridurre al minimo i costi, comprese le spese informatiche. Il Cloud Computing consente a un'organizzazione di ridurre tali costi attraverso un migliore utilizzo delle risorse, per ottenere sia costi amministrativi e infrastrutturali ridotti che cicli di implementazione più rapidi [4].

1.3 OLAP

OLAP (Online Analytical Processing) è una tecnologia fondamentale nell'ambito dell'analisi dei dati che consente agli utenti di esplorare grandi volumi di dati da molteplici prospettive in modo rapido e interattivo. I cubi OLAP hanno un ruolo cruciale di tale tecnologia: essi rappresentano una struttura dati multidimensionale che organizza le informazioni in un formato intuitivo e comprensibile per l'analisi aziendale. I cubi OLAP sono composti da tre elementi principali:

- Le **dimensions**, le quali rappresentano gli aspetti o le caratteristiche dei dati che si desidera analizzare. Ad esempio, nel contesto delle vendite al dettaglio, le dimensioni potrebbero includere il tempo (come anno, trimestre, mese), i prodotti (come categoria, marca), i canali di vendita (come negozio fisico, e-commerce) e i clienti (come regione, segmento di clientela).

- Le **measures**, le quali sono delle metriche numeriche che vengono analizzate all'interno del cubo. Queste possono includere valori come vendite totali, margine di profitto, quantità vendute e così via.

- Le **celle** rappresentano i punti di dati all'interno del cubo che corrispondono all'intersezione di una combinazione specifica di dimensioni. Ad esempio, una cella potrebbe rappresentare le vendite totali di un prodotto specifico in un determinato trimestre e in una determinata regione.

All'interno di questi cubi, gli utenti possono eseguire una serie di operazioni analitiche fondamentali, che consentono di estrarre informazioni significative dai dati. Alcune delle operazioni più rilevanti, che possiamo vedere graficamente in figura 1.6, includono:

- Drill-down e Roll-up: attraverso queste operazioni, gli utenti possono esplorare dati a diversi livelli di dettaglio o aggregarli per ottenere una visione più ampia;

- Slice-and-dice: questa operazione consente di "affettare" il cubo, selezionando una porzione specifica di dati in base a determinati criteri;

- Pivot: questa operazione consente agli utenti di scambiare dimensioni o aspetti del cubo per esaminare i dati da prospettive diverse;

- Calcolo di Metriche: gli utenti possono definire nuove metriche o indicatori di performance derivati da quelli esistenti, consentendo un'analisi più approfondita e personalizzata;

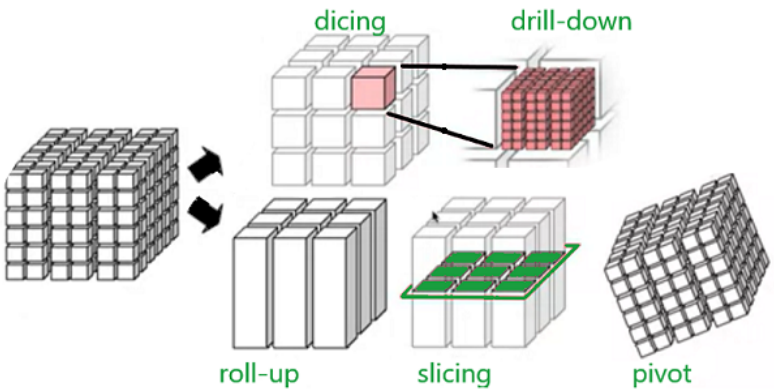

Figura 1.6: Operazioni sui Cubi OLAP

Utilizzando i cubi OLAP, è possibile identificare tendenze nel tempo e analizzare le variazioni nei dati per prendere decisioni informate.

1.4 Sistemi per l'analisi dei dati

In questa sezione andremo a esplorare com'è strutturata un l'architettura per l'analisi dei Big Data e quali sono le fasi che un sistema di analisi deve effettuare sui dati.

1.4.1 Architettura Big Data

Un'architettura Big Data è progettata per gestire l'acquisizione, l'elaborazione e l'analisi di dati troppo grandi o complessi per i sistemi di database tradizionali. Sempre più spesso, il termine Big Data fa riferimento al valore che si può estrarre dai dati attraverso analisi avanzate, piuttosto che solo alla dimensione dei dati, anche se in alcuni casi tendono ad essere molto grandi. Alcuni dati arrivano a un ritmo rapido, esigendo costantemente di essere esaminati. Altri dati arrivano più lentamente, ma in blocchi molto grandi. Queste sono sfide che le architetture Big Data cercano di risolvere [5].

1.4.1.1 Componenti di un'architettura Big Data

Il diagramma seguente (Figura 1.7) mostra le componenti che si inseriscono in un'architettura Big Data [5].

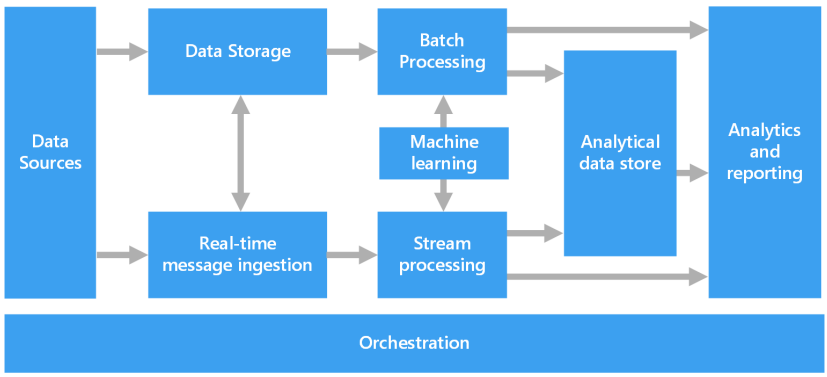

Figura 1.7: Architettura per i Big Data

"La maggior parte delle architetture Big Data includono alcuni o tutti i seguenti componenti:

- **Data sources**. Tutte le soluzioni Big Data acquisiscono i dati da una o più fonti. Come ad esempio database relazionali, file di log, dati in tempo reale provenienti da dispositivi IoT.
- **Data storage**. I dati per l'elaborazione batch vengono tipicamente archiviati in un DFS, ovvero un File System Distribuito, che riesce a contenere file di grandi volumi. Questo tipo di archivio è spesso chiamato un Data Lake o Data Warehouse.
- **Batch processing**. Spesso una soluzione offerta per elaborare i Big Data è quella dell'utilizzo di task batch a lunga durata per filtrare, aggregare e preparare i dati per l'analisi. Di solito queste task coinvolgono la lettura dei file sorgenti, la loro elaborazione e la scrittura dell'output in nuovi file.
- **Real-time message ingestion**. Se l'applicativo prevede l'acquisizione di fonti in tempo reale, l'architettura deve includere un modo per catturare e archiviare messaggi in tempo reale e permette l'elaborazione dei questi flussi. Spesso, questo tipo di soluzione necessita di un buffer dove immagazzinare i messaggi per poter supportare l'elaborazione distribuita e la consegna affidabile.
- **Stream processing**. Dopo aver catturato i messaggi in tempo reale, bisogna elaborarli filtrandoli, aggregandoli e preparandoli per l'analisi. I dati, dopo essere stati elaborati, vengono quindi scritti in un componente di output.
- **Machine learning**. Dopo aver letto i dati preparati per l'analisi (dall'elaborazione batch o di flusso), gli algoritmi di apprendimento automatico possono essere utilizzati per costruire modelli che possono prevedere risultati o classificare dati.
- **Analytical data store**. Molte volte i dati devono essere preparati per poter effettuare delle analisi su di essi, quindi vengono normalizzati in un formato strutturato che consente di effettuare ciò.

- **Analysis and reporting**. L'obiettivo della maggior parte delle soluzioni Big Data è quello di fornire approfondimenti sui dati attraverso analisi e la creazione di report ad Hoc.

- **Orchestration**. La maggior parte delle soluzioni per i Big Data vogliono poter offrire un modo facile di elaborare i dati, incapsulare i flussi, spostare dati da componenti in altre componenti o anche inviare dati a dashboard o report. Questo per aumentare la flessibilità e comprensione di tali dati." [5]

1.4.1.2 Tipi di architettura

Esistono diversi tipi di architetture utilizzabili a seconda delle nostre necessita, possiamo distinguere principalmente tra: Lambda architecture e Kappa architecture. Quando si lavora con insiemi di dati molto grandi, eseguire il tipo di interrogazioni di cui i clienti hanno bisogno può richiedere molto tempo. Queste interrogazioni non possono essere eseguite in tempo reale e spesso richiedono algoritmi che operano in parallelo sull'intero insieme di dati. I risultati vengono quindi memorizzati separatamente dai dati grezzi e utilizzati per le interrogazioni. Uno svantaggio di questo approccio è che introduce latenza: se l'elaborazione richiede alcune ore, un'interrogazione potrebbe restituire risultati che risalgono a diverse ore prima. Idealmente, si vorrebbe ottenere alcuni risultati in tempo reale (forse anche con una certa perdita di precisione) e combinare questi risultati con quelli provenienti dall'analisi batch. L'**architettura lambda**, proposta per la prima volta da Nathan Marz, affronta questo problema creando due percorsi per il flusso dei dati, come mostrato in Figura 1.8 [5].

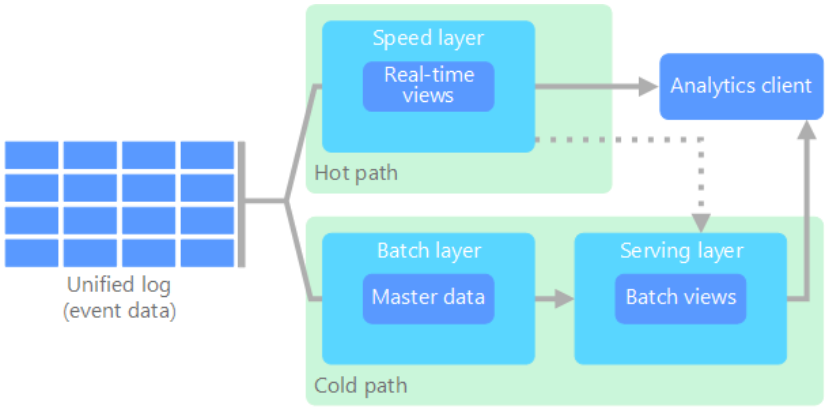

Figura 1.8: Architettura Lambda

Tutti i dati che entrano nel sistema passano attraverso questi due percorsi:

- Batch Layer (percorso "Cold") memorizza tutti i dati in arrivo nella loro forma grezza e esegue l'elaborazione batch sui dati. Il risultato di questa elaborazione viene memorizzato come una vista batch [5].
- Speed Layer (percorso "Hot") analizza i dati in tempo reale. Questo strato è progettato per una bassa latenza, a discapito della precisione [5].

Il Batch Layer alimenta il Serving Layer che indicizza la vista batch per interrogazioni efficienti. Lo Speed Layer aggiorna il Serving Layer con aggiornamenti incrementali basati sui dati più recenti [5].

I dati che scorrono nel percorso "Hot" sono limitati dai requisiti di latenza imposti dallo Speed Layer, in modo che possano essere elaborati il più rapidamente possibile. Spesso, questo richiede di trovare un compromesso tra la precisione e la velocità dei dati, in quanto inversamente proporzionali. I dati che fluiscono nel percorso "Cold", d'altra parte, non sono soggetti agli stessi requisiti di bassa latenza. Ciò consente un calcolo ad alta precisione su grandi insiemi di dati, che può richiedere molto tempo. Alla fine, i percorsi "Hot" e "Cold" convergono nell'applicazione di analisi del client. Se il client ha bisogno di visualizzare dati tempestivi, ma potenzialmente meno accurati, otterrà il suo risultato dal percorso "Hot". Altrimenti, selezionerà i risultati dal percorso "Cold" per visualizzare dati meno tempestivi ma più accurati[5].

Uno svantaggio dell'architettura lambda è la sua complessità. La logica di elaborazione appare in due luoghi diversi - i percorsi "Cold" e "Hot" - utilizzando framework diversi. Questo porta alla duplicazione della logica di calcolo e alla complessità della gestione dell'architettura per entrambi i percorsi. L'**architettura kappa** è stata proposta da Jay Kreps come alternativa all'architettura lambda. Ha gli stessi obiettivi di base dell'architettura lambda, ma con una distinzione importante: tutti i dati fluiscono attraverso un singolo percorso, utilizzando un unico sistema di elaborazione di flussi, come mostrato in Figura 1.9 [5].

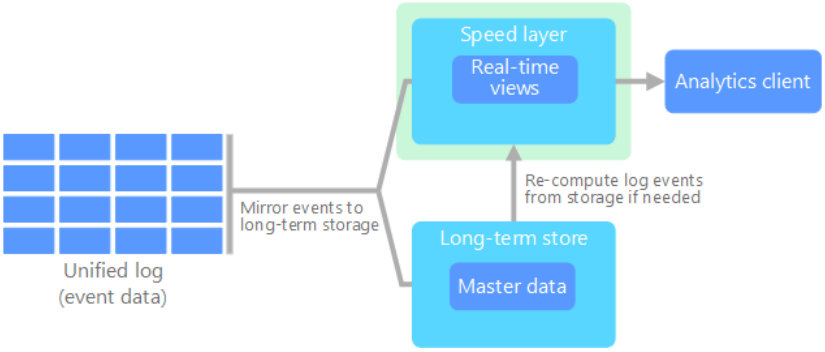

Figura 1.9: Architettura Kappa

Ci sono alcune somiglianze con il Batch Layer dell'architettura lambda, nel senso che i dati degli eventi sono immutabili, con la differenza che tutti i dati vengono salvati, piuttosto che una sottoparte. I dati vengono acquisiti come flusso di eventi e salvati in un registro distribuito. Questi eventi seguono l'ordine cronologico, e lo stato dell'evento corrente viene modificato solo quando viene aggiunto un nuovo evento. Similmente allo Speed Layer dell'architettura lambda, tutti i processi degli eventi vengono eseguiti in un unico flusso di input e memorizzati come vista in real-time. Se è necessario ricomputare l'intero insieme di dati (equivalentemente a ciò che fa il Batch Layer nell'architettura lambda), è sufficiente ridurre la dimensione dello stream per completare i calcoli in modo tempestivo attraverso l'utilizzo del calcolo parallelo [5].

1.4.2 Fasi dell'analisi dei dati

Nel panorama dell'analisi dati, la gestione efficace dei processi di acquisizione, elaborazione e archiviazione dei dati riveste un ruolo fondamentale. Queste fasi, interconnesse e complementari, costituiscono l'ossatura dei sistemi di analisi dati odierni. In questa sezione andremo ad approfondire le fasi che sono state introdotte nelle precedenti sezioni 1.1.1, 1.1.2 e 1.1.3.

1.4.2.1 Acquisizione dei Dati

L'acquisizione e l'integrazione dei dati rappresentano delle fasi fondamentali nei sistemi di analisi, richiedendo processi complessi per l'estrazione, il caricamento e la trasformazione dei dati (ELT). Questi processi, oltre a spostare e trasformare i

dati, potrebbero anche applicare su di essi dei processi per migliorarne la qualità. Gli strumenti di integrazione devono garantire la scalabilità e l'elaborazione parallela, garantendo il supporto per flussi di dati oltre alla classica elaborazione batch [6]. Possiamo far affidamento su tre strumenti, a seconda delle nostre necessità, per effettuare l'integrazione o l'acquisizione dei dati:

- **Strumenti di Integrazione dei Dati**
  Gli strumenti tradizionali di integrazione dei dati, utilizzati nel supporto al Data Warehousing, esistono da decenni. Questi strumenti consentono la creazione di pipeline attraverso un'interfaccia grafica, utilizzando componenti predefiniti per l'estrazione, la trasformazione e l'archiviazione dei dati. Tuttavia, si concentrano principalmente sull'elaborazione dei dati strutturati e sono orientati all'elaborazione batch [6].

- **Strumenti di Acquisizione dei Dati in Streaming**
  L'acquisizione dei dati in streaming rappresenta una sfida per gli strumenti di integrazione tradizionali a causa della loro natura orientata ai batch. Gli strumenti di elaborazione del flusso sono più adatti per l'uso con i Big Data, essendo scalabili e in grado di gestire grandi volumi di dati in tempo reale. Questi strumenti sono disponibili in varie forme, dai semplici strumenti basati su riga di comando ai più complessi con interfaccia grafica [6].

- **Strumenti di Elaborazione dei Dati nel Cloud**
  Gli strumenti di elaborazione dei dati nel Cloud rappresentano una terza categoria, che deve supportare sia l'elaborazione batch che quella in streaming senza limitazioni. Questi strumenti possono essere suddivisi in strumenti preesistenti spostati nel Cloud come servizi e servizi nativi del Cloud. Tuttavia, la maggior parte di essi offre poca o nessuna estensibilità oltre alle funzionalità integrate[6].

1.4.2.2 Elaborazione dei Dati

Alcune applicazioni fanno affidamento nella loro logica applicativa sul tempo, ad esempio il flusso di dati o la gestione, l'elaborazione e l'analisi dei dati in streaming. Queste applicazioni e le loro analisi devono gestire i dati in tempo reale e le decisioni devono essere prese entro un determinato intervallo di tempo. Un trasferimento continuo di informazioni da un luogo a un altro, per un lungo periodo di tempo, è un comune requisito nelle applicazioni in tempo reale. Possiamo vedere graficamente in Figura 1.10 il confronto tra i due approcci possibili:

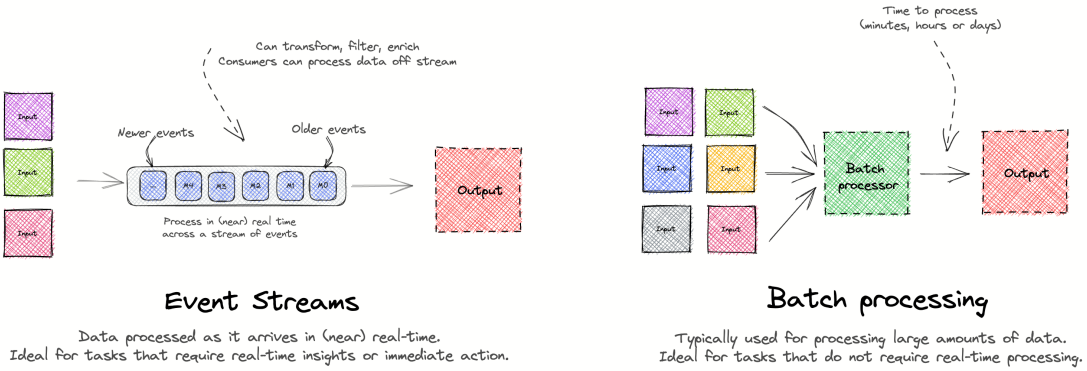

Figura 1.10: Batch Processing vs Real Time Processing

Quando una raccolta di transazioni viene accumulata nel tempo, parliamo di **Batch Processing**, ovvero si tratta di un approccio che consente di risparmiare tempo nell'elaborazione di enormi volumi di dati. I dati vengono raccolti, inseriti ed elaborati prima di generare i risultati del batch. Lo scopo fondamentale di un sistema di elaborazione batch è quello di mantenere in esecuzione contemporaneamente tutti i task di un batch. [7].

Quando invece parliamo di elaborazione di flussi di dati o eventi continui, parliamo di **Real Time Processing**. I sistemi di elaborazione in tempo reale sono estremamente rapidi e reattivi. Questi sistemi vengono impiegati in situazioni in cui è necessario accettare e gestire velocemente un numero enorme di eventi. L'elaborazione in tempo reale richiede transazioni rapide ed è caratterizzata da risposte tempestive. I dati devono essere scaricati in lotti prima di poter essere elaborati, archiviati o analizzati nei metodi di elaborazione dei dati batch, mentre i dati in streaming arrivano costantemente e possono essere trattati nello stesso momento[7].

1.4.2.3 Archiviazione dei Dati

I Data Warehouse raccolgono informazioni da fonti remote in un unico database. Tali sistemi sono stati creati per velocizzare le query riducendo il volume dei dati da scansionare. Il modello di dati più popolare per un Data Warehouse è un modello multidimensionale. Tale modello può esistere sotto forma di vari tipi di schemi, tra cui i più popolari sono lo schema a stella e lo schema a snowflake. Lo schema a stella è un tipo di database relazionale utilizzato per la rappresentazione di dati multidimensionali. In questo schema la tabella dei fact è in formato normalizzato e le tabelle delle dimension sono in formato denormalizzato. Lo schema snowflake è più complesso rispetto a uno schema a stella. Tale schema normalizza le dimension per eliminare la ridondanza [8].

Differenze tra lo schema Star e Snowflake

"Le differenze tra lo schema a stella e quello snowflake sono le seguenti:

1. Una tabella di uno schema a snowflake è facile da mantenere in memoria e consente di risparmiare spazio di archiviazione rispetto a una tabella di uno schema a stella, poiché una tabella di grandi dimensioni può diventare enorme quando la struttura dimensionale viene inclusa come colonna.

2. Qualora dividessimo lo schema a stella, otterremmo diverse tabelle con delle dimension, ottenendo di conseguenza uno schema a snowflake.

3. Le prestazioni saranno più alte con l'utilizzo dello schema a stella rispetto alle prestazioni ottenibili dallo schema a snowflake a causa delle join che quest'ultima dovrà effettuare.

4. Lo schema a stella è più semplice rispetto il snowflake che è più complesso. Inoltre, la struttura a snowflake può ridurre l'efficacia della navigazione poiché saranno necessari più join per eseguire una query. Pertanto, nella progettazione del Data Warehouse, lo schema a snowflake non è popolare come lo schema a stella.

5. La tabella delle dimension nello schema a stella non ha una tabella principale, a differenza dello schema snowflake dove la tabella delle dimensions ha diverse tabelle principali.

6. Lo schema a stella ha una tabella di facts circondata da tabelle dimension, mentre lo schema a snowflake ha una tabella di facts circondata da diversi livelli di tabelle dimension, a seconda della diversità del modello."[8]

Per capire meglio le differenze tra i due schemi, può essere esplicativa la seguente immagine (Figura 1.11).

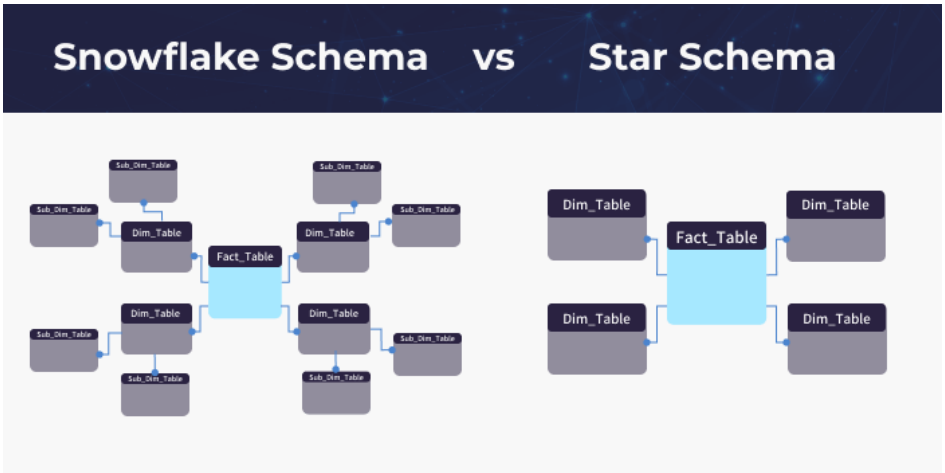

Figura 1.11: Differenze schema a snowflake e stella

CAPITOLO 2

Caso di Studio e Architettura Proposta

Dopo aver dato una panoramica dello stato dell'arte, con la quale si è capito e compreso quali sono i fondamenti teorici e architetturali del caso di studio, cerchiamo ora di approfondire il dominio applicativo e la mia proposta di soluzione. Possiamo partire da quali siano le esigenze da colmare, ovvero rendere facile e accessibile la consultazione di grandi quantità di dati in tempo reale, per aumentare le decisioni "data driven" e l'ottimizzare di processi.

2.1 Dominio Applicativo

Per il caso di studio è stato reperito un dataset contenente dei dati ospedalieri dalla piattaforma Kaggle, una piattaforma online dedicata all'analisi dei dati per algoritmi di machine learning. L'attenzione inizialmente si è focalizzata su due moduli chiave:

- L'estrazione e la pulizia dei dati (ETL): Il primo modulo riguarda la lettura dei dati da fonti aperte, il processamento e l'ingegnerizzazione dei dati. Verrà previsto il salvataggio di questi dati in una database.
- L'analisi multidimensionale (OLAP): Il secondo modulo vedrà invece l'integrazione e lo sviluppo della parte di Data Visualization, con lo scopo di fornire

analisi multidimensionali dei dati ingegnerizzati nella fase precedente.

L'obiettivo quindi è quello di andare a costruire un middleware che permette di semplificare le fasi di ETL, Data Ingestion, Data Processing e Data Visualization. Nel primo modulo, si è affrontato l'ingegnerizzazione dei dati e del loro caricamento nel database, mentre nel secondo modulo, si sono integrate tecnologie come Cube.js e Apache Superset per condurre analisi avanzate e visualizzazioni dei dati. Per ognuno dei seguenti moduli è previsto l'utilizzo delle tecnologie elencate:

- Data Ingestion, Data Processing:
  - Java EE
  - Postgresql
- Data Visualization e analisi multidimensionale:
  - Cube.js
  - Apache Superset
- Running environment:
  - Docker

2.1.1 Data Ingestion

Per la fase di Data Ingestion è stato sviluppato il modulo ETL che permette di partire dai dati grezzi presenti nel .csv e caricarli nel database PostGress. Prima di fare ciò si sono analizzati i dati presenti nel dataset. Tale dataset, comprende informazioni su oltre quattordicimila pazienti, ognuno caratterizzato da 55 attributi, accumulati nell'arco di due anni, ovvero dal 2017-2019. I dati sono stati raccolti presso il complesso ospedaliero denominato "Hero DMC Heart Institute" situato in Ludhiana, Punjab, India. In seguito sono state analizzate tutte le feature, ovvero i campi del dataset, per capire come fossero strutturate. Di seguito, riporto l'esito di questa analisi nel seguente formato (**Acronimo** - Nome - Tipo: Significato):

- **SNO** - Serial-number - Integer: pratica di ospedalizzazione.

- **MRD** - No.Admission number - Integer: numero di identificativo paziente.
- **D.O.A** - Date of Admission - Datetime: data di ospedalizzazione.
- **D.O.D** - Date of Discharge - Datetime: data di dimissione.
- **AGE** - Age - Integer: età del paziente.
- **GENDER** - Gender - Enum: sesso del paziente
  - [M] ale
  - [F]emale
  - [O]ther
- **RURAL** - Rural - Enum: area urbana di residenza
  - [U]rban
  - [R]ural
- **TYPE OF ADMISSION-EMERGENCY/OPD** - Type of admission-emergency/opd - Enum: ospedalizzazione per emergenza, oppure ospedalizzazione per visite cicliche o trattamenti
  - [E]mergency
  - [O]PD
- **DURATION OF STAY** - Duration of stay - Integer: giorni di ospedalizzazione.
- **DURATION OF INTENSIVE UNIT STAY** - Duration of intensive unit stay - Integer: giorni in terapia intensiva.
- **OUTCOME** - Outcome - Varchar: risultato dell'ospedalizzazione
  - [DAMA] Discharged Against Medical Advice
  - [DISCHARGE] Dimesso
  - [EXPIRY] Deceduto
- **SMOKING** - Smoking - Bool:

  - 0 paziente non tabagista;
  - 1 paziente tabagista;

- **ALCOHOL** - Alcohol - Bool:
  - 0 il paziente non assume periodicamente alcol;
  - 1 il paziente assume periodicamente alcol;
- **DM** - Diabetes mellitus - Bool:
  - 0 il paziente non ha il diabete mellito;
  - 1 il paziente ha il diabete mellito;
- **HTN** - Hypertension - Bool:
  - 0 il paziente non soffre di ipertensione;
  - 1 il paziente soffre di ipertensione;
- **CAD** - Coronary artery disease - Bool:
  - 0 il paziente non soffre di coronaropatia;
  - 1 il paziente soffre di coronaropatia;
- **PRIOR CMP** - Cardiomyopathy - Bool:
  - 0 il paziente non soffre di cardiomiopatia;
  - 1 il paziente soffre di cardiomiopatia;
- **CKD** - Chronic kidney disease - Bool:
  - 0 il paziente non soffre di malattia renale cronica;
  - 1 il paziente soffre di malattia renale cronica;
- **HB** - Haemoglobin - Double: valore rilevato di emoglobina.
- **TLC** - Total leukocytes count - Double: conteggio totale dei leucociti.
- **PLATELETS** - Platelets - Integer: conteggio delle piastrine.

- **GLUCOSE** - Glucose - Integer: valore rilevato del glucosio.
- **UREA** - Urea - Integer: valore rilevato dell'urea.
- **CREATININE** - Creatinine - Double: valore rilevato di creatinina.
- **BNP** - B-type natriuretic peptide - Integer: valore rilevato di peptide natriuretico di tipo B (anche detto peptide natriuretico cerebrale).
- **RAISED CARDIAC ENZYMES** - Raised cardiac enzymes - Bool:
  - 0 il paziente ha un numero di enzimi cardiaci nella norma;
  - 1 il paziente non ha un numero di enzimi cardiaci nella norma;
- **EF** - Ejection fraction - Integer: valore rilevato di frazione di eiezione.
- **SEVERE ANAEMIA** - Severe anaemia - Bool:
  - 0 il paziente non soffre di anemia grave;
  - 1 il paziente soffre di anemia grave;
- **ANAEMIA** - Anaemia - Bool:
  - 0 il paziente non soffre di anemia;
  - 1 il paziente soffre di anemia;
- **STABLE ANGINA** - Stable angina - Bool:
  - 0 il paziente non ha l'angina stabile;
  - 1 il paziente soffre di angina stabile;
- **ACS** - Acute coronary Syndrome - Bool:
  - 0 il paziente non soffre di sindrome coronarica acuta;
  - 1 il paziente soffre di sindrome coronarica acuta;
- **STEMI** - St elevation myocardial infarction - Bool:
  - 0 il paziente non ha avuto un infarto al miocardico con sopraslivellamento del tratto ST;

  - 1 il paziente ha avuto un infanto al miocardico con sopraslivellamento del tratto ST;

- **ATYPICAL CHEST PAIN** - Atypical chest pain - Bool:
  - 0 il paziente non soffre di un dolore atipico al petto;
  - 1 il paziente soffre di un dolore atipico al petto;
- **HEART FAILURE** - Heart failure - Bool:
  - 0 il paziente non ha avuto un'insufficienza cardiaca;
  - 1 il paziente ha avuto un'insufficienza cardiaca;
- **HFREF** - Heart failure with reduced ejection fraction - Bool:
  - 0 il paziente non ha un insufficienza cardiaca con frazione di eiezione ridotta;
  - 1 il paziente ha un insufficienza cardiaca con frazione di eiezione ridotta;
- **HFNEF** - Heart failure with normal ejection fraction - Bool:
  - 0 il paziente non ha un insufficienza cardiaca con frazione di eiezione normale;
  - 1 il paziente ha un insufficienza cardiaca con frazione di eiezione normale;
- **VALVULAR** - Valvular heart disease - Bool:
  - 0 il paziente non soffre di malattie cardiache valvolari;
  - 1 il paziente soffre di malattie cardiache valvolari;
- **CHB** - Complete heart block - Bool:
  - 0 il paziente non ha avuto un blocco cardiaco completo;
  - 1 il paziente ha avuto un blocco cardiaco completo;
- **SSS** - Sick sinus syndrome - Bool:
  - 0 il paziente non ha la sindrome del nodo del seno;

  - 1 il paziente ha la sindrome del nodo del seno;

- **AKI** - Acute kidney injury - Bool:
  - 0 il paziente non ha un danno renale acuto;
  - 1 il paziente ha un danno renale acuto;
- **CVA INFRACT** - Cerebrovascular accident infarct - Bool:
  - 0 il paziente non ha avuto un infarto da incidente cerebrovascolare;
  - 1 il paziente ha avuto un infarto da incidente cerebrovascolare;
- **CVA BLEED** - Cerebrovascular accident bleed - Bool:
  - 0 il paziente non ha un sanguinamento da incidente cerebrovascolare;
  - 1 il paziente ha un sanguinamento da incidente cerebrovascolare;
- **AF** - Atrial fibrilation - Bool:
  - 0 il paziente non ha una fibrillazione atriale;
  - 1 il paziente ha una fibrillazione atriale;
- **VT** - Ventricular tachycardia - Bool:
  - 0 il paziente non ha una tachicardia ventricolare;
  - 1 il paziente ha una tachicardia ventricolare;
- **PSVT** - Paroxysmal supra ventricular tachycardia - Bool:
  - 0 il paziente non ha una tachicardia parossissima sporaventricolare;
  - 1 il paziente ha una tachicardia parossissima sopraventricolare;
- **CONGENITAL** - Congenital heart disease - Bool:
  - 0 il paziente non ha una cardiopatia congenita;
  - 1 il paziente ha una cardiopatia congenita;
- **UTI** - Urinary tract infection - Bool:

  - 0 il paziente non ha un'infezione del tratto urinario;
  - 1 il paziente ha un'infezione del tratto urinario;

- **NEURO CARDIOGENIC SYNCOPE** - Neuro cardiogenic syncope - Bool:
  - 0 il paziente non ha una sincope neurocardiogenica;
  - 1 il paziente ha una sincope neurocardiogenica;
- **ORTHOSTATIC** - Orthostatic - Bool:
  - 0 il paziente non soffre di ipotenzione ortostatica;
  - 1 il paziente soffre di ipotenzione ortostatica;
- **INFECTIVE ENDOCARDITIS** - Infective endocarditis - Bool:
  - 0 il paziente non ha una endocardite infettiva;
  - 1 il paziente ha una endocardite infettiva;
- **DVT** - Deep venous thrombosis - Bool:
  - 0 il paziente non ha una trombosi venosa profonda;
  - 1 il paziente ha una trombosi venosa profonda;
- **CARDIOGENIC SHOCK** - Cardiogenic shock* - Bool:
  - 0 il paziente non ha avuto uno shock cardiogenico;
  - 1 il paziente ha avuto uno shock cardiogenico;
- **SHOCK** - Shock - Bool:
  - 0 il paziente non ha avuto uno shock;
  - 1 il paziente ha avuto uno shock;
- **PULMONARY EMBOLISM** - Pulmonary shock - Bool:
  - 0 il paziente non ha avuto uno shock polmonare;
  - 1 il paziente ha avuto uno shock polmonare;

- **CHEST INFECTION** - Chest infection - Bool:
  - 0 il paziente non ha un'infezione toracica;
  - 1 il paziente ha un'infezione toracica;

Dopo aver catalogato la vasta mole di informazioni presenti, si è provveduto a eseguire le fase di Data Processing.

2.1.2 Data Processing

La parte di Data Processing, ha al suo interno diverse operazioni, nel caso da me affrontato mi sono soffermato sulla parte di Data Cleaning. Analizzando il dataset, si è riscontrata la presenza di alcuni parametri che presentano delle istanze vuote/nulle, e sono:

- BNP, peptide natriuretico cerebrale (4% di istanze vuote, 54% di istanze nulle)
- EF, frazione di eiezione (10% di istanze nulle e 6 istanze vuote)
- Glucosio (5% di istanze nulle e 12 istanze vuote)
- Piastrine (2% di istanze nulle e 1 istanza vuota)

Si è deciso di tenere conto solamente dei record che presentano un numero inferiore o pari a due parametri inerenti alle analisi svolte durante l'ospedalizzazione, con valori uguali null/empty. Inoltre avendo notato la presenza di molte istanze con BNP e EF uguali a null, sarebbe opportuno capire quali azioni intraprendere, in quanto questi due parametri rappresentano un indicatore per una possibile insufficienza cardiaca. Si potrebbe pensare ad una rimozione di tali dati ma questa ipotesi è stata scartata in quanto, qualora si scartassero tutti i record che presentano questa caratteristica, si otterrebbe un dataset di dimensione dimezzata (dal 54% al 64% in meno). Si potrebbe ovviare a questo problema effettuando un'analisi statistica sui valori di BNP ed EF in base ai restanti valori delle analisi. Tuttavia, in questo caso, sarebbe necessario il supporto di uno specialista che indichi quali sono le regole per poter stabilire un valore approssimato. Si potrebbe pensare di applicare tecniche di Data Imputation, ma queste ovviamente non garantirebbero eterogeneità. Oppure utilizzare degli

algoritmi di machine learning, come i regressori, per stimare i valori mancanti, ma questo implicherebbe un investimento di tempo molto importante. Pertanto, si è deciso di non intervenire sui record che presentino valori null/empty sui parametri BNP ed EF.

2.2 Architettura Proposta

Per far si che l'applicazione risulti modulare, scalabile e per favorire lo sviluppo a microservizi si è deciso di modellare l'infrastruttura utilizzando Docker.

Docker

Docker è una piattaforma open-source per lo sviluppo, la distribuzione e l'esecuzione di varie applicazioni in modo rapido. Docker utilizza funzionalità di isolamento delle risorse come cgroup e kernel namespace per consentire l'esecuzione di "container" indipendenti all'interno di una singola istanza Linux, evitando il sovraccarico derivante dell'avvio di varie macchine virtuali. Docker fornisce un modo facile per eseguire quasi tutte le applicazioni, in modo sicuro e in un container. Più conteiner possono condividere lo stesso kernel, ma ciascun container può essere vincolato al utilizzo solo di una determinata quantità di risorse hardware disponibili dalla macchina host[9].

Il risultato finale è un container per ogni modulo, strutturato come un docker-compose(Figura 2.1).

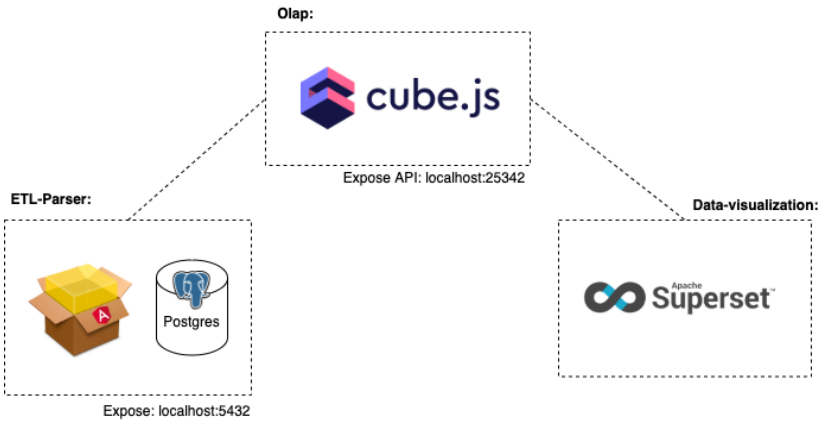

Figura 2.1: Architettura della soluzione

Per permettere la comunicazione tra i vari container, va creata una rete bridge attraverso l'apposito comando:

```
docker network create nomeRete
```

Si può controllare se la rete è stata effettivamente creata attraverso l'apposito comando:

```
docker network ls
```

Per facilitare lo start-up e shut-down, sono stati creati 2 script bash che inizializzano il docker-compose:

- start.sh
- stop.sh

Ricordiamo che per lanciare i comandi bash, vanno eseguiti nel terminale con l'apposito comando:

```
sudo bash nomescript.sh
```

2.2.1 ETL

Il modulo prevede una parte di Data Mining e pulizia del dataset, andando a effettuare tutti i processi indicati nella sezione di Data Ingestion e Data Processing. La selezione dei dei dati da fonti aperte avverrà controllando in prima istanza, il formato del file che verrà caricato, il quale se non sarà conforme (.csv) verrà automaticamente scartato. In seguito, verrà controllato se il file csv presenta le colonne necessarie per poter normalizzare le tuple in oggetti; qualora non fosse possibile, il file verrà scartato. Se i dati saranno correttamente formattati, avverrà il caricamento nel database attraverso JPA e verrà salvata una copia del file sul server in una cartella separata, seguendo il seguente formato: "./FileUploadX/namefile" dove X rappresenta il numero di file caricati e "namefile" il nome del file. Il salvataggio dei dati nel database relazione avverrà seguendo il seguente schema ERD (Figura 2.2), che è stato ottenuto dopo aver analizzato i dati. Questa struttura consentirà in seguito la corretta trasformazione dei dati in cubi. Per eseguire una corretta schematizzazione delle entità, si è deciso di creare delle classi Enumerated specifiche, ovvero:

- Gender:
  - Male
  - Female
  - Other
- Residenza:
  - Urban
  - Rural
- TOA:
  - OPD
  - Emergency
- Outcame:
  - DAMA
  - DISHARGE
  - EXPIRY

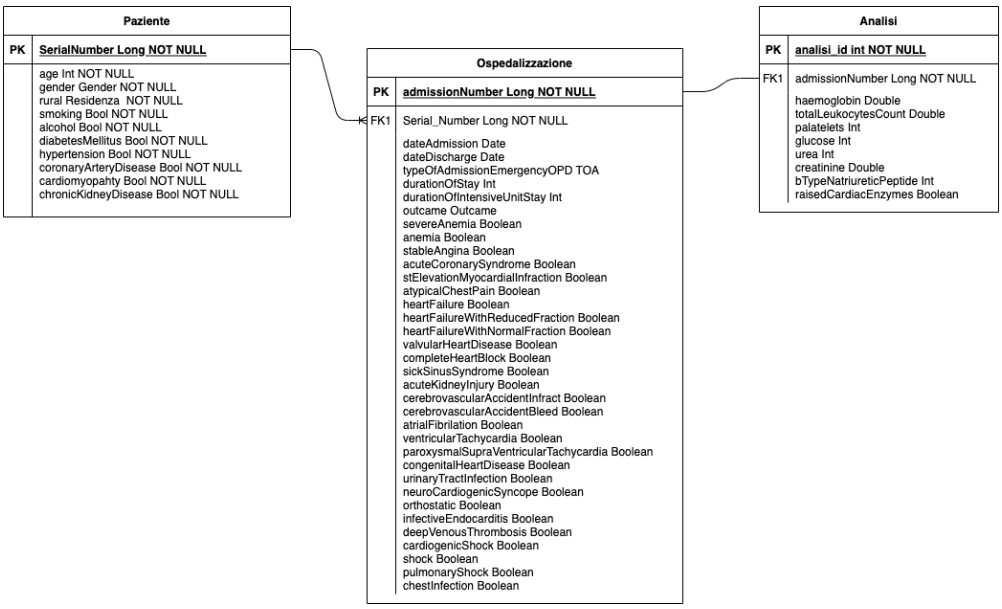

Figura 2.2: Schema ERD

Tutto ciò verrà offerto da un metodo REST esposto in localHost all'indirizzo:
"http://localhost:8080/parser/uploadCVS".
Il modulo non prevede una parte front-end, quindi il testing blackbox è avvenuto attraverso Swagger all'indirizzo
"http://localhost:8080/swagger-ui/index.html".
I test di unità e di integrazione sono stati svolti con l'utilizzo di JUnit e Mockito, e sono disponibili in
"/test/java/com.example.etlparser/".
La struttura del progetto è organizzata seguendo il formato aziendale (Figure 2.3).

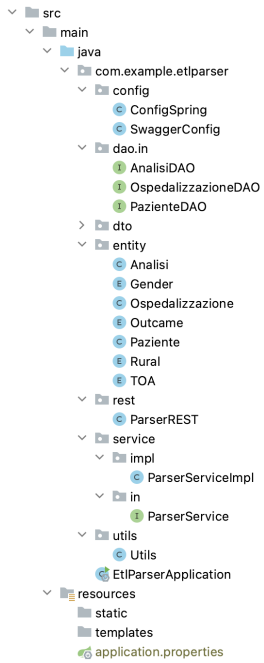

Figura 2.3: Struttura softwere

2.2.2 OLAP

Per implementare l'obiettivo del caso di studio, avevo bisogno di un framework che mi permettesse di aggiungere tutti i vantaggi derivanti dall'utilizzo dell'analisi multidimensionale. Dopo una lunga e accurata ricerca, sono state individuate 2 alternative:

- Apache kylin
- Cube.js

2.2.2.1 Apache kylin

Apache Kylin è un motore OLAP avanzato che garantisce prestazioni eccezionali su grandi volumi di dati. La versione 5.0 introduce importanti miglioramenti, tra cui un'architettura Cloud Native e una nuova interfaccia utente, rendendo Kylin una soluzione completa e all'avanguardia per le esigenze analitiche delle organizzazioni nel contesto dei Big Data. Il progetto inizialmente prevedeva l'utilizzo di Apache Kylin, il quale richiedeva molti prerequisiti:

- Software
  - Hadoop: cdh5.x, cdh6.x, hdp2.x, EMR5.x, EMR6.x, HDI4.x
  - Hive: 0.13 - 1.2.1+
  - Spark: 2.4.7/3.1.1
  - Mysql: 5.1.7 and above
  - JDK: 1.8+
  - OS: Linux only, CentOS 6.5+ or Ubuntu 16.0.4+ [10]
- Hardware
  - The minimum configuration of a server running Kylin is 4 core CPU, 16 GB RAM and 100 GB disk.
  - For high-load scenarios, a 24-core CPU, 64 GB RAM or higher is recommended.[10]

Dopo aver seguito la guida presente nella documentazione (`https://kylin.apache.org/5.0/docs/quickstart/deploy_kylin#download-and-install`), non è stato possibile effettuare la configurazione in locale, in quanto, dopo aver scaricato, installato e configurato tutto il necessario il comando "hadoop fs" produceva un errore. Si è pensato, quindi, di utilizzare Kylin in una configurazione docker-compose, seguendo la guida ufficiale: `https://hub.docker.com/r/apachekylin/apache-kylin-standalone`. La versione Docker di Kylin, invece riusciva ad essere eseguita, ma non configurata e di conseguenza era impossibile configurare il dataset postgress e/o mysql per poter creare i cubi. Considerato ciò si è provveduto alla ricerca di un gestore di cubi equivalente che non modificasse le finalità.

2.2.2.2 Cube.js

Cube.js è un framework open source che semplifica sia la connessione tra silos di dati, la creazione di metriche coerenti e sia l'accessibilità, attraverso API, a diverse applicazioni. Il punto chiave di Cube.js è la possibilità di creare "data-cube" utilizzando un approccio code-first, dove gli analisti definiscono modelli di dati in codice YAML o JavaScript. Questi "cubi" rappresentano entità aziendali e consentono di definire calcoli specifici tramite measures e dimensions. Inoltre, Cube.js supporta la creazione di view, semplificando l'interazione con il modello di dati. Questo approccio agile e code-first, facilita la gestione dei modelli di dati attraverso il controllo di versione, favorendo la collaborazione e la manutenibilità. Cube.js è stato adottato con successo da aziende come Cloud Academy, Cyndx e Cuboh, portando le applicazioni ad uno sviluppo più rapido, tempi ridotti di inattività dell'analisi e prestazioni ottimizzate. Cube.js mette a disposizione sia cube che le view per organizzare i dati:

- Cube: Rappresenta un set di dati multidimensionale. Può essere pensato come una raccolta di measure e dimension che possono essere esplorate e analizzate. Un "cubo" definisce quali dati sono disponibili per l'analisi e come possono essere aggregati in diverse dimensioni.

- View: È una rappresentazione specifica dei dati in un "cubo". Può essere considerata come una proiezione dei dati del cubo che specifica quali colonne e aggregazioni sono disponibili. Le viste consentono di definire insiemi specifici di dati e calcoli che possono essere utilizzati per rispondere a domande analitiche specifiche.

In breve, mentre il "cubo" rappresenta l'insieme complessivo di dati multidimensionali, la view è una rappresentazione di una specifica sezione di quei dati, progettata per rispondere a esigenze analitiche particolari. Entrambi giocano un ruolo chiave nell'organizzazione e nell'esposizione dei dati in Cube.js.
Di seguito riporto il lavoro effettuato per la creazione dei "cubi" e view.

Cubi:

Dopo aver inizializzato l'ambiente, si è provveduto alla creazione dei tre "cubi", sulla base degli oggetti salvati nel DB, che di seguito verranno riportati.

- Paziente.yml
- Ospedalizzazione.yml
- Analisi.yml

View:

Per una corretta esposizione dei dati, è stata creata un view da esporre a Superset. Il grafico seguente (Figura 2.4), può essere esplicativo per capire la struttura del modulo.

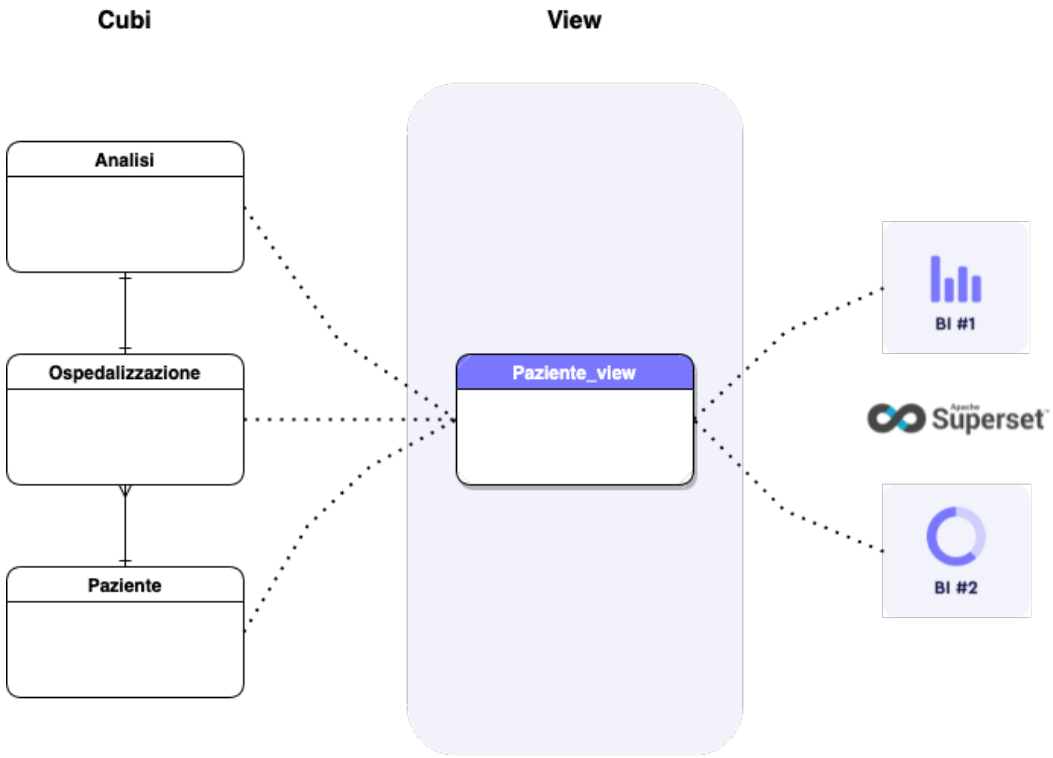

Figura 2.4: Struttura cubi

2.2.3 Superset

Per la parte finale del progetto, dedicata alla Data Visualization, si è deciso di utilizzare Apache Superset, una piattaforma open-source di Data Visualization e business intelligence progettata per semplificare l'analisi e la visualizzazione dei dati. Creato da Airbnb e successivamente donato alla Apache Software Foundation, Superset offre un'ampia gamma di funzionalità per esplorare, analizzare e condividere i dati in modo interattivo. Ecco alcuni punti chiave sull'Apache Superset che hanno portato a sceglierlo:

- Connessioni dati versatili: Superset può connettersi a una varietà di fonti dati, tra cui database SQL, PostgreSQL, Data Warehouse e fonti di dati come Apache Druid, Google Sheets e Cube.js. Questa flessibilità consente agli utenti di accedere a dati provenienti da diverse fonti, in un unico ambiente.

- Estensibilità: Essendo un progetto open source, Superset è altamente estensibile e offre la possibilità di integrare nuove funzionalità e connessioni dati. La comunità attiva di sviluppatori contribuisce continuamente al miglioramento e all'espansione delle capacità di Superset.

Complessivamente, Apache Superset è una soluzione potente per le esigenze di Data Visualization e business intelligence, offrendo un modo flessibile, intuitivo e collaborativo per esplorare e comunicare i dati aziendali. Dopo la configurazione, è stato possibile effettuare le query in maniera precisa potendo organizzare tutte le analisi in una dashboard consultabile in real time. Di seguito, viene riportato il risultato finale (Figura 2.5).

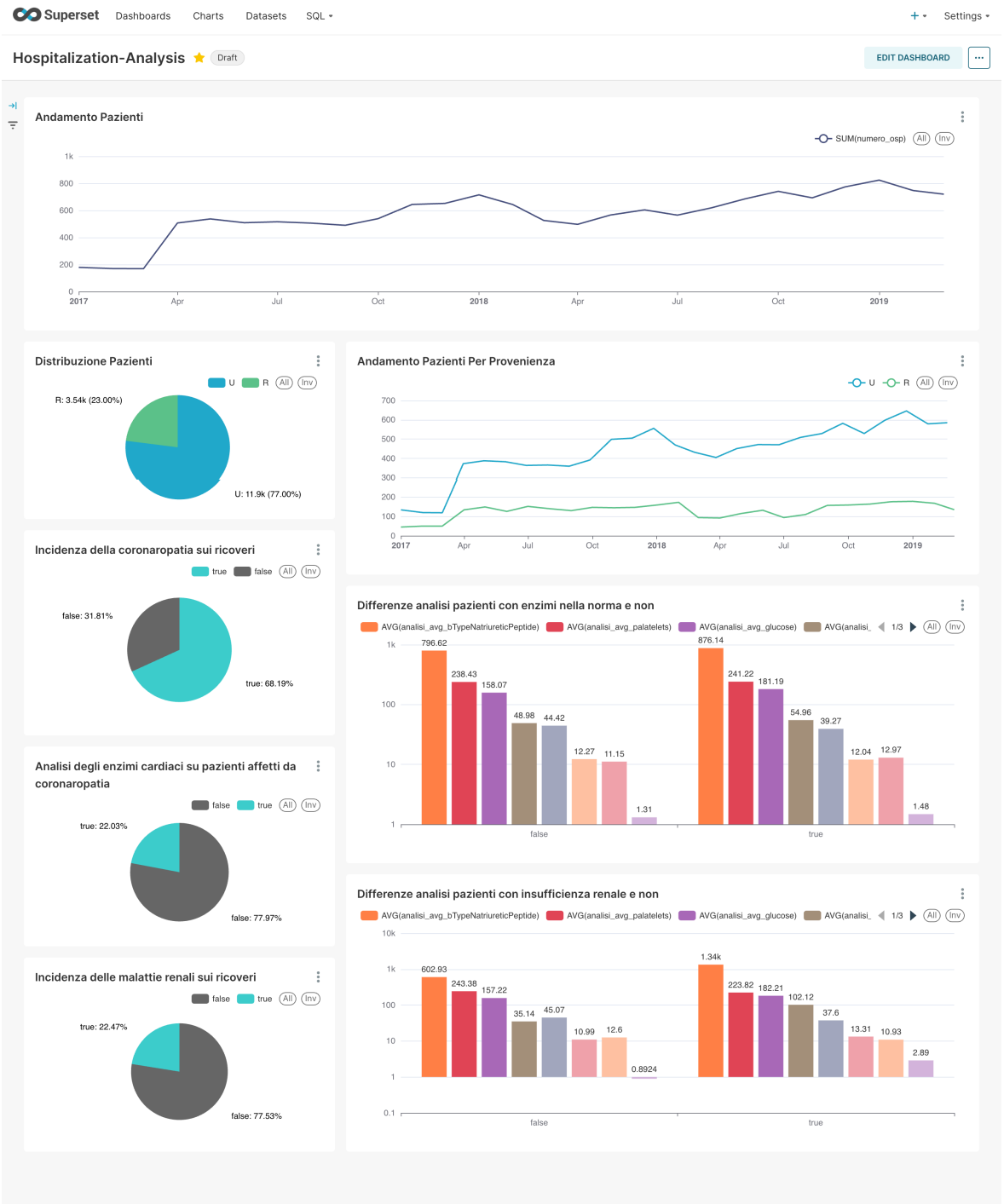

Figura 2.5: Dashboard Superset

CAPITOLO 3

Casi d'uso e Risultati

Di seguito, verranno riportati i casi d'uso che sono stati sviluppati e trasformati in query effettuate sui dati:

- Contare il numero di ospedalizzazioni per ciascun mese per i due anni dello studio
- Analizzare la provenienza geografica dei pazienti, ovvero Rural o Urban
  - Analizzare se ci siano dei trend in relazione alla provenienza
- Analizzare le patologie che colpiscono di più le persone, come
  - Coronaropatia [68.18%]
  - Ipertensione [48.59%]
  - Sindrome coronarica acuta [36.57%]
  - Insufficienza cardiaca [28.95%]
  - Insufficienza renale acuta [22.47%]

Primo caso d'uso

L'obiettivo del caso d'uso è capire se durante l'arco di due anni, il numero dei casi di ospedalizzazione è aumentato.

Secondo caso d'uso

L'obiettivo del caso d'uso è capire se l'incidenza di ospedalizzazione aumenta in relazione all'area di provenienza del paziente.

Terzo caso d'uso

L'idea era quella di soffermarsi su due patologie che, in ambiti diversi, coprono il maggior spettro dei ricoverati, sui quali si hanno dati che permettano di effettuare analisi, ovvero:

- Coronaropatia
- Insufficienza renale acuta

Inizialmente si volevano valutare anche i pazienti affetti da ipertensione, ma bisogna notare che il parametro fondamentale sull'analisi dell'ipertensione, è la pressione arteriosa, misurabile con lo sfigmomanometro, dato che però non è stato riportato nel dataset.

3.1 Risultati e Discussioni

Dopo aver configurato tutto il necessario, tra Postgress, Cube.js e Superset; si è riusciti a creare una dashboard per poter ricavare tutte le query dimensionali per assolvere ai vari caso d'uso. Analizziamo ora caso per caso i risultati ottenuti:

3.1.1 Primo caso d'uso

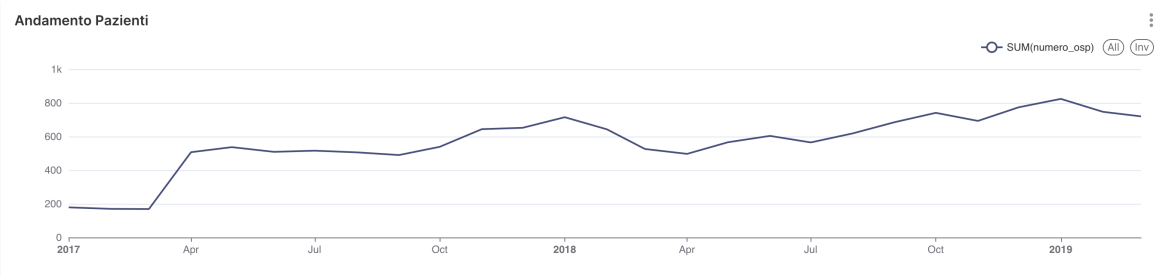

Figura 3.1: Andamento dei ricoveri tra 2017 e 2019

Il grafico (Figura 3.1) sembra confermare che al passare del tempo il numero di ospedalizzazione aumenti. Dopo aver visto questo trend, si è pensato di analizzare quale fosse la provenienza degli ospedalizzati.

3.1.2 Secondo caso d'uso

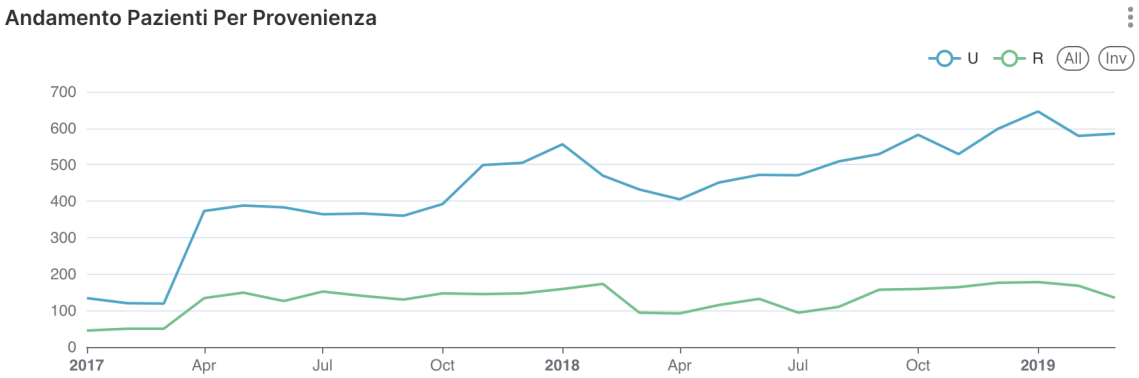

Figura 3.2: Andamento dei ricoveri suddivisi per provenienza tra 2017 e 2019

I dati riportati in Figura 3.2 sembrano mostrare che, per i contesti urbani, i casi di ospedalizzazione aumentino di anno in anno. D'altra parte, le ospedalizzazioni delle

persone provenienti dai contesti rurali, sembrano rimanere costati nel tempo. Quindi ci si presentano due scenari:

- Le persone che abitano in contesti rurali, sono più sane e tendono a necessitare meno degli ospedali.
- Le persone che abitano in contesti rurali, in quanto risedenti in aree periferiche, hanno meno accesso agli ospedali, di conseguenza non riescono ad accedere alle cure.

3.1.3 Terzo caso d'uso

Per capire su quale delle varie malattie soffermarmi, ho tenuto conto non solo dell'incidenza ma anche delle presenza di altri valori che potessero essere messi in correlazione a tali malattie. Quindi, tenendo conto di ciò, si sono prese in considerazione i pazienti affetti da coronaropatia e quelli affetti da insufficienza renale, che ricoprono rispettivamente la seguente incidenza sulle ospedalizzazioni (Figura 3.3 e 3.4):

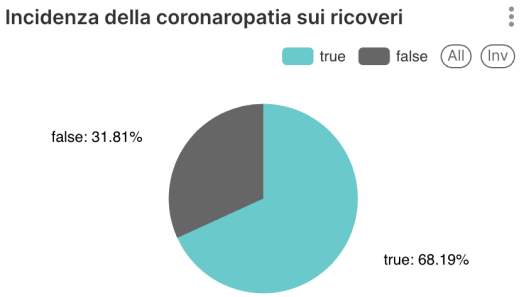

Figura 3.3: Incidenza "coronaropatia"

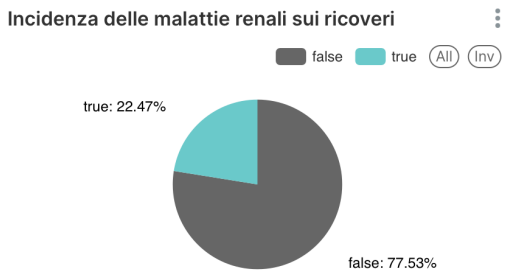

Figura 3.4: Incidenza "insufficienze renali"

La coronaropatia affligge il 68.18% dei pazienti, mentre l'insufficienza renale ricopre il 22.47% dei pazienti.

Coronaropatia:

Durante il trattamento della coronaropatia, è fondamentale avere valori di "Reised cardiac enzymis" nella norma, ma molto spesso questi nelle persone affette da coronaropatia non risultano essere nella norma. Verifichiamo.

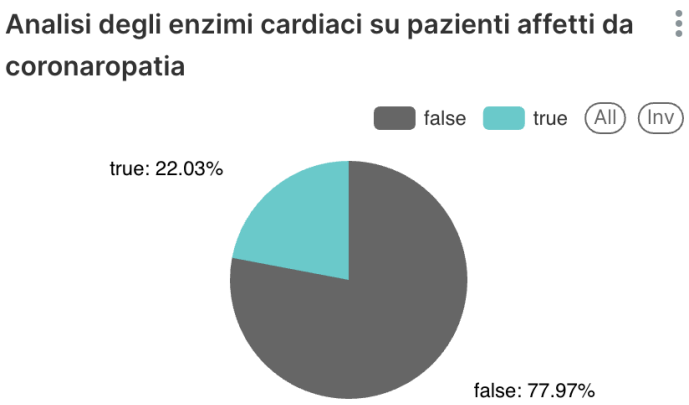

Figura 3.5: Analisi enzimi cardiaci nei pazienti affetti da coronaropatia

Si può vedere (Figura 3.5) come, chi non ha una giusta quantità di questi enzimi, risulti essere il 77.97% del totale di chi è afflitto da tale patologia; alla luce di questo dato, ho effettuato un confronto medio dei valori delle analisi (Figura 3.6):

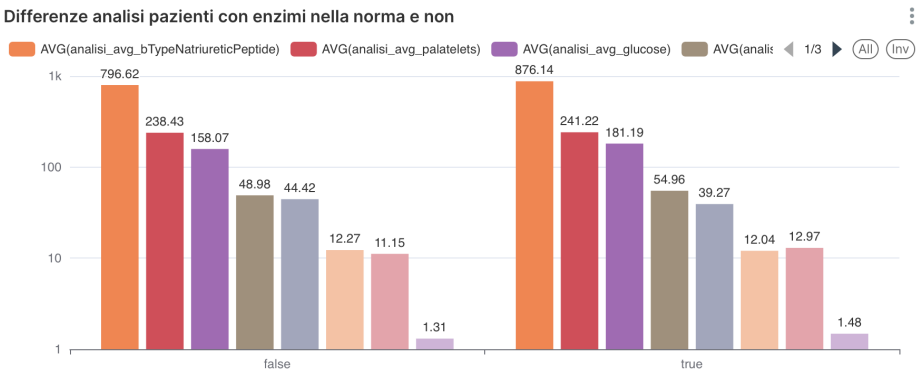

Figura 3.6: Comparazione analisi tra pazienti con enzimi cardiaci nella norma e non

Confrontando le analisi dei pazienti affetti da coronaropatia, correlandole agli enzimi cardiaci, riscontriamo delle differenze tra chi ha gli enzimi nella norma e chi no, ovvero: si è riscontrando che, chi soffre di coronaropatia e simultaneamente ha valori "raised cardiac enzymis" non nella norma, riporta valori inferiori per piastrine, glucosio, urea, creatinina, b type natruretic peptite e total leukocytes cont; mentre riporta valori superiori per ejection fraction e emoglobina. Non avendo conoscenze mediche e/o supporto specializzato, non posso determinare se questi risultati costituiscano un trend o meno.

Insufficienza renale acuta

Come noto in letteratura, i pazienti che soffrono di insufficienza renale acuta, hanno

valori per la creatinina superiore alla norma, quindi si è pensato di verificare tale trend, e perciò di confrontare i valori medi delle analisi fra chi soffre di insufficienza renale rispetto un paziente sano (Figura 3.7).

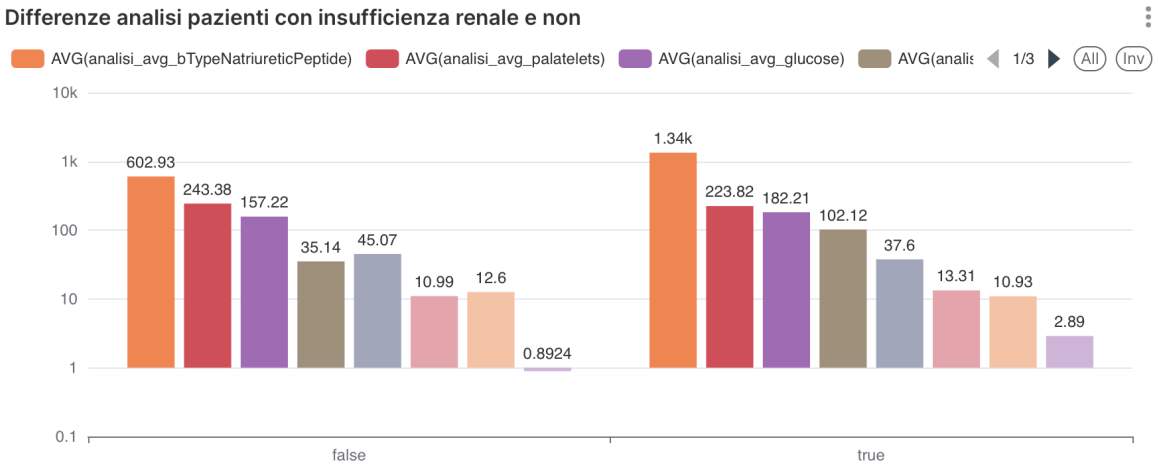

Figura 3.7: Comparazione analisi tra pazienti con insufficienza renale e non

Si può notare come chi soffre di insufficienza renale, ha valori superiori di glucosio, urea, b type natruretic peptite [222%] e creatinine [300%]; mentre valori inferiori in piastrine e ejection infraction. Nonostante la mancanza di competenze in ambito medico, possiamo però affermare che esiste una forte discrepanza tra le due categorie, di conseguenza, avremo quasi certamente delle motivazioni mediche dietro tale situazione.

Conclusioni e sviluppi futuri

Il mio lavoro di tesi si è concentrato sullo sviluppo di componenti software per facilitare la visualizzazione e l'analisi multidimensionale dei dati, consentendo di agire secondo ciò che i dati ci suggeriscono. In ambito sanitario potrebbe essere essenziale per la prevenzione e/o l'approvvigionamento di risorse, inoltre in accoppiata con i più recenti algoritmi di machine learng potrebbe essere ancor di più importante nell'individuare fattori di rischio a noi non visibili. Nel corso del mio lavoro, ho capito ed esplorato sia l'architettura dei sistemi per l'analisi dei Big Data, soffermandomi sulla la pulizia e la normalizzazione dei dati, oltre a tutte le altre fasi necessarie per renderli utilizzabili per l'analisi, che affrontato le sfide legate all'acquisizione di tale mole di dati. Spero che questa ricerca possa contribuire a comprendere l'importanza dell'analisi dei dati, sopratutto quelli ospedalieri e costituire un punto di partenza per migliorare nel campo medico-informatico.

Vista l'architettura della soluzione proposta, sono molteplici i miglioramenti applicabili, data la presenza di livelli multipli. Per la parte di Data Visualization reputo corretta la scelta di Apache Superset, si potrebbe scegliere un altro competitor, ma le soluzioni open source sono preferibili in ambito di ricerca. Possiamo sicuramente migliorare nei primi 2 livelli:

- ETL - Parser
- Cube.js

Per il primo livello, ovvero l'ETL - Parser, possiamo pensare di migliorare tutto il processo di Data Ingestion e di Data Processing. In primis, possiamo iniziare a migliorare come preleviamo il dato, attualmente le fonti che il parser preleva sono solo strutturate ed in un determinato formato. Potremmo aggiungere la lettura da fonti non strutturate, per fare ciò andrebbe aggiunto un database non relazionale, come potrebbe essere MongoDB, e aggiungere un layer che gestisca autonomamente questa fase, come Apache NiFi.

Per il secondo livello, attualmente le fasi di creazione dei cubi OLAP è fatta da un data analyst, che con la sua conoscenza del dominio capisce come strutturare questi cubi per massimizzare l'efficienza di tali strutture dati. Automatizzare questo processo, permetterebbe di facilitare ancora di più l'utilizzo di queste tipologia di strumenti, ma qualora non fosse fatta in maniera efficace verrebbe meno il senso di struttura in modo tridimensionale l'informazione.

Bibliografia

[1] Sahithreddy, last update: 17/09/2020. [Online]. Available: https://sahithreddy639.medium.com/what-is-big-data-and-how-facebook-is-using-big-data-d044fcd52948 (Citato a pagina 3)

[2] X. D. C. L. J. L. S. Z. X. Z. Jinchuan CHEN, Yueguo CHEN, "Big data challenge: a data management perspective," *Springer*, vol. 7, pp. 157–164, 2013. [Online]. Available: https://doi.org/10.1007/s11704-013-3903-7 (Citato alle pagine 4 e 5)

[3] K. T. . W. R. Herland, M., "A review of data mining using big data in health informatics," *Springer*, 2014. [Online]. Available: https://doi.org/10.1186/2196-1115-1-2 (Citato a pagina 5)

[4] D. Q. L. L. H. H. Greg Boss, Padma Malladi, "Cloud computing," *IBM white paper 321*, 2007. (Citato alle pagine 11 e 12)

[5] Z. Tejada, last update: 03/01/2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/databases/guide/big-data-architectures#lambda-architecture (Citato alle pagine 14, 16, 17 e 18)

[6] T. Hlupić and J. Puniš, "An overview of current trends in data ingestion and integration," in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 2021, pp. 1265–1270. (Citato a pagina 19)

[7] A. A. Hassan and T. M. Hassan, "Real-time big data analytics for data stream challenges: An overview," *European Journal of Information Technologies and Computer Science*, vol. 2, no. 4, p. 1–6, Jul. 2022. [Online]. Available: https://www.ej-compute.org/index.php/compute/article/view/62 (Citato a pagina 20)

[8] H. S. Aung and Y. K. Latt, "Data warehouse for student information system using star and snowflake schema," Ph.D. dissertation, MERAL Portal. (Citato alle pagine 21 e 22)

[9] P. E. N, F. J. P. Mulerickal, B. Paul, and Y. Sastri, "Evaluation of docker containers based on hardware utilization," in *2015 International Conference on Control Communication Computing India (ICCC)*, 2015, pp. 697–700. (Citato a pagina 32)

[10] A. Kylin, last update: 29/01/2024. [Online]. Available: https://kylin.apache.org/docs/install/index.html (Citato a pagina 36)