Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Corso di Laurea in Informatica per il Management

Architettura unificata per la Data Quality in una Data Platform: un'approccio configurabile per la gestione delle anomalie

Relatore:
Prof. Marco Di Felice

Correlatore:
Dott. Saverio Piermattei

Presentata da:
Michele Abruzzese

Anno Accademico 2023/2024

Indice

Abstract

L'obiettivo di questa tesi è la progettazione e l'implementazione di un sistema di controllo della qualità dei dati in un contesto di Data Analytics aziendale, utilizzando una piattaforma basata su Microsoft Azure. Il sistema è stato sviluppato per garantire un monitoraggio costante e automatizzato dei dati, rilevando anomalie e inviando notifiche tempestive agli amministratori del sistema, assicurando che i processi aziendali non vengano interrotti. La soluzione proposta è caratterizzata da un comportamento fail-safe, che consente di mantenere l'integrità del flusso di dati anche in presenza di errori.

Il sistema implementato è altamente configurabile, permettendo l'adattamento a diverse fonti di dati e scenari operativi senza necessità di interventi manuali. Viene gestito attraverso una serie di pipeline automatizzate e controlli centralizzati che possono essere personalizzati a seconda delle esigenze aziendali. Il progetto ha dimostrato di rispondere efficacemente agli obiettivi di miglioramento della governance dei dati, riducendo il rischio di decisioni aziendali basate su dati errati.

Listing

Elenco Figure

Capitolo 1

Introduzione

La qualità dei dati (Data Quality) rappresenta un aspetto fondamentale nelle moderne architetture di gestione e analisi dei dati industriali. In un contesto sempre più digitale e con volumi di dati in continua espansione, le aziende si trovano a dover garantire che i dati utilizzati nei processi decisionali siano accurati, completi e consistenti. Nell'ambito di questa soluzione un'**anomalia** del dato può essere definita come un'informazione che non rispetta le regole o i requisiti predefiniti del sistema, rendendo i dati potenzialmente inutilizzabili o inaccurati per le operazioni successive. Questi dati possono non essere conformi per vari motivi, come la presenza di valori errati, formati non corretti o violazione di vincoli di integrità. Un'anomalia può quindi causare errori nell'elaborazione, nelle analisi o nel decision-making aziendale.

La Data Quality è diventata un concetto chiave, poiché la presenza di dati errati o incompleti può influire negativamente su decisioni aziendali critiche, portando a inefficienze operative, riduzione della produttività o, peggio, a scelte strategiche sbagliate. Negli ultimi anni, grazie alla diffusione delle piattaforme cloud e dei sistemi di gestione dati distribuiti, sono stati sviluppati strumenti e tecniche per monitorare e migliorare la qualità dei dati attraverso sistemi di controllo, automazione e notifiche automatiche

delle anomalie.

In particolare, nel settore industriale, la qualità dei dati gioca un ruolo cruciale, in quanto la precisione e l'affidabilità dei dati possono influenzare non solo le decisioni operative, ma anche la sicurezza dei processi produttivi, la gestione delle risorse e il controllo delle performance delle infrastrutture. L'implementazione di controlli automatizzati e configurabili sui dati, inseriti in architetture cloud moderne come quelle basate su Microsoft Azure, rappresenta una soluzione strategica per affrontare queste sfide e garantire un monitoraggio continuo della qualità dei dati.

Questa tesi, frutto di un progetto aziendale sviluppato nel periodo di tirocinio presso l'azienda **Iconsulting**, si propone di rispondere a una delle sfide più rilevanti nel contesto delle infrastrutture di Data Analytics industriali: garantire la qualità dei dati attraverso un sistema automatizzato e configurabile che consenta di rilevare, gestire e notificare eventuali anomalie nei dati. Le aziende che operano su grandi volumi di dati, specialmente in contesti industriali, hanno bisogno di strumenti che non solo garantiscano la qualità del dato ma che lo facciano in maniera scalabile e automatizzata, riducendo al minimo gli interventi manuali e gli errori umani.

Gli obiettivi principali di questo lavoro sono i seguenti:

- **Automatizzazione del controllo della qualità dei dati:** L'obiettivo centrale è sviluppare un sistema che automatizzi il processo di verifica della qualità dei dati, riducendo il carico manuale su amministratori e data analyst. Grazie alla configurabilità del sistema, le regole di controllo possono essere facilmente aggiornate e applicate a diversi flussi di dati senza la necessità di interventi tecnici significativi.

- **Segnalazione delle anomalie:** Il sistema si propone di rilevare le anomalie nei dati e, in caso di errori, di inviare notifiche in tempo reale ai team responsabili. Questo approccio consente di ridurre i tempi di reazione e intervento in caso di

dati non conformi, evitando che gli errori si propaghino nei successivi processi di elaborazione o analisi.

- **Integrazione con flussi di dati esistenti:** Un altro obiettivo chiave è garantire che il sistema di controllo della qualità dei dati possa integrarsi agevolmente con le pipeline di dati già esistenti all'interno dell'infrastruttura aziendale, senza necessità di ristrutturare l'intero flusso di lavoro. Questa flessibilità consente una più rapida implementazione e una riduzione dei costi operativi.

- **Configurabilità e adattabilità:** Il sistema è progettato per essere configurabile, permettendo l'adattamento dei controlli a diverse esigenze aziendali o a nuove fonti di dati, senza richiedere modifiche al codice sottostante. Questa caratteristica rispecchia un sistema capace di adattarsi a un ambiente tecnologico in continua evoluzione.

- **Riduzione dei rischi operativi e miglioramento della governance:** Infine, il sistema contribuisce a migliorare la governance dei dati, riducendo il rischio che dati errati o incompleti possano influire negativamente sulle decisioni aziendali. Implementando un monitoraggio costante e controlli automatici, le organizzazioni possono garantire che solo dati di alta qualità vengano utilizzati per analisi e processi decisionali.

In sintesi, il sistema di Data Quality fail-safe descritto in questa tesi fornisce una soluzione pratica e innovativa per affrontare le sfide della qualità dei dati industriali.

Per quanto riguarda la struttura della tesi, essa è stata organizzata per seguire un percorso logico e lineare che accompagna il lettore dalla teoria alla pratica, descrivendo dapprima il contesto generale della qualità dei dati e poi scendendo nei dettagli dell'implementazione del sistema proposto. Nel Capitolo 2, viene esplorato lo stato dell'arte delle soluzioni esistenti per il controllo della qualità dei dati e il contesto nel quale si

va ad operare. Questo capitolo si conclude con una panoramica delle piattaforme di dati come Microsoft Azure, che giocano un ruolo cruciale nell'implementazione di sistemi di monitoraggio e controllo dei dati. Il Capitolo 3 si concentra sulla progettazione del sistema di Data Quality proposto in questa tesi. Vengono descritti i requisiti funzionali e non funzionali del sistema, seguiti dall'architettura del sistema stesso, che integra componenti come Azure Dedicated SQL Pool, Azure Data Factory, e Microsoft Power BI. Viene spiegato come questi strumenti interagiscono tra loro per garantire un monitoraggio costante della qualità dei dati, con un'attenzione particolare alla configurabilità e all'automazione dei controlli. Le scelte progettuali vengono giustificate in funzione degli obiettivi prefissati, come l'adattabilità e la scalabilità del sistema. Nel Capitolo 4, viene descritto il processo di implementazione del sistema di controllo della qualità dei dati. Qui si approfondiscono le tecnologie utilizzate e la struttura dell'implementazione, analizzando le pipeline create in Azure Data Factory per automatizzare i controlli e le notifiche in caso di anomalie. Viene descritta in dettaglio la Store Procedure Comune, che gestisce i controlli dei dati, e vengono presentati frammenti di codice che illustrano concretamente come il sistema è stato implementato. Nel Capitolo 5, viene eseguita la validazione del sistema proposto attraverso casi d'uso specifici e test sperimentali. Dopo aver descritto come il sistema risponde agli obiettivi prefissati, vengono presentati i risultati sperimentali, che mostrano l'efficacia del sistema nel rilevare e gestire le anomalie nei dati. Infine, sono inclusi alcuni screenshot che illustrano visivamente il funzionamento del sistema, mostrando le pipeline configurate, le notifiche automatiche e il monitoraggio delle anomalie, fornendo al lettore una chiara rappresentazione delle implementazioni tecniche. Infine, nel Capitolo 6, vengono riportate le conclusioni della tesi, insieme a possibili sviluppi futuri. Si riflette sui risultati ottenuti e si discutono le potenziali espansioni del sistema per includere ulteriori funzionalità o adattarsi a nuovi contesti aziendali.

Capitolo 2

Stato dell'Arte

Il capitolo affronta una panoramica critica della Data Quality e dei metodi esistenti per la sua gestione, con un focus particolare sugli approcci fail-safe e sulle piattaforme dati moderne. La Data Quality è una disciplina in continua evoluzione, particolarmente rilevante in contesti industriali e aziendali dove la precisione e l'affidabilità dei dati sono essenziali per decisioni operative e strategiche.

La prima parte del capitolo, Fondamenti e concetti della Data Quality, esplora le principali definizioni e dimensioni che caratterizzano la qualità dei dati, come accuratezza, completezza, e consistenza. Viene poi affrontata la questione delle sfide nella gestione dei Big Data, con l'obiettivo di evidenziare come la crescita esponenziale dei dati ponga nuovi problemi legati alla qualità e all'integrità delle informazioni. A conclusione di questa sezione, vengono presentate le tecniche e metodi esistenti per il controllo della qualità dei dati.

La seconda parte del capitolo, Approcci fail-safe nella gestione della Data Quality, introduce il concetto di fail-safe, ovvero quei meccanismi progettati per garantire che eventuali errori nei dati non compromettano l'integrità dei processi aziendali. Viene descritto come questo approccio sia applicato alla gestione della Data Quality e come

possa migliorare la robustezza dei sistemi di controllo. Vengono inoltre presentate alcune implementazioni e best practice esistenti, dimostrando l'efficacia degli approcci fail-safe in contesti pratici.

Infine, il capitolo si conclude con una descrizione delle Data Platform, fondamentali per la gestione e l'integrazione dei dati su larga scala. Questa sezione esplora le definizioni, l'architettura e le caratteristiche chiave delle piattaforme dati. Viene discusso anche il ruolo della Business Intelligence, che si appoggia su piattaforme di dati per fornire insight strategici attraverso l'analisi delle informazioni raccolte. La sezione si chiude con una riflessione su scalabilità e manutenzione di queste piattaforme, elementi cruciali per garantirne l'efficienza nel tempo.

2.1 Fondamenti e Concetti di Data Quality

In questo paragrafo ci proponiamo di esplorare i concetti fondamentali della Data Quality e di fornire una base solida su cui costruire le analisi successive. Includeremo una descrizione delle sue principali dimensioni, le sfide legate alla sua gestione nell'era dei Big Data, e il suo ruolo nel processo decisionale. Saranno inoltre esaminate le tecniche e i metodi più comuni utilizzati per garantire un controllo efficace della qualità dei dati, fornendo un quadro completo delle pratiche più diffuse e delle problematiche da affrontare.

2.1.1 Cos'è la Data Quality

La Data Quality (DQ) rappresenta uno degli aspetti più critici nella gestione dei dati all'interno delle organizzazioni moderne. Comunemente definita come "fitness for use", la DQ indica il grado in cui i dati sono adeguati all'uso per il quale sono destinati. Questa definizione implica che la qualità dei dati non sia un concetto assoluto, bensì relativo,

dipendendo dal contesto specifico e dalle esigenze degli utenti finali. Con l'aumento della centralità dei dati nelle strategie aziendali e la crescita esponenziale dei volumi di dati, la gestione della qualità dei dati ha assunto un ruolo sempre più strategico, influenzando direttamente la capacità di un'organizzazione di competere e innovare nel proprio settore [3].

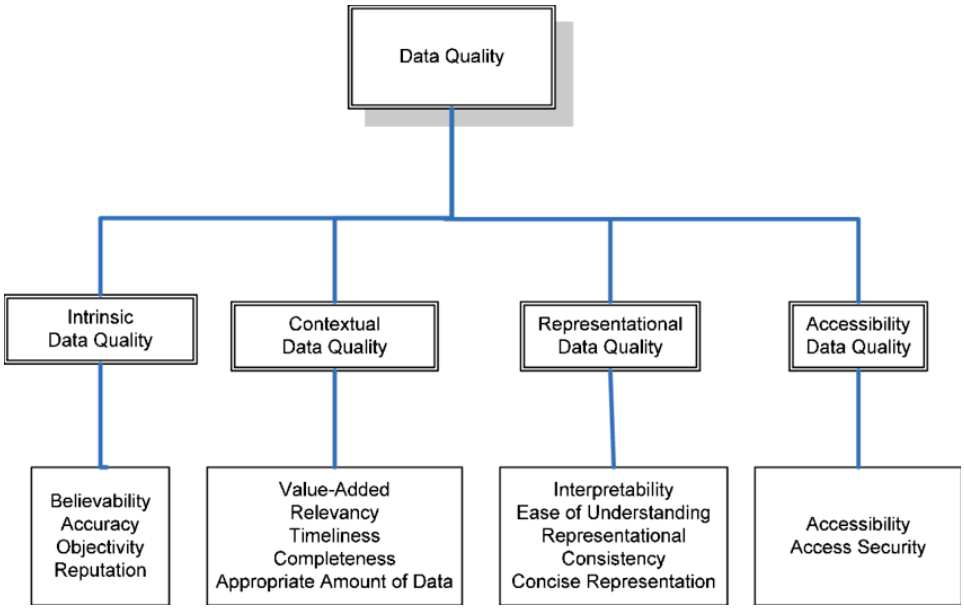

Figura 2.1: Framework concettuale della Data Quality (Wang e Strong 1996) [8]

Uno dei contributi più significativi alla comprensione della DQ è il lavoro di Wang e Strong (1996) [8], che hanno sviluppato un quadro concettuale per organizzare le diverse dimensioni della DQ in base all'importanza percepita dai consumatori di dati. Secondo il loro studio, la DQ non può essere ridotta a semplici criteri di accuratezza, ma deve essere vista come un concetto multidimensionale che include:

- **DQ Intrinseca:** Questa dimensione riguarda la qualità dei dati in sé e per sé, indipendentemente dal contesto in cui vengono utilizzati. Si concentra sulla misura in cui i dati sono corretti, accurati e affidabili. Sottocriteri:
  - Accuratezza: I dati rappresentano correttamente la realtà o i fatti.

  - Credibilità: I dati sono considerati veritieri e affidabili.
  - Obiettività: I dati sono imparziali e non influenzati da giudizi soggettivi.
  - Reputazione: La fonte dei dati è autorevole e rispettabile.

- **DQ Contestuale:** La qualità dei dati deve essere valutata in relazione all'uso specifico, assicurando che siano adeguati al compito previsto. I sottocriteti considerati sono:
  - Valore aggiunto: Capacità dei dati di apportare un contributo significativo alle attività aziendali o ai processi decisionali.
  - Rilevanza: I dati sono pertinenti e utili per le decisioni specifiche che devono essere prese.
  - Tempestività: I dati sono disponibili al momento giusto per il loro utilizzo.
  - Completezza: I dati contengono tutte le informazioni necessarie senza omissioni.
  - Adeguatezza del volume: La quantità di dati è sufficiente per lo scopo previsto.

- **DQ Rappresentazionale:** La chiarezza e la rappresentazione dei dati sono cruciali per assicurare che essi siano facilmente interpretabili dagli utenti. Sono inclusi i seguenti sottocriteri:
  - Interpretabilità: capacità dei dati di essere interpretati correttamente da chi li utilizza. I dati devono essere presentati in modo tale che gli utenti possano comprenderli facilmente e trarre conclusioni accurate.
  - Facilità di comprensione: quanto facilmente gli utenti possono comprendere i dati senza necessitare di sforzi significativi

  - Coerenza di rappresentazione: modo in cui i dati sono presentati uniformemente attraverso diversi sistemi o report. Questo include l'uso costante di formati, codici, e convenzioni
  - Rappresentazione concisa: presentazione dei dati in modo sintetico, evitando ridondanze e presentando solo l'informazione necessaria

- **DQ di Accessibilità:** I dati devono essere accessibili e recuperabili dagli utenti in modo efficace e tempestivo. Sottocriteri:
  - Accessibilità: I dati possono essere recuperati e utilizzati facilmente da chi ne ha bisogno.
  - Sicurezza dell'accesso: L'accesso ai dati è controllato e protetto per garantire che solo gli utenti autorizzati possano accedervi.

Questa visione olistica della DQ ha avuto un impatto significativo sul modo in cui le organizzazioni percepiscono e gestiscono la qualità dei loro dati. Wang e Strong hanno sottolineato che le esigenze dei consumatori di dati dovrebbero guidare le iniziative di miglioramento della DQ, piuttosto che un approccio puramente teorico o tecnico.

Parallelamente, la letteratura sulla DQ identifica ulteriori dimensioni fondamentali per una valutazione completa della qualità dei dati, come la consistenza, la conformità e la tracciabilità. Queste dimensioni non sono isolate, ma interconnesse; migliorare una dimensione senza considerare le altre può non essere sufficiente per garantire un'elevata qualità complessiva dei dati. Ad esempio, dati accurati ma non tempestivi possono risultare inutili se non disponibili quando necessario. [9]

Inoltre, la natura dinamica della DQ richiede un approccio continuo e sistematico alla gestione della qualità. I dati possono deteriorarsi nel tempo a causa di vari fattori, come l'obsolescenza delle informazioni o l'introduzione di nuovi sistemi. Pertanto, la DQ

deve essere monitorata e migliorata costantemente per mantenere la sua adeguatezza e rilevanza nel tempo. [3]

In sintesi, la DQ è un concetto complesso e multidimensionale che richiede un approccio caratterizzato da una visione globale di tutto ciò che è connesso ai dati, in modo tale che i dati stessi, rimangano un asset strategico per le organizzazioni moderne.

2.1.2 Dimensioni della Data Quality

Le dimensioni della Data Quality (DQ) sono essenziali per valutare e migliorare la qualità dei dati all'interno di un'organizzazione. Diversi studi [8][9] chiave nella letteratura hanno identificato una serie di dimensioni fondamentali che sono ampiamente riconosciute e utilizzate per misurare la DQ. Queste dimensioni forniscono un quadro di riferimento per garantire che i dati siano "fit for use" e rispondano alle esigenze operative e strategiche dell'organizzazione.

- **Accuratezza** (Accuracy): Questa dimensione misura il grado in cui i dati riflettono correttamente la realtà o una fonte verificabile. È una delle dimensioni più critiche e spesso la più discussa nella letteratura sulla DQ. La correttezza dei dati è fondamentale per prendere decisioni aziendali informate e ridurre i rischi associati a dati errati o incompleti[9]. La misurazione dell'accuratezza implica il confronto dei dati con una fonte di verità riconosciuta o un benchmark. Questo può includere l'uso di controlli incrociati con dati esterni o verificabili e la validazione da parte di esperti del dominio.

- **Completezza** (Completeness): La completezza si riferisce alla presenza di tutte le informazioni necessarie in un dataset. Dati incompleti possono portare a interpretazioni errate o decisioni non ottimali, compromettendo l'efficacia delle operazioni aziendali. Questa dimensione è spesso valutata in parallelo con l'accuratezza,

poiché entrambe influenzano direttamente l'usabilità dei dati [9]. Per misurare la completezza bisogna controllare un campione di dati per verificare se tutte le informazioni richieste sono state registrate

- **Consistenza** (Consistency): La consistenza riguarda la coerenza dei dati tra diversi sistemi e all'interno dello stesso sistema. Dati inconsistenti possono generare conflitti e discrepanze che minano la fiducia negli stessi. Studi come quello di Wang e Strong (1996) evidenziano l'importanza della consistenza per mantenere l'integrità delle informazioni nel tempo [3]

- **Tempestività** (Timeliness): Questa dimensione valuta quanto i dati sono aggiornati e disponibili al momento giusto. La tempestività è cruciale in contesti in cui le decisioni devono essere prese rapidamente, e la disponibilità di dati obsoleti può avere un impatto negativo sulle operazioni aziendali[9]. Analizzare la tempestività significa monitorare i tempi di aggiornamento dei dati e confrontarli con le esigenze operative.

- **Conformità** (Conformance): La conformità si riferisce all'aderenza dei dati a standard, normative e requisiti predefiniti. Questa dimensione è particolarmente rilevante in settori regolamentati dove la non conformità può portare a sanzioni legali o problemi di governance [3]

- **Tracciabilità** (Traceability): La tracciabilità è la capacità di seguire il percorso dei dati attraverso il loro ciclo di vita, dalla creazione all'uso finale. Questa dimensione supporta la trasparenza e la fiducia nei dati, facilitando l'identificazione di errori e il monitoraggio delle trasformazioni subite dai dati stessi [3]

- **Accessibilità** (Accessibility): Riguarda la facilità con cui i dati possono essere recuperati e utilizzati dagli utenti. È importante che i dati di alta qualità siano

non solo corretti e completi, ma anche prontamente accessibili per coloro che ne hanno bisogno. [9]

Queste dimensioni non sono indipendenti l'una dall'altra, ma sono spesso interconnesse. Ad esempio, dati accurati e completi ma non tempestivi potrebbero comunque essere inutili se non disponibili quando necessario.

2.1.3 Sfide nella Gestione della Data Quality nei Big Data

I Big Data rappresentano un concetto chiave nell'era digitale moderna. Si riferiscono a grandi volumi di dati che vengono generati a una velocità elevata, provenendo da una varietà di fonti e presentando una complessità intrinseca. Questi dati possono includere tutto, dai dati strutturati (come i tradizionali database relazionali), ai dati semi-strutturati (come i file JSON e XML) fino ai dati non strutturati (testi, immagini, video, ecc.). La crescita esponenziale dei dati negli ultimi anni ha portato all'emergere di concetti come i "4V" (Volume, Velocità, Varietà, Veridicità) che descrivono le caratteristiche principali dei Big Data. Il volume si riferisce all'enorme quantità di dati generati, la velocità riguarda la rapidità con cui i dati vengono creati e processati, la varietà si riferisce ai diversi tipi di dati (strutturati, semi-strutturati e non strutturati) che devono essere gestiti, mentre la veridicità si riferisce alla qualità e all'affidabilità dei dati. L'av-

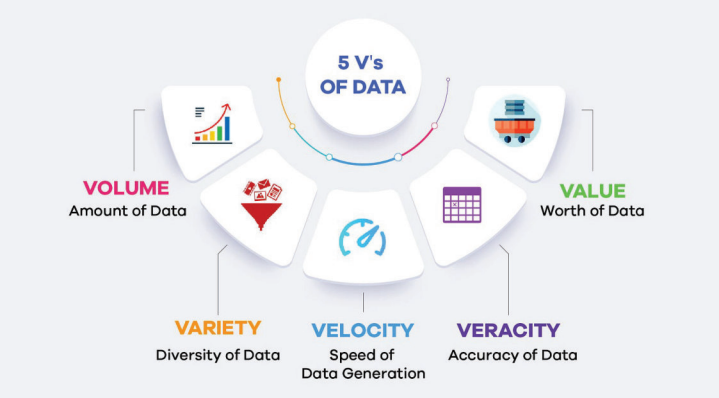

Figura 2.2: Le 5v dei Big Data (Medium)

vento dei Big Data ha portato a nuove e complesse sfide nella gestione della qualità dei dati (DQ). Con l'aumento esponenziale del volume, della varietà e della velocità dei dati, le tradizionali tecniche di gestione della qualità si sono rivelate inadeguate, richiedendo approcci innovativi e più dinamici.

Il **volume** dei Big Data è uno dei principali fattori che influiscono sulla qualità dei dati. La quantità di dati generati e raccolti è così vasta che i metodi di controllo tradizionali, basati su operazioni manuali o su tecnologie che non possono scalare efficacemente, non sono più sufficienti. Questo scenario impone la necessità di soluzioni di gestione della qualità che possano operare su scala massiva, senza compromettere l'efficienza. La difficoltà principale è che i sistemi devono gestire petabyte di dati e garantire che i controlli di qualità siano eseguiti in modo tempestivo e accurato [7].

La **varietà** dei dati è un'altra sfida significativa. I Big Data non si limitano ai dati strutturati, ma comprendono anche dati semi-strutturati e non strutturati, come testi, immagini e dati generati da dispositivi IoT. Questa eterogeneità complica enormemente l'integrazione dei dati e la loro uniformità. L'integrazione di dati provenienti da fonti disparate richiede tecniche avanzate per mantenere la consistenza e garantire che le informazioni siano complete e corrette. Per esempio, i dati provenienti da sensori IoT possono avere formati e strutture completamente diversi rispetto ai dati dei database tradizionali, richiedendo metodologie di integrazione sofisticate.

La **velocità** con cui i dati vengono generati e devono essere elaborati è altrettanto cruciale. In molti casi, i dati devono essere elaborati e analizzati in tempo reale o quasi reale. Questo richiede che i sistemi di gestione della qualità dei dati siano in grado di operare in modo rapido e dinamico per garantire che i dati siano accurati e aggiornati. Tecniche come l'elaborazione in streaming e i controlli di qualità on-the-fly sono diventate essenziali per gestire i flussi continui di dati e mantenere la loro integrità.

La **veridicità** si riferisce alla qualità e affidabilità dei dati, indicando quanto i dati

possano essere considerati accurati, consistenti e degni di fiducia. In molti contesti, la veridicità implica verificare che i dati non siano rumorosi, distorti o parziali, poiché l'uso di dati non affidabili può portare a decisioni sbagliate. La sfida principale per le aziende è quella di garantire che i dati raccolti provengano da fonti attendibili e che siano adeguatamente filtrati prima di essere analizzati.

Inoltre, la consistenza e l'integrità dei dati sono continuamente messe alla prova in ambienti di Big Data. Con la proliferazione dei dati tra diversi sistemi e fonti, mantenere una visione coerente e integrata delle informazioni può essere estremamente complesso. Le tecniche di sincronizzazione e validazione devono essere avanzate e adattabili per gestire le discrepanze e garantire che i dati rimangano affidabili nel tempo.

Infine, l'adozione di tecnologie avanzate come il machine learning e l'intelligenza artificiale ha portato all'adozione di una quinta V, la V del **Valore**. Questa dimensione riguarda la capacità di estrarre informazioni utili dai dati. Non basta infatti raccogliere e gestire enormi volumi di dati; il vero valore deriva dalla capacità di trasformarli in insight che possano supportare decisioni aziendali o strategiche.

2.1.4 Tecniche e Metodi Esistenti per il Controllo della Qualità dei Dati

Avendo compreso l'importanza cruciale della qualità dei dati nell'era del big data, è fondamentale esaminare le tecniche e i metodi specifici che sono stati sviluppati per garantire l'integrità, la coerenza e l'affidabilità delle informazioni gestite. Questi approcci, che spaziano dalla validazione dei dati alla loro pulizia, profilazione, e monitoraggio continuo, rappresentano le fondamenta su cui si basa un'efficace gestione della qualità dei dati. Ciascuna tecnica offre strumenti distintivi che affrontano particolari aspetti delle sfide inerenti alla gestione di grandi volumi di dati eterogenei, consentendo di mantenere standard elevati e di garantire che le decisioni prese sulla base di questi dati siano

affidabili e precise

1. **Validazione dei Dati**: La validazione è una tecnica fondamentale che verifica se i dati soddisfano specifici criteri e regole predefinite. Questo processo include controlli come il formato, l'intervallo e il tipo di dato, assicurandosi che i dati inseriti siano coerenti con le aspettative del sistema. Ad esempio, una data di nascita non può essere futura e un indirizzo email deve avere una sintassi corretta. La validazione dei dati aiuta a prevenire errori sistematici, anche Wand e Wang (1996)[8] ancorano la ricerca sulla qualità dei dati a una prospettiva di sistemi informativi, evidenziando l'importanza della validazione nel mantenere la precisione dei dati.

2. **Pulizia dei Dati (Data Cleansing)**: La pulizia dei dati è un processo essenziale per garantire l'accuratezza e la coerenza all'interno di un dataset, ed è una fase cruciale del processo di ETL (Extract, Transform, Load) che caratterizza i sistemi di data warehousing. Durante questa fase, i dati errati, incompleti o duplicati vengono identificati e corretti, migliorando la qualità complessiva delle informazioni. Le attività di pulizia includono la rimozione di duplicati, la correzione di errori di ortografia e la gestione dei valori mancanti. In un contesto di data warehousing, la pulizia dei dati assicura che le informazioni siano affidabili e pronte per essere utilizzate in analisi approfondite, garantendo che le decisioni basate sui dati siano fondate su informazioni precise e consistenti.

3. **Profilazione dei Dati**:La profilazione dei dati è il processo di analisi dei dati per comprendere la loro struttura e il contenuto. Rappresenta un passo fondamentale per garantire la validità e l'usabilità dei dataset. Questo processo consente di rilevare anomalie, identificare pattern ricorrenti e valutare la distribuzione dei

valori, offrendo un quadro chiaro delle caratteristiche intrinseche dei dati. Attraverso la profilazione, è possibile scoprire incongruenze e problemi che potrebbero compromettere la qualità dei dati. [7].

4. **Monitoraggio della Qualità dei Dati**: Il monitoraggio della qualità dei dati è una pratica fondamentale per garantire che i dati continuino a soddisfare gli standard di qualità nel tempo. Questa tecnica implica l'uso di strumenti e processi per controllare continuamente la qualità dei dati e rilevare eventuali degradi o anomalie. Durante il monitoraggio, vengono impostati indicatori e metriche per valutare aspetti come l'accuratezza, la completezza e la consistenza dei dati. Gli strumenti di monitoraggio possono generare allerta in caso di anomalie o violazioni delle regole di qualità, permettendo interventi tempestivi per correggere i problemi prima che influiscano sulle analisi e decisioni basate sui dati. La letteratura suggerisce che un monitoraggio continuo e sistematico è essenziale per mantenere la qualità dei dati, specialmente in ambienti dinamici dove i dati cambiano frequentemente.

5. **Tecniche di Normalizzazione**: La normalizzazione dei dati si riferisce alla standardizzazione delle informazioni per garantire la loro coerenza e comparabilità. Questo processo è cruciale per evitare problemi derivanti da dati non uniformi, come incongruenze nei nomi dei campi o nei formati delle date. La normalizzazione facilita la manipolazione e l'analisi dei dati, rendendoli più facili da integrare e confrontare. Tecniche di normalizzazione includono la conversione dei dati a un formato uniforme e la standardizzazione dei nomi dei campi.

6. **Tecniche di Integrazione dei Dati**: L'integrazione dei dati implica la combinazione di dati provenienti da diverse fonti in modo coerente e significativo. Questo processo può comportare la risoluzione di conflitti tra dati disomogenei e la mappatura dei dati tra sistemi diversi. L'integrazione efficace garantisce che i dati

siano allineati e pronti per essere utilizzati in analisi e reportistica. Le tecniche di integrazione possono includere la fusione di dataset, l'unificazione dei formati e la gestione delle discrepanze tra i dati provenienti da fonti diverse.

7. **Utilizzo di Metriche di Qualità dei Dati**:L'applicazione di metriche per misurare la qualità dei dati è fondamentale per valutare aspetti come l'accuratezza, la completezza, la consistenza e la tempestività dei dati. Le metriche forniscono una misura quantitativa della qualità dei dati, permettendo alle organizzazioni di monitorare e migliorare continuamente le proprie pratiche di gestione dei dati. L'uso di metriche consente di identificare aree di miglioramento e di garantire che i dati siano sempre allineati agli standard di qualità richiesti.

8. **Tecniche di Data Provenance**: La provenienza dei dati riguarda la tracciabilità e la documentazione dell'origine e delle trasformazioni dei dati nel tempo. Questo è particolarmente importante per garantire la trasparenza e la fiducia nei dati utilizzati per le decisioni. Le tecniche di data provenance consentono di seguire il percorso dei dati dalla loro origine fino alla loro destinazione finale, documentando ogni trasformazione e modifica subita. Questo processo facilita la verifica della qualità dei dati e la risoluzione di eventuali problemi che potrebbero emergere. [6].

In conclusione, l'adozione di tecniche e metodologie per il controllo della qualità dei dati è essenziale per garantire l'affidabilità e l'efficacia dei sistemi informativi. L'integrazione di queste tecniche permette di affrontare e mitigare le sfide associate alla qualità dei dati, migliorando così le decisioni basate su dati affidabili e accurati.

2.2 Approcci Fail-safe nella gestione della Data Quality

L'integrazione di approcci fail-safe nella gestione della Data Quality (DQ) è diventata sempre più rilevante, soprattutto in contesti in cui la qualità dei dati è fondamentale per supportare decisioni strategiche e operative. Un sistema fail-safe è progettato per prevenire, rilevare e mitigare gli effetti di eventuali errori o anomalie che possono verificarsi durante il processo di gestione dei dati, garantendo che le operazioni possano proseguire in modo sicuro e senza interruzioni significative.

Il fail-safe, pur avendo origine nel campo dell'ingegneria dei sistemi, trova una sua applicazione naturale anche nella gestione della qualità dei dati. In questo contesto, si tratta di progettare meccanismi che non solo proteggano l'integrità dei dati, ma che garantiscano anche la continuità operativa [2]. Quando parliamo di approcci fail-safe in relazione alla Data Quality, ci riferiamo a sistemi e procedure che intercettano e gestiscono i dati errati o non conformi in tempo reale, evitando che possano propagarsi all'interno della pipeline di dati e causare danni più estesi.

Una delle sfide principali nella gestione della Data Quality, soprattutto in ambienti di Big Data, è la velocità con cui i dati vengono generati, elaborati e utilizzati. In questi contesti, i dati possono diventare rapidamente obsoleti, incompleti o inaccurati, compromettendo la validità delle analisi e delle decisioni che ne derivano. È qui che un approccio fail-safe dimostra la sua efficacia, consentendo di intercettare gli errori sul nascere e di attivare automaticamente le azioni correttive necessarie, come il blocco della propagazione dei dati errati, la loro correzione o la segnalazione ai team di supporto.

Questi sistemi non solo migliorano la qualità complessiva dei dati, ma rafforzano anche la fiducia nelle decisioni basate su di essi. L'adozione di un approccio fail-safe nella gestione della DQ, quindi, non è solo una misura di sicurezza, ma una strategia

fondamentale per mantenere elevati standard di qualità in un contesto aziendale sempre più data-driven. I successivi paragrafi esploreranno in dettaglio il concetto di fail-safe, la sua relazione con la Data Quality, e le migliori pratiche per la sua implementazione, fornendo un quadro completo che supporta la soluzione proposta in questa tesi.

2.2.1 Concetto di Fail-safe

Il concetto di fail-safe fonda le sue radici nell'ingegneria dei sistemi, in particolare nei settori in cui la sicurezza è di importanza critica, come l'ingegneria aerospaziale e la progettazione di dispositivi elettronici. Qui, un sistema fail-safe è progettato per prevenire danni significativi anche in caso di guasto. L'idea centrale è che, qualora un componente critico fallisca, il sistema possa continuare a funzionare in modo sicuro o, se necessario, fermarsi in modo controllato per evitare conseguenze gravi. Questo principio è stato essenziale per garantire la sicurezza e l'affidabilità di sistemi complessi fin dai primi sviluppi della tecnologia moderna.

Nel tempo, il concetto di fail-safe è stato adattato e applicato anche nell'ambito della gestione della qualità dei dati (Data Quality, DQ). Con l'evoluzione delle tecnologie informatiche e la crescente dipendenza dalle decisioni basate sui dati, è diventato cruciale assicurare che i dati siano non solo accurati, ma anche continuamente affidabili e sicuri. In questo contesto, un sistema fail-safe implica la capacità di rilevare e gestire automaticamente errori o anomalie nei dati, prevenendo la propagazione di informazioni errate lungo la pipeline dei dati e minimizzando l'impatto sull'operatività aziendale.
La capacità di un sistema di gestione della DQ di operare in modalità fail-safe non solo previene la propagazione di errori, ma consente anche di mantenere la fiducia nei processi decisionali basati sui dati. Questo è particolarmente rilevante in ambienti di Big Data, dove la velocità e il volume dei dati rendono essenziale la presenza di meccanismi automatici per la rilevazione e la correzione degli errori.

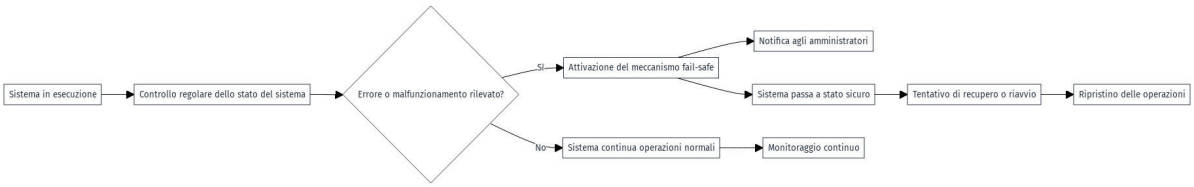

Figura 2.3: Flusso fail-safe

In letteratura vengono esplorate diverse metodologie per valutare e migliorare la qualità dei dati, evidenziando come queste possano essere integrate in sistemi che non solo rilevano ma anche correggono automaticamente i problemi di qualità. Tali approcci sono essenziali per costruire un'infrastruttura dati resiliente, in grado di operare in modo affidabile anche in presenza di anomalie. In un mondo dove i dati guidano decisioni critiche, l'adozione di principi fail-safe nella gestione della DQ non è solo una misura di sicurezza, ma una strategia fondamentale per mantenere elevati standard di qualità e operatività aziendale [2].

2.2.2 Fail-safe e Data Quality

Il concetto di fail-safe è ampiamente implementato nella gestione della qualità dei dati, specialmente in contesti dove la robustezza e l'affidabilità del sistema sono cruciali. Nei sistemi di Data Analytics, dove i volumi elevati e la complessità dei dati possono facilmente generare errori e incongruenze, un approccio fail-safe è essenziale per garantire l'integrità e l'affidabilità dei dati. Gli obiettivi principali nell'era dell'analisi dei dati sono principalemte i seguenti:

1. **Rilevamento e correzione degli errori**. Il fail-safe viene spesso utilizzato per rilevare e correggere automaticamente gli errori nei dati, prevenendo interruzioni operative. Per evitare il fallimento del sistema e garantire la qualità e la continuità dei dati, è importante l'utilizzo di strumenti automatizzati per il monitoraggio in tempo reale delle anomalie e per l'attivazione di meccanismi di correzione.

2. **Resilienza del sistema**. Un sistema fail-safe contribuisce alla resilienza complessiva della piattaforma, permettendo al sistema di continuare a funzionare anche in presenza di dati problematici. La progettazione di questi sistemi si focalizza sulla capacità di mantenere operativa la piattaforma riducendo al minimo l'impatto sugli utenti finali, un aspetto cruciale per le piattaforme che richiedono operatività continua.

3. **Miglioramento continuo della qualità dei dati**. Oltre alla gestione degli errori, bisigna supportare il miglioramento continuo della qualità dei dati. Sistemi ben progettati possono integrare meccanismi di feedback per apprendere dagli errori passati, migliorando la qualità del dato nel tempo e riducendo la necessità di interventi manuali. Questo approccio è fondamentale per affrontare le sfide poste dall'evoluzione costante dei dati e delle tecnologie.

   L'adozione di un approccio fail-safe nella gestione della qualità dei dati è cruciale per proteggere le organizzazioni dai rischi associati a dati problemativi, migliorando al contempo la resilienza e l'affidabilità del sistema, fattori essenziali in un contesto di Big Data.

2.2.3 Implementazioni e best practice

Per non compromettere l'integrità del sistema tramite dati non conformi, bisogna adottare pratiche e soluzioni tecniche specifiche, consolidate nell'ambito fail-safe per una profiqua gestione della qualità dei dati. Di seguito esploreremo alcune pratiche standard esistenti nel modo fail-safe, dando una descrizione del **trattamento** dei dati non conformi dei metodi utilizzati per prevenire la **propagazione degli errori**, delle strategie per la gestione delle **segnalazioni** e degli spunti sull'**assistenza tecnica**.

1. **Trattamento dei Dati Non Conformi:** Il trattamento dei dati che non rispettano determinate regole o vincoli predefiniti, rigurda una delle implementazioni più comuni in ambito fail-safe. Lo scarto, in questo caso, è l'operazione che viene presa in considerazione per prima insieme all'isolamento in modo tale da non incidere sul flusso principale, evitando così che i dati possano causare errori più gravi nel sistema.Ad esempio, in molte architetture di Big Data, quando vengono rilevati dati anomali, essi vengono automaticamente etichettati e inviati a un sistema di *quarantine* dove possono essere ulteriormente analizzati senza influenzare l'elaborazione dei dati corretti[1].

2. **Prevenzione della Propagazione degli Errori:** Un'altra best practice fondamentale è l'uso di checkpoint e rollback. Questa tecnica consente al sistema di tornare a uno stato precedente nel caso in cui venga rilevato un errore, prevenendo la propagazione di dati corrotti attraverso la pipeline di elaborazione. I checkpoint sono salvati periodicamente in modo tale che, in caso di errore, il sistema può ripristinare uno di questi punti, riducendo al minimo l'impatto dell'anomalia.

3. **Segnalazione e Gestione delle Anomalie:** Un'altra componente importante di un sistema fail-safe è la gestione delle segnalazioni. L'essenziale è notificare immediatamente agli operatori del sistema, tramite un sistema di *alerting*, quando un errore o un'anomalia viene rilevata. Le piattaforme più avanzate integrano meccanismi di escalation, dove gli errori critici generano avvisi di alta priorità, attivando team di supporto per interventi immediati. Le segnalazioni spesso vengono avvalorate da dettagli diagnostici che sono utili a identificare la causa del problema.

4. **Automatizzazione del Supporto Tecnico:** L'automatizzazione del supporto tecnico rappresenta una strategia avanzata nella gestione della Data Quality, dove l'uso di tecnologie come l'intelligenza artificiale (AI) e il machine learning (ML)

permette di analizzare automaticamente le cause delle anomalie e di proporre soluzioni correttive. In molti sistemi moderni, l'integrazione di algoritmi intelligenti consente di rilevare pattern di errore ricorrenti e di attivare processi di correzione automatizzati, riducendo così la necessità di interventi manuali e accelerando i tempi di risoluzione. Questo approccio non solo migliora l'efficienza operativa, ma aumenta anche la capacità del sistema di adattarsi e rispondere rapidamente alle problematiche emergenti, minimizzando l'impatto sulle operazioni aziendali quotidiane[1]

Seguendo queste best practice, le organizzazioni possono non solo prevenire la propagazione degli errori, ma anche migliorare la loro capacità di rispondere rapidamente e in modo efficace alle anomalie, minimizzando l'impatto sulle operazioni quotidiane.

2.3 Data Platforms

Le Data Platform sono il cuore pulsante delle moderne infrastrutture aziendali. Queste piattaforme fungono da fondamenta per la raccolta, l'integrazione, l'analisi e la gestione di dati provenienti da diverse fonti. La crescente necessità di gestire e sfruttare nell'immediato grandi volumi di dati ha reso queste piattaforme essenziali per qualsiasi organizzazione che si voglia definire **data-driven**. Grazie a una Data Platform, i dati sono centralizzati e resi accessibili a diversi team, dagli analisti ai manager, in modo tale da poter prendere decisioni informate. L'importanza dell'implementazione di sistemi di Data Quality e di monitoraggio fail-safe, che andremo a discutere nei capitoli a venire, emerge nei contesti di Data Platforms. Questi sistemi richiedono un ambiente scalabile, sicuro e in grado di gestire stream di dati complessi: caratteristiche che solo una piattaforma dati ben strutturata può assicurare. Nel contesto del mio lavoro, la Data Platform su cui è stato sviluppato il sistema di monitoraggio della qualità dei dati è Microsoft

Azure. Azure rappresenta un esempio di piattaforma cloud avanzata che offre una suite di strumenti integrati per l'acquisizione, l'elaborazione, la memorizzazione e l'analisi dei dati. Tuttavia, la panoramica che seguirà non si limita solo a Microsoft Azure, ma presenta le caratteristiche chiave che definiscono le moderne Data Platform, esplorando i loro concetti fondamentali e le sfide che affrontano.

In primo luogo, analizzeremo lo scopo principale di una Data Platform: la capacità di raccogliere dati da una vasta gamma di fonti e di renderli disponibili per l'elaborazione e l'analisi, supportando decisioni operative e strategiche. Successivamente, affronteremo l'architettura generale di una Data Platform, discutendo i vari componenti che permettono di orchestrare l'ingestione, la trasformazione e l'elaborazione dei dati. Vedremo come tali processi siano essenziali per mantenere la qualità dei dati, prevenendo errori e garantendo la tracciabilità delle informazioni. Un aspetto cruciale delle Data Platform, che verrà discusso, è la loro capacità di gestire fonti di dati eterogenee. Che si tratti di database relazionali, dati in streaming o file di log, queste piattaforme devono essere in grado di armonizzare e integrare dati provenienti da diverse origini, garantendo coerenza e qualità lungo tutta la pipeline. Infine, verranno discusse le soluzioni di analisi dei dati e gli strumenti di business intelligence, come Azure Synapse e Power BI, che consentono di trarre valore dai dati attraverso visualizzazioni e report. Analizzeremo anche aspetti chiave come la sicurezza, la governance e la scalabilità, elementi indispensabili per garantire la continuità e la manutenzione della piattaforma nel lungo termine.

Questa introduzione alla panoramica della Data Platform servirà da base per comprendere meglio come l'architettura su cui è stato sviluppato il sistema di Data Quality fail-safe sia in grado di affrontare le sfide legate alla qualità e alla gestione dei dati nelle moderne organizzazioni.

2.3.1 Definizione e scopo di una Data Platform

Una Data Platform rappresenta un'infrastruttura tecnologica complessa, progettata per centralizzare, gestire, elaborare e distribuire i dati all'interno di un'organizzazione. Nel contesto attuale, con l'aumento esponenziale delle fonti di dati e della loro varietà, il concetto di Data Platform è diventato un pilastro per le imprese moderne. Essa non è solo un sistema di archiviazione, ma un insieme di tecnologie e processi integrati che permettono di trasformare dati grezzi in informazioni utili e fruibili in vari contesti aziendali. Secondo IBM, una Data Platform è una suite di prodotti software che consentono la raccolta, la pulizia, la trasformazione e l'analisi dei dati di un'organizzazione per contribuire a migliorare il processo decisionale.[4]

Lo scopo principale di una Data Platform è quello di creare un'**infrastruttura centralizzata** in grado di fornire accesso continuo e scalabile ai dati, garantendo allo stesso tempo la loro qualità, sicurezza e disponibilità. L'obiettivo è quello di migliorare la business intelligence e facilitare decisioni informate basate sui dati aggiornate. La piattaforma permette alle organizzazioni di raccogliere dati da varie fonti, trasformarli in informazioni utili attraverso strumenti di analisi e fornire i risultati a chi ne ha bisogno in modo tempestivo. La flessibilità e l'elasticità della data platform sono fondamentali per consentire alle organizzazioni di adattarsi ai continui cambiamenti tecnologici e di mercato.

Oltre alla centralizzazione, le Data Platforms si prefiggono di garantire una elevata **qualità dei dati**. La qualità è un fattore critico: informazioni errate o obsolete possono portare a decisioni sbagliate. Per questo motivo, una piattaforma dati efficace deve essere in grado di mantenere i dati accurati, coerenti e completi, monitorando costantemente le fonti e i processi di acquisizione per rilevare e correggere eventuali anomalie.

Un altro obiettivo chiave è la **scalabilità**. Trovandosi di frone a volumi crescenti di dati, una Data Platform deve essere in grado di crescere di pari passo all'organizzazione,

gestendo carichi di lavoro sempre maggiori senza compromettere le prestazioni. La capacità di scalare in modo dinamico consente di adattarsi non solo alla crescita dei dati, ma anche all'introduzione di nuove tecnologie e requisiti aziendali, come l'intelligenza artificiale e l'apprendimento automatico.

La **sicurezza** e la **governance** sono altrettanto fondamentali per una piattaforma di dati moderna. La protezione dei dati, specialmente in settori altamente regolamentati, è un aspetto cruciale. Le piattaforme devono garantire che i dati siano accessibili solo a chi ha l'autorizzazione necessaria e che siano conformi alle normative sulla privacy e la gestione dei dati. Questo include non solo l'implementazione di rigidi controlli di accesso, ma anche la definizione di politiche di gestione e di monitoraggio continuo.

2.3.2 Architettura di una Data Platform

L'architettura di una data platform può essere vista come una struttura stratificata e modulare, progettata per rispondere alle necessità di raccolta, gestione, elaborazione e analisi dei dati in ambienti aziendali complessi. Questo tipo di architettura è generalmente suddivisa in vari livelli che operano insieme per garantire una gestione efficiente e sicura delle informazioni. Partiamo dal **Source Layer**, che rappresenta il punto di origine per tutti i dati che alimentano la piattaforma. Le fonti di dati possono variare notevolmente, da sistemi transazionali tradizionali a data warehouse, flussi di dati in tempo reale, eventi generati da applicazioni o dispositivi, e persino feed di terze parti. L'obiettivo principale di questo livello è quello di raccogliere dati da una vasta gamma di fonti, sia interne che esterne, preparandoli per i processi successivi. I dati raccolti passano quindi attraverso **l'Acquisition Layer**, che si occupa di trasformare queste informazioni grezze in un formato utilizzabile per la piattaforma. Questo livello utilizza una serie di strumenti tecnici come API, pipeline ETL (Extract, Transform, Load) e processori di stream. Grazie a questi componenti, i dati vengono acquisiti, trasformati, e

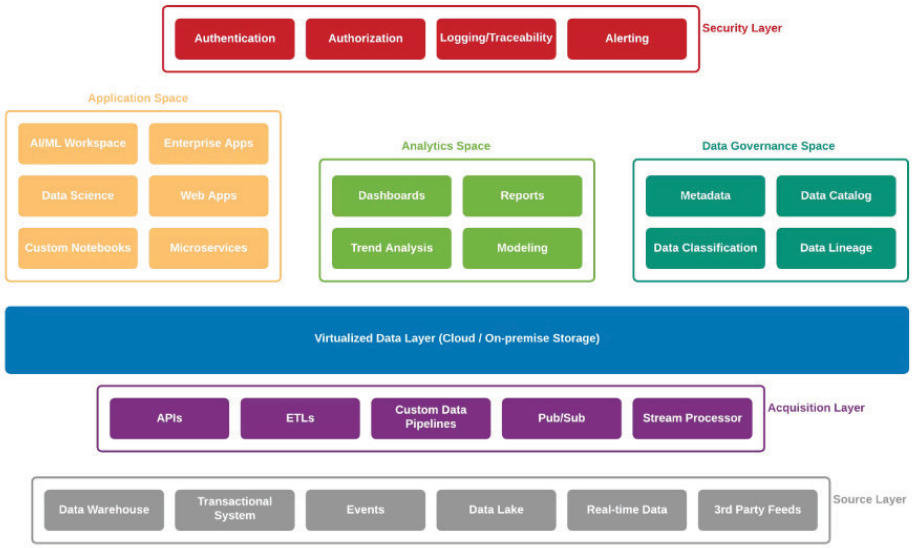

Figura 2.4: Architettura di una Data Platform (mongoDB)

preparati per l'elaborazione in modo che possano essere utilizzati dalle applicazioni e dagli strumenti di analisi. Una volta acquisiti, i dati si spostano nel Virtualized Data Layer, un componente cruciale che funge da punto centrale per la memorizzazione e l'accesso. Questo livello virtualizza lo storage, consentendo alle aziende di gestire dati su cloud, in ambienti on-premise o in una combinazione ibrida. I data warehouse e i data lake presenti in questo strato non solo memorizzano i dati ma li rendono facilmente accessibili per processi di analisi, reportistica o ulteriori trasformazioni. Mentre i dati si muovono verso l'alto nell'architettura, entrano nel cosiddetto **Application Space**, dove applicazioni e strumenti specifici per le aziende interagiscono con essi. Questo spazio ospita una varietà di applicazioni, dai workspace per progetti di machine learning e intelligenza artificiale, fino agli strumenti di microservizi e alle applicazioni web. L'Application Space rappresenta l'area in cui gli utenti finali, come analisti e data scientist, possono effettivamente utilizzare i dati per creare modelli, algoritmi e dashboard personalizzati. Di pari passo all'Application Space precedente abbiamo **l'Analytic Space**, dove i dati raccolti e trasformati vengono sfruttati per creare insight strategici. Gli strumenti analitici qui inclusi consentono di costruire report, dashboard e modelli avanzati di analisi dei trend. Questo

spazio è essenziale per trasformare i dati grezzi in informazioni fruibili, fondamentali per supportare il processo decisionale aziendale. A supporto di queste operazioni troviamo il **Data Governance Space**, che gestisce tutti gli aspetti legati alla qualità, sicurezza e conformità dei dati. In questo livello vengono implementati strumenti di classificazione dei dati, catalogazione, lineage e gestione dei metadati. Il Data Governance Space garantisce che i dati siano utilizzati nel rispetto delle normative, proteggendo la loro integrità e permettendo la tracciabilità delle informazioni lungo l'intero ciclo di vita. Infine, l'intera architettura è protetta da un **Security Layer** che garantisce la sicurezza e la riservatezza dei dati. Questo livello include funzioni di autenticazione e autorizzazione per controllare l'accesso ai dati, così come strumenti di logging e tracciabilità per monitorare le attività. Inoltre, vengono implementati meccanismi di alert per prevenire e rispondere tempestivamente a eventuali minacce o anomalie. In questo modo, la struttura di una data platform offre un ecosistema completo e ben integrato, che consente di gestire il flusso dei dati dal momento in cui vengono acquisiti fino alla loro analisi finale, garantendo sicurezza, qualità e facilità di accesso lungo tutto il processo.

2.3.3 Integrazione delle fonti dati e storage

L'integrazione delle fonti dati e lo storage rappresentano due aspetti centrali nella gestione di una moderna data platform, strettamente interconnessi per garantire la disponibilità e l'efficacia dei dati all'interno dell'organizzazione. Integrare le fonti, come discusso in precedenza, si traduce nell'acquisire, trasformare e consolidare i dati provenienti da diverse origini. La chiave di un'integrazione efficace risiede nella capacità di orchestrare questo flusso di dati eterogenei attraverso pipeline robuste e flessibili. Alla base dell'integrazione dei dati abbiamo il processo di **ETL (Extract, Transform, Load)**[5]:

- **Estrazione (Extract)** L'integrazione delle fonti dati inizia con l'estrazione dei dati dalle diverse sorgenti. Le sorgenti possono includere database relazionali (SQL), fonti di dati non strutturati come i file JSON o CSV, dati in tempo reale provenienti da sensori IoT o streaming come Apache Kafka, e fonti esterne come servizi cloud o API. Ogni sorgente ha le sue caratteristiche uniche, e il processo di estrazione deve essere configurato per affrontare problemi di formattazione, velocità di acquisizione e accessibilità ai dati. Ad esempio, strumenti come Azure Data Factory permettono l'estrazione automatizzata di dati da svariate fonti utilizzando connettori predefiniti che semplificano il processo. Microsoft Azure offre numerosi strumenti per la connessione, gestione e automazione dell'estrazione, consentendo l'acquisizione di dati da più fonti contemporaneamente

- **Trasformazione (Transform)** Dopo l'estrazione, i dati devono essere trasformati per diventare coerenti e utilizzabili per le fasi successive. Questa fase può includere la normalizzazione dei dati, il mapping di campi da sorgenti diverse, la rimozione dei duplicati, il controllo della qualità dei dati e il loro arricchimento. In un ambiente come Microsoft Azure, l'Azure Data Factory consente di creare pipeline di trasformazione complesse che integrano funzionalità come il controllo della qualità e la validazione.

- **Caricamento (Load)** Infine, i dati trasformati vengono caricati nei sistemi di archiviazione della data platform. Questa fase prevede che i dati siano collocati in data warehouse, data lake o altre architetture di archiviazione. Il processo di caricamento deve essere gestito in modo efficiente per evitare colli di bottiglia, specialmente quando si trattano grandi volumi di dati.

Una volta affrontata l'ultima parte del processo di ETL entra in gioco lo **storage** che consete di gestire e archiviare i dati provenienti da diverse fonti, garantendo che

siano disponibili per le successive analisi e trasformazioni. Nelle piattaforme moderne, lo storage viene suddiviso tipicamente in due categorie principali: **Data Lake** e **Data Warehouse**.

La migliore definizione di **Data Lake** lo descrive come un luogo destinato all'archiviazione, analisi e correlazione di dati strutturati e non strutturati (da quelli del CRM ai post dei social media, dai dati ERP alle info delle macchine di produzione), in formato nativo. La sua peculiarità è di consentire il recupero e l'organizzazione del dato secondo il tipo di analisi che si intende effettuare. Questa novità, rispetto ai tradizionali sistemi di Big Data Analytics, rappresenta una semplificazione e un notevole potenziamento dello strumento.

Un **Data Warehouse** è un database ottimizzato per analizzare dati relazionali provenienti da sistemi transazionali e applicazioni line of business. La struttura dei dati e lo schema sono definiti preventivamente per ottimizzare le query SQL veloci, in cui i risultati sono utilizzati di solito per il resoconto operativo e l'analisi. I dati vengono riordinati, arricchiti e trasformati in modo da poter agire come "unica fonte di verità" a cui gli utenti possono fare affidamento.

Nell'utimo periodo si sta parlando di **Lakehouse**, una nuova architettura aperta per la gestione dei dati che unisce la flessibilità, l'economicità e la scalabilità dei data lake alle funzionalità di gestione dei dati e alle transazioni ACID dei data warehouse, per realizzare attività di business intelligence (BI) e machine learning (ML) su tutti i dati. L'obiettivo è quello di implementare strutture e funzionalità per la gestione dei dati simili a quelle di un data warehouse direttamente su un sistema di organizzazione dei dati analogo a quello dei data lake. Queste due caratteristiche vengono applicate ad un unico sistema, quindi, i team di gestione dei dati possono agire più velocemente, in quanto possono utilizzare i dati senza bisogno di accedere a diversi sistemi. Inoltre, i data lakehouse fanno sì che i team abbiano sempre a disposizione i dati più completi e

aggiornati per progetti di data science, machine learning e business analytics.

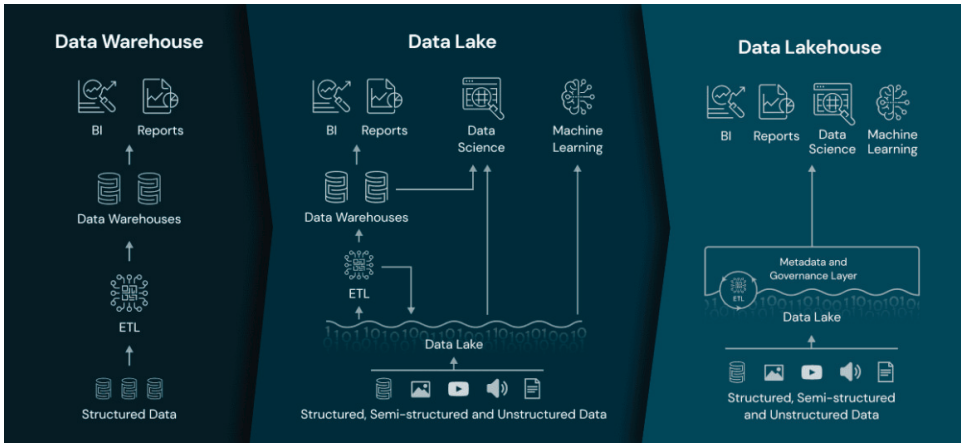

Figura 2.5: Panoramica Data Warehouse Data Lake Data Lakehouse (Databricks)

2.3.4 Analisi dei dati e business intelligence

Sebbene costruite con l'obiettivo principale di raccogliere, gestire e utilizzare i dati, le data platforms manifestano il loro vero valore tramite la capacità di analizzare i dati. L'analisi dei dati consente alle organizzazioni di trasformare grandi quantità di dati grezzi in informazioni preziose per prendere decisioni strategiche e operative. La fase di analisi è ciò che permette alle imprese di andare oltre il semplice immagazzinamento delle informazioni, permettendo loro di ottenere insight significativi che possono influenzare ogni aspetto dell'organizzazione, dall'ottimizzazione delle operazioni fino alla personalizzazione dei servizi. L'analisi dei dati nelle data platforms si articola su diversi livelli, dai semplici report descrittivi alle analisi avanzate basate su intelligenza artificiale e machine learning, permettendo un'esplorazione continua e profonda dei dati aziendali.

Dopo aver introdotto l'importanza dell'analisi dei dati all'interno delle data platforms, possiamo passare alla descrizione delle diverse tipologie di analisi che vengono effettuate per estrarre valore dai dati. Queste tipologie di analisi sono essenziali per comprendere

come vengono utilizzati i dati e quali strumenti di business intelligence sono necessari per ciascuna di esse.

- **Analisi Descrittiva**: L'analisi descrittiva è il punto di partenza dell'analisi dei dati, si concentra sulla comprensione di "cosa è successo" in un determinato contesto. Si basa sulla raccolta e sulla visualizzazione di informazioni storiche e produce report, grafici e dashboard che riassumono le informazioni. È la tipologia utilizzata più comunemente dalle aziende per monitorare le performance e ottenere una visione immediata dello stato delle operazioni. Il suo scopo è quello di fornire un contesto che può essere facilmente interpretato, identificando tendenze o pattern storici.

- **Analisi Diagnostica**: Questa tipologia è più sofisticata dell'analisi descrittiva perchè approfondisce il dettaglio di "cosa è successo" e cerca di rispondere alla domanda "perché è successo?". Attraverso l'analisi diagnostica si indagano le cause sottostanti a determinati fenomeni o risultati osservati nei dati, utilizzando tecniche come il drill-down e l'esplorazione dei dati per individuare correlazioni o fattori scatenanti. L'obiettivo è identificare relazioni causali che possano spiegare eventi specifici, fornendo così indicazioni utili per correggere errori o ottimizzare processi.

- **Analisi Predittiva**: Con l'analisi predittiva si cerca di prevedere "cosa succederà" in futuro, basandosi su modelli statistici e algoritmi di machine learning. Questa tipologia di analisi utilizza dati storici per identificare pattern e tendenze che permettono di prevedere scenari futuri. È particolarmente utile per settori che necessitano di anticipare la domanda di mercato, come il retail o la logistica, o per la gestione del rischio in ambiti come la finanza. I modelli predittivi sono complessi e richiedono una buona qualità dei dati per fornire risultati accurati.

- **Analisi Prescrittiva**: Infine, l'analisi prescrittiva risponde alla domanda "cosa dobbiamo fare?". Va oltre la previsione e offre suggerimenti concreti sulle azioni da intraprendere per ottenere il miglior risultato possibile. Viene utilizzata per ottimizzare processi decisionali e operativi, suggerendo le strategie più efficaci in base alle informazioni disponibili. L'analisi prescrittiva sfrutta modelli avanzati di ottimizzazione e simulazione per consigliare azioni specifiche, come aggiustare le scorte o intervenire su un processo produttivo.

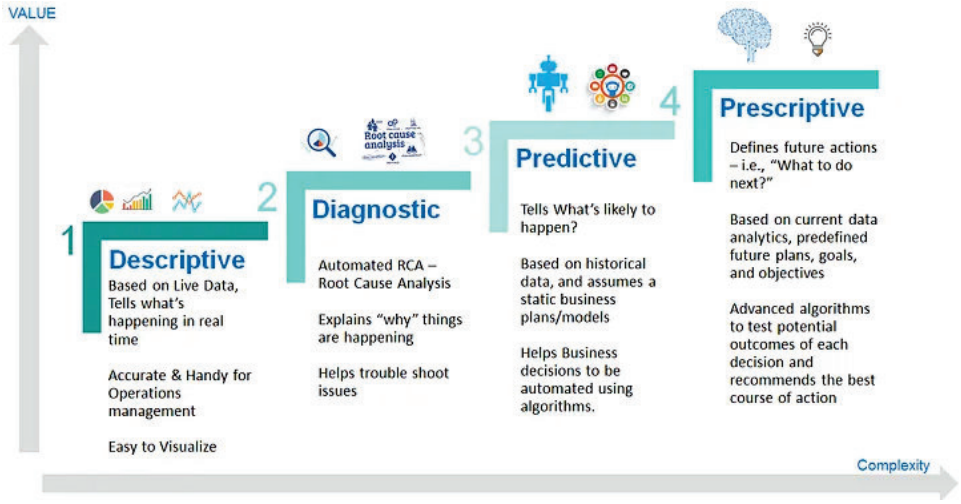

Figura 2.6: Tipi di analisi dati (Medium)

Proseguendo dal nostro esame dell'analisi dei dati, è fondamentale comprendere come la Business Intelligence (BI) si inserisca nel contesto delle data platforms e come completi il quadro analitico. Mentre l'analisi dei dati si concentra sull'approfondimento e sull'interpretazione dei dati per scoprire tendenze e previsioni, la BI svolge un ruolo complementare, trasformando questi risultati in strumenti pratici per le decisioni aziendali. La BI si occupa principalmente di raccogliere, aggregare e visualizzare i dati in modo che possano essere facilmente interpretati dagli utenti finali. Utilizzando strumenti di BI, le organizzazioni possono prendere i risultati dell'analisi dei dati e presentarli attraverso dashboard interattive, report e visualizzazioni che rendono le informazioni più accessibili

e comprensibili. Questo processo di visualizzazione è essenziale per tradurre i complessi risultati analitici in insight utili e immediatamente fruibili per i decision makers. I risultati delle analisi avanzate, come le previsioni di vendite o le correlazioni tra variabili, vengono integrati nelle visualizzazioni di BI, creando un ciclo continuo di analisi e decisione. Mentre l'analisi dei dati fornisce la base per comprendere i "perché" e i "cosa potrebbe accadere", la BI utilizza questi risultati per rispondere al "cosa sta succedendo adesso" e al "come possiamo reagire". Gli strumenti di Business Intelligence (BI) si articolano principalmente in tre categorie, ognuna delle quali svolge un ruolo chiave nel processo di trasformazione dei dati in informazioni utili per la presa di decisioni.

- **Strumenti di Reporting e Dashboarding**: Questi strumenti sono progettati per presentare i dati in modi visivamente accattivanti e comprensibili. Il reporting consente di generare report dettagliati che offrono una panoramica dei dati in formati standardizzati, mentre il dashboarding fornisce visualizzazioni interattive che permettono agli utenti di esplorare i dati in tempo reale. I report e i dashboard possono includere grafici, tabelle e indicatori di performance, facilitando la comprensione delle tendenze e dei risultati chiave.

- **Strumenti di Analisi Avanzata**: Questi strumenti si occupano di esaminare i dati con tecniche più sofisticate. L'analisi avanzata può includere analisi statistica, modelli predittivi e algoritmi di machine learning. L'uso di queste tecniche permette di identificare pattern nascosti, fare previsioni basate su dati storici e ottenere insights dettagliati sui fattori che influenzano le performance aziendali. Questi strumenti sono essenziali per comprendere non solo ciò che è successo, ma anche per anticipare le future tendenze e risultati.

- **Strumenti di Preparazione e Integrazione dei Dati**: Questi strumenti gestiscono la raccolta, la trasformazione e la combinazione dei dati provenienti da

diverse fonti. La preparazione dei dati comprende attività come l'estrazione dei dati da sistemi diversi, la loro pulizia e la loro trasformazione in formati compatibili. L'integrazione dei dati consente di combinare e armonizzare dati disparati in un'unica vista coerente. Questi strumenti sono fondamentali per garantire che i dati utilizzati per l'analisi siano accurati e pronti per essere esplorati.

In sintesi, la Business Intelligence utilizza una combinazione di tecniche di reporting e dashboarding, analisi avanzata e preparazione dei dati per trasformare i dati grezzi in informazioni utili. Questi strumenti lavorano insieme per offrire una comprensione approfondita delle informazioni aziendali, migliorando la capacità di prendere decisioni basate su evidenze e facilitando una gestione più efficace delle operazioni aziendali.

2.3.5 Governance dei dati

La **governance** dei dati si occupa di stabilire le regole e le pratiche per la gestione e l'uso delle informazioni. Questo include assicurarsi che i dati siano di alta qualità, accurati e completi. Le tecniche per garantire la qualità dei dati comprendono la profilazione dei dati e la gestione dei dati, che aiutano a identificare e correggere errori e incongruenze nei dati.

La **classificazione e catalogazione** dei dati sono pratiche importanti nella governance. Queste attività comportano l'organizzazione dei dati in categorie e la documentazione delle loro caratteristiche, come la loro origine e struttura. Questo facilita l'accesso e l'uso dei dati, migliorando la loro gestione e garantendo che siano utilizzati in modo appropriato. Inoltre, la governance dei dati deve garantire la **conformità alle normative**, come il GDPR, che stabilisce requisiti specifici per la protezione dei dati personali. Ciò implica che le organizzazioni devono adottare pratiche che rispettano le leggi e riducono il rischio di sanzioni. Una buona governance dei dati non solo protegge l'integrità e la riservatezza delle informazioni, ma migliora anche la qualità dei dati e facilita la

conformità alle normative. Si sottolinea l'importanza di integrare pratiche di sicurezza e governance per una gestione efficace dei dati nelle moderne data platforms. Mentre la sicurezza dei dati si concentra sulla protezione delle informazioni da accessi non autorizzati e minacce, la governance dei dati garantisce che i dati siano gestiti e utilizzati in modo corretto e conforme. Entrambi sono essenziali per mantenere l'affidabilità e la sicurezza delle data platforms, assicurando che le informazioni siano non solo protette ma anche di alta qualità e ben gestite.

2.3.6 Scalabilità e manutenzione

La scalabilità di una data platform è una caratteristica fondamentale, specialmente in un contesto dove i volumi di dati e le esigenze analitiche crescono in modo esponenziale. Parlare di scalabilità non significa solo aumentare la capacità di archiviazione o elaborazione, ma adottare una strategia flessibile e dinamica che permette all'infrastruttura di adattarsi rapidamente a nuove esigenze, ottimizzando al contempo l'uso delle risorse. La scalabilità, in questo senso, può essere vista su tre dimensioni principali: scalabilità orizzontale, verticale e funzionale.

La **scalabilità orizzontale** consiste nell'aggiungere nuovi nodi o server per distribuire il carico di lavoro, in modo tale da gestire volumi crescenti di dati senza creare colli di bottiglia. È qui che le piattaforme cloud giocano un ruolo fondamentale. Servizi come **Azure**, **AWS** o **Google Cloud** permettono alle aziende di espandere la loro capacità in modo dinamico e senza interruzioni. A differenza di un'infrastruttura tradizionale, dove sarebbe necessario acquistare e configurare nuovi hardware, il cloud fornisce risorse on-demand, consentendo alle aziende di crescere con costi proporzionali alle risorse effettivamente utilizzate

La **scalabilità verticale**, invece, consiste nell'aumentare la capacità delle risorse già esistenti, come aumentare la potenza di elaborazione o la memoria di un singolo nodo.

Questo approccio è meno comune in ambienti moderni a causa dei limiti hardware, ma rimane utile in contesti dove la scalabilità orizzontale non è immediatamente fattibile. In ambienti cloud, questa operazione può essere eseguita senza dover sospendere i servizi, grazie a soluzioni di **autoscaling** che permettono l'adattamento delle risorse in tempo reale.

La **scalabilità funzionale** riguarda la capacità di adattare le funzionalità della piattaforma in base alle nuove esigenze aziendali o di mercato. Ad esempio, l'integrazione di nuovi strumenti di analisi avanzata, l'aggiunta di moduli di machine learning o l'espansione verso nuove aree come l'analisi predittiva. Questi sviluppi richiedono una piattaforma scalabile non solo in termini di risorse, ma anche a livello di interoperabilità e flessibilità. Ad esempio, una piattaforma con un'infrastruttura flessibile può facilmente adottare nuove tecnologie senza doversi preoccupare di ricostruire tutto il sistema da zero.

La **scalabilità nel cloud** diventa dunque il perno di una strategia di crescita sostenibile. Grazie alla sua flessibilità e capacità di ridurre i costi fissi, le soluzioni cloud consentono alle organizzazioni di espandere le loro operazioni in maniera fluida, rispondendo velocemente a nuovi bisogni di mercato. Le piattaforme come **Microsoft Azure**, grazie alle loro funzioni di elasticità e autoscaling, permettono di adattare dinamicamente il numero di macchine virtuali o la capacità di storage senza la necessità di un intervento manuale continuo, semplificando l'operatività e migliorando l'efficienza complessiva.

Una piattaforma scalabile richiede una **manutenzione continua** e strutturata per mantenere prestazioni ottimali e garantire che le operazioni procedano senza interruzioni. Con l'aumento del volume dei dati e della complessità delle architetture, diventa essenziale adottare strategie di manutenzione che siano non solo **reattive**, ma soprattutto **preventive**.

La **manutenzione preventiva** permette di identificare potenziali problemi prima che

si verifichino guasti o rallentamenti, riducendo così i tempi di inattività. Questo approccio implica il monitoraggio regolare delle prestazioni, l'applicazione di aggiornamenti di sistema e la gestione accurata dei backup per garantire la protezione dei dati. Inoltre, l'uso di strumenti automatizzati di monitoraggio consente di rilevare anomalie in tempo reale, come picchi anomali di traffico o utilizzo sproporzionato delle risorse, rendendo possibile l'intervento tempestivo. In ambienti cloud, la manutenzione è resa più efficace grazie all'automazione. Le piattaforme cloud permettono di eseguire aggiornamenti automatici, bilanciare i carichi di lavoro e distribuire le risorse in modo dinamico, senza richiedere un intervento manuale costante. Questa automazione non solo migliora l'efficienza, ma riduce anche il rischio di errore umano, garantendo tempi di inattività minimi e massimizzando l'affidabilità del sistema.

Infine, la **manutenzione correttiva** interviene laddove si verificano errori o guasti non previsti, assicurando che i problemi siano risolti nel minor tempo possibile. La combinazione di manutenzione preventiva e correttiva permette di garantire una piattaforma performante e affidabile, capace di supportare in modo continuo le esigenze in crescita dell'azienda.

Capitolo 3

Progettazione

In questo capitolo viene delineata la progettazione del sistema di controllo della qualità dei dati all'interno di un'infrastruttura di Data Analytics basata su Microsoft Azure. Questo lavoro si inserisce nell'ambito del mio tirocinio, durante il quale è stato sviluppato un sistema mirato a garantire una gestione efficace e automatizzata del monitoraggio e controllo dei dati, assicurando al tempo stesso flessibilità e configurabilità in un contesto aziendale complesso.

La sezione 3.1 Architettura approfondisce la struttura tecnologica e i componenti principali che costituiscono l'architettura del sistema, mettendo in luce il ruolo di strumenti come Azure Synapse Dedicated SQL Pool, Data Factory e Power BI. La sezione 3.2 Requisiti descrive i requisiti funzionali e non funzionali necessari per la realizzazione del sistema, specificando le esigenze fondamentali a cui esso deve rispondere. Nel paragrafo 3.3 Funzionalità, si analizzano le principali funzionalità del sistema, evidenziando come esso consenta il controllo continuo della qualità dei dati e la gestione delle anomalie. Infine, la sezione 3.4 Scelte progettuali spiega le ragioni alla base delle decisioni architetturali e tecnologiche, con un focus su aspetti come l'automazione e la configurabilità del sistema.

3.1 Architettura

Il sistema di controllo va ad inserirsi nell'architettura utilizzata dall'azienda, che si basa su **Microsoft Azure**, per gestire i dati del cliente finale in modo tale da utilizzare le stesse tecnologie. Infatti lde strutture e le procedure implementate, che varrano mostrate nei capitoli successivi, sono stati creati tramite l'utilizzo dei componenti che vediamo nell'architettura presente nell'immagine. L'architettura sul quale lavora il nostro sistema è composta da vari livelli e componenti specializzati che interagiscono tra loro per fornire una pipeline di gestione dei dati che va dall'estrazione fino alla visualizzazione e analisi. Una delle caratteristiche fondamentali di questa architettura è la capacità di supportare diverse fasi del ciclo di vita dei dati, dalla loro acquisizione e trasformazione fino all'archiviazione e alla fruizione tramite strumenti di Business Intelligence. In questo senso, l'architettura è progettata per essere modulare, con componenti dedicati per ciascuna fase del processo, e per garantire che le informazioni siano sempre disponibili in tempo reale e nel formato più adatto all'analisi.

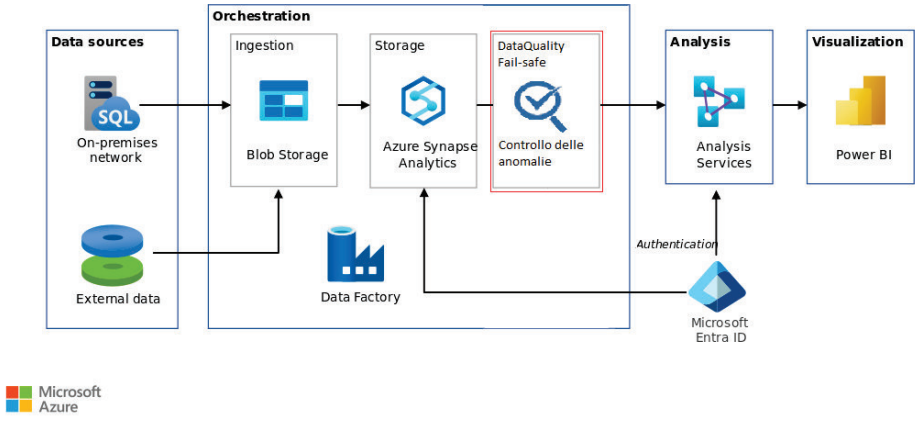

Figura 3.1: Architettura azure

La soluzione proposta si integra nell'architettura illustrata nell'immagine attraverso l'uso di componenti come **Azure Dedicated SQL Pool, Azure Data Factory, Power BI**.

Azure Dedicated SQL Pool viene utilizzato per creare le strutture necessarie per il controllo e la gestione delle anomalie. Azure Data Factory gestisce il caricamento e la trasformazione dei dati. Per quanto riguarda Power BI invece, anche se attualmente non viene utilizzato nella nostra soluzione, è già integrato nell'architettura per il monitoraggio dei caricamenti di determinate tabelle.

Il nostro sistema di controllo si iterpone, in fase di **orchestrazione**, fra la memorizzazione del dato nel sistema e l'effettivo consumo del dato in fase di analisi.
Diamo a questo punto una descrizione di quelli che sono i componenti utilizzati nell'architettura e un cenno iniziale al loro scopo.

- **Azure Dedicated SQL Pool – Storage dei dati.** Azure Dedicated SQL Pool, nella soluzione proposta, viene utilizzato per creare le strutture necessarie per il controllo e la gestione delle anomalie. Le strutture principali sono la tabella di configurazione dei controlli e la tabella di esito, metre per la gestione delle anomalie, dal punto di vista del coordinamento, implementiamo una store procedure. L'Azure Dedicated SQL Pool rappresenta una componente essenziale per lo storage dei dati nelle architetture di data platform. Si tratta di un database distribuito, parte del servizio Azure Synapse Analytics, progettato per gestire enormi quantità di dati strutturati. Il database è dedicato, il che significa che le risorse sono assegnate specificamente al pool, garantendo un controllo preciso sulle prestazioni e sulla capacità di elaborazione. Una delle caratteristiche chiave di SQL Pool è la sua architettura **massivamente parallela (MPP)**, che consente l'esecuzione simultanea di operazioni su più nodi. Questo approccio incrementa drasticamente la velocità di elaborazione di query complesse, soprattutto su dataset di grandi dimensioni. Inoltre, il sistema supporta la scalabilità elastica, permettendo alle aziende di aumentare o diminuire la capacità di storage e calcolo in base alle esigenze in tempo reale. Questa flessibilità rende SQL Pool particolarmente adatto

a scenari di Data Analytics, dove i volumi di dati possono variare notevolmente. La distribuzione su più nodi garantisce l'esecuzione parallela delle operazioni, aumentando notevolmente la velocità di accesso ai dati e migliorando le prestazioni durante i carichi di lavoro più impegnativi. La configurazione cloud consente inoltre di ridurre i tempi di inattività e i costi operativi, rendendo SQL Pool una soluzione ideale per gestire sia dati operativi che analitici.

- **Azure Data Factory.** Nel contesto della nostra soluzione, viene utilizzato per inserire un'activity che richiama la stored procedure subito dopo il caricamento delle tabelle oggetto di monitoraggio, per eseguire i controlli sui dati e una pipeline per l'invio di notifiche email in caso di rilevamento di anomalie, informando gli amministratori del sistema. L'Azure Data Factory (ADF) è la piattaforma di integrazione dati di Azure che svolge un ruolo cruciale nell'orchestrazione dei processi ETL (Extract, Transform, Load). Il suo scopo principale è quello di raccogliere dati da diverse fonti (on-premises, cloud, database relazionali, file non strutturati) e trasformarli in modo da renderli utilizzabili per l'analisi. ADF è altamente configurabile, supporta diversi tipi di trasformazioni dati, e consente di creare pipeline complesse che gestiscono il flusso dati end-to-end. Queste pipeline possono essere eseguite in modalità batch o in tempo reale, consentendo l'elaborazione di dati in streaming. Uno degli aspetti distintivi di Azure Data Factory è l'integrazione con servizi di intelligenza artificiale e machine learning, che consente l'applicazione di modelli predittivi durante la fase di trasformazione. ADF è anche pensato per il monitoraggio e la gestione automatica. Attraverso una dashboard centralizzata, gli utenti possono tenere traccia dello stato delle pipeline, ricevere notifiche in caso di errori o fallimenti, e monitorare la qualità dei dati elaborati. Questo strumento si integra perfettamente con altre risorse di Azure, come il SQL Pool o Azure Blob Storage, rendendo fluido il passaggio dei dati attraverso diversi sistemi.

- **Microsoft Power BI.** Microsoft Power BI è una soluzione di business intelligence che permette di visualizzare i dati tramite dashboard interattivi e report personalizzati. È ampiamente utilizzato per fornire insight in tempo reale e per prendere decisioni aziendali basate su dati accurati e aggiornati. Power BI consente di collegarsi a diverse fonti di dati, tra cui database SQL, fogli di calcolo, applicazioni cloud e piattaforme di big data. La sua integrazione con servizi cloud come Azure Synapse Analytics o Azure SQL Pool permette di creare report basati su dati aggiornati in tempo reale, ottimizzando l'analisi dei dati aziendali. Oltre alle funzionalità di base di reporting, Power BI supporta modelli di analisi avanzata attraverso l'integrazione con linguaggi come DAX (Data Analysis Expressions) e la possibilità di eseguire operazioni di machine learning direttamente all'interno della piattaforma. Per ora questo strumento viene utilizzato per monitorare il caricamento di determinate strutture, ma non è ancora integrato nella nostra soluzione.

Il vero punto di forza dell'architettura risiede nell'**interazione** tra i tre componenti principali dell'architettura che stiamo descrivendo (Azure Dedicated SQL Pool, Azure Data Factory, Microsoft Power BI). Azure Data Factory rappresenta il cuore del flusso ETL, governando l'orchestrazione dei dati dall'ingestione alla trasformazione. Una volta che i dati sono passati attraverso le pipeline di Data Factory, essi vengono stoccati nel Dedicated SQL Pool, che funge da repository centralizzato e ottimizzato per l'analisi massiva. Qui, i dati possono essere organizzati, processati in parallelo e resi disponibili per l'analisi successiva. Successivamente, Microsoft Power BI accede direttamente ai dati elaborati e immagazzinati nel SQL Pool per fornire insight significativi. Grazie all'integrazione nativa tra questi strumenti, i dati aggiornati nel pool vengono immediatamente riflessi nei report e nelle dashboard creati in Power BI, permettendo una visualizzazione dinamica e in tempo reale. Questo processo rende fluido il passaggio dalle fasi di trattamento tecnico

dei dati alle analisi a valore aggiunto, supportando decisioni aziendali strategiche.

3.2 Requisiti

Per la progettazione di un sistema di fail-safe all'interno di una data platform, è di primaria importanza stabilire chiaramente i requisiti funzionali e non funzionali. I requisiti funzionali definiscono le operazioni specifiche che il sistema deve eseguire per gestire e mitigare le anomalie nei dati, assicurando una risposta adeguata ai problemi identificati. I requisiti non funzionali, invece, riguardano le caratteristiche come la performance, la sicurezza e la scalabilità, che determinano la capacità del sistema di mantenere l'efficienza e l'affidabilità anche in condizioni di carico variabile e situazioni impreviste. La seguente sezione esplorerà in dettaglio questi requisiti, fornendo una base solida per la progettazione e l'implementazione del sistema.

3.2.1 Requisiti funzionali

Nell'ambito della progettazione di un sistema di controllo e monitoraggio dei dati, i requisiti funzionali tracciano le specifiche necessarie affinché il sistema esegua correttamente le sue operazioni. Questi requisiti si concentrano sulle funzionalità pratiche, come la configurazione, l'automazione dei controlli, la gestione dei dati anomali, l'invio di notifiche e l'aggiornamento incrementale del sistema, tutti elementi cruciali per garantire la qualità e l'affidabilità del processo di elaborazione dati nel contesto di una data platform.

1. **Configurabilità del controllo dati:** Il sistema deve permettere la configurazione di controlli tramite l'inserimento di parametri come l'ID del controllo, la tabella da monitorare, la colonne chiave, la query di controllo, la tipologia del controllo e il messaggio di errore.

La configurabilità è un aspetto fondamentale in un sistema di monitoraggio dei dati, poiché consente agli utenti di definire e adattare i controlli senza dover riprogettare l'intera infrastruttura. Questa capacità è cruciale in ambienti aziendali complessi e in continua evoluzione, dove le fonti di dati, le tabelle e le regole di controllo possono cambiare nel tempo. La configurabilità permette di ridurre il tempo necessario per implementare nuove verifiche e consente agli amministratori di sistema di rispondere rapidamente a nuove esigenze aziendali o a cambiamenti normativi. Avere una **tabella comune di configurazione** consente di gestire tutto da un unico punto, migliorando la tracciabilità e la governance delle modifiche ai controlli. La configurazione tramite un'interfaccia centralizzata garantisce una gestione semplificata, permettendo agli amministratori di apportare modifiche con minima interferenza operativa. La facilità di configurazione evita interventi su codice e permette una gestione più agile dei requisiti di monitoraggio.

2. **Automatizzazione delle Procedure di Controllo:** Il sistema deve eseguire automaticamente il controllo dei dati in fase di caricamento, integrando le pipeline di Azure Data Factory per gestire l'esecuzione della **Stored Procedure (SP) di controllo** e la segnalazione via mail in caso di scarti.

   L'automatizzazione gioca un ruolo essenziale nel garantire che i controlli di qualità sui dati siano eseguiti in modo costante e senza intervento manuale. Automatizzare questi controlli tramite strumenti come Azure Data Factory o altre pipeline di elaborazione dati significa garantire un monitoraggio continuo e in tempo reale della qualità dei dati. La configurazione di una Stored Procedure (SP) che gestisce l'esecuzione dei controlli offre un approccio modulare e riutilizzabile, riducendo la necessità di scrivere codice personalizzato per ogni controllo. L'automazione riduce significativamente il rischio di errore umano e migliora la coerenza dei controlli applicati ai dati. Una pipeline automatizzata permette di identificare e isolare

immediatamente i dati non conformi prima che possano propagarsi a valle del processo di analisi. Questo è particolarmente importante nei contesti di grandi volumi di dati, dove l'analisi manuale è impraticabile e inefficace.

3. **Gestione dei Dati Anomali:** Il sistema deve individuare, tramite la query di controllo, le righe di dati anomali dalle pipeline di caricamento e memorizzarle nella **tabella di esito/scarto.**

   La capacità di gestire i dati anomali in modo proattivo è un elemento essenziale di un sistema di monitoraggio. Questa capacità evita che i dati anomali influenzino negativamente le analisi o i risultati operativi. L'architettura proposta prevede l'uso di una tabella di esito/scarto, in cui i dati anomali vengono isolati per ulteriori revisioni o correzioni.

   Grazie all'informazione che ci indica la **tipologia del controllo** è possbile gestire i dati anomali in tre modi differenti:

   - Eseguendo la query di controllo SQL, si vanno a salvare i valori problematici nella tabella di esito/scarto, ma non si escudono automaticamente dalla tabella sulla quale viene effettuato il controllo. Se necessario, si aggiornerà manualmente la struttura di caricamnto dei dati (come viste, stored procedure, ecc.) affinché vengano escluse le righe segnalate nella tabella di esito.

   - La seconda tipologia di controllo prevede l'esecuzione della query di controllo SQL, il salvataggio dei valori problematici nella tabella di esito/scarto e dell'intera riga della tabella in una parallela tabella di errore. Successivamnete avviene la rimozione automatica dei valori problematici dalla struttura oggetto di controllo.

- Come ultima tipologia di controllo si prevede la possibilità di eseguire una StoreProcedure al posto di una semplice query SQL per i casi in cui è necessario poter eseguire un elenco di operazioni articolato e più complesso rispetto ad una query di controllo. I valori saranno salvati nella tabella degli esiti, e l'eventuale rimozione dei valori anomali dalla tabella oggetto di controllo è definibile all'interno delle logiche della SP stessa.

Non solo i dati scartati vengono monitorati e archiviati, ma si garantisce anche che l'intero processo resti stabile, con una chiara visibilità su dove e perché i dati siano stati rifiutati. Questo approccio consente un'ulteriore analisi dei motivi di scarto e facilita l'intervento correttivo da parte degli amministratori.

4. **Invio di Notifiche di Anomalie:** Se si verificano anomalie, il sistema deve inviare notifiche via mail con informazioni personalizzate sul messaggio di errore.

   Il sistema proposto include l'invio automatico di notifiche via e-mail agli amministratori o ai responsabili, permettendo loro di agire immediatamente sugli errori identificati. L'utilizzo del messaggio di errore personalizzato configurato nella tabella di controllo consente di fornire informazioni specifiche e contestuali, migliorando la comprensione dell'anomalia e accelerando la risoluzione.

5. **Aggiornamento Incrementale del Sistema:** Il sistema deve supportare l'aggiunta di nuovi controlli sia su strutture di dati nuove che su quelle già monitorate, senza dover modificare significativamente la struttura di caricamento.

   Questa funzionalità garantisce la continuità operativa e mantiene l'efficienza nel lungo periodo. In contesti di dati complessi, nuovi controlli possono essere richiesti a seguito di cambiamenti nei requisiti aziendali o nelle normative di settore. Il sistema proposto offre un metodo semplice per aggiungere questi controlli e aggiornarli all'interno della pipeline di elaborazione dati esistente, senza richiedere

modifiche estese. L'approccio modulare e incrementale garantisce che il sistema possa evolversi e adattarsi rapidamente ai nuovi scenari operativi.

3.2.2 Requisiti non funzionali

I requisiti non funzionali riguardano aspetti di qualità come la scalabilità, la sicurezza e la manutenibilità del sistema. Tali requisiti assicurano che il sistema possa crescere e adattarsi ai volumi di dati crescenti, mantenendo al contempo un elevato livello di prestazioni e una facile gestione nel lungo termine.

1. **Affidabilità:** Il sistema deve garantire che i controlli siano eseguiti in maniera robusta e che i dati anomali vengano gestiti in modo da non compromettere la qualità complessiva del processo di elaborazione. L'affidabilità si traduce non solo nella capacità di monitorare e segnalare errori, ma anche nel mantenimento della continuità operativa, indipendentemente da eventuali anomalie riscontrate nei dati.

2. **Manutenibilità:** Il sistema deve permettere aggiornamenti e modifiche continue ai controlli senza impattare negativamente sulle operazioni esistenti, facilitando l'aggiunta di nuovi controlli. Un sistema facilmente manutenibile permette di aggiungere nuovi controlli e monitoraggi, o modificare quelli esistenti, senza la necessità di intervenire sull'intera struttura. Ciò implica che ogni nuovo controllo può essere configurato e integrato nel flusso di lavoro esistente in modo semplice, riducendo al minimo il rischio di errori e abbassando i costi legati all'assistenza tecnica.

3. **Scalabilità:** Deve essere possibile adattare il sistema a volumi crescenti di dati senza perdita di performance, mantenendo la velocità e l'accuratezza delle pipeline di controllo. Il sistema deve poter gestire un aumento del carico di lavoro sia in termini di quantità di dati processati che di nuovi controlli implementati. Questa

scalabilità non deve però compromettere le prestazioni del sistema, che devono rimanere costanti anche con una maggiore complessità operativa.

4. **Sicurezza:** Il sistema deve garantire che solo utenti autorizzati possano accedere e modificare le configurazioni di controllo, oltre a proteggere le informazioni sensibili all'interno delle tabelle di esito e configurazione.

3.3 Funzionalità

L'obiettivo del sistema sviluppato è assicurare che i dati siano monitorati costantemente, eseguendo controlli di qualità in tempo reale. Grazie a un approccio configurabile e automatizzato, l'architettura proposta facilita la gestione delle anomalie, avvisando tempestivamente e supportando le azioni correttive da parte degli amministratori del sistema. Di seguito, esploreremo due principali funzionalità: **Configurazione di controllo su nuova struttura** e **Configurazione di nuovo controllo su struttura già monitorata.**

Queste funzionalità nascono per supportare i team di manutenzione della data platform nelle operazioni di gestione dei dati anomali. Prima della nascita di questo sistema, quando si ricontravano errori o anomalie nelle pipeline dovute alla qualità dei dati che atterravano nella piattaforma, il team di manutenzione doveva per prima cosa individuare il punto critico nel quale il dato anomalo veniva caricato e successivamente doveva scartare le righe che causavano l'errore. Queste operazioni avvenivano in maniera manuale senza nessun supporto e senza nessun meccanismo di log che potesse in qualche modo documentare quanto successo, per poi essere condiviso anche con chi generava la sorgente dati al fine di correggere l'errore a monte. Da queste criticità nasce l'esigenza del sistema sviluppato, per avere una struttura di monitoraggio configurabile che sia in

grado di supportare il team di manutenzione, e non solo, a comprendere e a condividere le cause e le soluzioni alle problematiche riscontrate.

3.3.1 Configurazione di controllo su nuova struttura.

Si descrive il processo per configurare un controllo dei dati su una nuova struttura non monitorata precedentemente. Il sistema permette di definire controlli personalizzati e configurabili, che vengono inseriti in una tabella dedicata e successivamente richiamati attraverso pipeline di elaborazione dei dati. Ogni passaggio, dalla configurazione del controllo all'aggiornamento delle pipeline, è pensato per garantire che eventuali anomalie nei dati vengano rilevate e gestite in modo tempestivo.

Fase 1: Configurazione del controllo. Inizialmente, per monitorare una nuova struttura, è necessario inserire una riga nella **tabella comune di configurazione.** Questo permette di definire il controllo che verrà eseguito, specificando i dettagli chiave come l'ID del controllo, la tabella su cui si eseguirà il controllo, la colonna o le colonne chiave interessate, il messaggio di errore per le segnalazioni, la tipologia di controllo e infine la query di controllo che valuterà i dati secondo le regole configurate.

Fase 2: Aggiornamento della pipeline in Azure Data Factory. Dopo aver configurato il controllo, è necessario aggiornare la pipeline di Azure Data Factory. In questa fase, viene aggiunto manualmente un richiamo alla **Stored Procedure comune**, che verrà eseguita dopo il caricamento dei dati della struttura da monitorare. La SP richiamerà l'ID del controllo impostato nella tabella di configurazione, eseguendo i check di qualità sui dati.

Fase 3: Notifiche tramite email. Successivamente, viene aggiunta una seconda activity per inviare una **notifica email.** Questa notifica, impostata per essere inviata immediatamente dopo l'esecuzione della SP, segnala se il controllo ha prodotto delle righe

di scarto. Come testo dell'email, viene utilizzato il messaggio di errore precedentemente configurato nella tabella.

Fase 4: Aggiornamento della struttura di caricamento. Questa fase dipende dalla tipologia di controllo che viene settata nella fase 1. Esiste la possibilità di scartare automaticamente i dati anomali scegliendo una determinata tipologia di controllo o, altrimenti, bisogna aggiornare manualmente la struttura di caricamento dei dati se vi è la necessità di intervenire su viste o su StoreProcedure particolari.

3.3.2 Configurazione di nuovo controllo su struttura già monitorata.

In questo caso si illustrano le varie fasi necessarie all'estensione del sistema di controllo dei dati su strutture già sottoposte a monitoraggio. Viene spiegato come aggiungere nuovi controlli alle pipeline esistenti, integrando nuovi criteri di monitoraggio senza interferire con i controlli già attivi. Anche in questo caso, il sistema prevede la possibilità di notifiche email in caso di dati anomali, ma sfrutta la struttura già esistente per semplificare l'implementazione.

Fase 1: Configurazione del controllo. La prima fase consiste nell'inserire una nuova riga nella tabella di configurazione comune, specificando i parametri necessari, come l'ID del controllo, la tabella oggetto del controllo, e la colonna chiave che si intende monitorare. Nel caso di una struttura già monitorata, la **colonna chiave** può coincidere con quella già in uso per controlli precedenti, riducendo la necessità di modificare ulteriormente la struttura esistente.

Anche in questa fase viene definito il messaggio di errore che sarà incluso nella notifica email in caso di rilevazione di anomalie. La tipologia di controllo e la query di controllo rappresentano gli aspetti chiave di questa configurazione, poiché viene progettata per

intercettare specifiche casistiche di dati anomali che potrebbero non essere state coperte dai controlli esistenti.

Fase 2: Aggiornamento della pipeline in Azure Data Factory. Per integrare il nuovo controllo nella pipeline di Azure Data Factory, occorre intervenire manualmente. La pipeline esistente, già configurata per eseguire altri controlli, viene aggiornata aggiungendo il richiamo alla Stored Procedure comune (SP) con **l'ID del nuovo controllo.** In questo modo, la SP eseguirà sia il controllo precedente che quello nuovo in sequenza, garantendo un'architettura modulare e facilmente espandibile.

Fase 3: Notifiche tramite email. Come per le strutture non monitorate, viene aggiornato il nodo di invio delle email di notifica. Il sistema è configurato per includere il nuovo messaggio di errore nel caso in cui il nuovo controllo produca righe di scarto. Questo meccanismo assicura che eventuali anomalie nei dati vengano comunicate tempestivamente, permettendo interventi correttivi mirati.

Fase 4: Aggiornamento della struttura di caricamento Se il nuovo controllo impone criteri di esclusione bisogna aggiornare la struttura di caricamento dei dati (vista, stored procedure, ecc.) con le metodologie illustrate nella fase 4 della configurazione precedente.

3.4 Scelte progettuali

Nel progettare il sistema di qualità del dato, è stato fondamentale integrare la soluzione in un'infrastruttura già esistente e ottimizzare il processo che in passato richiedeva interventi manuali da parte degli amministratori. L'obiettivo principale era quello di rendere il monitoraggio e la gestione delle anomalie nei dati più **automatizzati** e **configurabili**, senza stravolgere la struttura esistente, ma introducendo strumenti per una maggiore efficienza e controllo.

Le **scelte architetturali** si sono basate sulla necessità di integrare un sistema di controllo dei dati in una piattaforma esistente, senza richiedere modifiche invasive. L'architettura si è costruita attorno all'uso di una struttura centralizzata per la gestione dei controlli, con una singola tabella di configurazione e una tabella di esito/scarto. Questo approccio garantisce una separazione chiara tra la logica di business e l'infrastruttura di controllo, mantenendo una chiara distinzione tra i dati gestiti e i controlli sui dati stessi.

L'utilizzo di una Stored Procedure centralizzata permette l'esecuzione dei controlli in maniera uniforme, indipendentemente dalla specifica tabella o struttura dati monitorata. La scelta di utilizzare una procedura centralizzata permette di avere una gestione univoca dei controlli, semplificando l'amministrazione e riducendo la duplicazione di codice. In questo modo, ogni nuovo controllo può essere facilmente aggiunto semplicemente aggiornando le tabelle di configurazione, lasciando invariata la logica applicativa.

La struttura scelta permette inoltre di scalare orizzontalmente il sistema, aggiungendo nuovi flussi di dati senza impatti sulla gestione dei controlli, garantendo così flessibilità e riduzione del carico operativo. Questa modularità, quindi, facilita la manutenibilità del sistema, rendendolo adatto anche per ambienti di crescita rapida.

Dal punto di vista **dell'automazione**, il sistema è stato progettato per ridurre l'intervento manuale da parte degli amministratori del sistema, grazie all'integrazione con le pipeline di Azure Data Factory. In particolare, l'automazione si concretizza nella gestione delle notifiche di errore via email, che informano immediatamente gli operatori in caso di anomalie nei dati, eliminando la necessità di monitorare manualmente i processi.

L'automazione dei controlli non elimina, però, la possibilità di intervenire manualmente dove necessario. Il processo di scarto dei dati non conformi viene segnalato in automatico, ma rimane sotto il controllo dell'operatore, che può decidere quando e come escludere i dati anomali. Questo **approccio ibrido** tra automazione e intervento umano garantisce una maggiore sicurezza e controllo sui processi critici.

La **configurabilità** del sistema è stata un punto chiave. Attraverso una semplice modifica nelle tabelle di configurazione, è possibile aggiungere nuovi controlli senza toccare il codice sorgente. Tuttavia, un altro aspetto importante della configurabilità è stato mantenere un bilanciamento tra flessibilità e semplicità. Per esempio, il sistema è progettato per essere facile da usare per i team operativi, che possono modificare parametri di controllo senza bisogno di supporto tecnico avanzato, rendendo così la soluzione accessibile anche per utenti non tecnici.

Capitolo 4

Implementazione

In questo capitolo viene descritta nel dettaglio l'implementazione del sistema di controllo della qualità dei dati, che rappresenta una parte fondamentale dell'infrastruttura di gestione dei dati. L'implementazione è stata progettata per essere modulare, automatizzata e altamente configurabile, sfruttando tecnologie di Microsoft Azure, come Azure Dedicated SQL Pool e Azure Data Factory.

Nel paragrafo dedicato alle **tecnologie utilizzate**, viene illustrata la scelta e il ruolo delle tecnologie principali impiegate nell'implementazione. Ognuna di queste tecnologie è stata selezionata per soddisfare requisiti specifici del sistema, come la gestione di grandi volumi di dati, l'automazione dei flussi ETL (Extract, Transform, Load), e la visualizzazione dei risultati. A seguire si mostra la **struttura dell'implementazione**, viene descritto il flusso dei dati e l'architettura implementata. Si illustrano i vari livelli e componenti coinvolti, spiegando come ciascuno di essi interagisce con gli altri per assicurare l'integrità e la qualità dei dati elaborati. Vengono presentati i diversi flussi operativi basati sul tipo di controllo (come SQL, SP, o SQL_KEYS), ognuno dei quali gestisce scenari specifici di verifica e registrazione degli esiti. Dopo aver descritto quali

sono le tecnologie alla base e com'è strutturata l'implementazione, si passa alla **descrizione dell'implementazione** che si concentra sugli elementi chiave implementati, come le tabelle di anagrafica e di scarto, la Store Procedure Comune che esegue i controlli e le pipeline configurabili in Azure Data Factory per la gestione automatizzata dei flussi di dati e delle notifiche. Viene data particolare attenzione ai processi di controllo, come la gestione dei controlli SQL standard, l'esecuzione delle Stored Procedure personalizzate, e la verifica delle chiavi in SQL_KEYS. Infine,tramite i **frammenti di codice**, si forniscono esempi pratici dei componenti chiave dell'implementazione. Vengono inclusi i codici di creazione delle tabelle, la Store Procedure Comune che esegue i controlli, e i frammenti di codice delle pipeline di Azure Data Factory, fornendo una visione tecnica del funzionamento del sistema.

4.1 Tecnologie utilizzate

L'implementazione del sistema di qualità dei dati ha richiesto l'integrazione di tre tecnologie principali che hanno contribuito alla costruzione di una soluzione completa: Azure Dedicated SQL Pool e Azure Data Factory. Questi strumenti, già presentati nel contesto architetturale, sono stati utilizzati in modo specifico e mirato durante la fase di implementazione per garantire la corretta gestione, monitoraggio dei dati problematici.

- **Azure Dedicated SQL Pool.** In Azure Dedicated SQL Pool, la fase implementativa ha visto la creazione di tre componenti principali legati al sistema di controllo della qualità dei dati:

    1. **Tabella di anagrafica dei controlli:** Questa tabella contiene le informazioni su tutti i controlli configurati. Ogni controllo ha un set di parametri specifici, come il tipo di controllo e le regole applicate. La sua funzione è quella di centralizzare e rendere configurabili i controlli su diversi dataset,

permettendo così una gestione flessibile e scalabile. È qui che vengono definiti i dati relativi a ogni controllo utili all'esecuzione corretta dell'intero flusso.

2. **Tabella dei log degli esiti del controllo:** Questa tabella raccoglie i risultati di ogni esecuzione dei controlli. Viene popolata dopo che le procedure di controllo sono state eseguite, registrando gli esiti sia positivi che negativi. Questo sistema di logging consente agli amministratori e ai tecnici di monitorare l'andamento dei controlli nel tempo e di individuare eventuali criticità nei dati. Inoltre, permette di avere una tracciabilità storica dei controlli effettuati.

3. **Tabella parallela di errore** (utilizzata solo in determinati casi): Questa tabella entra in gioco esclusivamente quando il tipo di controllo (definito tramite il campo "CheckType" nella tabella di anagrafica) è di tipo "SQL_KEYS". In questo scenario, la tabella di errore serve a registrare le righe dei dati che non passano i controlli delle chiavi SQL, consentendo così di isolare e eliminare le anomalie in modo mirato. Questo approccio garantisce che i dati non conformi non entrino nel sistema principale fino a che non vengono risolti.

Questi elementi del sistema implementato in Azure Dedicated SQL Pool assicurano un controllo robusto e adattabile sulla qualità dei dati, permettendo di gestire facilmente anomalie e fornendo un livello di automazione che prima non era possibile.

- **Azure Data Factory (ADF).** In Azure Data Factory (ADF), l'implementazione ha riguardato principalmente l'orchestrazione delle pipeline per gestire il flusso di dati e l'esecuzione automatizzata dei controlli di qualità sui dati. Nella pipeline di caricamento dati, vengono utilizzate due Activity configurabili, che svolgono un ruolo chiave nella verifica dei dati caricati.

1. **Activity di gestione del controllo:** Questa attività è configurabile e viene inserita subito dopo l'attività di caricamento dati. Serve per eseguire la procedura di controllo dei dati appena caricati. Il controllo utilizza la store procedure comune che interroga le tabelle di configurazione precedentemente create. In questo modo, viene garantito che i dati caricati rispettino le regole definite nella tabella di configurazione.

2. **Activity di notifica invio email:** L'activity controlla l'esito del controllo e ne caso in cui il controllo ha lo status "KO", invia un'email contenente i dettagli delle anomalie riscontrate. Questo passaggio automatizzato assicura che il team di gestione riceva notifiche tempestive, permettendo loro di intervenire in maniera rapida e risolvere eventuali problematiche con i dati.

Azure Data Factory (ADF) non solo consente l'automazione dei processi di controllo e di gestione dei dati, ma permette anche di semplificare la gestione di scenari più complessi, come i controlli basati su chiavi. In questo contesto, oltre alle due Activity descritte in precedenza, è stata implementata una pipeline aggiuntiva che automatizza l'avvio dei controlli di chiave (CheckType "SQL_KEYS"). Questa pipeline è stata progettata per avviare il controllo "SQL_KEYS", se presente, per la tabella che viene passata come parametro di ingresso. Questo approccio centralizzato facilita l'esecuzione automatica di controlli complessi, riducendo la necessità di intervenire manualmente ogni volta che è richiesto un controllo di integrità basato su chiavi. In questo modo, ADF gestisce non solo i controlli di qualità standard sui dati, ma anche scenari più complessi legati alla verifica delle chiavi, ottimizzando i processi di monitoraggio e controllo all'interno della data platform.

4.2 Struttura dell'implementazione

L'implementazione del sistema di controllo della qualità dei dati si basa su una serie di passaggi configurabili e automatizzati all'interno di Azure Data Factory (ADF). Il processo di controllo prevede due flussi distinti, dipendenti dal tipo di controllo definito nella tabella di configurazione, ovvero **SQL/SP** o **SQL_KEYS.**

- **Flusso di controllo per CheckType = "SQL" o CheckType = "SP".**

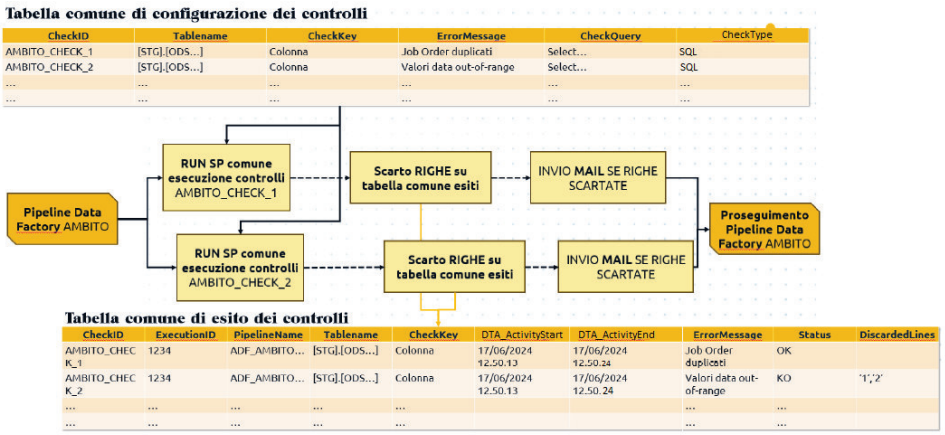

Figura 4.1: Diagramma di flusso tipo 1

Il primo flusso è attivato quando il **CheckType** è impostato su **SQL** o **SP**, il che indica che il controllo viene eseguito tramite query SQL o Stored Procedure personalizzate. Questo flusso segue i seguenti passaggi:

1. **Esecuzione del Controllo:** Dopo il caricamento dei dati, ADF esegue una Stored Procedure (SP) comune che gestisce l'attività di controllo configurata, eseguendo una query SQL per verificare l'integrità o la coerenza dei dati in base ai criteri definiti nella tabella di configurazione dei controlli.
2. **Scarto delle Righe Non Conformi:** Se il controllo rileva anomalie nei dati, vengono tracciati gli ID dele righe problematiche e registrati nella tabella comune degli esiti dei controlli, dove vengono memorizzati l'ID del controllo,

il nome della pipeline, l'intervallo temporale di esecuzione, e lo stato finale (esito OK o KO). In caso di fallimento, vengono identificate le righe che hanno causato problemi.

3. **Invio Email:** Se durante il controllo vengono scartate righe, viene generata automaticamente una email che notifica l'errore e include i dettagli sui dati anomali.

4. **Proseguimento della Pipeline:** Dopo il controllo e l'eventuale invio della mail, la pipeline di ADF procede al passaggio successivo nel flusso, che potrebbe includere ulteriori trasformazioni o attività.

- **Flusso di controllo per CheckType = "SQL_KEYS"**

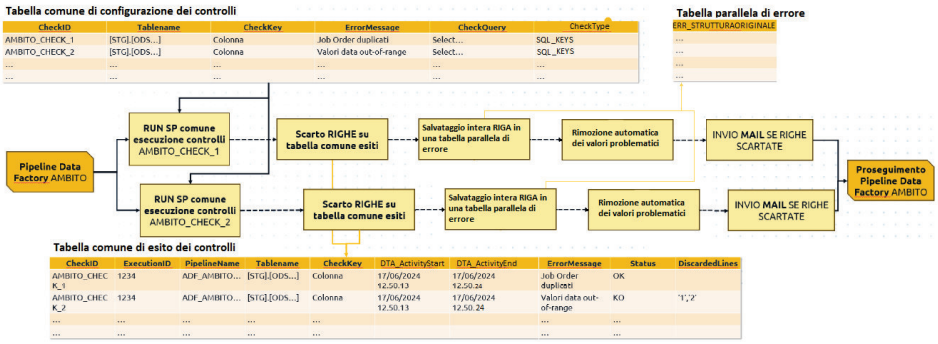

Figura 4.2: Diagramma di flusso tipo 2

Il secondo flusso viene attivato nel caso in cui il tipo di controllo sia SQL_KEYS, utilizzato per controlli su chiavi primarie o duplicati. In questo scenario, oltre ai passaggi sopra descritti, viene eseguita un'attività aggiuntiva:

1. **Esecuzione del Controllo:** Come nel flusso precedente, la pipeline esegue una Stored Procedure per effettuare il controllo SQL definito.

2. **Tabella Parallela di Errore:** In caso siano rilevati errori, non solo vengono tracciati gli ID delle righe problematiche nella tabella degli esiti, ma le righe sono anche salvate in una tabella parallela di errore. Questa tabella

contiene l'intera struttura della riga problematica, permettendo un'analisi più dettagliata degli errori.

3. **Rimozione Automatica:** Le righe contenenti errori vengono automaticamente rimosse dai dati in corso di elaborazione.

4. **Invio Email:** Come per il flusso SQL/SP, viene generata un'email di notifica con i dettagli delle righe anomale.

5. **Proseguimento della Pipeline:** La pipeline prosegue solo dopo che le righe errate sono state salvate e scartate, garantendo che i dati siano stati correttamente trattati prima delle successive fasi di elaborazione.

I diagrammi mostrano visivamente l'interazione tra i diversi componenti del sistema, come la Stored Procedure comune, la tabella degli esiti e la tabella parallela di errore. Questi elementi garantiscono che il flusso dei dati sia monitorato in modo continuo e che ogni anomalia sia prontamente identificata e gestita. Le email di notifica consentono inoltre di mantenere trasparenza e tracciabilità su eventuali problematiche emerse durante il processo.

4.3 Descrizione dell'implementazione

L'implementazione del sistema di controllo della qualità dei dati si basa su una serie di componenti chiave che interagiscono tra loro per assicurare che i dati vengano validati e che eventuali anomalie vengano gestite in modo efficiente. Ogni componente svolge un ruolo specifico all'interno della pipeline di gestione dei dati e viene configurato per garantire l'automazione e la scalabilità delle operazioni.

4.3.1 Tabella di anagrafica dei controlli

La tabella **[CTL].[D_ANAG_CHECKS]** di anagrafica dei controlli rappresenta il cuore del sistema di configurazione. Essa contiene le informazioni necessarie per definire quali controlli devono essere eseguiti sui dati, quali colonne devono essere validate, e il tipo di controllo da applicare. Nella fase di implementazione, è stata creata una tabella dedicata, e successivamente sono state popolate le righe con i dati relativi ai controlli. Questo permette al sistema di eseguire controlli configurabili e modificabili dinamicamente.

La tabella funge da anagrafica dei controlli, all'inserimento di un nuovo controllo viene richiesto di popolare le seguenti informazioni:

- **CheckID:** Identificativo sintetico del controllo.
- **Tablename:** Schema e nome della tabella oggetto del controllo, corredati di parentesi quadre.
- **CheckKey:** Colonne della tabella che sono oggetto del controllo, separate da una virgola.
- **ErrorMessage:** Messaggio sintetico di errore, che viene concatenato al nome della tabella in fase di invio della mail di notifica.
- **CheckQuery:** Query di controllo vera e propria che restituisce in output una sola riga con tutti i valori problematici tramite il meccanismo di **"string_agg"** che popola una variabile **"@retvalOUT"**. In questo campo si può eventualmente indicare una SP da avviare, al posto di una query.
- **CheckType:** Indica la tipologia di controllo. Al momento sono previsti:

1. **SQL.** Tipologia di controllo che prevede l'esecuzione della query SQL, il salvataggio dei valori problematici nella tabella di esito, ma non la loro esclusione automatica dalla tabella oggetto di controllo.
2. **SQL_KEYS.** Tipologia di controllo che prevede l'esecuzione della query SQL, il salvataggio dei valori problematici nella tabella di esito, dell'intera riga della tabella in una parallela tabella di errore e la rimozione automatica dei valori problematici dalla tabella stessa.
3. **SP.** Tipologia di controllo che prevede l'esecuzione di una StoredProcedure, il salvataggio dei valori problematici nella tabella di esito, ma non la loro esclusione automatica dalla tabella stessa (per ora previsto ma non utilizzato).

4.3.2 Tabella di esito

La tabella **[CTL].[LOG_Checks_Discarded_Lines]** di esito viene utilizzata per registrare gli esiti dei controlli eseguiti sui dati. Ogni volta che un controllo viene eseguito, viene inserita una nuova riga nella tabella, la quale riporta l'esito dell'operazione. In particolare, se il controllo non va a buon fine, nella colonna "discardedLines" vengono elencati gli ID delle righe che non hanno superato il controllo, specificando esattamente quali record sono stati scartati. Questo approccio permette un tracciamento dettagliato degli errori, facilitando la successiva analisi e correzione delle anomalie.

La tabella riporta le seguenti infrormazioni:

- **CheckID.** Identificativo del controllo avviato.
- **ExecutionID.** RunID della pipeline da cui è stato avviato il controllo.
- **PipelineName.** Nome della pipeline da cui è stato avviato il controllo.
- **Tablename.** Schema e nome della tabella oggetto del controllo, corredati di parentesi quadre.

- **CheckKey.** Colonne della tabella che sono oggetto del controllo, separate da una virgola.

- **DTA_ActivityStart.** Data e ora di avvio della SP di controllo.

- **DTA_ActivityEnd.** Data e ora di termine della SP di controllo.

- **Status.** OK o KO, in base al rilevamento di valori problematici o meno.

- **ErrorMessage.** Messaggio sintetico di errore, che viene concatenato al nome della tabella in fase di invio della mail di notifica.

- **DiscardedLines.** Valori problematici, popolata in caso di KO del controllo. I valori sono riportati separati da un trattino ("-") e sono riportati nello stesso ordine delle colonne oggetto del controllo, specificate al campo "CheckKey" dell'anagrafica. I valori sono separati da virgole e terminano con un "
" (tag di "a capo" in HTML) per migliorarne la leggibilità da mail di notifica

4.3.3 Store procedure

La stored procedure comune rappresenta il nucleo operativo del sistema di controllo. Essa viene richiamata all'interno delle pipeline di Azure Data Factory e si occupa di eseguire i controlli definiti nella tabella di anagrafica. La procedura gestisce l'intero processo, dall'esecuzione del controllo alla registrazione degli esiti nella tabella di log o di scarto, a seconda dei risultati. L'implementazione di questa procedura consente di centralizzare e riutilizzare il codice, garantendo consistenza e manutenzione facilitata.

Viene eseguita come parte delle pipeline di Azure Data Factory e gestisce diversi tipi di controlli (SQL, SP, SQL_KEYS) in modo centralizzato. Di seguito una descrizione delle operazioni eseguite:

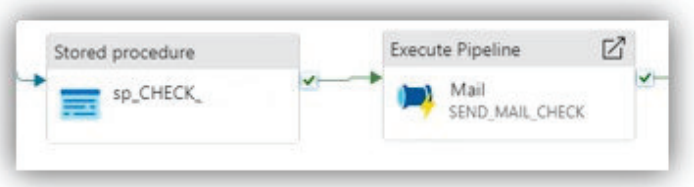

Figura 4.3: Pipeline con richiamo della store procedure e invio mail

1. **Recupero dei parametri di configurazione:** Dopo aver ricevuto in input i parametri: **CheckID, ExecutionID, PipelineName,** inviati dalla pipeline, la procedura inizia recuperando le informazioni necessarie dalla tabella di anagrafica dei controlli. Questo include la query SQL da eseguire **CheckQuery**, il nome della tabella da controllare **TableName**, la chiave del controllo **CheckKey**, il tipo di controllo **CheckType**, e il messaggio di errore da utilizzare in caso di fallimento **ErrorMessage**.

2. **Esecuzione del controllo SQL (CheckType = 'SQL'):** Se il controllo è di tipo SQL, la procedura esegue la query definita nella configurazione per verificare la qualità dei dati. Se vengono rilevati problemi (anomalie o righe che non soddisfano i criteri di controllo), gli ID di queste ultime sono salvati nel risultato della query.

3. **Esecuzione di una Stored Procedure personalizzata (CheckType = 'SP'):** Per i controlli di tipo SP, viene eseguita una Stored Procedure specifica configurata nel sistema. Questa Stored Procedure può contenere logiche più complesse e personalizzate per il controllo dei dati, offrendo una maggiore flessibilità rispetto al controllo SQL diretto.

4. **Gestione dei controlli sulle chiavi (CheckType = 'SQL_KEYS'):** In questo caso, viene eseguita una query che verifica l'integrità delle chiavi primarie o duplicati nei dati. Se vengono trovate anomalie, le righe corrispondenti alle chiavi

problematiche vengono salvate in una tabella parallela di errore. Questo permette di isolarle e successivamente rimuoverle dalla tabella principale.

5. **Popolamento della tabella di esito:** Dopo l'esecuzione del controllo, la procedura aggiorna la tabella di log degli esiti, registrando l'ID del controllo, l'ID dell'esecuzione della pipeline, il nome della pipeline, il nome della tabella su cui è stato eseguito il controllo, la chiave di controllo, la data di inizio e fine del controllo, e lo stato finale (OK o KO).

6. **Aggiornamento del messaggio di errore e righe scartate:** Se il controllo rileva errori, viene aggiornato il messaggio di errore nella tabella di log e vengono registrate le righe scartate nella colonna **DiscardedLines**. Questo fornisce una tracciabilità completa dei controlli falliti e consente agli amministratori di analizzare i dati problematici.

4.3.4 Pipeline di invio mail

Una parte essenziale del sistema è la pipeline che gestisce l'invio di email in caso di errori o anomalie nei dati. Se il controllo dei dati fallisce, la pipeline invia una notifica via email con i dettagli delle righe scartate, permettendo una gestione immediata degli errori. Questa pipeline è altamente configurabile e può essere utilizzata per avvisare i responsabili dei dati o i team operativi in modo automatico.

La pipeline di invio mail utilizza i seguenti parametri:

- **mailBody.** Testo del messaggio inviato tramite mail che viene popolato dal capo ErrorMessage + DiscardedLines se lo status è KO.

- **mailReceipt.** In questo campo inseriamo gli indirizzi email delle fiugure interessate al conrollo dei dati.

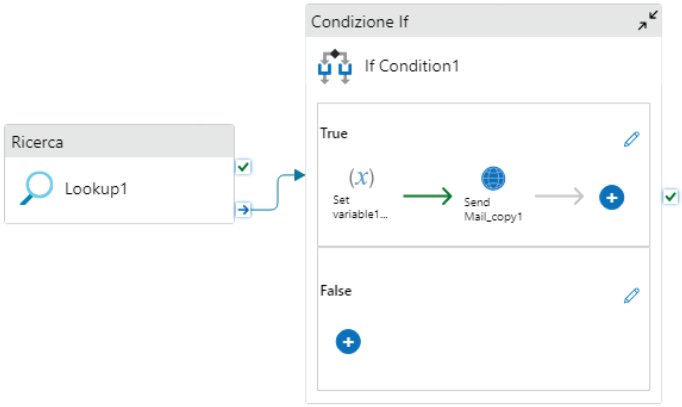

Figura 4.4: Pipeline di invio mail

- **mailSubject.** Inseriamo l'oggetto della mail pertinente con il controllo eseguito.
- **executionID.** il RunId che identifica la pipeline.
- **checkID.** l'id del controllo effettuato.

In fase di lookup tramite il CheckID e l'ExecutionID, viene estratto l'ErrorMessage con la concatenazione del DiscardedLines e dello Status. Quest'ultimo viene controllato nella condizione dell'if e se è KO viene settata la variabile del mailBody con il messaggio di errore e le righe scartate.

4.3.5 Pipeline per i check chiave

In presenza di controlli di tipo **SQL_KEYS**, è stata implementata una pipeline dedicata che gestisce questo tipo di controllo, garantendo che eventuali duplicati o chiavi problematiche vengano identificati e correttamente trattati. Questa pipeline si avvia automaticamente quando nella tabella di anagrafica è presente un controllo di tipo SQL_KEYS e prevede ulteriori passaggi di registrazione e gestione degli errori.

Utilizza i seguenti parametri:

- **DestSchemaName.** Lo schema della tabella di destinazione.

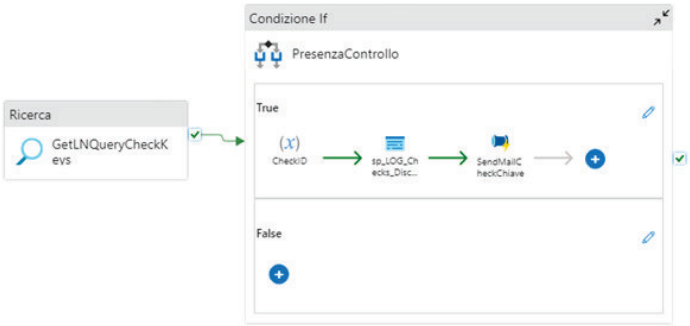

Figura 4.5: Pipeline check chiave

- **runID.** RunID della pipeline da cui è stato avviato il controllo.
- **DestTableName.** Nome della tabella di destinazione.

La pipeline di check chiave si occupa di avviare il controllo di tipo “SQL_KEYS”. Viene assicurata la presenza del controllo di tipo "SQL_KEYS", per la tabella passata come parametro di ingresso, tramite l’activity **GetLNQueryCheckKeys** e successivamente si avvia la store procedure che esegue il controllo con CheckID impostato tramite l’activity **CheckID**. In fine si esegue la pipeline di send mail.

4.4 Frammenti di codice

In questa sezione vengono presentati i principali frammenti di codice che sono stati utilizzati durante l’implementazione del sistema di controllo della qualità dei dati. Questi esempi forniscono una panoramica pratica su come sono state create le tabelle di configurazione e di scarto, come funziona la Store Procedure Comune per l’esecuzione dei controlli, e come le pipeline di Azure Data Factory orchestrano il processo di verifica e notifica. Attraverso questi frammenti di codice, si evidenziano i passaggi chiave dell’implementazione, offrendo una visione dettagliata delle logiche applicative e delle soluzioni adottate per garantire un sistema automatizzato, scalabile e configurabile.

4.4.1 CREATE TABLE

Codice per la creazione della tabella di anagrafica dei controlli:

```
CREATE TABLE [CTL].[D_ANAG_Checks] (
    [CheckID] [NVARCHAR](50) NOT NULL,
    [TableName] [NVARCHAR](150) NULL,
    [CheckKey] [NVARCHAR](200) NULL,
    [ErrorMessage] [NVARCHAR](200) NULL,
    [CheckQuery] [NVARCHAR](4000) NULL,
    [CheckType] [NVARCHAR](100) NULL
)
```

4.1 Codice create table anagrafica controlli

Codice per la creazione della tabella di esito/scarto:

```
 CREATE TABLE [CTL].[LOG_Checks_Discarded_Lines] (
    [CheckID] [NVARCHAR](50) NOT NULL,
    [ExecutionID] [NVARCHAR](100) NOT NULL,
    [PipelineName] [NVARCHAR](100) NOT NULL,
    [TableName] [NVARCHAR](150 NOT NULL,
    [CheckKey] [NVARCHAR](200) NULL,
    [DTA_ActivityStart] DATETIME NOT NULL,
    [DTA_Activity_End] DATETIME NULL,
    [Status] [NVARCHAR](50) NOT NULL,
    [ErrorMessage] [NVARCHAR](2000) NULL,
    [DiscardedLines] [NVARCHAR](MAX) NULL
)
```

4.2 Codice create table tabella esiti

4.4.2 CREATE PROC

```
CREATE PROC [CTL].[sp_LOG_Checks_Discard_Lines] @CheckID [nvarchar](50)
   ↪ ,@ExecutionID [nvarchar](100),@PipelineName [nvarchar](100) AS

BEGIN
        DECLARE @sql as nvarchar(4000)
        DECLARE @TableName as nvarchar(150)
        DECLARE @CheckKey as nvarchar(200)
        DECLARE @CheckType as nvarchar(100)
        DECLARE @ErrorMessage as nvarchar(2000)
        DECLARE @Status as nvarchar(50)

        DECLARE @CheckDateStart datetime
        set @CheckDateStart = (CONVERT(datetime, SYSDATETIMEOFFSET() AT
           ↪  TIME ZONE 'W. Europe Standard Time'))

        IF OBJECT_ID(N'tempdb..##anag_checks', N'U') IS NOT NULL
                DROP TABLE #anag_checks;
        create table #anag_checks with (DISTRIBUTION=ROUND_ROBIN) as
        SELECT * from [CTL].[D_ANAG_Checks] where [CheckID] = @CheckID

        set @sql = (select CheckQuery from #anag_checks)
        set @TableName = (select TableName from #anag_checks)
        set @CheckKey = (select CheckKey from #anag_checks)
        set @CheckType = (select CheckType from #anag_checks)
        set @ErrorMessage = (select REPLACE(TRIM(ErrorMessage),'''',''''
           ↪ ''') as ErrorMessage from #anag_checks)

        print @sql
        print @TableName
```

```
print @CheckKey
print @CheckType
print @ErrorMessage

DECLARE @ParmDefinition nvarchar(500);
DECLARE @retval NVARCHAR(max)
SET @ParmDefinition = N'@retvalOUT NVARCHAR(MAX) OUTPUT';

IF @CheckType = 'SQL'
BEGIN
        print 'Ho rilevato questo CheckType - ' + @CheckType
        EXEC sp_executesql @sql, @ParmDefinition, @retval
           ↪ OUTPUT;
        print @retval
        set @retval = REPLACE(TRIM(@retval),'''','''''')
        print @retval
END

IF @CheckType = 'SP'
BEGIN
        print 'Ho rilevato questo CheckType - ' + @CheckType
        EXEC(@sql, @retval OUTPUT);
        print @retval
        set @retval = REPLACE(TRIM(@retval),'''','''''')
        print @retval
END

IF @CheckType = 'SQL_KEYS'
BEGIN
        print 'Ho rilevato questo CheckType - ' + @CheckType
```

```
EXEC sp_executesql @sql, @ParmDefinition, @retval
    ↪ OUTPUT;
print @retval
set @retval = REPLACE(TRIM(@retval),'''','''''')
print @retval

DECLARE @deleteSQL as nvarchar(max)
DECLARE @KeysToDelete as nvarchar(max)
DECLARE @errTable as nvarchar(500)

IF (@retval is not null)
BEGIN

        set @KeysToDelete = cast(CONCAT('''',REPLACE(
            ↪ REPLACE(@retval, '<br>', ''), ',', '''
            ↪ union all select '''), '''') as NVARCHAR(
            ↪ MAX))
        print @KeysToDelete

        set @errTable = REPLACE(@TableName, '].[', '].[
            ↪ ERR_')
        print @errTable

        DECLARE @dropSQL nvarchar(max)
        set @dropSQL= 'IF OBJECT_ID(N'''+ @errTable+'''
            ↪ , N''U'') IS NOT NULL DROP TABLE '+
            ↪ @errTable+';'
        print @dropSQL
        EXEC sp_executesql @dropSQL

        DECLARE @createSQL nvarchar(max)
```

```
                    set @createSQL= 'create table '+ @errTable+'
                        ↪ with (DISTRIBUTION=ROUND_ROBIN) as SELECT
                        ↪  * FROM ' +@TableName + '
                    where ' + CONCAT('CONCAT(COALESCE(', REPLACE(
                        ↪ @CheckKey, ',', ',''''),''-'',COALESCE(')
                        ↪ , ','''''))') + ' in (select '+
                        ↪ @KeysToDelete + ')';
                    print @createSQL
                    EXEC sp_executesql @createSQL

                    --END
                    set @deleteSQL = 'DELETE FROM ' +@TableName + '
                    where ' + CONCAT('CONCAT(COALESCE(', REPLACE(
                        ↪ @CheckKey, ',', ',''''),''-'',COALESCE(')
                        ↪ , ','''''))') + ' in (select '+
                        ↪ @KeysToDelete + ')';
                    print @deleteSQL
                    EXEC sp_executesql @deleteSQL

            END
    END

    SET @Status = 'KO';
    IF (@retval is null) SET @Status = 'OK';
    print @Status

    DECLARE @CheckDateEnd datetime
    set @CheckDateEnd = (CONVERT(datetime, SYSDATETIMEOFFSET() AT
        ↪ TIME ZONE 'W. Europe Standard Time'))

    DECLARE @insertSql as nvarchar(4000)
```

```
set @insertSql = 'INSERT INTO [CTL].[LOG_Checks_Discarded_Lines
   ↪ ] (CheckId, ExecutionID, PipelineName, TableName,
   ↪ CheckKey, DTA_ActivityStart, DTA_ActivityEnd, Status)
SELECT '''+@CheckID+''' AS [CheckID]
,'''+@ExecutionID+''' AS [ExecutionID]
,'''+@PipelineName+''' AS [PipelineName]
,'''+@TableName+''' AS [TableName]
,'''+@CheckKey+''' AS [CheckKey]
,'''+convert(varchar(25), @CheckDateStart, 120)+''' as [
   ↪ DTA_ActivityStart]
,'''+convert(varchar(25), @CheckDateEnd, 120)+''' as [
   ↪ DTA_ActivityEnd]
,'''+@Status+''' AS [Status]'
print @insertSql
EXEC sp_executesql @insertSql

DECLARE @updateSQL as nvarchar(4000)
set @updateSQL = 'UPDATE [CTL].[LOG_Checks_Discarded_Lines] SET
 ErrorMessage='''+@ErrorMessage+'''
where CheckID = '''+ @CheckID+''' and ExecutionID = '''+
   ↪ @ExecutionID + '''
';
print @updateSQL
EXEC sp_executesql @updateSQL

DECLARE @updateSQL2 as nvarchar(max)
set @updateSQL2 = 'UPDATE [CTL].[LOG_Checks_Discarded_Lines]
   ↪ SET
 DiscardedLines='''+@retval+'''
where CheckID = '''+ @CheckID+''' and ExecutionID = '''+
   ↪ @ExecutionID + '''
```

```
        ';
        print @updateSQL2
        EXEC sp_executesql @updateSQL2

        drop table #anag_checks

END
GO
```

4.3 Codice store procedure

4.4.3 Codice in Azure

Operazione di lookup nella pipeline di invio mail (vedi 4.3.4 Pipeline di invio mail):

```
SELECT ErrorMessage +' '+ TableName +'<br><br>'+ COALESCE(
   ↪ DiscardedLines,'') as DiscardedLines, Status
    FROM [CTL].[LOG_Checks_Discarded_Lines]
    WHERE CheckID='@{pipeline().parameters.checkID}' AND ExecutionID =
       ↪ '@{pipeline().parameters.executionID}'
```

4.4 Codice operazione di lookup per l'invio della mail

IF per controllare se lo status del controllo è KO:

```
@equals(activity('lookup1').output.firstRow.status,'KO')
```

4.5 Condizione if sullo status

Operazione di lookup nella pipeline di check delle chiavi per controllare se sulle tabelle che si stanno caricando ci siano dei controlli di chiave (vedi 4.3.5 Pipeline per il check chiave):

```
SELECT [CheckID], [Tablename], [CheckKey], [ErrorMessage], [CheckQuery
   ↪ ], [CheckType]
FROM [CTL].[D_ANAG_Checks]
WHERE [TableName] = CONCAT('[',UPPER('@{pipeline().parameters.
   ↪ DestSchemaName}'),) ']', '.', '[', UPPER('@{pipeline().parameters
   ↪ .DestTableName}'), ']')
AND [CheckType] = 'SQL_KEYS'
```

4.6 Codice operazione di lookup per i check chiave

Condizione IF nella pipeline di check delle chiavi che segue la lookup precedente, in modo tale da attivare i controlli chiave solo ed esclusivamente se la tabella in esame li richiede (vedi 4.3.5 Pipeline per il check chiave):

```
@greater(indexOf(string(activity('GetLNQueryCheckKeyes').output),'
   ↪ firstRow'), -1)
```

4.7 Codice if per i check chiave

Capitolo 5

Validazione

Il capitolo dedicato alla validazione ha lo scopo di dimostrare, attraverso scenari reali, il corretto funzionamento del sistema di controllo della qualità dei dati sviluppato. La validazione è suddivisa in due sezioni principali, ciascuna delle quali approfondisce un aspetto specifico dell'utilizzo e delle performance del sistema.

Nella prima sezione, 5.1 Casi d'uso, vengono presentati due esempi pratici di configurazione ed esecuzione dei controlli sui dati. In questi casi d'uso, si illustrano le funzionalità del sistema nel monitorare nuove strutture di dati e nell'aggiungere controlli su strutture già monitorate, mostrando come l'utente può configurare i controlli, aggiornare le pipeline e gestire le notifiche di anomalie. Gli scenari includono l'inserimento di dati reali e la dimostrazione delle operazioni necessarie per garantire l'integrità e l'accuratezza dei dati.

La seconda sezione, 5.2 Risultati sperimentali, è dedicata all'analisi dei risultati ottenuti durante i test dei casi d'uso. Vengono presentate tabelle degli esiti dei controlli effettuati, che mostrano in dettaglio i dati analizzati e gli eventuali scarti dovuti alle anomalie rilevate. Inoltre, viene mostrata la mail di notifica generata dal sistema, dimostrando la tempestività delle segnalazioni inviate agli amministratori. Questa parte

fornisce una prova concreta dell'efficacia del sistema e della sua capacità di migliorare la governance dei dati.

Insieme, queste due sezioni offrono una visione completa dell'efficacia del sistema, mostrando come esso gestisca la configurazione, l'esecuzione e il monitoraggio dei controlli in modo automatizzato e configurabile.

5.1 Casi d'uso

In questa sezione mostreremo alcuni esempi concreti di utilizzo del sistema che fanno riferimento alle funzionalità descritte in precedenza utilizzando tutti gli elementi necessari descritti fin ora. Per ogni caso d'uso si descriverà lo scenario che mostra tutte le attività pratiche che vengono eseguite dall'utente tramite il sistema.

5.1.1 Caso d'uso 1: Configurazione di controllo su una nuova struttura

Questo caso d'uso illustra il processo di configurazione di un controllo per una nuova struttura non ancora monitorata dal sistema. L'esempio mostra come sia possibile definire i controlli, aggiornarli all'interno della pipeline di Azure Data Factory e gestire le notifiche via email.

Scenario: Il dipartimento IT ha deciso di monitorare una nuova tabella contenente dati di ordini provenienti da un sistema esterno. A causa della variabilità dei dati importati, esiste la possibilità che alcuni record non rispettino i requisiti di accuratezza e consistenza richiesti per l'elaborazione successiva. Il sistema di controllo deve verificare che non ci siano record che a parità di OrID riportano stato e data diversi.

- **Fase 1: Configurazione del controllo**

  Per iniziare, il responsabile del sistema inserisce una nuova riga nella tabella comune di configurazione [CTL].[D_ANAG_CHECKS]. I dettagli del controllo sono i seguenti:

  - **CheckID:** 'ORDER_DUPLICATES';
  - **Tablename:** '[STG].[ORDER]';
  - **CheckKey:** 'OrID';
  - **ErrorMessage:** 'I seguenti OrID sono duplicati a causa di Status o Date multipli sulla tabella';
  - **CheckQuery:**

```
select @retvalOUT = string_agg(cast(''''+OrID+'''' as nvarchar
    ↪ (max),',')
from(select OrID,status,date
        from [STG].[ORDER]
        group by OrID,status,date)dupli
        group by OrID having count(*)>1
```

5.1 Codice CheckQuery caso d'uso 1

  - **CheckType:** 'SQL'

La query verifica se ci sono OrID che appaiono con combinazioni diverse di status e date nella tabella [STG].[ORDER]. Prima elimina i duplicati esatti (stessa combinazione di OrID, status, date), poi raggruppa per OrID e seleziona quelli che hanno più di una combinazione unica. Infine, concatena gli OrID duplicati in una stringa separata da virgole.

- **Fase 2: Aggiornamento della pipeline in Azure Data Factory**

  Dopo aver configurato il controllo, viene aggiornata la pipeline in Azure Data Factory che gestisce il caricamento dei dati. Viene aggiunta un'activity per richiamare la Stored Procedure Comune con l'ID del controllo configurato subito dopo aver caricato i dati.

  Il primo dato richiesto è il nome della store procedure da richiamare, nel nostro caso è "[CTL].[sp_LOG_Checks_Discard_Lines]". Successivamente vengono inseriti i parametri necessari alla store procedure:

  - **CheckID** = 'ORDER_DUPLICATES'
  - **ExecutionID** = @pipeline().runId
  - **PipelineName** = @pipeline().pipeline

  Il responsabile del sistema utilizza "sp_CHECK_ORDER DUPLICATES" come nome dell'activity.

- **Fase 3: Notifiche tramite email**

  Successivamente viene aggiunta una nuova pipeline per l'invio di una notifica email, in caso di righe scartate, subito dopo l'activity di richiamo della store procedure.

  Questa pipeline, come abbiamo visto nei capitoli precedenti, riceve i seguenti parametri:

  - **mailBody**: 'Dati errati nella tabella Ordini'
  - **mailReceipt**: 'ams@department.it'
  - **mailSubject**: 'OrID duplicati'
  - **executionID**: @pipeline.RunId
  - **checkID**: 'ORDER_DUPLICATES'

A questo punto, il responsabile del sistema, inserisce un'activity di **lookup** per ricavare il messaggio di errore, il nome della tabella, eventuali righe scartate e lo status del controllo. Il comando di lookup viene settato come segue:

```
SELECT ErrorMessage +' '+ TableName +'<br><br>'+ COALESCE(
   ↪ DiscardedLines,'') as DiscardedLines, Status
    FROM [CTL].[LOG_Checks_Discarded_Lines]
    WHERE CheckID='@{pipeline().parameters.checkID}' AND
       ↪ ExecutionID = '@{pipeline().parameters.executionID}'
```

5.2 Codice operazione di lookup per l'invio della mail scenario

A seguito dell'activity di lookup, viene inserita un'activity di **controllo condizionale** con la seguente condizione:

```
@equals(activity('lookup1').output.firstRow.status,'KO')
```

5.3 Condizione if sullo status scenario

Il responsabile del sistema effettua questo controllo sullo status per decidere se inviare la mail di notifica a seconda del risultato. Infatti, se il controllo è 'KO' e quindi siamo nella condizione true, il responsabile utilizza un'activity di **set variable** per aggiungere al body della mail le righe scartate utilizzando le informazioni ricavate dall'activity di lookup.

Dopo aver settato la variabile del body, il responsabile del sistema utilizza un'activity di **send mail** per inviare effettivamente la mail all'indirizzo impostato precedentemente nei parametri della pipeline.

In questo caso abbiamo mostrato come un responsabile di sistema può configurare un nuovo controllo su una struttura partendo da zero. Dal punto di vista pratico, si

ricorda, che se esiste già almeno un controllo su una qualsiasi struttura, un utente può velocizzare le operazioni descritte nello scenario, soprattutto quelle che riguardano la fase 2 e 3 della configurazione. L'utente, avendo a disposizione almeno un controllo, può copiare l'activity di richiamo della store procedure modificando solo i parametri e il nome dell'activity, discorso analogo per quanto rigurda la pipeline di invio mail, anche in quel caso l'utente può copiare l'intera pipeline e modificare solo i parametri della stessa.

5.1.2 Caso d'uso 2: Configurazione di controllo su una struttura già monitorata

Questo caso d'uso illustra il processo di configurazione di un controllo per una struttura già oggetto di monitoraggio da parte del sistema. L'esempio mostra come sia possibile definire l'aggiunta di un altro controllo, aggiornarlo all'interno della pipeline di Azure Data Factory e gestire le notifiche via email.

Scenario: Il dipartimento IT ha deciso di aggiungere un nuovo controllo su una tabella contenente dati di reclami provenienti da un sistema esterno. A causa della compilazione manuale dei dati importati, esiste la possibilità che alcuni record non rispettino i requisiti di accuratezza e consistenza richiesti per l'elaborazione successiva. Il sistema di controllo deve verificare che non ci siano record che contengono ClID con una data di ricezione precedednte al 1970. Questa struttura al momento presenta già un controllo su un altro campo.

- **Fase 1: Configurazione del controllo**

  Per iniziare, il responsabile del sistema inserisce una nuova riga nella tabella comune di configurazione [CTL].[D_ANAG_CHECKS]. I dettagli del controllo sono i seguenti:

  - **CheckID:** 'CLAIM_PAST_YEARS_RECEP';

- **Tablename:** '[STG].[CLAIM]';
- **CheckKey:** 'ClID,ReceptionDate';
- **ErrorMessage:** 'I seguenti ClID-ReceptionDate presentano ReceptionDate precedenti al 1970 sulla tabella ';
- **CheckQuery:**

```
select @retvalOUT = string_agg(cast(''''+ClID+'-'+cast([
    ↪ ReceptionDate] as varchar(10))+'''' as nvarchar(max)), '
    ↪ ,')
from(select distinct ClID,[ReceptionDate]
        from [STG].[CLAIM]
        where YEAR([ReceptionDate])<1970) sq
```

5.4 Codice CheckQuery caso d'uso 2

- **CheckType:** 'SQL'

La query verifica se ci sono ClID che appaiono con ReceptionDate precedenti al 1970. Viene creata una lista di record che contengono un ClID con una ReceptionDate antecedente al 1970, concatenandoli in una singola stringa separata da virgole

- **Fase 2: Aggiornamento della pipeline in Azure Data Factory**

  Dopo aver configurato il controllo, viene aggiornata la pipeline in Azure Data Factory che gestisce il caricamento dei dati. Viene aggiunta un'activity per richiamare la Stored Procedure Comune con l'ID del controllo configurato subito dopo aver caricato i dati. Dato che è già presente un controllo sulla struttura, l'activity di richiamo alla pipeline vine aggiunta parallelamente all'activity già presente per il controllo esistente.

Il primo dato richiesto è il nome della store procedure da richiamare, nel nostro caso è "[CTL].[sp_LOG_Checks_Discard_Lines]". Successivamente vengono inseriti i parametri necessari alla store procedure:

- **CheckID** = 'CLAIM_PAST_YEARS_RECEP'
- **ExecutionID** = @pipeline().runId
- **PipelineName** = @pipeline().pipeline

Il responsabile del sistema utilizza "sp_CHECK_CLAIM PAST YEARS" come nome dell'activity.

- **Fase 3: Notifiche tramite email**

  Successivamente viene aggiunta una nuova pipeline per l'invio di una notifica email, in caso di righe scartate, subito dopo l'activity di richiamo della store procedure.

  Questa pipeline, come abbiamo visto nei capitoli precedenti, riceve i seguenti parametri:

  - **mailBody**: 'Dati errati nella tabella Claim'
  - **mailReceipt**: 'ams@department.it'
  - **mailSubject**: 'Controllo ReceptionDate values - tabella STG.CLAIM'
  - **executionID**: @pipeline.RunId
  - **checkID**: 'CLAIM_PAST_YEAR_RECEP'

  Dopo aver settato i parametri della pipeline, il responsabile del sistema **copia** il contenuto della pipeline di invio mail del controllo già esistente sulla struttura e lo utilizza anche per questa pipeline. In questo modo il responsabile sfrutta a pieno le potenzialità di configurabilità delle pipeline.

5.2 Risultati sperimentali

In questa sezione andiamo ad analizzare i risultati derivanti dall'esecuzione dei due scenari illustrati nella sezione precedente. Per ognuno di essi mostrimao alcune righe della tabella di esito/scarto [CLT].[LOG_Checks_Discarded_Lines] e, nel caso in cui ci sono stati esiti con status = KO, mostriamo il contenuto della mail inviata ai tecnici.

5.2.1 Risultati scenario 1

Il primo scenario ha l'obiettivo di garantire l'integrità dei dati importati dal sistema esterno, verificando che non ci siano incongruenze nei record degli ordini. In particolare, si intende evitare la presenza di record con lo stesso identificativo dell'ordine (OrID), ma con combinazioni diverse di stato e data, che potrebbero indicare anomalie o errori nella gestione degli ordini. Questo controllo è fondamentale per assicurare che i dati siano coerenti e accurati prima di essere utilizzati nei processi successivi.

Dopo che il responsabile del sistema ha configurato il controllo, la pipeline di caricamento dati è stata rieseguita automaticamente in base alla pianificazione di Azure che determina tutte le tempistiche di avvio delle pipeline.

- **Contenuto di [CLT].[LOG_Checks_Discarded_Lines] filtrato per il nostro CheckID**

CheckID	ExecutionId	PipelineName	TableName	CheckKey	DTA_ActivityStart	DTA_ActivityEnd	Status	ErrorMessage	DiscardedLines
ORDER_DUPLICATES	65300d03-e89c-4965-9a75-84f71f3775aa	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-15 03:10:11.000	2024-10-15 03:10:13.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	9b1b3ba9-b070-4f13-b593-3b9217dfb76a	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-14 13:31:35.000	2024-10-14 13:31:38.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	d09e03e2-0fe1-44c9-9861-2d7e97292aea	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-14 03:10:37.000	2024-10-14 03:10:40.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	5df68cff-ea98-4a5f-8499-820413b506e0	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-13 04:00:44.000	2024-10-13 04:00:45.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	25ec193f-d8fd-44bb-91f9-635b379f5e19	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-11 03:15:51.000	2024-10-11 03:15:57.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	c19e9bb2-4134-44f8-a1d0-492bfa2425f6	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-10 03:10:57.000	2024-10-10 03:11:01.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	f53e97b6-86dd-4cd9-920b-54f2ad255616	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-09 03:11:59.000	2024-10-09 03:12:05.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	6a302d82-8718-4ecb-983a-4e58387c31a1	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-08 03:54:57.000	2024-10-08 03:55:02.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	5912bc6c-b95b-43a5-9d53-c97077292e5b	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-07 03:10:10.000	2024-10-07 03:10:14.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL
ORDER_DUPLICATES	d5b859bb-fe75-4f53-b05a-50a70748af52	DP 01 STG Order	[STG].[ORDER]	OrID	2024-10-06 03:12:40.000	2024-10-06 03:12:42.000	OK	I seguenti OrID sono duplicati a causa di Status...	NULL

Figura 5.1: Tabella di esito scenario 1

Come possiamo notare dalla tabella, l'ultimo controllo con

CheckID = ORDER_DUPLICATES non ha prodotto righe di scarto, infatti, lo

status del controllo è OK. Oltre a considerare questo ultimo controllo, possiamo prendere visione anche dello storico dei controlli con lo stesso CheckID che sono stati eseguiti periodicamente dal sistema e che non hanno prodotto righe di scarto.

Lo storico dei controlli ci può dare una visione panoramica sia dello stato dei controlli ma anche del **tempo di esecuzione dei controlli**, che è un'informazione molto importante da considerare. Analizzando i vari tempi di esecuzione possiamo affermare infatti che si impiegano mediamente **3,6 secondi** per eseguire un controllo. Spendere alcuni secondi per fare dei controlli essenziali in fase di caricamento dei dati può fare risparmiare ore di lavoro in futuro, soprattutto se si scartano delle righe e si va a circoscrivere il problema.

Tramite queste visualizzazioni di dati e di considerazioni abbiamo potuto **validare** l'esecuzione di un caso d'uso del sistema utilizzando dei risultati concreti.

5.2.2 Risultati scenario 2

Nel secondo scenario Il dipartimento IT ha introdotto un nuovo controllo su una tabella contenente dati di reclami provenienti da un sistema esterno. Poiché i dati vengono inseriti manualmente, c'è il rischio che alcuni record non rispettino gli standard di accuratezza e coerenza necessari per le fasi di elaborazione successive. L'obiettivo del nuovo controllo è verificare che nessun record contenga un ClID con una data di ricezione antecedente al 1970, garantendo così la validità temporale dei dati. Attualmente, la struttura dispone già di un controllo attivo su un altro campo per monitorare ulteriori incongruenze.

Dopo che il responsabile del sistema ha configurato il controllo aggiungendolo parallelamente a quello già esistente, la pipeline di caricamento dati è stata rieseguita automaticamente in base alla pianificazione di Azure che determina tutte le tempistiche di avvio delle pipeline.

- **Contenuto di [CLT].[LOG_Checks_Discarded_Lines] filtrato per il nostro CheckID**

CheckID	ExecutionId	PipelineName	TableName	CheckKey	DTA_ActivityStart	DTA_ActivityEnd	Status	ErrorMessage	DiscardedLines
CLAIM_PAST_YEARS_RECEP	52a1cb5c-b351-488d-87be-53f173f228a7	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-15 02:25:48.000	2024-10-15 02:25:52.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	d2c0df6f-06b8-4b1c-a990-8ee21b7adc20	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-14 12:44:36.000	2024-10-14 12:44:39.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	92cfdaf2-e4ec-4494-be1d-a624fe303c9e	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-14 02:24:21.000	2024-10-14 02:24:56.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	f7c9932c-ed97-4af7-a785-40ff3d3b36c7	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-13 02:27:37.000	2024-10-13 02:27:39.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	5570c9ff-e491-4688-ab79-067e32911775	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-12 02:28:26.000	2024-10-12 02:28:30.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	6a741ec1-c803-4bdd-a447-61b199df4b56	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-11 02:37:04.000	2024-10-11 02:37:17.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	6dc18ff4-155b-4fe2-a27e-4ef808a1c13c	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-10 02:23:54.000	2024-10-10 02:23:56.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	edb32784-5412-43e1-8c82-6b821b0c7a24	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-09 02:27:27.000	2024-10-09 02:27:47.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	9a8c874f-0ab9-4fe5-99c0-d43a16af70e8	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-08 02:26:30.000	2024-10-08 02:26:42.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'
CLAIM_PAST_YEARS_RECEP	3e239394-7d13-4559-b367-1c48dfd7b4d7	DP 01 STG CLAIM	[STG].[CLAIM]	CIID,ReceptionDate	2024-10-07 02:27:11.000	2024-10-07 02:27:32.000	KO	I seguenti CIID presentano ReceptionDate prec...	'CL2426465-0062-01-22'

Figura 5.2: Tabella di esito scenario 2

Dalla tabella di esito che restituisce lo storico dei controlli, filtrata per il nostro CheckID, notiamo che tutti i controlli effettuati nel tempo hanno prodotto delle righe di scarto. Infatti il reclamo che fa riferimento al codice scartato ha una data di reclamo antecedente al 1970. Avendo segnalato una riga problematica, troveremo nella casella di posta una mail che ci segnala l'errore.

- **Contenuto della mail**

Figura 5.3: Email scenario 2

Come possiamo vedere dalla mail, viene indicato nel subject il tipo di controllo e la tabella sul quale viene eseguito. Nel corpo della mail abbiamo il messaggio di errore seguito dalla tabella nel quale si è verificato l'errore e in fine le informazioni necessarie ad individuare le righe scartate che abbiamo indicato nella CheckQuery. A questo punto, dato che il tipo di query è SQL, il team di AMS interverrà sull'errore per risolverlo tramite l'utilizzo di tutte le informazioni raccolte durante il controllo.

Anche per questo scenario abbiamo **validato**, tramite risultati tangibili, le configurazioni che l'utente ha scelto per questo controllo.

Capitolo 6

Conclusioni e Sviluppi Futuri

Il percorso intrapreso con questa tesi ha avuto come obiettivo la progettazione e l'implementazione di un sistema di controllo della qualità dei dati basato su un'architettura fail-safe integrata in una piattaforma dati. Il sistema sviluppato si propone di gestire in maniera automatizzata i controlli sulla qualità dei dati, intervenendo in caso di anomalie e garantendo che il flusso di lavoro possa proseguire senza interruzioni, minimizzando i rischi operativi.

Il progetto ha raggiunto tutti gli obiettivi prefissati. Innanzitutto, è stato creato un sistema configurabile e scalabile che permette di eseguire controlli su dati provenienti da diverse fonti, senza la necessità di interventi manuali da parte degli amministratori. Il sistema offre una struttura modulare, in cui ogni controllo è definito in maniera centralizzata attraverso una tabella di configurazione, garantendo una facile gestione e aggiornamento dei criteri di verifica. Questo ha reso il sistema altamente flessibile e adatto a diverse esigenze operative.

Inoltre, il sistema consente un monitoraggio continuo dei dati. Quando viene rilevata un'anomalia, il sistema invia una notifica automatica agli amministratori, permettendo loro di intervenire tempestivamente. Questo approccio proattivo è fondamentale per

ridurre i tempi di inattività e prevenire l'ingresso di dati errati nei processi aziendali successivi. Grazie alla combinazione tra automazione e configurabilità, il sistema si dimostra non solo affidabile ma anche facile da gestire, riducendo significativamente il carico di lavoro manuale.

Oltre agli aspetti tecnici, il sistema offre vantaggi significativi anche dal punto di vista della governance dei dati. Un maggiore controllo della qualità dei dati si traduce infatti in una riduzione degli errori nei processi decisionali e operativi, garantendo che le decisioni aziendali si basino su dati accurati e affidabili. Questo migliora non solo l'efficienza operativa ma anche la competitività dell'organizzazione.

Per quanto riguarda gli **sviluppi futuri**, ci sono diverse direzioni in cui il sistema potrebbe essere ulteriormente migliorato. Un primo possibile sviluppo riguarda l'integrazione di strumenti di intelligenza artificiale e machine learning per potenziare l'identificazione delle anomalie. L'utilizzo di modelli predittivi potrebbe infatti migliorare la capacità del sistema di riconoscere pattern di errore complessi o non evidenti tramite controlli tradizionali, offrendo suggerimenti per l'automazione della correzione.

Infine, potrebbe essere utile sviluppare ulteriori strumenti di monitoraggio e reporting. Un modulo che permetta agli amministratori di analizzare nel dettaglio i controlli eseguiti, visualizzando le statistiche sugli errori rilevati e le performance del sistema, aiuterebbe a migliorare ulteriormente il livello di controllo e di governance dei dati.

In conclusione, il sistema proposto si è dimostrato efficace nel rispondere alle sfide poste dalla gestione della qualità dei dati in una piattaforma aziendale. Ha raggiunto l'obiettivo di automatizzare i controlli, minimizzare gli errori e offrire una soluzione scalabile e configurabile per gestire i dati in modo sicuro e accurato. Gli sviluppi futuri suggeriti, come l'integrazione di intelligenza artificiale e l'espansione verso nuove tipologie di dati, possono rendere questo sistema ancora più potente e adattabile alle esigenze tecnologiche in continua evoluzione.

Bibliografia

[1] Mohammed Al-Ruithe, Elhadj Benkhelifa, and Khawar Hameed. Big data quality framework: a holistic approach to continuous quality management. *Journal of Big Data*, 8(1):1–29, 2021.

[2] T. Dahlberg and F. Norrgren. Fail-safe approaches to data quality management. In *Proceedings of the International Conference on Data Quality*, pages 75–84, 2014.

[3] Gustavo Gracioli, Gustavo Bueno, Ryan Motz, Icaro Bertotti, Bianca Stein, and Karina Becker. A survey of data quality measurement and monitoring tools. *Frontiers in Big Data*, 5, 2022.

[4] IBM. Ibm's perspective on modern data platforms. *IBM Cloud Insights*, 2022.

[5] Ralph Kimball and Joe Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley, Hoboken, NJ, 2004.

[6] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[7] Michael Stonebraker. The challenges of big data. *Communications of the ACM*, 58(7):24–26, 2015.

[8] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems*, 12(4):5–33, 1996.

[9] Mohamed Yassine, Anuradha Sinha, and Maik Benedick. Big data quality framework:

a holistic approach to continuous quality management. *Journal of Big Data*, 8(1):1–28, 2021.