ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in
Ingegneria e Scienze Informatiche

Selezione di un data catalog per l'azienda Bonfiglioli S.p.A.

Relatore:
Prof.
Stefano Rizzi

Correlatore:
Prof.
Enrico Gallinucci

Presentata da:
Yuqi Sun

Sessione III

Anno Accademico 2023/2024

A tutti coloro che mi hanno accompagnato in questo percorso

Sommario

La *self-service business intelligence* (SSBI) è una possibile strategia che le aziende possono applicare per trarre il maggior vantaggio competitivo possibile dai loro dati. Considerabile come un'evoluzione della *business intelligence* (BI) tradizionale, che a oggi fatica a stare dietro l'aumento di domanda di informazioni e analisi, la SSBI mira a semplificare l'uso della BI affinché tutti gli utenti possano effettuare le proprie analisi in autonomia, senza richiedere il supporto di professionisti IT.

Uno dei maggiori ostacoli che impedisce un'azienda a implementare con successo la SSBI è la mancanza di una visione globale dei dati, che spesso sono frammentati in *silos* e poco documentati. Uno strumento che potrebbe permettere di superare questo problema è il *data catalog*, un software che raccoglie e mantiene i metadati provenienti da varie sorgenti e che, attraverso un motore di ricerca integrato, permette agli utenti di esplorare e cercare i dati desiderati, comprenderli meglio e individuare dove sono effettivamente localizzati.

Per tutti questi motivi, l'azienda di metalmeccanica Bonfiglioli S.p.A. vorrebbe inserire un data catalog all'interno del loro stack tecnologico. L'obiettivo di questa tesi è individuare il data catalog che meglio risponde ai bisogni dell'azienda, effettuando test e analisi approfondite delle funzionalità più desiderate.

Indice

Capitolo 1

Introduzione

Molte organizzazioni adottano la *business intelligence* per supportare il loro processo decisionale: i sistemi di BI provvedono alle aziende informazioni accurate e tempestive, con le quali possono prendere decisioni e reagire velocemente ai cambiamenti del mercato e ai bisogni dei clienti [44]. Tuttavia, negli ultimi decenni, con l'avvento dei Big Data da una parte, e l'estensione della BI anche nelle decisioni più operazionali dall'altra, è cresciuta sempre di più la necessità di facilitare i processi di analisi affinché anche gli utenti non tecnici possano applicare la BI in modo autonomo e senza l'assistenza dal reparto IT [3].

La *Self-Service BI* (SSBI) è un nuovo approccio il cui obiettivo è proprio quello di creare un ambiente dove tutti gli utenti hanno gli strumenti necessari per ottenere le informazioni che vogliono, quando li vogliono e come meglio preferiscono visualizzarli [19]. Fiduciosi di queste promesse, anche l'azienda di metalmeccanica Bonfiglioli S.p.A. vorrebbe che la SSBI diventasse un punto di riferimento per la propria reportistica e per effettuare delle prime sperimentazioni ha già creato dei dataset per alcune aree di business specifiche. Un ostacolo che stanno incontrando, e che è citato anche da alcune ricerche sulla SSBI [23] [19], è la difficoltà a rendere i dati facilmente accessibili: i dati della Bonfiglioli sono separati in molti silos; manca inoltre una documentazione scritta, esaustiva e comprensiva dei propri dati e del contesto in cui questi operano e molti asset di BI e *analytics* non sono tracciati, con il rischio di riprodurre dataset già esistenti.

Per risolvere questi problemi, la Bonfiglioli ha pensato di introdurre un *data catalog*, un software che è in grado di importare i metadati da varie sorgenti, ottenendo un catalogo di *data asset* indicizzati, organizzati e arricchiti di descrizioni che gli utenti possono liberamente esplorare grazie al motore di ricerca nativamente integrato. Con un data catalog, un'azienda è in grado di fornire ai propri dipendenti un inventario centralizza-

to e ben documentato dei dati, rimuovendo così uno degli ostacoli all'implementazione della SSBI, ma può anche venire in aiuto ad aree come il *data management* e la *data governance*. L'obiettivo di questa tesi è proprio individuare un data catalog intuitivo e facile da mantenere che soddisfi i bisogni della Bonfiglioli: la tecnologia che alla fine sarà valutata come migliore verrà collaudata con un ristretto numero di reparti prima di essere condivisa con il resto dell'azienda.

La tesi procederà come segue. Il capitolo due presenterà il gruppo Bonfiglioli S.p.A. e la sua organizzazione, mettendo a fuoco i processi aziendali e le tecnologie attualmente in uso con cui il data catalog dovrà integrarsi. Il capitolo tre provvederà il background teorico, spiegando quali sono le ragioni che hanno portato alla nascita della *self service business intelligence* e come un *data catalog* può venire a supporto di questa nuova pratica, definendo le sue caratteristiche principali e analizzando i vantaggi che promette di apportare dentro un'azienda. Seguirà il capitolo quattro, dove verranno elencati i requisiti espressi dalla Bonfiglioli e introdotti i data catalog, illustrando le loro architetture ad alto livello e principali funzionalità. Nel capitolo cinque verranno esposti gli esperimenti effettuati, descrivendo in dettaglio i risultati ottenuti e confrontandoli tra di loro. La tesi si concluderà con una riflessione sul lavoro svolto.

Capitolo 2

L'azienda e i suoi dati

2.1 Bonfiglioli S.p.A.

Il gruppo Bonfiglioli è un'azienda specializzata nella produzione di "motoriduttori", dispositivi formati da un motore e un riduttore che ne modula la velocità a seconda delle necessità.

Il gruppo nasce da Clementino Bonfiglioli nel 1956, fondando quella che era allora la "Costruzioni Meccaniche Bonfiglioli". L'azienda inizia come produttrice di riduttori per le principali industrie locali del settore imballaggi, ma già dopo venti anni di attività comincia ad ampliarsi: nei decenni successivi, acquisisce varie aziende, sia locali sia estere, specializzate in diversi prodotti metalmeccanici e si espande aprendo filiali e stabilimenti in paesi europei ed extra-europei. A oggi, il gruppo Bonfiglioli progetta, produce e distribuisce una vasta gamma di motoriduttori, dispositivi di azionamento, riduttori di tipo epicicloidali e inverter in grado di soddisfare qualunque richiesta dei settori industriali dell'automazione, delle macchine mobili e dell'energia rinnovabile. [8]

In totale il gruppo Bonfiglioli conta 15 stabilimenti produttivi, 24 filiali commerciali e oltre 550 distributori in tutto il mondo. È composta da tre distinte unità business:

- *Mobility and Wind Industries*: offre una vasta gamma di riduttori per la realizzazione di soluzioni nei settori delle costruzioni, dell'energia eolica, della logistica, nell'agricoltura e nei settori marino e offshore;
- *Discrete Manufacturing and Process Industries*: offre riduttori, motoriduttori e motori elettrici per qualunque tipo di applicazione e settore industriale, come imballaggi, alimenti e bevande, minerario, logistica e intralogistica;

- *Motion and Robotics*: offre soluzioni dedicate ai processi e all'automazione industriale in settori quali lavorazione dei materiali, logistica, intralogistica, robotica, imballaggio e tessile.

Internamente, la Bonfiglioli è divisa in varie *competence center*, ciascuna con un proprio responsabile. In particolare, il reparto *Information Technology* (IT) si divide in due unità, *IT Core* e *IT Digital*: l'unità IT Core si occupa dei processi aziendali informatici più tradizionali, come l'infrastruttura, la sicurezza e la *business intelligence* (BI); l'unità IT Digital si preoccupa, invece, di aspetti più recenti, come i *big data*, l'intelligenza artificiale e l'*internet of things* (IOT).

2.2 Ciclo di vita dei dati

2.2.1 Le sorgenti dati

A supporto delle proprie attività giornaliere, la Bonfiglioli fa uso di vari software. Questi software sono le sorgenti di dati grezzi che verranno poi raccolti, trasformati e analizzati. Le sorgenti più importanti sono:

- SAP ERP (*Enterprise Resource Planning*) [36]: è un software che permette una gestione integrata di tutte le attività chiave di business di un'azienda, come acquisti, vendite, gestione magazzino, contabilità;
- Salesforce [32]: è un CRM (*Customer Relationship Management*), ovvero un software che gestisce le relazioni e le interazioni di un'azienda con i clienti per tutto il loro ciclo di vita, dal marketing, alla vendita fino al servizio cliente;
- MES (*Manufacturing Execution System*) [10]: è un sistema software che gestisce la produttività di un'azienda; supporta il controllo, il monitoraggio e l'esecuzione dei processi di produzione;
- PLC (*Programmable Logic Controller*) [7]: è un elaboratore specializzato per i processi di automazione industriale, come catene di montaggio, macchinari e robotica.

Esistono diverse differenze tra queste sorgenti: prima di tutto, le prime tre producono dati strutturati, mentre le sorgenti PLC producono dati sia strutturati sia non strutturati e contengono le informazioni riguardanti le macchine dei vari stabilimenti produttivi; in secondo luogo, i dati provenienti da ERP, CRM e MES sono gestiti principalmente dal reparto IT Core, mentre i dati provenienti dai PLC dal reparto IT Digital. Anche se

esistono comunque delle interazioni tra i due diversi team e quindi non è una separazione netta e rigida, questa distinzione comporta l'esistenza di due flussi di dati paralleli che non solo fanno uso di strumenti di memorizzazione e trasformazione diversi, ma soddisfano esigenze e scopi differenti:

- i vari dati provenienti da ERP, CRM e MES subiscono trasformazioni attraverso strumenti di ETL (Extract, Trasformation, Load) più tradizionali; i dati da PLC vengono trasformati attraverso script in Python;
- il reparto IT Core si occupa di BI tradizionale, mentre IT Digital di analisi predittiva dei guasti delle macchine e *data profiling*.

I flussi di dati che sono stati oggetto di studio durante la tesi hanno come sorgente dati principalmente SAP ERP, MES e PLC.

2.2.2 Flusso di dati da SAP

Benché faccia uso anche di altre sorgenti, come il sistema MES della Bonfiglioli, il team di business intelligence dell'azienda usa come sorgente principale SAP ERP, i cui dati seguono il flusso raffigurato in figura 3.1.

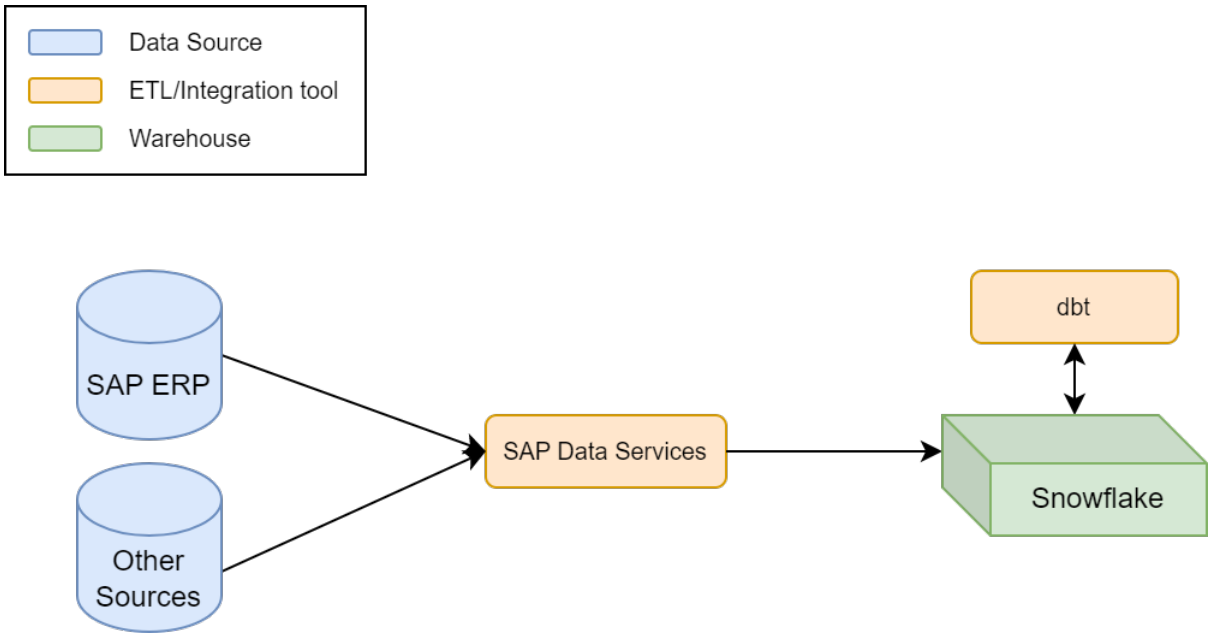


Figura 2.1: Flusso di trasformazioni che subiscono i dati provenienti da SAP ERP.

SAP Data Services

Per estrarre e trasformare i dati da SAP, si usa **SAP Data Services**, un software per l'integrazione e l'elaborazione dei dati. [34]

SAP Data Services è un componente centrale dello stack tecnologico dell'azienda. I sistemi SAP, infatti, oltre ad avere diverse sorgenti e formati dati, utilizzano ABAP (*Advanced Business Application Programming*), un linguaggio di sviluppo proprietario, caratterizzato da una logica applicativa complessa e creato specificamente per consentire l'elaborazione di massa dei dati nelle applicazioni aziendali SAP. [33] Poiché è un linguaggio proprietario, usare un software esterno all'ambiente SAP per estrarre i dati può risultare difficile, mentre SAP offre ai suoi cliente molti strumenti nativi, tra cui proprio *Data Services*.

In Data Services, i vari processi di elaborazione dei dati sono costruiti con un'interfaccia grafica *drag-and-drop*. L'estrazione dei dati dalle varie sorgenti avviene tramite **job**, che possono essere raggruppati in *project*; un *job* a sua volta è composto da vari *step* che vengono eseguiti insieme; lo *step* di un *job* può essere un:

- *data flow*: possono essere sorgenti, destinazioni, trasformazioni; è l'entità che estrae, trasforma e carica i dati;
- *work flow*: oltre a *data flow*, possono essere script, `while loop`, blocchi `try-catch`, condizioni; è l'entità che definisce e controlla come eseguire i *data flow*.

Con un *data flow* ABAP, la Bonfiglioli estrae e trasforma i dati dalle tabelle SAP, produce un data set finale e lo salva come file. Eventualmente questo data set può divenire un input per ulteriori trasformazioni.

Tramite un altro *data flow*, il file viene caricato prima su una cartella locale d'appoggio e poi sull'area di *stage* di Snowflake, sovrascrivendo eventuali copie già presenti nella cartella locale o su Snowflake. Durante questo processo è importante specificare:

1. il nome del file creato da Data Services;
2. il nome dello *stage* dove verrà caricato il file (es: `STG_SAP.STAGE_SAP`);
3. il nome della tabella finale di Snowflake (es: `COMPANY_GROUP`).

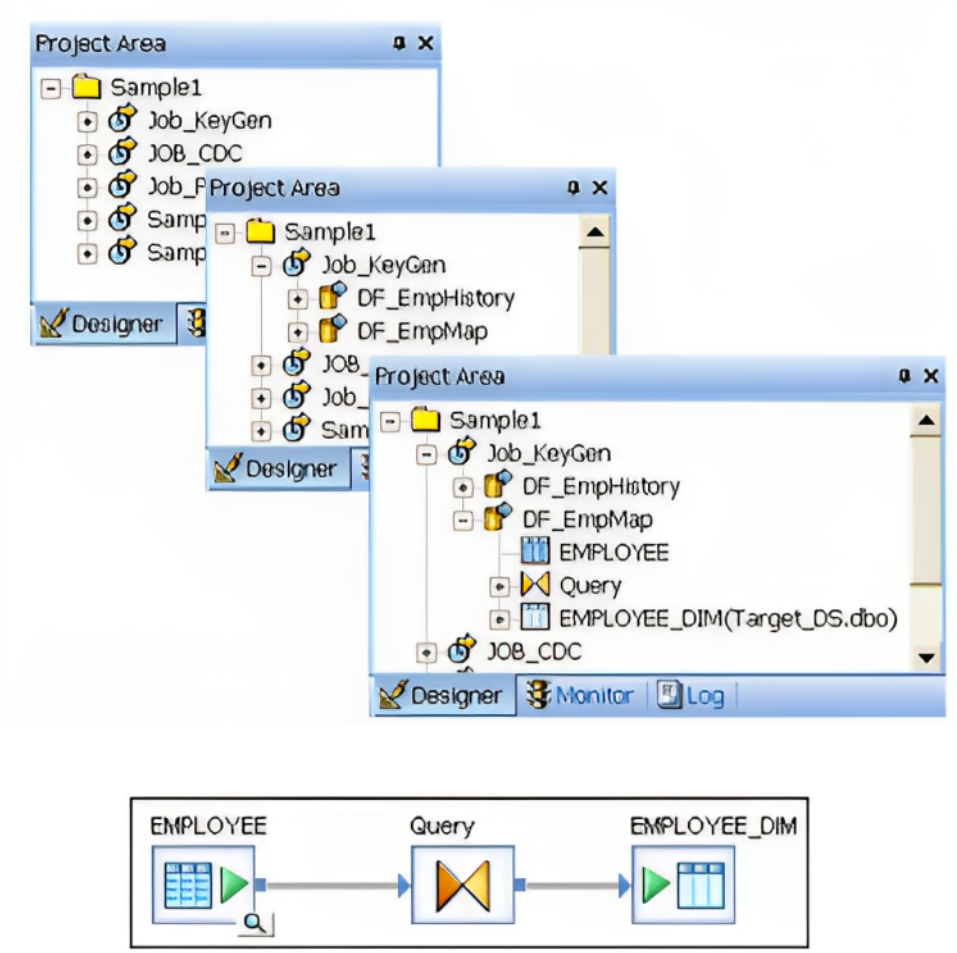


Figura 2.2: Esempio di *job* `Job_KeyGen` composto da due data flow. Il *dataflow* `DF_EmpMap` contiene a sua volta una tabella, una dimensione e un'interrogazione. [35]

Snowflake

Snowflake è una piattaforma dati che funziona completamente su infrastruttura cloud pubblica, con la quale è possibile salvare e gestire i propri dati, senza la necessità di installare o configurare hardware o software aggiuntivi. [40]

Un aspetto molto importante di Snowflake è che separa l'aspetto dell'archiviazione dei dati da quello dell'elaborazione, in modo che le risorse possano scalare in modo indipendente. La sua architettura può essere divisa in tre livelli:

- **archiviazione**: i dati vengono ottimizzati, compressi e salvati in formato colonnare in un repository centrale accessibile da tutti i nodi;
- **elaborazione**: le query sono elaborate utilizzando "*virtual warehouse*", cluster indipendenti di calcolo MPP (*massively parallel processing*), in modo che la performance

di uno non impatti gli altri;

- **servizi cloud**: una collezione di servizi che coordina le varie attività gestite da Snowflake.

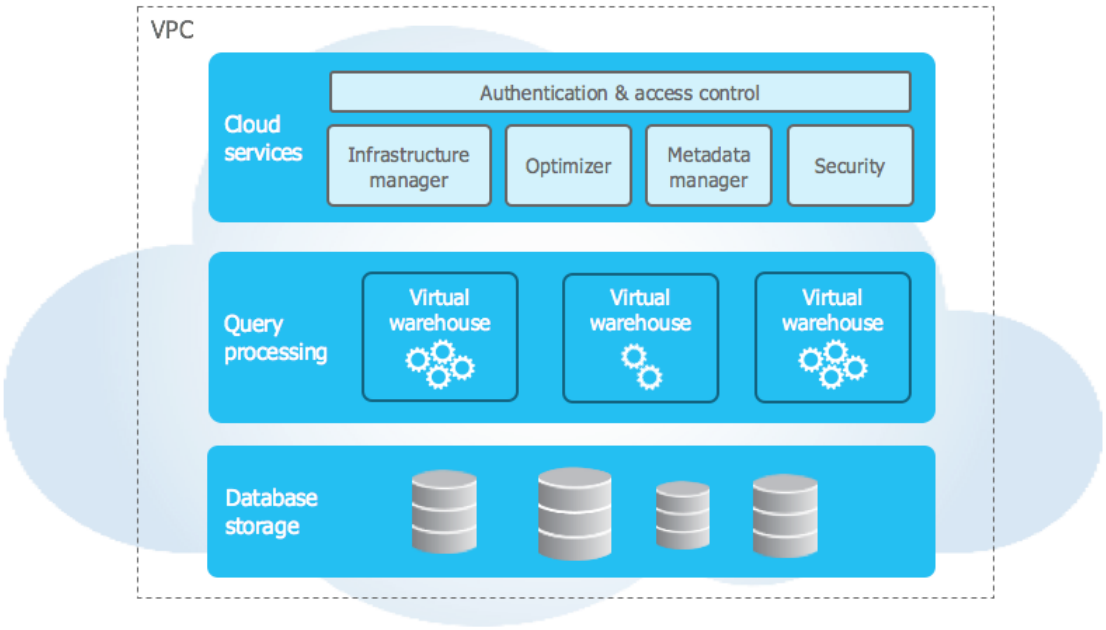


Figura 2.3: Architettura di snowflake [40].

Nel gruppo Bonfiglioli, Snowflake è il *data warehouse* di destinazione delle tabelle di SAP: dopo che queste sono state scritte sull'area di *stage* sotto forma di file, vengono trasformate nelle tabelle finali tramite *dbt*, uno strumento di trasformazione dati, passando per diverse fasi. Una volta importati su Snowflake, i dati vengono organizzati in **database** e **schema**:

- il database è un raggruppamento logico di schemi; ogni database appartiene a un singolo account Snowflake;
- uno schema è un raggruppamento logico di oggetti di database (tabelle, viste, ecc.); ogni schema appartiene a un singolo database.

Il database principale della Bonfiglioli è denominato `DWH`, al cui interno si trovano tre diversi schemi:

- **STG**: è lo schema dove sono caricati i file dello *stage* dopo che sono stati trasformati in tabelle; solitamente non si applicano trasformazioni e quindi le nuove tabelle Snowflake sono uguali alle tabelle SAP originali;
- **WRK**: è lo schema che viene usato per caricare eventuali tabelle di appoggio;
- **DWH**: è lo schema dove vengono caricate le tabelle nella loro forma finale.

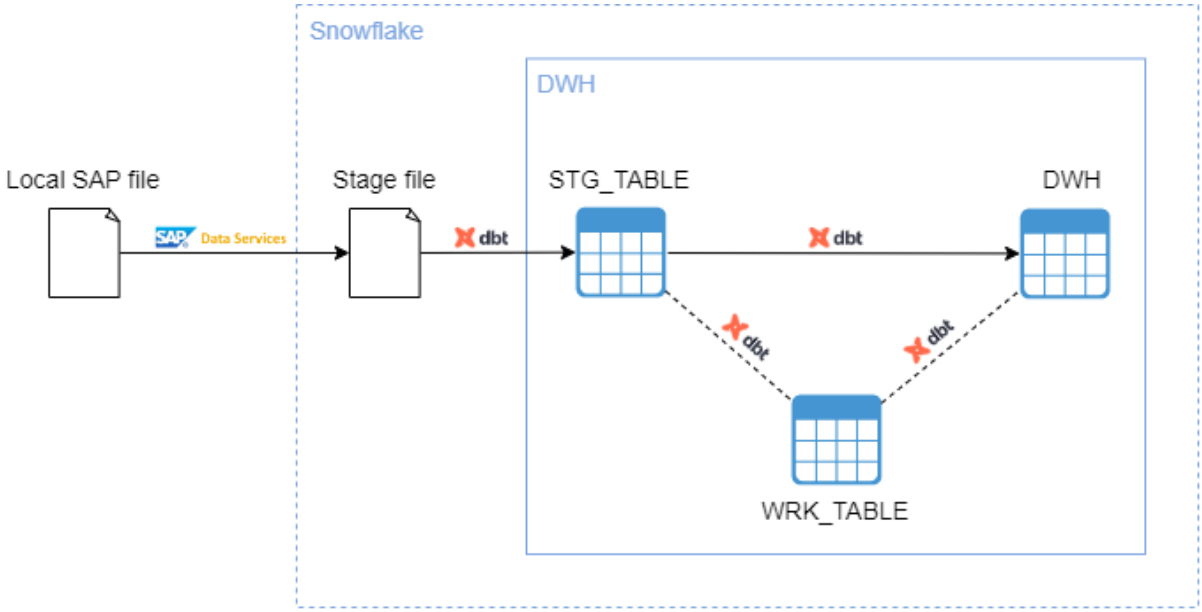


Figura 2.4: Schema dei vari passaggi per ottenere le tabelle finali.

Eventuali ulteriori trasformazioni sono eseguite direttamente su Snowflake.

dbt

dbt (data build tool) è lo strumento che la Bonfiglioli usa per prelevare i file dall'area di *stage* e trasformarli in tabelle Snowflake.

A differenza di un tradizionale strumento ETL, che estrae, trasforma e carica i dati, dbt si conforma meglio in un'architettura ELT (Extract, Load, Transform): data warehouse come Snowflake sono estremamente performanti e molto scalabili, tanto che nella maggior parte dei casi la trasformazione dei dati può essere gestita in modo molto più efficace all'interno del data warehouse piuttosto che in uno strumento di elaborazione esterno.

dbt si occupa della fase di trasformazione nell'ELT: prende il codice, lo compila in SQL e poi lo esegue sul data warehouse. Presenta molte funzionalità che permette all'utente di

scrivere trasformazioni riusabili e modulari, permettendo di avere un codice più leggibile e pulito: si possono definire e usare funzioni, `if` e cicli `for`, variabili d'ambiente ecc. Un progetto dbt è un insieme di file `.sql` e `.yml` e deve avere almeno:

- un file `dbt_project.yml` con le configurazione del progetto;
- uno o più **modelli**, ovvero una trasformazione dei dati, espressa con un'istruzione `SELECT`.

```
name: 'dwh'
version: '1.0.0'
config-version: 2

profile: 'bonfiglioli'

models:
  dwh:
    +schema: DWH
    +materialized: table
    +copy_grants: true

    PLM:
      +tags: ['PLM']
      Staging_PLM:
        +materialized: staging
        +schema: STG_PLM
      KPI:
        WRK:
          +schema: WRK
```

Listing 2.1: Estratto del progetto dbt della Bonfiglioli

Esecuzione

I vari modelli definiti con dbt vengono schedulati ed eseguiti tramite SAP Data Services, divenendo parte integrante dei vari *work flow*.

Poiché la Bonfiglioli è dislocata su diversi continenti, se la tabella da caricare è molto grande, viene divisa per continenti prima di scriverla sugli schemi `STG` e `WRK`, in modo che gli *work flow* durino di meno; le varie partizioni verranno poi unite prima di caricare la tabella finale su `DWH`.

2.2.3 Flusso di dati da PLC

Il team di big data usa come sorgente dati principale i PLC e seguono il flusso raffigurato in figura 2.5.

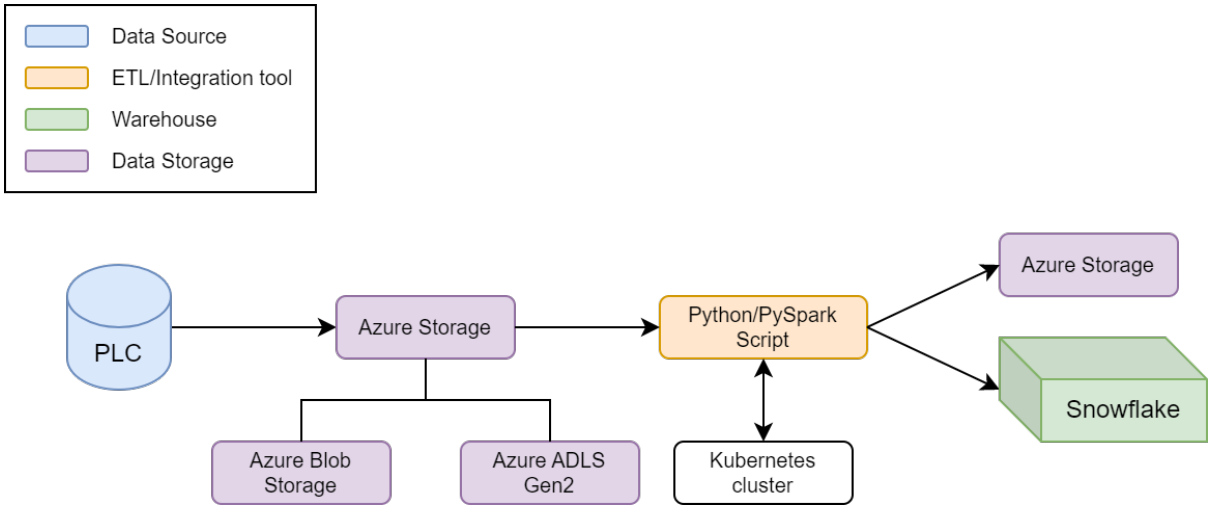


Figura 2.5: Flusso di trasformazioni che subiscono i dati provenienti da PLC.

Azure Storage

I dati provenienti dai vari PLC presenti nelle macchine vengono salvati direttamente su **Azure Storage** [24].

Azure Storage è un servizio cloud di Microsoft che offre diversi servizi di archiviazione. Ciascun servizio è progettato per soddisfare specifiche esigenze, come l'archiviazione di oggetti, la condivisione di file, l'accodamento di messaggi ecc. Azure Storage permette di scalare la capacità di archiviazione a secondo delle necessità degli utenti; per garantire disponibilità e resilienza, vengono offerte diverse opzioni di replicazione dei dati.

Dei vari servizi offerti, la Bonfiglioli usufruisce di:

- **Azure Blob Storage**: è una soluzione di tipo *object storage*, un tipo di archiviazione dove i dati sono gestiti come oggetti **blob** (binary large object);
  - i blob sono organizzati in *containers*, concettualmente assimilabili alle directory in un file system;
  - è ottimizzato per memorizzare dati non strutturati;
  - gli oggetti sono facilmente accessibili tramite richieste API REST;

- **Azure Data Lake Storage Gen2** (ADLS Gen2): è un insieme di funzionalità dedicate all'analisi dei big data che costituisce un data lake basato su cloud costruito sopra il *blog storage*. È progettato principalmente per usare Hadoop e gli framework Apache che fanno uso di HDFS.

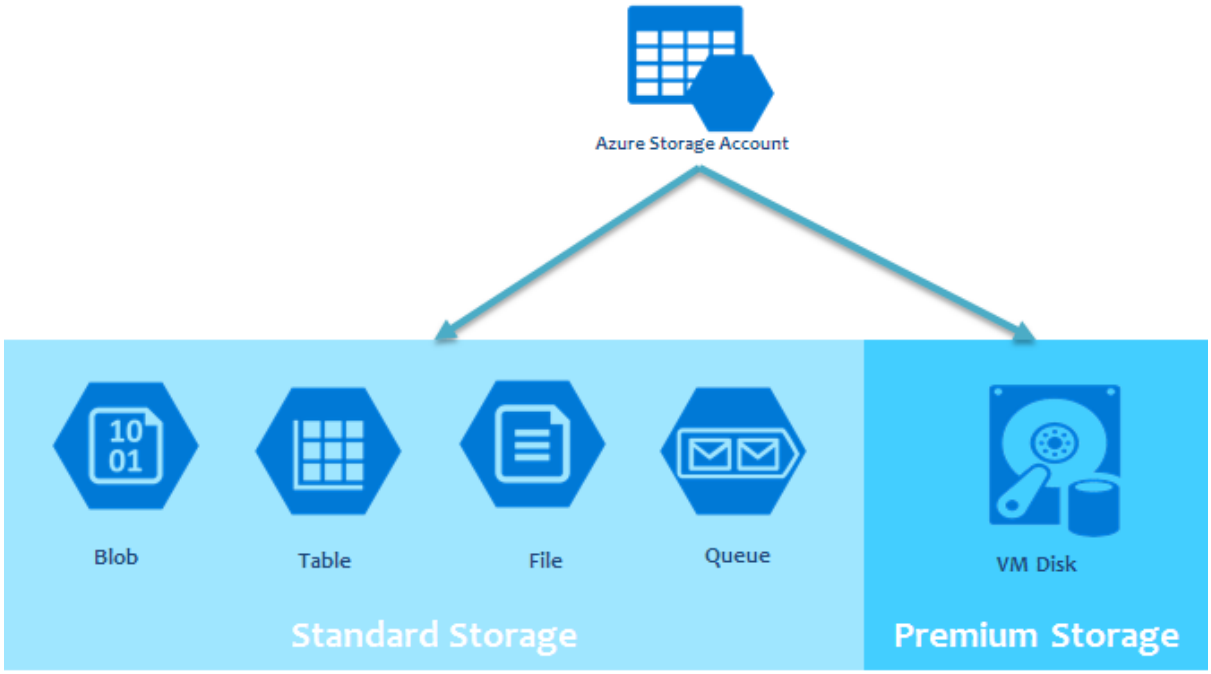


Figura 2.6: I vari tipi di archiviazione offerti da Azure Storage

Script

Dopo essere stati salvati su Azure, i dati subiscono varie trasformazioni attraverso **script** in Python/PySpark.

Questi script fanno uso di **Apache Spark**, un framework per l'elaborazione di grandi volumi di dati in un ambiente distribuito: quando viene lanciato un job Spark, questo esegue in parallelo, su diverse macchine, un insieme di computazioni su un certo partizionamento dei dati. Dividere i dati in blocchi più piccoli, per poi essere processati in più macchine, permette di avere un'elaborazione più veloce e scalabile [41].

Apache Spark quindi, tramite uno specifico connettore, legge i dati da Azure ed esegue le computazioni su un cluster Kubernetes. I risultati vengono poi salvati sull'Azure Storage/ADLS Gen2 oppure su Snowflake.

2.2.4 Reporting

Durante la sua attività, la Bonfiglioli ha usato vari strumenti di reportistica. La tesi si è focalizzata principalmente su **PowerBI** [26], un software di Microsoft e quindi nativamente integrato con gli altri prodotti dell'azienda, e in maniera minore su **Tableau** [45], recentemente acquisito da Salesforce.

Entrambi sono software di business intelligence che forniscono avanzate funzionalità di collezione, analisi e presentazione dei dati.

PowerBI

Le sorgenti che possono essere importate su PowerBI possono essere di vario tipo: file di testo, fogli Excel e diversi database, sia interni all'ecosistema Microsoft sia esterni, grazie a specifici connettori. L'insieme delle sorgenti, opportunamente rielaborato, diviene un unico dataset, pronto per essere usato nei vari *report*.

Un report può avere una o più pagine e ciascuna pagina può contenere uno o più grafici. Una dashboard invece è composta da una singola pagina e i suoi elementi grafici provengono da altri report. I vari dataset, report e dashboard possono essere organizzati in *workspace*, aree di lavoro per la collaborazione tra utenti.

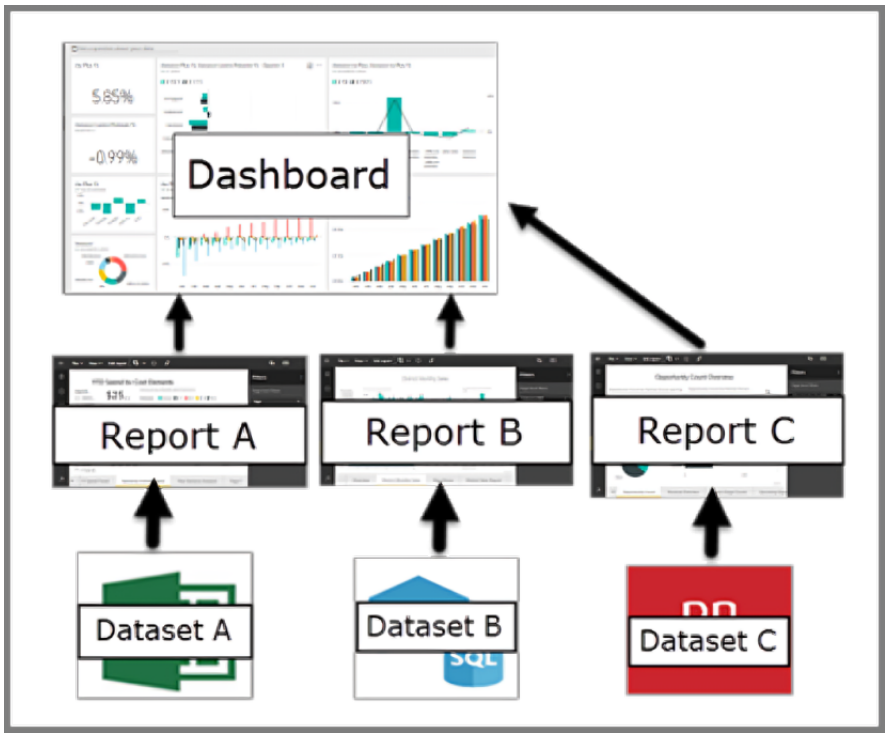


Figura 2.7: Diagramma esemplificativo dei vari livelli che compongono una dashboard PowerBI [26].

La Bonfiglioli usa poco le dashboard e generalmente report e dataset sono separati in due *workspace* distinti.

Tableau

Tableau segue una struttura non troppo dissimile da Microsoft Excel, ma con una terminologia diversa:

- un *workbook* può contenere uno o più *sheet*;
- un sheet può essere:
  - un *worksheet*, contenente una sola *view*;
  - una *dashboard*, una collezione di più *view*, proveniente da multipli *worksheet*;
  - una *story*, contenente una sequenza di *worksheet* o *dashboard* con lo scopo di comunicare una specifica narrazione.

Capitolo 3

Self-service BI e Data Catalog

Con gli sviluppi tecnologici degli ultimi decenni vengono collezionati e analizzati sempre più dati; parallelamente, sempre più organizzazioni realizzano come i dati siano una risorsa strategica fondamentale per la loro competitività. Per sfruttare al meglio queste opportunità, le organizzazioni devono integrare e armonizzare i dati in loro possesso [31]. Possiamo definire la **business intelligence (BI)** come "l'insieme di strumenti e procedure che consentono a un'azienda di trasformare i propri *dati aziendali* in **informazioni** fruibili a diversi livelli di dettaglio e, quindi, in **conoscenza** utili al processo decisionale" [16]. Le informazioni ottenute sono utilizzate dai decisori aziendali per definire e supportare le strategie di business, così da operare decisioni consapevoli e informate con l'obiettivo di trarre vantaggi competitivi, migliorare le prestazioni operative e la profittabilità e, più in generale, creare valore per l'azienda.

Poiché l'estrazione di conoscenza è un processo complesso e richiede capacità tecniche specifiche, tradizionalmente la BI viene servita come uno scambio di richieste tra i cosiddetti "*power user*" e i "*business user*" [44]:

- i *power user* hanno le capacità di navigare e analizzare i dati, creare report e dashboard; sono quindi tutti gli utenti in grado di generare informazioni, per sé stessi o per altri;
- i *business user* sono coloro che consumano le informazioni prodotte dai *power user*.

Poiché i *business user* non hanno la conoscenza e le abilità necessarie per usare strumenti di BI avanzati, possono avere bisogno di un frequente supporto da parte dei *power user* per soddisfare le proprie esigenze: i *power user* cercano di rispondere alle richieste dei *business user*; quando le risposte non sono sufficientemente complete o chiare, i *business user* chiedono ulteriore supporto. Lo scambio tra i due gruppi può continuare iterativamente e ciò può avere diversi costi aggiuntivi sull'azienda, che non può prendere decisioni

informate in tempi brevi [44]. Non solo, negli ultimi decenni, con la diffusione dei *Big Data*, sono aumentati i volumi e la velocità con cui le aziende acquisiscono dati e la BI è stata estesa dalle decisioni più strategiche anche a quelle più operazionali [3]. Tutto ciò ha fatto aumentare ancor di più la domanda di informazioni e analisi: il carico di lavoro cresce per i *power user*, che di fatto diventano un collo di bottiglia, mentre i *business user*, nell'impazienza, prendono decisioni critiche per il business senza sfruttare appieno i dati a loro disposizione [3].

È in questo contesto che viene introdotta la **self service BI** (SSBI): l'obiettivo della SSBI è quello di creare un ambiente che consenta ai *business users* di eseguire analisi personalizzate e di ricavare informazioni utili dai dati senza coinvolgere professionisti BI; non solo, la SSBI dovrebbe venire in aiuto anche dei *power users* stessi, rendendo il loro lavoro più semplice e veloce rispetto al passato [3]. Secondo un sondaggio di 1.999 risposte da parte di 587 professionisti che lavorano nel settore, condotto nel 2011 da Imhoff e White e incluso nel loro report sulla self-service BI, alcuni fattori che spingono a investire sulla SSBI sono [19]:

- l'incapacità di stare al passo con la velocità con cui cambiano i bisogni aziendali;
- l'incapacità dell'IT di soddisfare le nuove richieste in modo tempestivo;
- il bisogno di diventare un'organizzazione più *data-driven*;
- l'accesso a dati più completo e veloce.

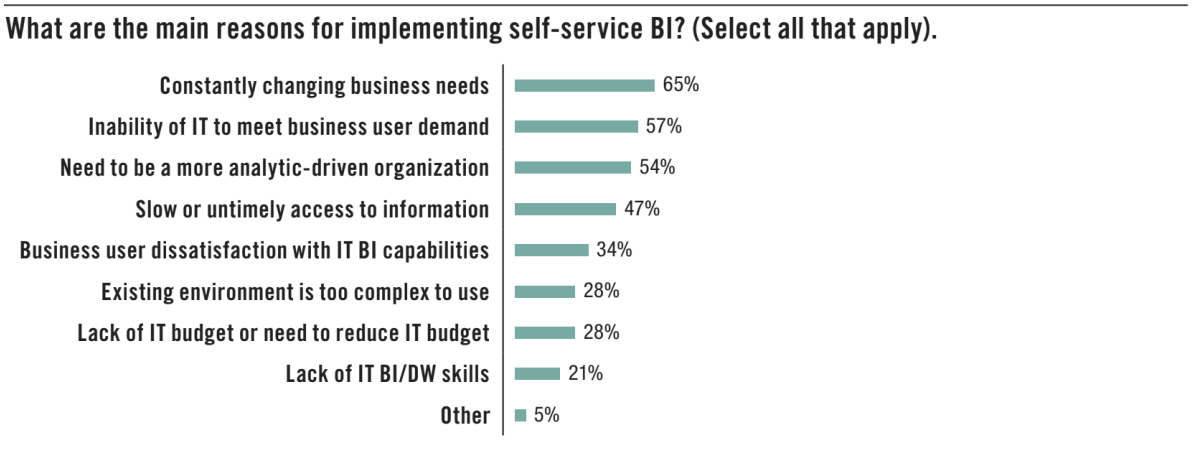


Figura 3.1: Risposte alla domanda "Quali sono i principali motivi per implementare la SSBI?" [19]

Secondo Imhoff e White, quindi, la SSBI dovrebbe focalizzarsi su 4 punti chiave.

- **Rendere le sorgenti facili da accedere**: è importante poter accedere facilmente a tutte le sorgenti che permettono agli utenti di comprendere il quadro completo del business. Questo potrebbe anche prevedere l'accesso a dati contestuali che risiedono al di fuori del *data warehouse*: di conseguenza è necessario un meccanismo che federi i dati e offri una vista globale delle varie sorgenti.

- **Semplificare l'uso di strumenti BI**: già da anni i fornitori di soluzioni BI cercano di rendere facile l'uso dei propri prodotti, un obiettivo che si può considerare raggiunto se si considerano le interfacce di *reporting* e *analytics* più semplici; se si vuole introdurre la SSBI però, è opportuno aumentare il supporto anche per gli *analytics* più sofisticati, tuttora ancora molto complessi, e le modalità di pubblicazione dei risultati.

- **Migliorare le modalità di consumo di risultati BI**: informazioni, report e *analytics* devono essere facili da scoprire, accedere e condividere. Non solo, avere la possibilità di personalizzare le *dashboard*, di inviarli al dispositivo di propria scelta e nel formato preferito, e di interagire con i risultati a proprio piacimento promuove un ambiente dove gli utenti possono essere più indipendenti e prendere decisioni più velocemente.

- **Provvedere un'infrastruttura ottimale**: affinché la SSBI possa raggiungere gli obiettivi promessi, tutto l'ambiente sottostante deve provvedere buoni livelli di performance e scalabilità per qualunque grado di carico di lavoro e volume di dati.

La SSBI promette quindi di sopperire alle difficoltà che la BI tradizionale non riesce a superare, offrendo un ambiente più flessibile e rendendo così gli utenti più auto-sufficienti. Tuttavia l'effettivo tasso di adozione è abbastanza basso. Seconda una review del 2018 [23], in letteratura sono presenti diversi articoli e sondaggi che sottolineano i problemi delle organizzazioni ad applicare con successo la SSBI, problemi legati principalmente al difficile accesso e uso dei dati e alla scarsa autonomia degli utenti. Non solo, per facilitare la SSBI, è importante che le aziende applichino anche politiche ottimali di *data management* e *data governance* [11].

Un possibile strumento che potrebbe venire in aiuto alle aziende per gestire meglio i propri dati è il **data catalog**. Fondamentalmente, un data catalog è un inventario organizzato dei dati di un'azienda: provvede una panoramica a livello dei metadati di un'organizzazione, integrato con un motore di ricerca che rende più facile trovare i dati desiderati [28].

Analizziamo nelle sezioni successive cos'è un data catalog e quali sono le sue caratteristiche desiderate.

3.1 Data Catalog

Possiamo definire un data catalog come un "inventario organizzato dei *data asset* — inteso come un qualsiasi dato in possesso dall'azienda, quali dati strutturati, non strutturati, dashboard, modelli di *machine learning* e così via — che, attraverso la raccolta e la manutenzione di metadati provenienti da sorgenti eterogenee, facilita gli utenti nella ricerca e nell'uso dei dati" [28] [18] [31].

Un data catalog ha quindi tre funzionalità essenziali:

- è un database di metadati che sono stati estratti dalle sorgenti;
- provvede un visione organizzata degli asset;
- è un motore di ricerca che permette di cercare gli asset catalogati.

3.1.1 Metadati

Alla base di tutto ci sono i metadati. I **metadati** sono tutte quelle informazioni che non sono i dati stessi: sono pervasivi in qualunque sistema informativo, possono assumere forme diverse e con il tempo hanno assunto un ruolo sempre più importante per la definizione dei dati, la loro integrazione e il loro uso [37]. Per esempio, i metadati possono essere utili per tenere traccia dei dati durante tutto il loro ciclo di vita, dalla sorgente, ai processi di trasformazione che subiscono, fino alla loro rappresentazione all'utente finale; oppure possono essere usati per analizzare l'impatto che un cambiamento può avere sui dati [37].

Popolamento del data catalog

Le sorgenti da cui un data catalog può estrarre i metadati possono essere di diverso tipo: un sistema IT, un'applicazione, un data warehouse, una piattaforma o anche dei semplici fogli Excel.

Solitamente i data catalog caricano i metadati in modalità *pull* attraverso l'uso di *connettori* nativi [28]. Un connettore è un programma progettato per interrogare una sorgente specifica ed estrarne tutti i metadati rilevanti; sorgenti diverse richiederanno connettori diversi. Considerando il numero enorme di sorgenti esistenti nel panorama

informatico, raramente un data catalog sarà già fornito nativamente di tutti i possibili connettori di cui un'azienda potrebbe aver bisogno e quindi il numero di sorgenti supportate è un aspetto importante da tenere conto durante la scelta di un data catalog. In mancanza di un connettore, si possono usare le API provviste dalla sorgente per estrarre i metadati e caricarli manualmente all'interno del catalog.

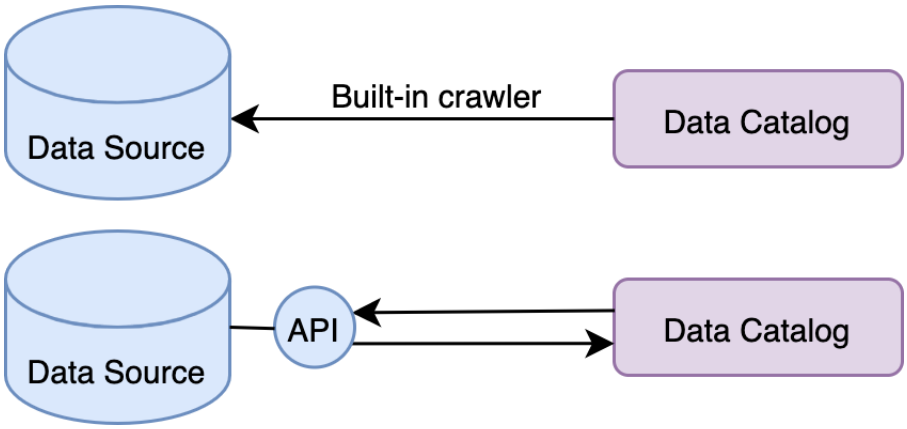


Figura 3.2: Possibili modalità per popolare un data catalog

Tipologie di informazioni

I metadati raccolti dal data catalog possono essere di diverso tipo [31] [38].

- **Metadati descrittivi**: provvedono informazioni in lingua naturale sull'asset stesso, come nomi e definizioni di tabelle e colonne o descrizioni del dominio di business.
- **Metadati amministrativi**: includono informazioni utili per la corretta gestione o creazione dell'asset; possono essere:
  - informazioni tecniche sulla struttura fisica, come il formato dei dati, dell'encoding o il tipo di storage;
  - informazioni legate all'*ownership*, ai diritti d'accesso o alla sensibilità dei dati;
  - informazioni di *data provenance*, come l'origine dei dati e tenere traccia della trasformazioni che queste subiscono, detto anche *data lineage*.
- **Metadati strutturali**: rappresentano le relazioni e le interazioni tra gli asset e le sue parti, come le relazioni tra le chiavi in tabelle relazionali.

Ulteriori arricchimenti

A volte i metadati che provengono dalle sorgenti non sono sufficienti. Di conseguenza il data catalog deve essere ulteriormente arricchito affinché provveda valore all'azienda, per esempio aggiungendo termini al *business glossary*, integrando le descrizioni, classificando gli asset e tanto altro.

Business glossary

Poiché i dati possono essere usati da un gruppo vario di analisti e interpretati da diversi stakeholders, avere definizioni consistenti e facilmente accessibili è importante affinché tutti siano allineati sullo stesso significato. Se ciò non accade, due persone potrebbero assumere che un certo termine voglia indicare un concetto invece di un altro, un errore che può propagarsi e venire identificato solo dopo che la reportistica è già stata presentata e vengono messi in dubbio i risultati [47].

Per comprendere quindi al meglio il contesto dei dati e il loro corretto utilizzo, è utile avere una visione chiara delle parole chiave del business, condivisa trasversalmente da tutta l'azienda. Per questo motivo, molti data catalog offrono agli utenti la possibilità di creare una lista di parole e le loro rispettive definizioni, chiamata appunto *business glossary*.

Classificazione

Sempre nell'ottica di provvedere ulteriori informazioni sui dati, un data catalog potrebbe classificare gli asset sulla base del loro contenuto, confidenzialità e sensibilità, tramite l'identificazione automatica di certi pattern e regole nei dati o l'aggiunta manuale di *label* [28].

- Contenuto: i valori di un asset rispettano certi pattern — come codice IBAN, numero di carta d'identità, URL ecc. — o più in generale l'asset è associato a un certo dominio di business, quali finanza o marketing.
- Confidenzialità: esplicita quanto gli asset sono "segreti"; ciò facilita la definizione delle regole d'accesso ai dati.
- Sensibilità: indica se gli asset contengono dati sensibili o meno, permettendo di avere una visione migliore di quali sono i dati che necessitano di politiche di protezione di dati.

Data tagging

È possibile aggiungere "tag" o "label" agli asset. Questi tag possono essere termini dal *business glossary* o provenienti dalla classificazione dei dati [28].

Questi tag possono essere usati per filtrare gli asset durante la ricerca.

Persone

È possibile associare all'asset le persone ad esso rilevanti. Per esempio, un asset può avere un *owner*, colui che possiede il dato originale e definisce chi può accedere alla sorgente e per cosa i dati possono essere usati, o un *steward*, colui che mantiene e gestisce l'asset [28].

3.1.2 Organizzazione

Una volta che il data catalog è stato popolato da metadati, è possibile organizzarli in domini.

La suddivisione in domini è a scelta dell'azienda: si possono unire in un unico dominio tutti i dati che supportano un certo processo aziendale, ad esempio per la creazione di un prodotto o un servizio; in alternativa, un dominio può essere costituito dai dati che supportano un certo reparto [28]. L'organizzazione a domini permette di unificare quei dati che sono accomunati da conoscenze e obiettivi simili.

Un data catalog visualizza questa organizzazione tramite una struttura simile a quella di un albero, che può essere ampliata a proprio piacimento in larghezza e profondità:

- in cima c'è la radice, punto di partenza di tutti i domini;
- un dominio può avere eventuali sottodomini;
- ciascun (sotto)dominio può essere alimentata da una o più sorgenti, ovvero le componenti tecnologiche che supportano il processo o la capacità, come database, data lake, data warehouse o applicazioni.
- ogni sorgente può avere uno o più asset, appropriatamente descritti da metadati.

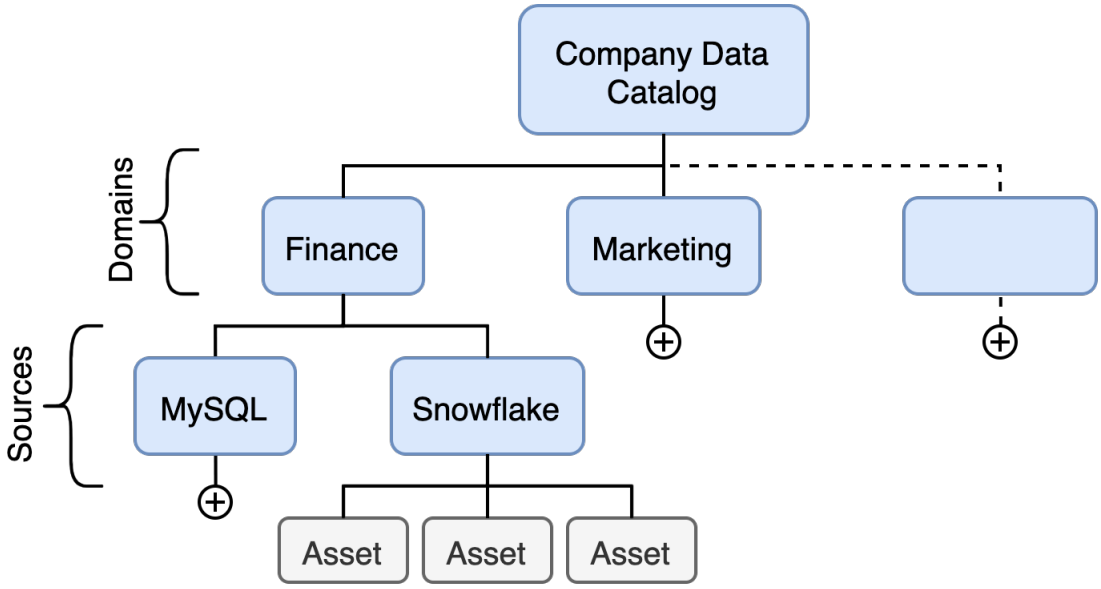


Figura 3.3: Un esempio di organizzazione degli asset di un'azienda per dominio di business

3.1.3 Ricerca

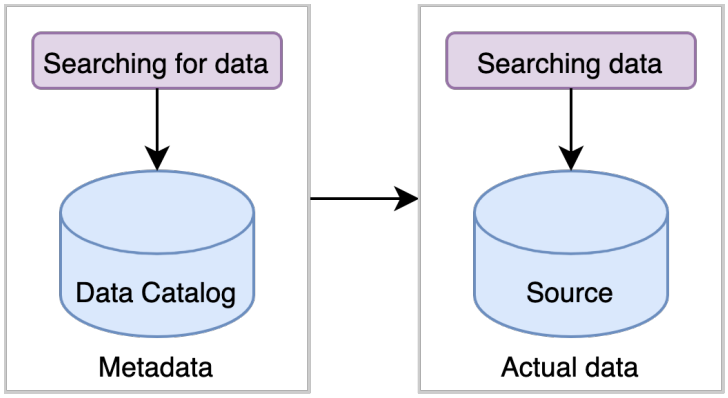
Arricchire i dati aziendali con metadati e il contesto di business nel quale il dato risiede e opera, e organizzare tutto ciò in un catalogo facilmente accessibile, agevola la fruibilità dei dati stessi, in quanto gli utenti hanno maggiore consapevolezza su dove si trova il dato e come deve essere usato [38] [28]. Un aspetto chiave dei data catalog, infatti, è proprio la possibilità di poter cercare i dati in possesso da un'azienda come se fosse un motore di ricerca.

Cercare dentro un data catalog vuol dire applicare tecniche di *information retrieval* per trovare gli asset che sono più rilevanti al bisogno informativo espresso dall'utente, asset che rimanderanno poi all'effettiva sorgente, in quanto stiamo cercando dentro un database di metadati e non la sorgente stessa. L'efficacia della ricerca dipenderà dalla qualità stessa del data catalog: un data catalog con asset ben arricchiti ritornerà risultati migliori rispetto ad asset con poche descrizioni e tag [28].

In modo da coprire il più possibile le esigenze degli utenti, da quelli con richieste più semplici a quelle più complesse, il data catalog deve fornire diversi meccanismi di ricerca. Ciò può comprendere [28]:

- ricerche semplici digitando parole in una barra di ricerca;
- navigare gli asset verticalmente, da dominio a dominio, oppure orizzontalmente, seguendo il *data lineage*;

- filtrare gli asset per termini del glossario, classificazione, dominio o per le persone associate;
- usare operatori per effettuare ricerche più complesse.



3.1.4 Altri requisiti funzionali

Il data catalog può offrire una serie di funzionalità molto vario a seconda del fornitore. Per esempio, per ampliare ulteriormente le modalità di fruizione dei dati, alcuni data catalog mettono a disposizione anche la possibilità di accedere direttamente ai dati e di interrogarli, oppure di visualizzarne dei campioni.

Da un'analisi [31] della letteratura scientifica attualmente presente, sono stati estratti 13 requisiti funzionali che un data catalog dovrebbe idealmente possedere quando questo è inserito in un contesto aziendale.

Escludendo quelli legati a popolamento e ricerca degli asset, i restanti requisiti possono essere suddivisi come segue.

1. Data governance
    - Un data catalog dovrebbe adeguarsi a linee guida, interne o esterne all'azienda, chiare e coerenti di *data governance*, compresi la definizione di ruoli non ambigigui, *ownership* dei dati, politiche di qualità dei dati.
    - Un data catalog dovrebbe fornire meccanismi affidabili di *access control* e altre politiche che regolano l'uso, la gestione e l'accesso dei dati in termini di sicurezza e privacy.
    - Un data catalog dovrebbe provvedere meccanismi che controllino la qualità dei dati; inoltre dovrebbe permettere all'utente di definire standard e metriche di qualità che possano essere riviste continuamente per evitare dati errati o non affidabili.

2. Interoperabilità

    - Un data catalog dovrebbe incorporare API per interrogare i propri asset e tutte le loro proprietà, in modo che possano essere facilmente integrate nelle infrastrutture già esistenti.

3. User experience

    - Un data catalog dovrebbe facilitare l'uso e la navigazione dello strumento da parte di utenti tecnici e non attraverso interfacce intuitive.
    - Un data catalog dovrebbe inserire funzionalità che favoriscano la collaborazione e la comunicazione tra utenti per migliorare lo scambio collettivo di informazioni.

4. Funzionalità avanzate

    - Un data catalog dovrebbe essere dotato di funzionalità di intelligenza automatica che riduca le attività manuali di *data discovery*, analisi, *data management* e manutenzione da parte degli utenti.

3.2 Le promesse: discovery, management, governance, provenance

L'ambiente tipico dentro il quale si potrebbe sentire il bisogno di introdurre un data catalog è un ambiente dove i dati sono frammentati e archiviati in numerosi sistemi IT eterogenei e isolati fra di loro: diventa difficile trovare l'origine dei dati, valutare la loro qualità e affidabilità o individuare quali sono le relazioni presenti tra di loro. Tutto ciò impedisce all'azienda non solo di usare i dati a proprio vantaggio, ma rende più difficile adottare iniziative come la SSBI [31][38][11] .

Il data catalog potrebbe essere quindi lo strumento appropriato con il quale i *business users* possono essere in grado di esplorare autonomamente i dati. Questo processo di ampliamento della disponibilità dei dati a un pubblico più ampio rispetto ai tradizionali utenti esperti [21] viene detto "data democratization" e viene spesso correlato proprio con i data catalog [31] [22]: la possibilità di cercare, navigare, documentare i dati sono tutte funzionalità molto richieste nei data catalog e che permettono agli utenti di sapere dove e come trovare i dati e il loro significato.

Comprendere i dati è un requisito fondamentale per applicare la SSBI. Gli autori del paper "Self-Service Business Intelligence" [3] definiscono l'esistenza di diversi livelli di self-service.

- Accesso a risorse o report già preparati: può essere adatto per gli utenti che non hanno competenze avanzate, ma risulta poco flessibile per chi desidera fare analisi più approfondite.

- Accesso diretto ai dati al livello meno aggregato: solo gli utenti stessi hanno la visione completa delle proprie esigenze e bisogni, che non sempre risulta chiara ai *power users* quando questi devono soddisfare le richieste dei *business user*; c'è necessità quindi di avere strumenti visuali che permettano ai *business user* di fare report sul momento; il rischio di questi strumenti è che gli utenti non comprendano appieno le relazioni che legano i dati e usino dati incorretti.

- Accesso alle funzioni: a volte bisogna fare analytics molto complesse (es: predictive analytics). È possibile rendere disponibili queste funzioni, portando più autonomia, ma allo stesso tempo utenti inesperti potrebbero non essere in grado di valutare correttamente i risultati.

- Creazione di nuove risorse: offrire la possibilità di creare nuove risorse a partire da dati che non sono già stati preprocessati da IT è il livello ultimo di self-service; anche in questo caso si presentano dei rischi: bisogna prestare massima attenzione alla qualità dei dati, assicurarsi che le regole di accesso siano rispettate e che i dati siano compresi appieno.

Da notare quindi come all'aumento di self-service, aumenti anche la necessità che gli utenti conoscano bene i dati con cui lavorano. Gli stessi autori affermano anche che:

- diffondere l'uso di BI sia possibile attraverso la collaborazione tra utenti tramite *enterprise social network* e *wiki*, così che gli utenti possano riusare e adattare report;

- si possono usare tecniche di *information retrieval* e sistemi di *recommendation* per supportare gli utenti nel trovare le informazioni rilevanti;

- data governance e sicurezza dovrebbero essere aumentate affinché solo gli utenti autorizzati abbiano diritto ad accedere, modificare, aggiungere dati;

tutte affermazioni che sono in linea con la definizione e le promesse di un data catalog. Il data catalog è compatibile anche con l'importanza posta da Imhoff e White nel loro

report sulla SSBI sul "poter accedere facilmente a tutte le sorgenti che permettono agli utenti di comprendere il quadro completo del business" [19].

3.2.1 Data Management & Governance

Generalmente il termine *data management* viene usato per indicare l'insieme delle procedure e dei processi usati per gestire i dati [43]. Il manuale "Data Management Body of Knowledge" (DAMA-DMBOK) [20] definisce il *data management* come "lo sviluppo, l'esecuzione e la supervisione di piani, politiche, programmi e pratiche che forniscono, controllano, proteggono e valorizzano i dati e gli asset informativi durante tutto il loro ciclo di vita". Oggi il data management è reso ancor più complesso da diversi fattori come [43]:

- analisi avanzate, che forzano la ridistribuzione di dati in varie applicazioni, poiché domini di business diversi necessitano di dati che vengono plasmati per rispondere alle esigenze specifiche di analisi;
- il passaggio da infrastrutture monolitiche e centralizzate ad architetture più distribuite, che distribuiscono i dati in componenti più piccoli;
- il cloud, che porta alla duplicazione dei dati per muoverli più vicini alla potenza computazionale.

In un ambiente come questo, la sicurezza, la privacy e la corretta gestione e consumo dei dati diventano un aspetto chiave [43].

La *data governance*, in particolare, che il DAMA-DMBOK pone al centro delle pratica di data management 3.4, specifica i diritti decisionali e le responsabilità all'interno del processo decisionale di un'organizzazione sui propri dati: determina quali decisioni è necessario prendere sui dati, come queste decisioni sono prese e chi ha il diritto di prenderle. Inoltre formalizza le politiche, gli standard e le procedure relative ai dati e ne monitora la conformità [1].

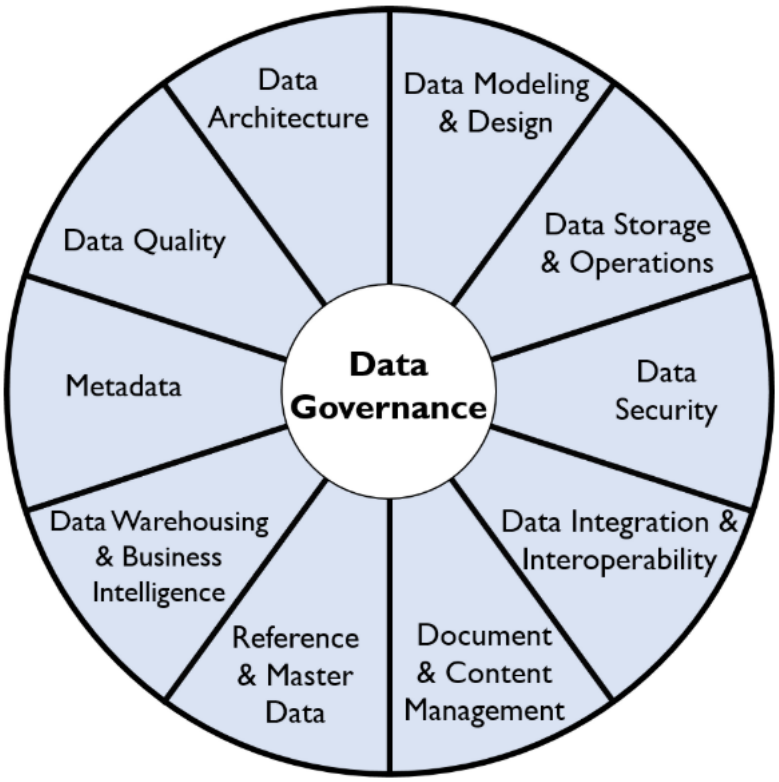


Figura 3.4: Le 11 aree funzionali del data management, al cui centro troviamo la *data governance* [20].

Analizzare la natura dei dati, il modo in cui vengono usati e da chi è importante per capire quali sono le politiche di data governance più appropriate da mettere in atto. Anche qui però la mancanza di una visione d'insieme dei dati ostacola l'applicazione di governance [47] [9] [6], problema che può essere risolto con un data catalog; non solo, funzionalità come la classificazione dei dati, il business glossary e il lineage possono essere d'aiuto durante la definizione, applicazione e monitoraggio della data governance.

3.2.2 Data Provenance

In generale, il termine **provenance** indica la collezione di informazioni che descrivono l'origine e i cambiamenti che subiscono un prodotto all'interno di un certo processo di produzione [46]. In ambito informatico, invece, la provenance può essere declinata come *data provenance*, intesa come la collezione di metadati a livello dei dati, considerati nella loro granularità più fine e inclusiva delle operazioni che queste subiscono [17].

La stragrande maggioranza dei data catalog, oltre ai metadati, estrae anche le informazioni riguardanti le trasformazioni che un asset subisce e le visualizza in un grafico, comunemente riferito come *data lineage*. Avere uno strumento visuale facilmente naviga-

bile supporta gli utenti nel raccogliere e analizzare la *provenance*, permettendo di avere maggiore [39][17] [46]:

- comprensibilità, poiché viene spiegato come sono stati ottenuti i risultati e con quali processi;
- responsabilità, perché esplicita chi è l'*owner* delle sorgenti e delle elaborazioni effettuate;
- riproducibilità, perché è possibile verificare se un risultato può essere ricreato a partire dalle stesse fonti e metodi;
- opportunità di miglioramento, facilitando il monitoraggio e il debugging dei processi;
- qualità, in quanto permette di analizzare le sorgenti e le elaborazioni attualmente utilizzate e valutarne l'affidabilità e la correttezza.

Capitolo 4

Il caso Bonfiglioli

4.1 Progetto

Appartenendo al settore metalmeccanico, il reparto di BI non è il core del gruppo Bonfiglioli. Tuttavia i dati svolgono comunque una funzione molto importante, soprattutto quando avvengono problemi e nasce la necessità di comprenderne le cause: i dati coprono quindi sia un ruolo descrittivo e predittivo, per la costruzione di dashboard e report, sia un ruolo diagnostico.

Per questo motivo, la Bonfiglioli ha espresso il desiderio di innovare l'attuale processo di gestione dei propri dati. In particolare, l'azienda ha mostrato interesse nell'inserire un data catalog nel proprio stack tecnologico, un'aggiunta che si colloca nell'obiettivo più ampio di adottare pratiche SSBI: l'idea è che il data catalog funga da supporto per i dipendenti nella ricerca e comprensione dei dati.

4.1.1 Requisiti

1. Il data catalog deve provvedere business glossary e data lineage, le funzionalità più di interesse per l'azienda.
2. È desiderabile che il data catalog offra anche data profiling.
3. Il data catalog deve supportare Snowflake, PowerBI e Tableau.
4. Possibilmente, lo strumento deve essere in grado di:
    - connettersi ad Azure Storage;
    - popolare il data catalog e costruire il data lineage a partire da script in Python/PySpark;

5. Il supporto per dbt è opzionale, in quanto dbt provvede nativamente un data lineage, che tuttavia è molto complesso e può risultare difficile da navigare per utenti non esperti.

6. Il data catalog deve dimostrasi uno strumento intuitivo e facile da usare.

7. Il data catalog non deve richiedere un'elevata manutenzione.

4.1.2 Obiettivo

L'obiettivo della tesi è trovare un data catalog che risponda il più possibile a questi requisiti. Verrà posta particolare attenzione all'esaustività del data lineage e al livello di dettaglio con cui è possibile arricchire il data catalog, in termini di business glossary e descrizioni a tutti gli livelli di un asset.

Gli strumenti che sono stati presi in considerazione sono stati:

- **Apache Atlas**;
- **Microsoft Purview**;
- **OpenMetadata**;
- **Atlan**;
- **Spline**.

4.2 Apache Atlas

Apache Atlas [4] è un framework open-source di *data governance* che provvede diversi servizi per aiutare le aziende a catalogare, classificare e governare i propri dati, e facilitare la collaborazione tra data scientist, analisti e consumatori di dati.

I componenti di Atlas possono essere classificati in macro-categorie.

- **Core**
  - **Type system**: per gestire i metadati, gli utenti devono definirne il loro modello; un modello è composto da definizioni chiamate "*types*", mentre istanze di un 'types' sono dette 'entity' e rappresenta il metadato vero e proprio. Viene chiamato *type system* il componente che permette agli utenti di definire e gestire tipi ed entità.

  - **Graph engine**: internamente i metadati vengono salvati con un modello a grafo, in quanto questo tipo di persistenza offre maggiore flessibilità e permette di gestire in modo efficiente le relazioni tra metadati. Il *graph engine* non è altro che il componente responsabile della traduzione tra i tipi e le entità del *type system* e il modello sottostante.
  - **Ingest/Export**: *ingest* è il componente che consente di aggiungere metadati ad Atlas, mentre *export* è il componente che espone eventuali cambiamenti ai metadati rilevati da Atlas come eventi. Questi eventi possono essere consumati dai *consumer* per reagire in tempo reale ai cambiamenti.

- **Integrazione**
  - **API**: tutte le funzionalità di Atlas sono esposte agli utenti tramite una REST API che permette di creare, aggiornare, eliminare, interrogare e navigare tipi ed entità.
  - **Messaging**: Atlas può essere integrato con un'interfaccia a messaggi basato su Kafka. Ciò permette di consumare i cambiamenti ai metadati come eventi.

- **Sorgenti**: nativamente Atlas supporta il consumo e la gestione di metadati dalle seguenti sorgenti: HBase, Hive, Sqoop, Storm, Kafka. L'utente può integrare Atlas con ulteriori sorgenti aggiungendo i *type* necessari al *type system*.

- **Applicazioni**: i metadati gestiti da Atlas possono essere consumati da diverse applicazioni a seconda delle necessità che si vuole soddisfare. Di base, esiste l'**Admin UI**, un'applicazione web che permette gli utenti di navigare e gestire i metadati.

Essendo open-source, adottare Apache Atlas non implica costi monetari. Tuttavia, non supportando nativamente nessuna delle sorgenti richieste da Bonfiglioli, richiede un alto costo di messa in opera e manutenzione, in termini di tempi e risorse umane: è necessario aggiungere tutti i *type* necessari e sviluppare componenti che tramite l'API inseriscono i metadati all'interno del data catalog. Da valutare comunque che, a un alto costo di installazione e gestione, si affianca una maggiore personalizzazione, potendo liberamente interagire con l'API aggiungendo tutte le sorgenti desiderate, cosa non possibile con prodotti proprietari.

A seguito una breve descrizione delle funzionalità chiave.

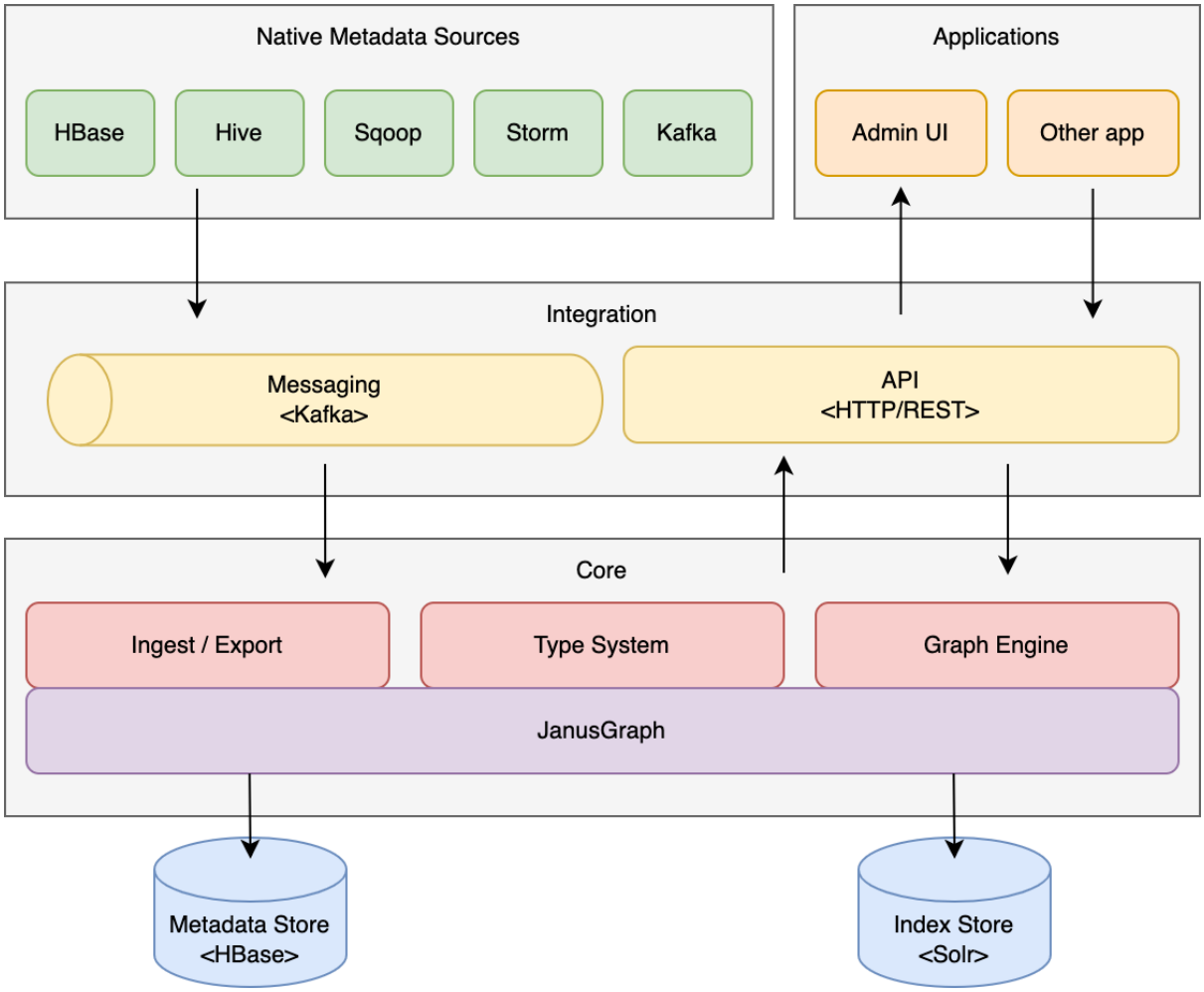


Figura 4.1: Architettura ad alto livello di Apache Atlas.

4.2.1 Type

Similmente alla definizione di una classe nella programmazione a oggetti, o allo schema di una tabella relazionale, un tipo è una collezione di attributi che definisce le proprietà di un metadato e come questo viene salvato e acceduto.

Un tipo è identificato univocamente da un nome e ha un metatipo. In Atlas i metatipi possono essere:

- primitivi: bool, int, long, date, string ecc;
- enumerazioni;
- collezioni: array, map;
- composti: entity, struct, classification, relationship.

I tipi *entity* e *classification* possono estendere da uno o più supertipi e di conseguenza ereditare gli attributi definiti nel supertipo. Atlas ha diversi tipi predefiniti da cui l'utente può estendere per supportare ulteriori sorgenti:

- **Referenceable**: rappresenta un *tipo* che può essere cercato usando un attributo univoco chiamato *qualifiedName*;
- **Asset**: estende *Referenceable*; aggiunge attributi quali nome, descrizione e owner.
- **DataSet**: estende *Referenceable*; può essere usato per rappresentare un *tipo* in grado di contenere dati e partecipare in trasformazioni che possono essere acquisite da Atlas attraverso il *lineage*.
- **Process**: estende *Asset*; può essere usato per rappresentare una qualunque operazione di trasformazione dati; ha due attributi, *input* e *ouput*, di tipo array di entità *DataSet*.

Se volessimo quindi aggiungere una nuova sorgente ad Atlas, sarebbe necessario definire nuovi tipi che estendono *DataSet* e *Process*, inserendo tutti gli attributi necessari.

```
Name:          hive_table
TypeCategory:  Entity
SuperTypes:    DataSet
Attributes:
    name:              string
    db:                hive_db
    owner:             string
    createTime:        date
    lastAccessTime:    date
    comment:           string
    retention:         int
    sd:                hive_storagedesc
    partitionKeys:     array<hive_column>
    aliases:           array<string>
    columns:           array<hive_column>
    parameters:        map<string>
    viewOriginalText:  string
    viewExpandedText:  string
    tableType:         string
    temporary:         boolean
```

Listing 4.1: Tipo di una tabella Hive

4.2.2 Funzionalità

In Atlas è possibile cercare le entità per testo o filtrandole per tipo, classificazione o termine di glossario. In alternativa, è possibile usare un DSL (Domain Specific Language), un linguaggio sintatticamente simile a SQL, o l'API per effettuare ricerche più avanzate.

Glossario

È possibile creare uno o più glossari dove gli utenti possono inserire termini che possono essere poi associati alle varie *entity*. Un glossario può essere diviso in categorie e una catagoria può essere ulteriormente strutturata in gerarchie.

Si può interagire con il glossario tramite UI o API.

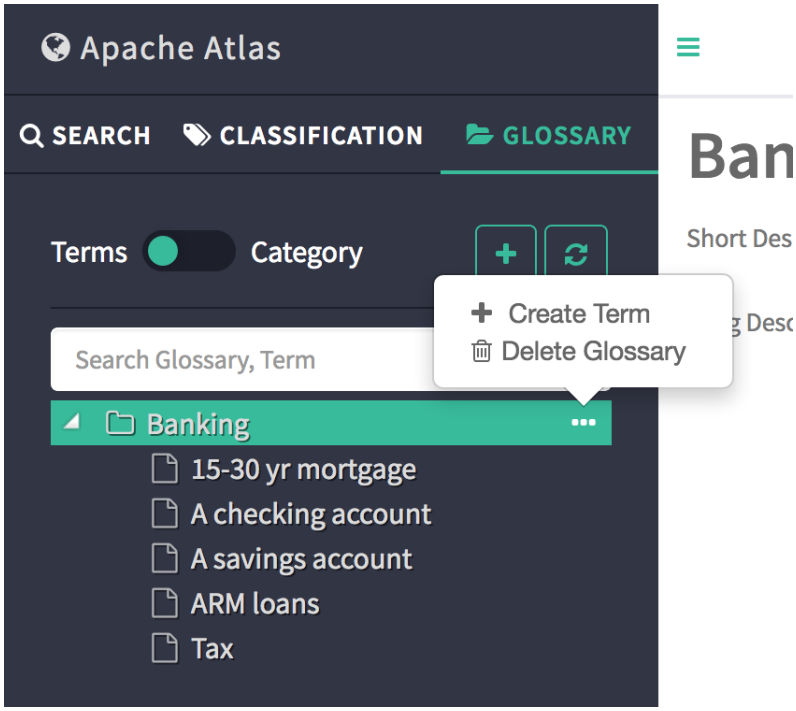


Figura 4.2: UI per inserire un nuovo termine o eliminare il glossario [4]

Lineage

Atlas provvede a mostrare le trasformazioni di una entità come un grafo. Quando un'entità viene classificata, questa classificazione può essere automaticamente propagata alle entità figlie seguendo il lineage.

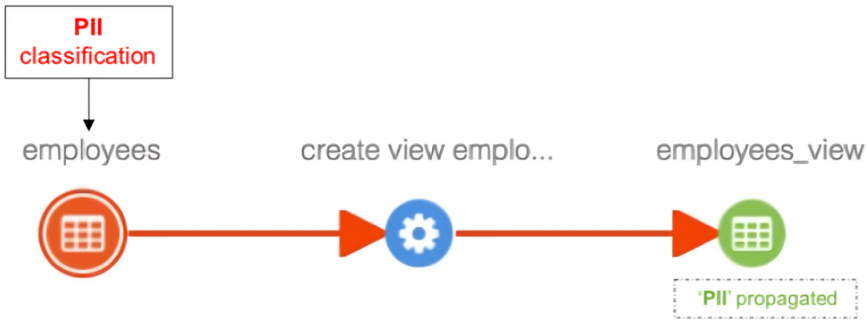


Figura 4.3: Propagazione di una classificazione sul lineage di una tabella [4]

4.3 Microsoft Purview

Microsoft Purview [27] viene presentato come "una soluzione unificata per la *data governance* che aiuta a gestire e governare i propri dati; permette di creare facilmente una mappa olistica e aggiornata dei dati con *data discovery* automatico, classificazione dei dati sensibili e *data lineage*, offrendo agli utenti una gestione affidabile dei dati".

Microsoft Purview è basato su Apache Atlas ed estende e supporta tutte le sue API. Oltre a data catalog, data lineage e business glossary, offre diversi servizi, quali data estate insights, data map e data sharing.

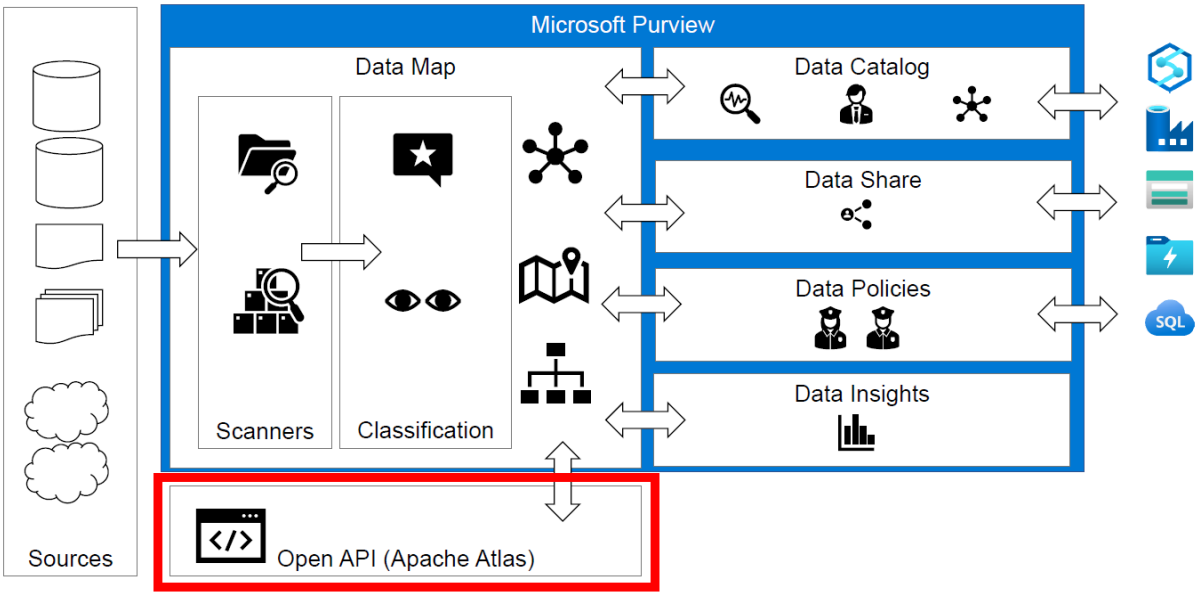


Figura 4.4: Microsoft Purview & Apache Atlas APIs (Wolfgang Strasser 2023) [42]

Essendo un prodotto Microsoft, supporta molti prodotti interni, tra cui Azure Storage e PowerBI, e le politiche d'accesso sono già nativamente integrate con Azure. Tra i prodotti esterni, supporta unicamente Snowflake, ma non Tableau, scripting o dbt.

4.3.1 Data catalog e Lineage

Microsoft Purview è integrato con diversi connettori con la quale è possibile popolare il data catalog: è sufficiente registrare la sorgente su Purview inserendo i dati necessari, come indirizzo e credenziali; in automatico, Purview stabilisce una connessione con la sorgente ed estrae i metadati.

Già durante questo caricamento, Purview effettua una classificazione degli asset, aggiungendo eventuali tag quando il suo valore corrisponde a una certa regola di classificazione, come può essere il caso per il numero di una carta di identità o un numero di telefono.

Dai metadati estratti, viene inferito anche il data lineage tra i vari asset: dove Purview fallisce a ricostruire il lineage, è sempre possibile usare l'API per aggiornarlo manualmente.

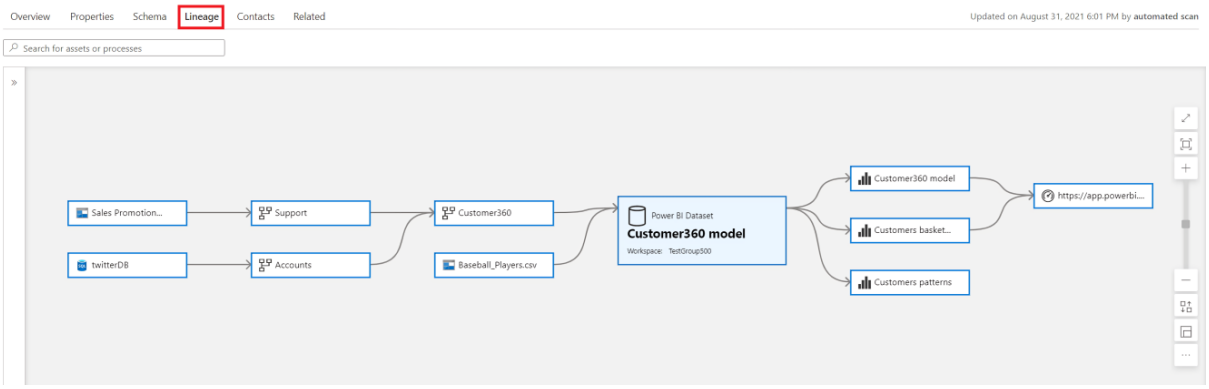


Figura 4.5: Lineage di un dataset di PowerBI [27]

4.3.2 Altre funzionalità

Un aspetto interessante di Purview è la possibilità di assegnare una certa sorgente, all'atto della sua registrazione, a un dominio o una collezione: ciò permette di organizzare gli asset e le rispettive sorgenti ricalcando l'organizzazione dell'azienda stessa, provvedendo una vista navigabile dei propri dati. Questa funzionalità viene chiamata "*data map*".

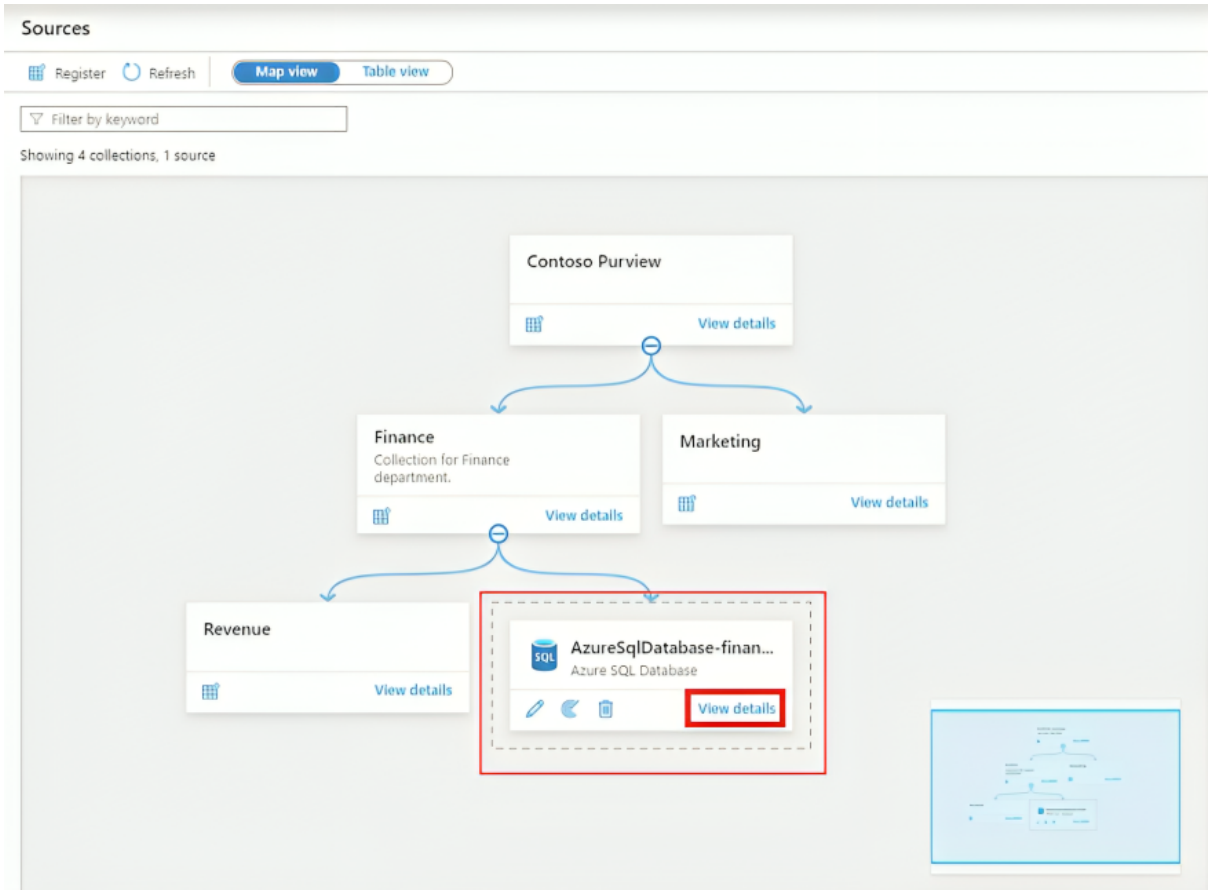


Figura 4.6: Esempio di data map con dati mock [27]

Oltre al *data map*, abbiamo anche il *data estate insights*, che offre dashboard e KPI sullo stato corrente di Purview, come livello di arricchimento degli asset, percentuali di classificazione, numero di utenti attivi ecc.

4.4 OpenMetadata

OpenMetadata (OM) [30] è una soluzione open-source per il data management. Iniziato nel 2021, alla fine del 2022, con la versione 0.13.0, risulta già uno strumento che soddisfa diversi requisiti della Bonfiglioli, in quanto supporta Snowflake, dbt, PowerBI, Tableau. Non supporta, invece, Azure Storage o scripting.

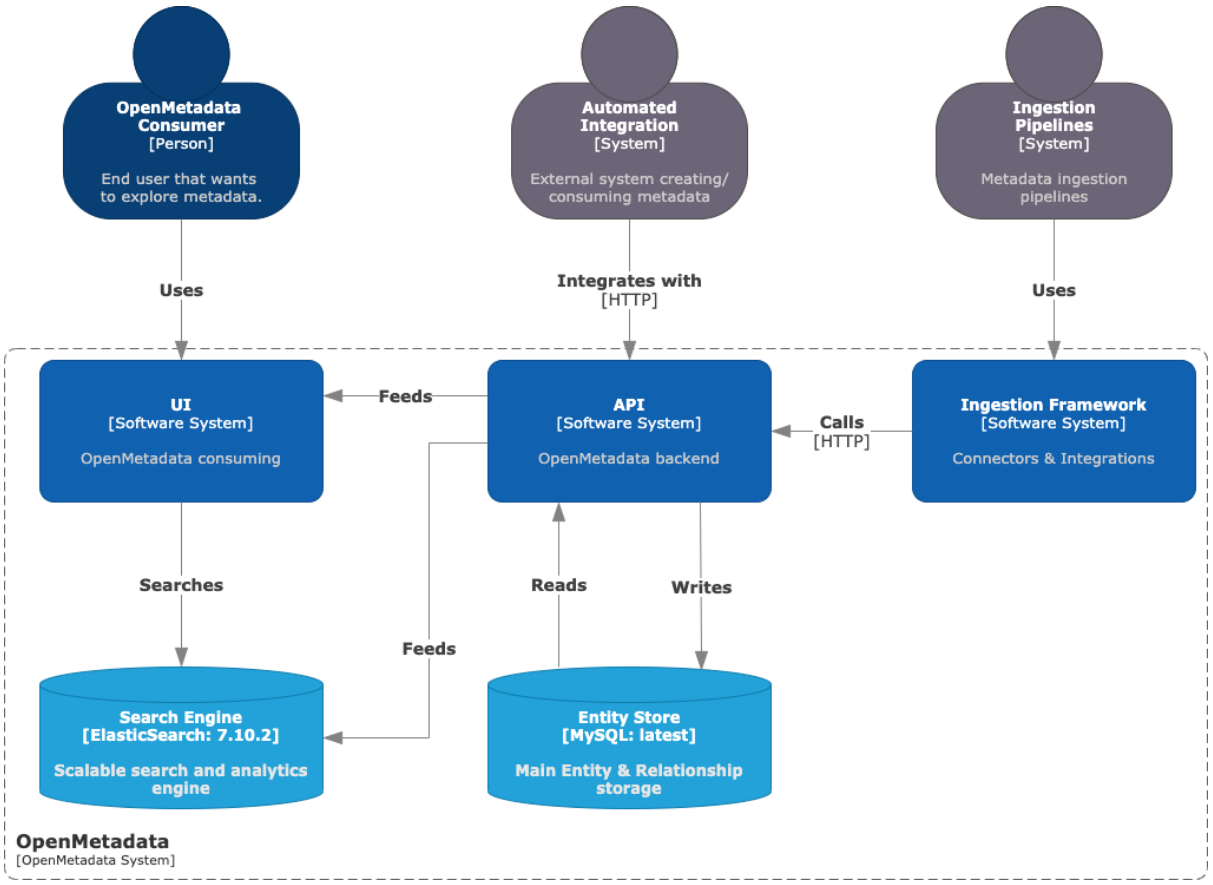


Figura 4.7: Design ad alto livello di OpenMetadata [30]

L'architettura di OM ha i seguenti componenti.

- **API**: gestisce le chiamate interne ed esterne tra i vari sistemi; aggiorna i metadati delle entità e tutte le loro relazioni.
- **Ingestion Framework**: è l'insieme di connettori e integrazioni che definiscono le comunicazioni tra OM e le sorgenti esterne di cui si vuole estrarre i metadata.
- **Entity Store**: è lo storage MySQL dove sono memorizzati gli asset;
  - esistono tabelle specifiche per rappresentare gli asset, i tag, i termini di glossario e le relazioni tra tutti questi elementi;
  - per avere maggiore flessibilità, l'asset in sé è salvato come un documento JSON all'interno di una colonna.
- **Search Engine**: costruito con ElasticSearch, un motore di ricerca open-source, indicizza le entità in modo che possano essere cercabili.

- **UI**: è l'interfaccia grafica con cui gli utenti possono interagire per esplorare gli asset.

Essendo open-source, OM non ha costi di licenza e richiede meno manutenzione rispetto ad Apache Atlas, poiché molti connettori sono già presenti o pianificate per le prossime release.

4.4.1 Funzionalità

Anche OM ha un business glossary e un ricco motore di ricerca che permette di cercare gli asset per sorgente, tag e vari altri filtri. Per popolare il catalog, è necessario registrare le sorgenti, inserendo informazioni come nome e credenziali. Dopodiché, è possibile scegliere il tipo di *ingestion* che si desidera effettuare:

- *metadata ingestion*: importa i metadata dalla sorgente;
- *lineage ingestion*: importa il lineage dalla sorgente;
- *profiler ingestion*: effettua analisi sui dati, mostrando all'utente diverse informazioni come numero di colonne, numero di righe, numero valori nulli, numero valori distinti ecc.

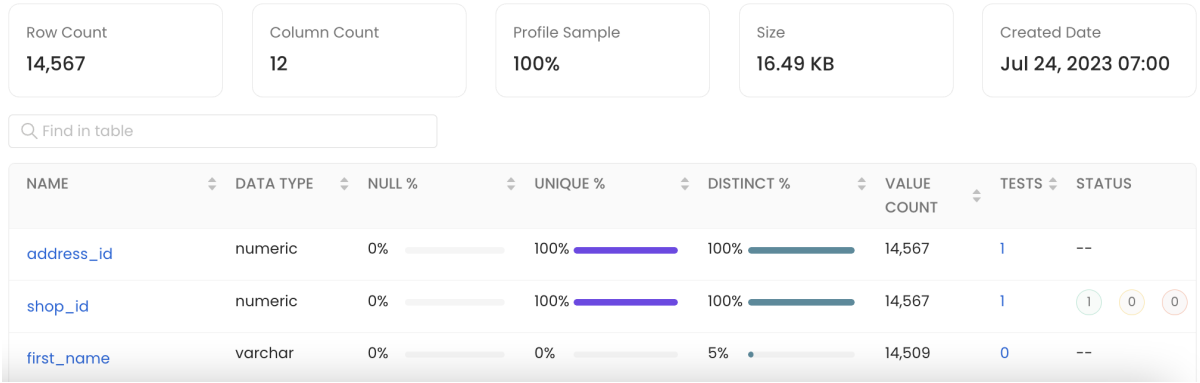


Figura 4.8: Data profiling [30]

Presenta anche diverse feature che rendono OM quasi un social media: per esempio, è possibile "seguire" un asset, ricevere notifiche e inviare annunci.

4.5 Atlan

Atlan [5] è una "piattaforma di metadati per team moderni, che li supporta nella scoperta, comprensione, fiducia e collaborazione sugli asset". Essendo un prodotto commerciale, è

richiesta una licenza a pagamento; l'istanza può essere hostata su diversi cloud, tra cui Azure.

Atlan è composto da molti componenti che interagiscono tra loro per gestire i diversi aspetti della piattaforma, quali l'API, la gestione delle credenziali, l'elaborazione di log e metriche ecc. In particolare, Atlan ha un *metastore* per memorizzare i metadati: basato su Apache Atlas, usa Apache Cassandra come database orientato agli oggetti, ElasticSearch come motore di ricerca e Apache Zookeeper per coordinare i servizi del *metastore*.

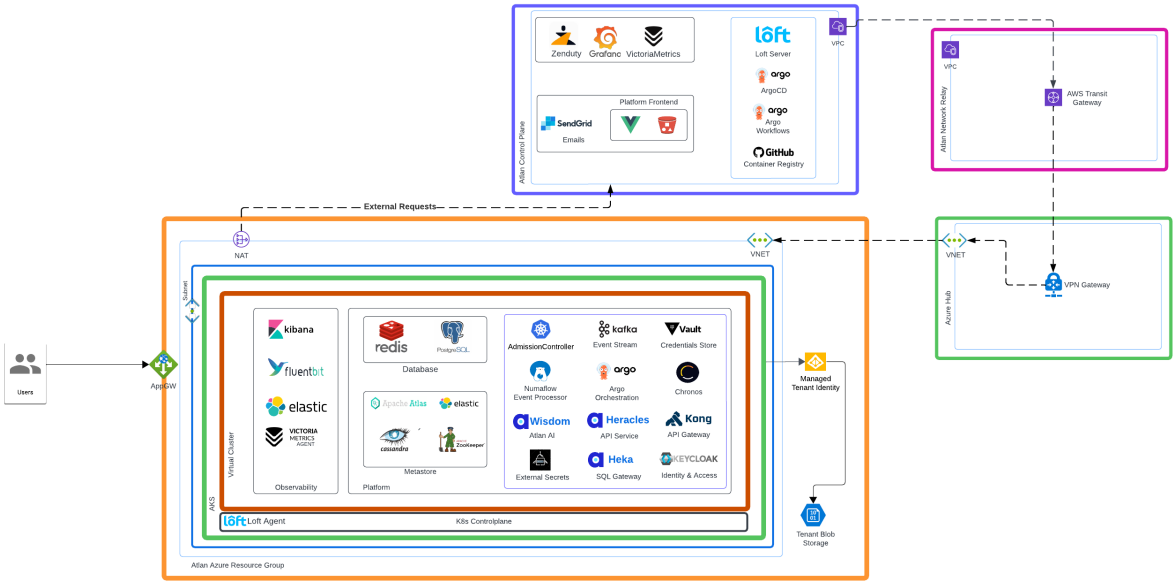


Figura 4.9: Architettura di Atlan su Microsoft Azure [5]

4.5.1 Funzionalità

Similmente a OM, Atlan supporta Snowflake, dbt, PowerBI e Tableau, ma non supporta Azure Storage o scripting.

Rispetto agli altri data catalog, che permettono di aggiungere una sola descrizione agli asset, in Atlan esiste un ulteriore campo *ReadMe* dove è possibile aggiungere maggiori dettagli, allegati, tabelle ecc. È presente anche un motore di suggerimenti che consiglia descrizioni in caso di similarità tra i nomi dei campi e sono previsti diversi altri sviluppi sul fronte dell'AI.

Un'altra funzionalità molto interessante è l'estensione Chrome con cui è possibile accedere ad Atlan da Snowflake, PowerBI, Tableau ecc.

4.6 Spline

Spline [2] è uno strumento open-source per tracciare in modo automatico il data lineage e le strutture delle pipeline di una organizzazzione. Sviluppato inizialmente per supportare solo Apache Spark, in seguito è stato esteso per accomodare anche altre tecnologie.

Spline è composto da tre componenti principali:

- *spline server*: riceve il lineage dagli agenti tramite il *Producer API* e li memorizza in un database; provvede il *Consumer API* per leggere e interrogare il lineage.
- *spline agent*: un programma che cattura il lineage dalle pipeline di trasformazioni e lo invia al server;
- *UI*: un'applicazione web per visualizzare il lineage.

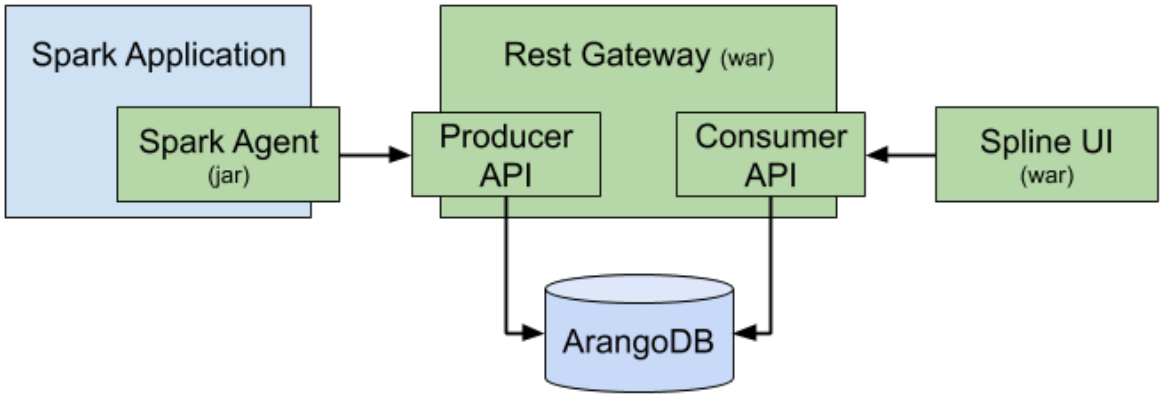


Figura 4.10: Architettura di Spline [2]

Capitolo 5

Test e analisi comparativa

La fase di test si è focalizzata principalmente su Microsoft Purview, Open Metadata e Atlan, mentre gli esperimenti effettuati per Apache Atlas e Spline sono stati interrotti a causa di mancanze evidenti fin dal loro primo utilizzo. Poiché i test sono stati effettuati tra settembre e dicembre del 2022, le osservazioni presentate qui di seguito sono valide per le sole versioni disponibili in quel periodo.

Per testare Purview, OpenMetadata, Atlan, sono stati usati tre schema Snowflake.

- DWH: una selezione di tabelle anagrafiche usate per report PowerBI e Tableau.
- PBI: una collezione di tabelle e viste usate per diversi report PowerBI. In particolare, sono state esaminate le tabelle che alimentano i seguenti report:
  - *Quality MES*: report sulla qualità di alcuni stabilimenti produttivi, frutto di un'elaborazione dati provenienti da MES;
  - *Pricing*: report sul monitoraggio dei prezzi, utile al reparto vendite;
  - *Group Costing*: report sull'area *sales* con le principali misure d'analisi (fatturato, numero spedizioni e ordini, backlog, budget).
- PUBLIC: una collezione di tabelle e viste usate per i report Tableau.

Per ciascun data catalog è stata applicata la stessa metodologia:

1. caricamento delle sorgenti;
2. analisi del data lineage;
3. esplorazione delle funzionalità di arricchimento degli asset e il loro livello di dettaglio;

4. aggiunta di termini al *business glossary*.

5.1 Microsoft Purview

5.1.1 Attività preliminari

Infrastruttura

L'istanza dell'ambiente di test è stata preparata dai consulenti Microsoft contattati dalla Bonfiglioli e non sono stati quindi richiesti ulteriori passi. Essendo un prodotto Microsoft, si accede tranquillamente con il tenant della Bonfiglioli.

Creazione di una collection

Il primo passo per popolare Purview è creare una *collection*, un "contenitore" in cui verranno inseriti le varie sorgenti e asset che verranno caricati.

Una *collection* può essere utile per separare tra loro domini aziendali: queste *collection* sono infatti riunite nel *data map* che l'utente può navigare, offrendogli quindi una visione ad alto livello di quali sono i dati che vengono utilizzati in un reparto rispetto a un altro.

Essendo di poco interesse, la funzionalità di *data map* non è stata esplorata ed è stata semplicemente creata una *collection* di default contenente tutte le sorgenti di test.

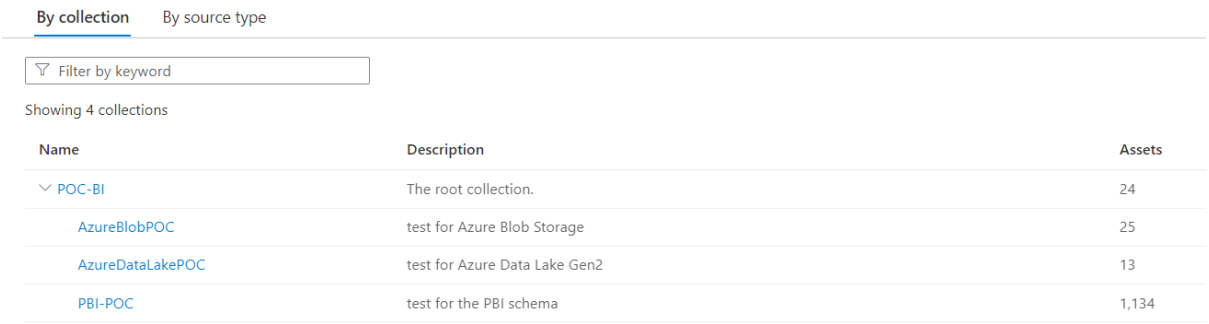


Figura 5.1: Oltre alla vista a mappa, è possibile visualizzare gli asset divisi per collezione.

Registrazione di una sorgente

Creata una *collection*, è necessario registrare la sorgente. La registrazione implica:

- assegnare un nome alla sorgente;
- specificare il server o l'endpoint della sorgente;

- selezionare la collezione a cui destinare gli asset della sorgente.

Questa procedura è valida per Azure Storage e Snowflake. Per PowerBI, invece, bisogna abilitare dal tenant Azure dell'azienda alcune impostazioni che danno a Purview il permesso di usare le *admin API* di PowerBI.

Scan

Lo *scan* è il processo dove Purview stabilisce una connessione con una sorgente registrata ed estrae i metadati necessari per popolare il data catalog.

Per creare uno *scan*, è necessario provvedere le seguenti informazioni base:

- nome dello *scan*;
- *integration runtime* con cui Purview si connette alle sorgenti;
  - *integration runtime* è l'infrastruttura di calcolo usata da Azure per offrire le funzionalità di integrazione dati in diversi ambienti di rete [25];
- credenziali con cui connettersi alla sorgente.

A seconda della sorgente che si vuole scannerizzare, ci possono essere campi aggiuntivi da compilare: per esempio, per PowerBI si può scegliere di includere o meno i workspace, oppure si può decidere di limitare o meno il numero di asset da importare da un database o data warehouse specificando il solo schema o database su cui eseguire lo *scan*.

Lo *scan* può anche essere schedulato giornalmente, settimanalmente o mensilmente all'orario desiderato.

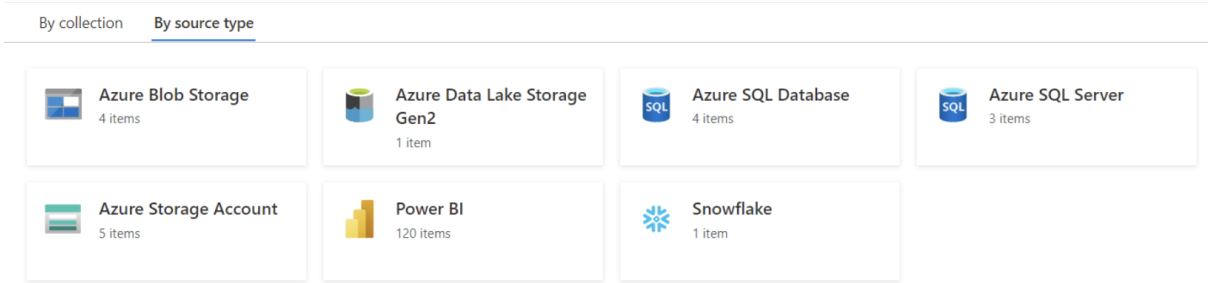


Figura 5.2: È possibile visualizzare gli asset divisi per sorgente.

5.1.2 Analisi

Le sorgenti importante su Purview sono state Azure Blob Storage, Azure Data Lake Storage Gen2, Snowflake e PowerBI.

- Purview è in grado di scannerizzare ed estrarre i metadati da entrambe le sorgenti di Azure Storage. I dati presenti nel Blob Storage e nel ADLS Gen 2 provengono dai macchinari di produzione dell'azienda e vengono successivamente rielaborati con script in Python e Spark, che però non sono attualmente supportate da Purview. Purview quindi non ha costruito alcun data lineage tra le varie entità di Azure Storage.

  Essendo sorgenti di secondaria importanza e non avendo un lineage, una volta confermato che gli asset sono stati correttamente importati nel data catalog, non sono state effettuate ulteriori analisi.

- È possibile aggiungere uno o più *glossary*.
  - Per ogni termine, si può aggiungere definizione, sinonimi, termini correlati e termini figli.
  - Quando si associano termini agli asset, questi sono visibili nella pagina del termine.
  - È possibile associare eventuali persone come *steward* o *experts* nel tab *Contacts*.

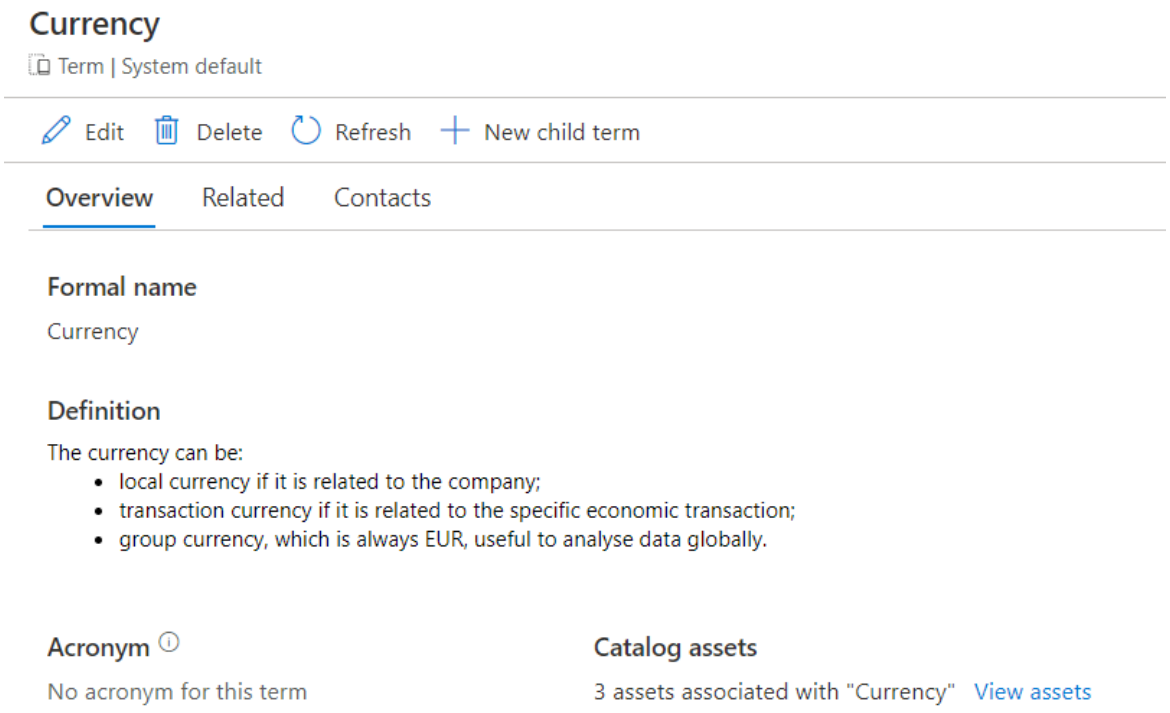


Figura 5.3: Overview di un termine di glossario.

- Purview è in grado di scannerizzare ed estrarre i metadati delle viste e delle tabelle Snowflake e degli asset PowerBI in tutti i suoi livelli di dettaglio (workspace, dashboard, report e dataset).
- Attualmente Purview non è in grado di costruire il lineage tra le tabelle Snowflake e i dataset PowerBI.
- Un asset presenta le seguenti tab.
  - *Overview*: è possibile aggiungere la descrizione dell'asset, *glossary terms* o classificazioni. Al momento, Purview non supporta la classificazione né per Snowflake né per PowerBI.

    Nell'*overview* sono elencati anche il *qualified name*, la *collection* a cui appartiene l'asset e la sua eventuale gerarchia. Per una tabella Snowflake, la gerarchia potrebbe essere: server, database, schema, tabella.
  - *Properties*: proprietà dell'asset automaticamente compilate da Purview, come timestamp dell'ultima modifica, *qualified name*, tipo dell'asset ecc.

– *Schema*: elenco delle eventuali colonne e il loro tipo che compongono l'asset, a cui si possono aggiungere *glossary terms*, descrizioni, classificazioni o etichette sulla sua sensibilità.

  La descrizione associata alle colonne è tagliata, ma è possibile leggerla nella sua interezza portando il cursore sopra il testo; in alternativa, cliccando sul nome della colonna si apre una tab laterale con ulteriori dettagli sulla colonna.

– *Lineage*: vista navigabile con il lineage dell'asset, fino al livello di colonna, con eventuali dettagli sulle query e operazioni effettuate.

– *Contacts*: è possibile aggiungere persone esperte o *owner* dell'asset.

– *Related*: eventuali asset correlati.

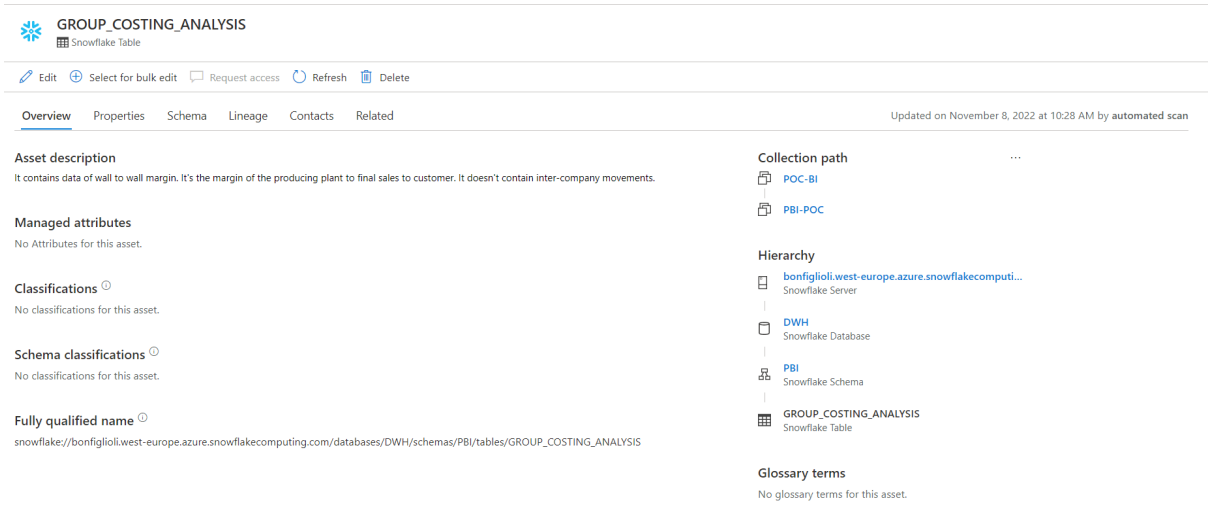


Figura 5.4: Overview della tabella `GROUP_COSTING_ANALYSIS`. Al momento, la classificazione e la *sensitivity label* non sono supportate per Snowflake.

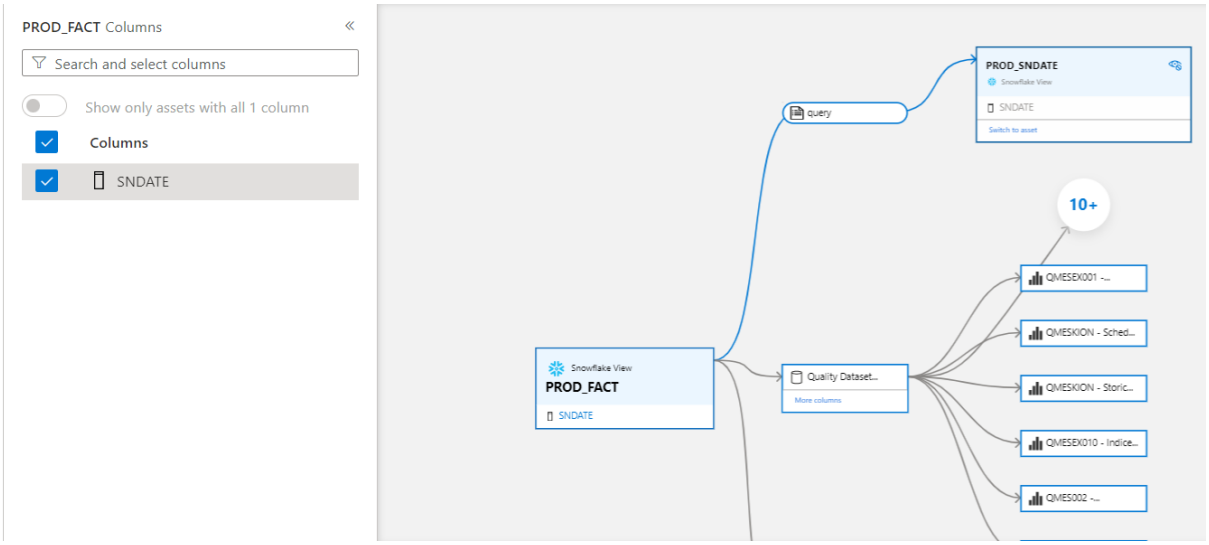


Figura 5.5: Il lineage della colonna `SNDATE`.

- È presente una barra di ricerca con cui è possibile cercare gli asset. I risultati possono essere ulteriormente raffinati con filtri per tipo di dato, collezione, termini di glossario, contatti ecc.

  Purview permette anche di usare operatori come *OR*, *AND*, *NOT*, raggruppare parole chiave con parentesi o cercare dentro un attributo specifico dell'asset. Per esempio, `group AND (term:currency OR term:Customer)` ritornerà:

  – gli asset che contengono "group" e hanno *glossary terms* che includono *currency* o *customer*;

  – i *glossary terms* che corrispondono a *currency* o *customer*.

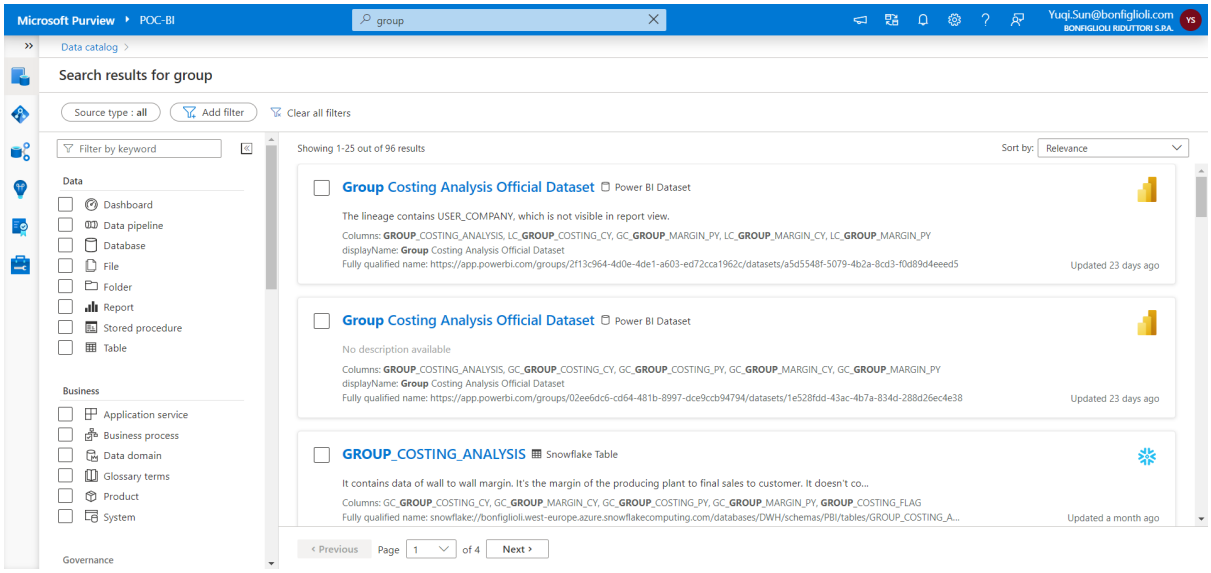


Figura 5.6: I risultati ritornati da Purview quando si cerca “group”.

Snowflake-PowerBI

Purview non è in grado di costruire automaticamente il lineage tra le tabelle Snowflake e i dataset PowerBI. È però possibile usufruire dell’API di Purview per aggiungere manualmente il lineage.

È stato quindi sviluppato uno script Python che: recupera le tabelle PowerBI che compongono il dataset PowerBI; cerca le tabelle e le viste Snowflake il cui nome corrisponde alle tabelle PowerBI e che sono attualmente nel data catalog; costruisce il lineage tra gli asset Snowflake e il dataset PowerBI.

- Per aggiungere il lineage, si crea una nuova entità *AtlasProcess*, una classe Python che rappresenta un processo di Apache Atlas.
  - Questa entità ha come nome e *qualified name* gli stessi del dataset PowerBI da aggiornare; il suo tipo è *powerbi_dataset_process*.
  - Nell’attributo *input* vengono inseriti gli asset di Snowflake individuati precedentemente.
- Le tabelle e le viste Snowflake sono modellate come classi *AtlasEntity*, corrispondenti a un’entità Atlas.

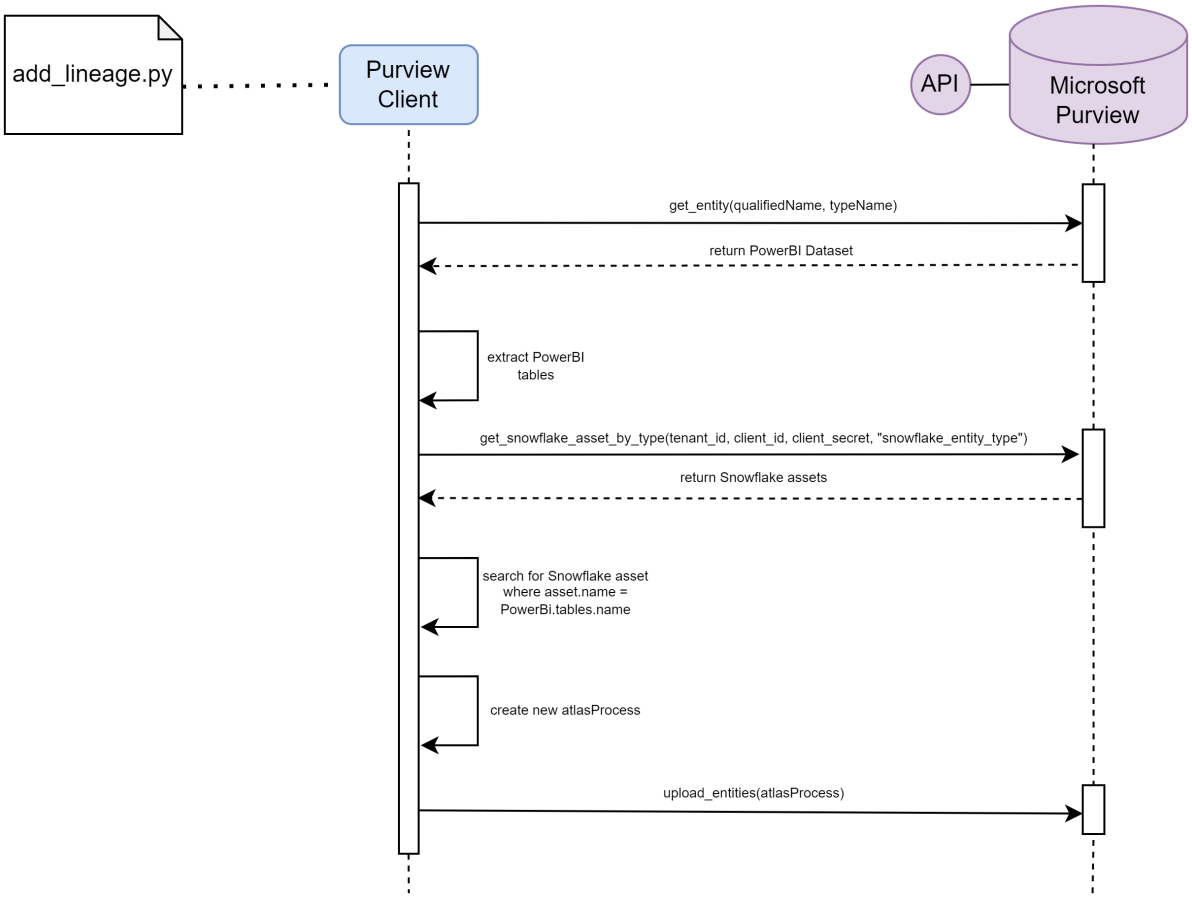


Figura 5.7: Diagramma di sequenza per inserire il lineage tra Snowflake e PowerBI.

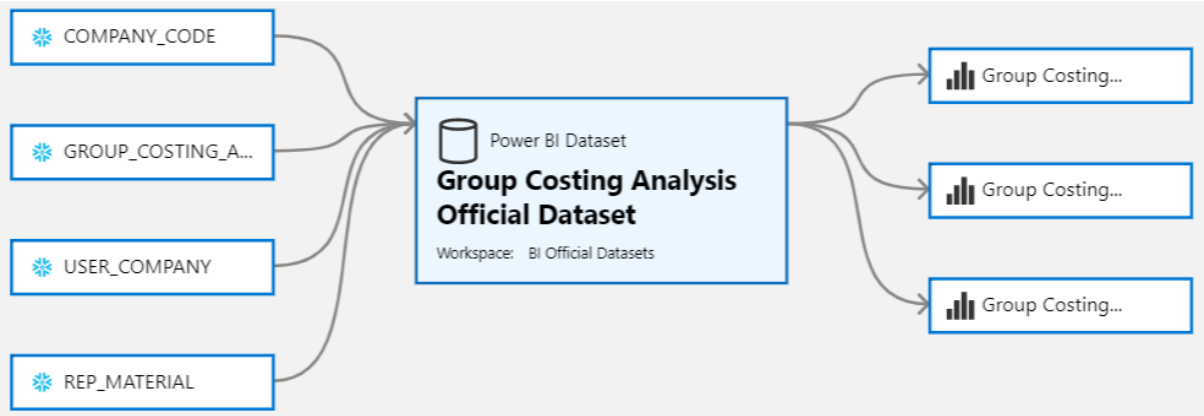


Figura 5.8: Lineage, creato con lo script Python, tra le tabelle di Snowflake che compongono il dataset `Group Costing Analysis Official Dataset` e i report alimentati da esso.

5.2 OpenMetadata

5.2.1 Attività preliminari

Infrastruttura

Per testare OM, è stata installata un'istanza locale con Docker seguendo il tutorial provvisto dalla documentazione. In un secondo momento, OM è stato installato su un cluster Kubernetes della Bonfiglioli, dove è stato abilitato il SSO tramite Microsoft.

La prima versione usata è stata la `0.12.2`, successivamente aggiornata alla `0.13.0-Release`.

Creazione di un service

In OM le varie istanze di sorgenti che possono essere registrate sono dette *service*. Per aggiungere un nuovo *service* bisogna:

- selezionare la sorgente che si desidera registrare (Snowflake, PowerBI ecc);
- assegnare un nome al *service*;
- specificare l'endpoint e le credenziali con cui connettersi alla sorgente;
- altri campi specifici alla sorgente scelta;
  - per esempio, se la sorgente è Snowflake, è necessario indicare il warehouse e opzionalmente anche il database, in caso si desideri limitare l'estrazione dei metadati a determinati asset.

Ingestion

Creata una *service*, è possibile aggiungere un'*ingestion*. L'ingestion è il processo vero e proprio dove OM si connette alla sorgente per popolare il data catalog. Si possono eseguire diversi tipi di *ingestion*.

- *Metadata ingestion*: comune a tutte le sorgenti, estrae i metadati. Per database e data warehouse, è possibile includere i file dbt `manifest.json` e `catalog.json` come ulteriore fonte da cui estrarre metadati e lineage.

  Il *metadata ingestion* deve essere effettuato prima degli altri.

- *Usage ingestion*: applicabile per database e data warehouse, aggiunge statistiche su quanto è stata utilizzata una tabella; queste statistiche sono calcolate a partire dalla *query log* e dai dettagli di creazione delle tabelle.
- *Lineage ingestion*: applicabile per database e data warehouse, costruisce il lineage tra gli asset presenti in OM, anche tra sorgenti differenti, sulla base delle informazioni estratte durante il *metadata ingestion*.
- *Profiler ingestion*: applicabile per database e data warehouse, produce informazioni quali numero di righe e colonne, valori distinti o nulli ecc.

Ogni tipo di *ingestion* può essere schedulato a intervalli orari, giornalieri, settimanali o mensili.

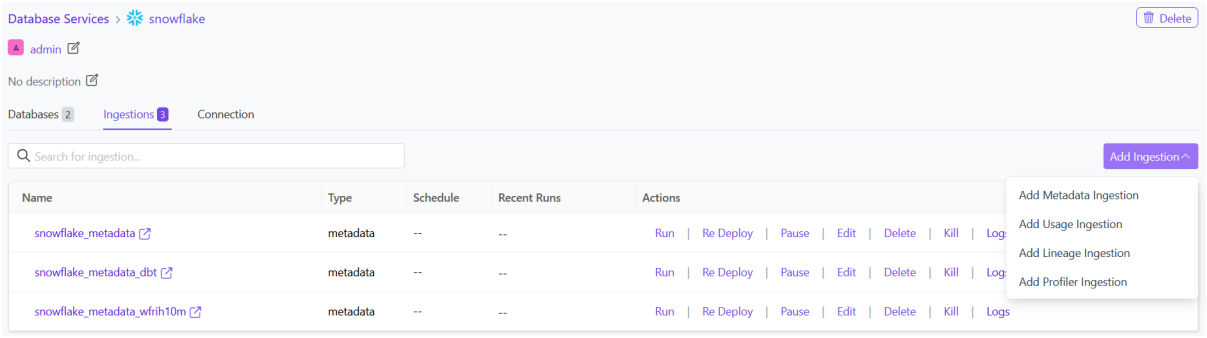
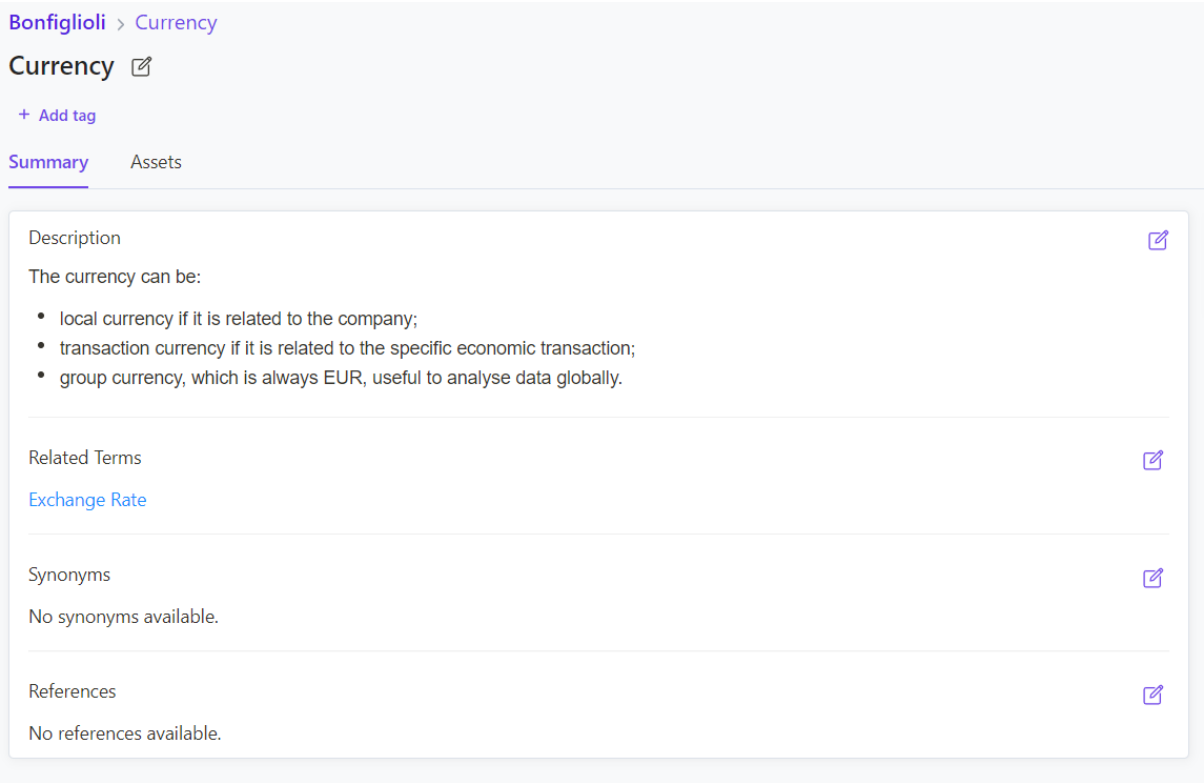


Figura 5.9: Tab contenenti le ingestion eseguite su Snowflake.

5.2.2 Analisi

Le sorgenti importate su OM sono state Snowflake, PowerBI e Tableau.

- OM è in grado di scannerizzare ed estrarre i metadati di Snowflake, PowerBI e Tableau. Tuttavia il lineage che riesce a individuare tra di essi è limitato.
- È possibile creare uno o più *glossary*.
  - Per ogni termine si può aggiungere la definizione, sinonimi, termini correlati e etichette di classificazione.
  - È possibile indicare dei *reviewer*, utenti che hanno il compito di revisionare e accettare eventuali cambiamenti subiti da un termini.
  - Gli asset associati a un termine sono visibili in una tab separata.



- Un asset presenta le seguenti informazioni.
  - La gerarchia dell'asset è rappresentata come una *breadcrumb*, seguita da una lista con diverse informazioni, come *owner* dell'asset, numero di colonne e righe.
  - Se l'asset è una tabella o una vista, il tab *Schema* elenca le colonne e il loro tipo, a cui è possibile aggiungere descrizioni, *glossary terms* e classificazioni.
  - Se l'asset è una dashboard, il tab *Details* elenca i report che lo compongono.
  - Nel tab *Lineage* è possibile navigare il lineage dell'asset. Anche OM riesce a individuare il lineage colonnare e dove possibile estraendo le trasformazioni subite.
  - Se impostato durante il *metadata ingestion*, nel tab *Sample Data* si può visualizzare un campione di dati estratto dalla sorgente.
  - Le statistiche sulla sulla *data quality* dell'asset calcolate durante il *profiler ingestion* sono visibili nel tab *Profiler & Data Quality*.
  - Grazie al *usage ingestion*, OM mostra agli utenti dati come percentili d'uso e numero di query eseguite che comprendono l'asset.

– Per andare ulteriormente incontro alle necessità degli utenti, OM permette di creare delle *custom properties*.

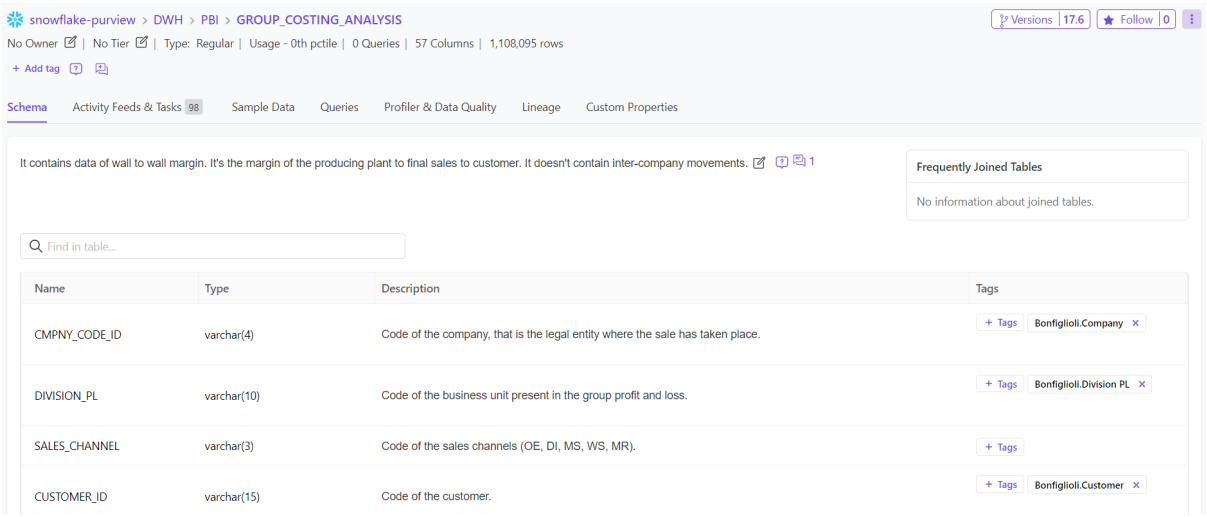


Figura 5.10: Overview della tabella `GROUP_COSTING_ANALYSIS`

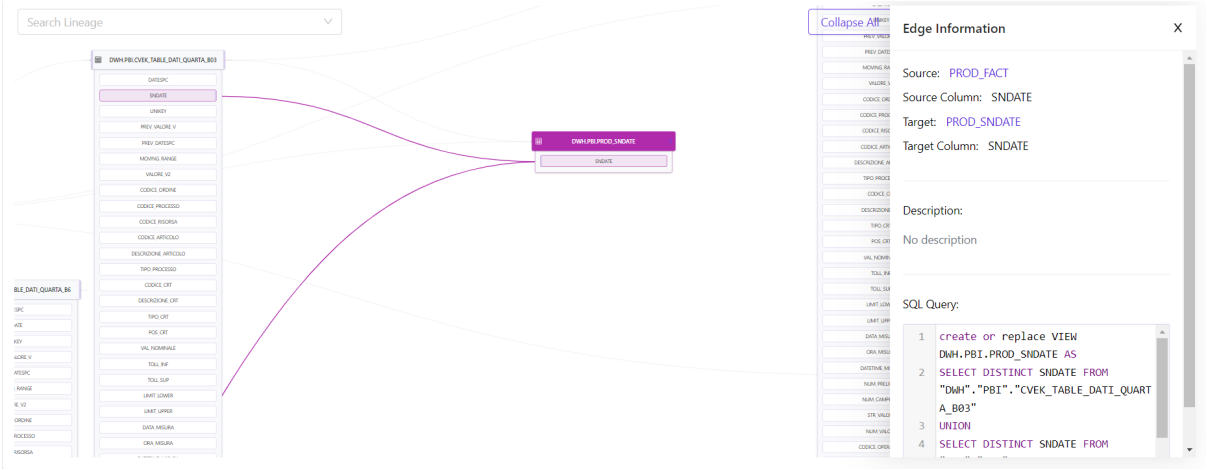


Figura 5.11: Lineage della colonna '`SNDATE` e relativa query utilizzata.

- OM presenta diverse funzionalità di collaborazione.
  - Un utente può "seguire" un asset o aggiungere commenti alle descrizioni.
  - Nel tab *Activity Feeds & Tasks*, può vedere quali modifiche ha subito un'asset, da chi sono state effettuate e quali task l'asset necessita ancora.

- È presente una barra di ricerca con cui è possibile cercare gli asset. I risultati sono già divisi in tab a seconda della loro tipologia (tabella, topic, dashboard ecc) e possono essere ulteriormente raffinati con filtri (per tipo di dato, *glossary terms*, database ecc).

  OM ha anche una ricerca avanzata. Tramite una dialog, guida gli utenti a selezionare il campo (colonna, proprietà, *owner*), l'operatore (uguaglianza, minore, maggiore) e il valore. Queste condizioni possono essere messe in AND o OR tra di loro.

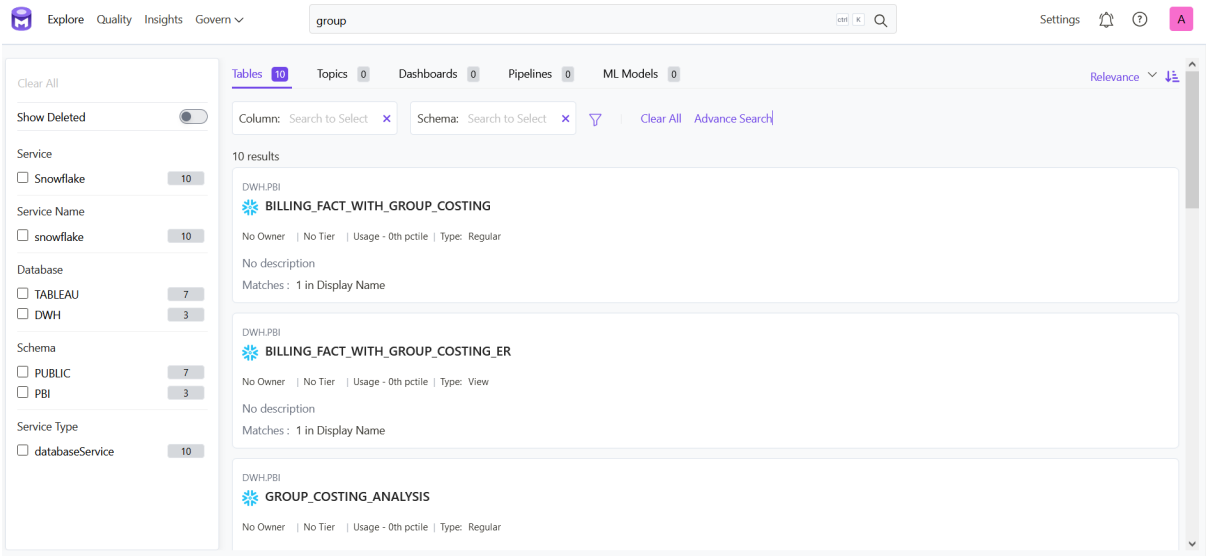


Figura 5.12: I risultati ritornati da OpenMetadata quando si cerca "group".

- Gli esperimenti hanno evidenziato alcune problematiche minori:
  - nel *business glossary* non è possibile creare un termine figlio [13];
  - OM ha avuto problemi a parsare tabelle il cui nome presenta la sequenza di caratteri \"[14].

PowerBI

OM supporta sia PowerBI sia Tableau, ma l'analisi si è concentrata principalmente su PowerBI.

Durante i test, è stato evidenziato che mancavano diversi elementi che Purview aveva invece individuato: solo le dashboard vengono importate come asset distinti; i dataset PowerBI non sono trovabili nell'interfaccia di OM, mentre i report vengono mostrati come

componente delle dashboard, nella loro pagina *Details*. Di conseguenza anche il lineage collega le tabelle Snowflake alle sole dashboard PowerBI.

Dopo test più approfonditi, sono state raggiunte le seguenti conclusioni.

- Avere le *dashboard* come il solo componente con cui popolare il data catalog è una precisa scelta implementativa di OM.
- Il lineage viene costruito esclusivamente se dataset, report e dashboard sono tutti e tre presenti nel stesso workspace:
  - ciò è dovuto dal fatto che OM cerca i dataset che compongono un report `X` solo all'interno del workspace che contiene `X`;
  - analogamente, OM cerca i report che compongono una dashboard `Y` solo dentro il workspace che contiene `Y` [29].

Questa è una forte limitazione, in quanto su PowerBI la Bonfiglioli tende a utilizzare un approccio diverso: i dataset sono raggruppati in un workspace, mentre i report sono costruiti in un workspace separato, spesso senza usare dashboard.

Se la scelta di modellare solo le dashboard come asset distinti e cercabili può essere semplicemente considerata come una visione non compatibile con i processi attuali della Bonfiglioli, l'obbligo di collocare dataset, report e dashboard nello stesso workspace per ottenere il lineage ci è sembrato un limite vero e proprio del prodotto: per portare attenzione su questa tema, quindi, è stata aperta una GitHub issue nella repository di OM [12].

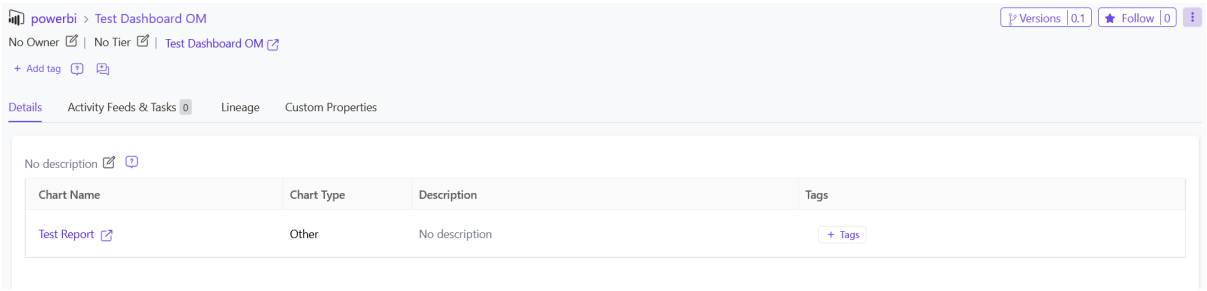


Figura 5.13: I report sono visibili nella pagina *Details* della dashboard.

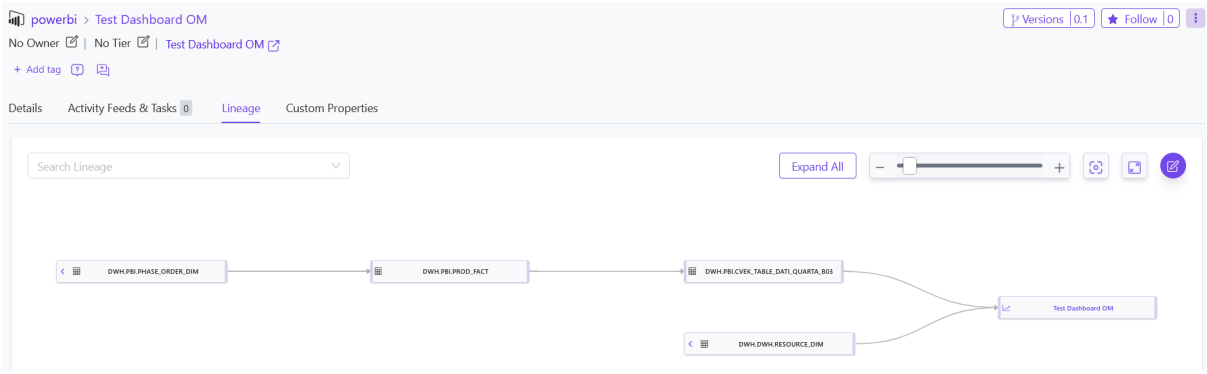


Figura 5.14: Se si creano report, dashboard e dataset nello stesso workspace, OpenMetadata è in grado di individuare il lineage automaticamente.

Un altro problema riscontrato su OM è che non è in grado di importare tutti gli workspace effettivamente presenti su PowerBI. Questo problema è probabilmente causato dal fatto che l'API che ritorna i workspace, `GetGroupsAsAdmin`, richiede un parametro `top` che attualmente è impostato a 100 nella versione `0.13.0`, tale per cui vengono estratti solo i primi cento workspace. Anche in questo caso è stata aperta una issue [15].

Lineage

A volte il lineage identificato da OM non è corretto. Questo errore è dovuto dal fatto che, per identificare la tabella Snowflake usata per alimentare un dataset PowerBI, si confronta semplicemente il nome della tabella con gli asset già presenti in OM. Esempio: per il dataset `B6 Quality Dataset Official`, la dashboard è stata collegata alla tabella `DWH.DWH.RESOURCE_DIM`, ma la tabella corretta è `DWH.PBI_TEST.RESOURCE_DIM`.

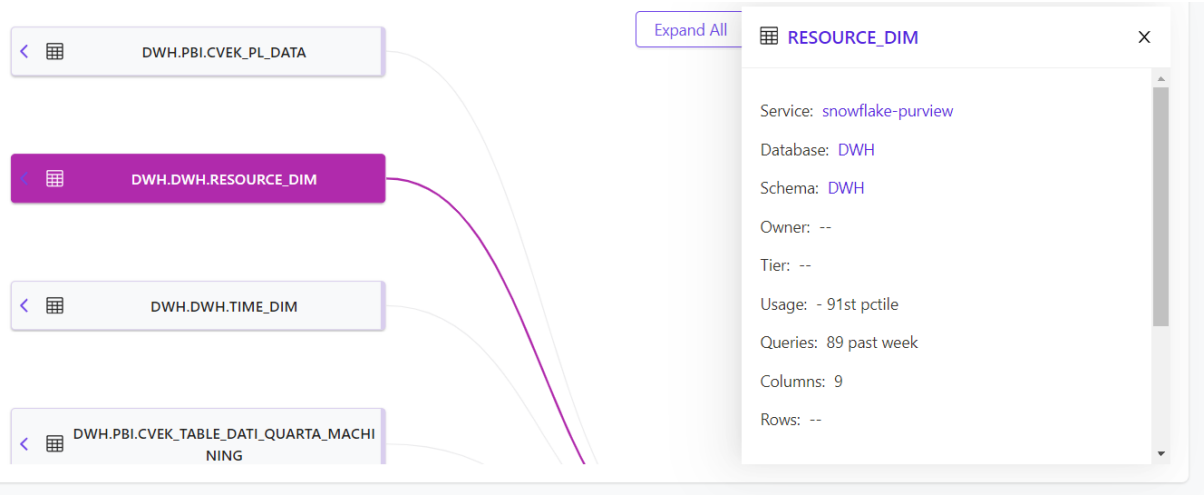


Figura 5.15: La tabella che OM ha usato per costruire il lineage.

Figura 5.16: Su PowerBI la tabella usata per alimentare il dataset `B6 Quality Dataset Official` è `DWH.PBI_TEST.RESOURCE_DIM`

5.3 Atlan

5.3.1 Attività preliminari

Infrastruttura

L'istanza dell'ambiente di test è stata preparata dai consulenti Atlan e non sono stati quindi richiesti ulteriori passi. Anche qui è possibile autenticarsi con SSO tramite Microsoft.

Creazione di un workflow

Il processo attraverso il quale viene popolato il data catalog si chiama *workflow*. Per estrarre gli metadati, è necessario impostare un *workflow* di tipo *connector*, inserendo informazioni quali:

- nome della connessione;
- endpoint della sorgente;
- credenziali d'accesso;
- eventuali campi specifici alla sorgente in oggetto.

Solo dopo che è stato eseguito il *connector* e importato gli asset, è possibile creare un *workflow* di tipo *miner*, con il quale Atlan analizza la *query history* della sorgente per generare il lineage e l'*usage*.

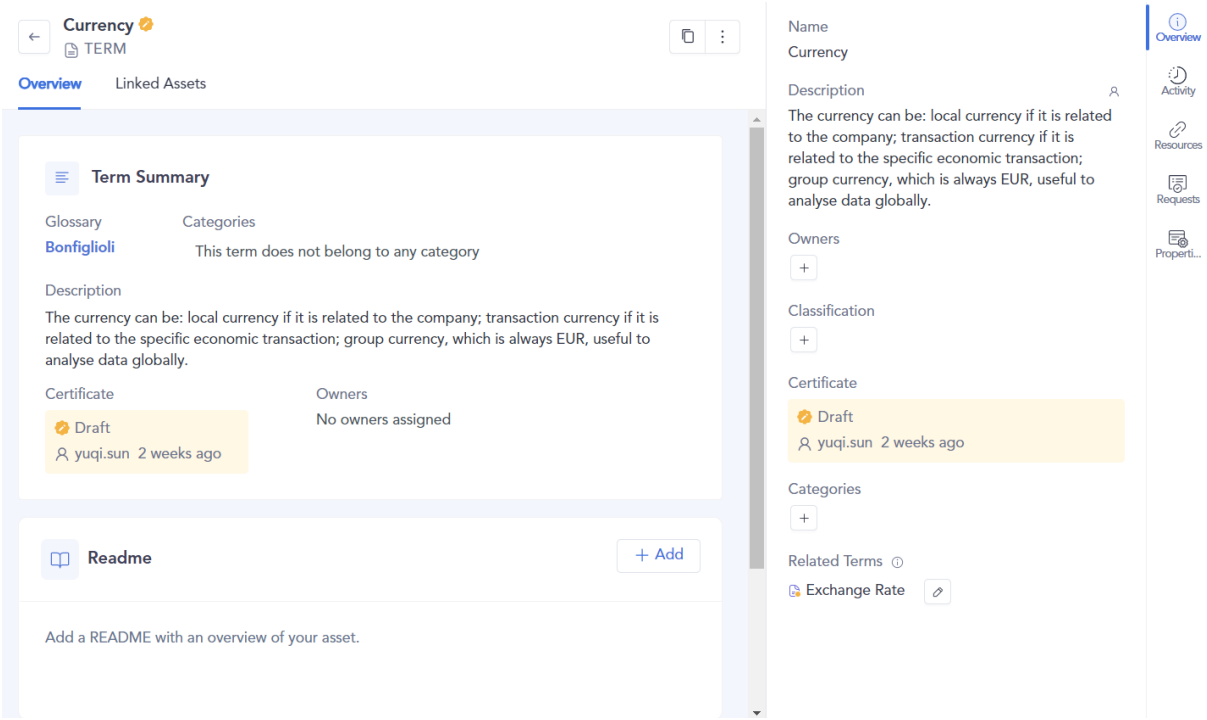
Sia il *connector* sia il *miner* possono essere schedulati a intervalli regolari.

5.3.2 Analisi

Le sorgenti importate su Atlan sono state Snowflake e PowerBI. Poiché la versione di Tableau in uso dalla Bonfiglioli non era compatibile con Atlan, non è stato possibile effettuare test con i report Tableau.

- Atlan è in grado di scannerizzare ed estrarre i metadata di Snowflake e PowerBI, costruendo correttamente il lineage tra di essi.
- È possibile creare uno o più *glossary*. Come per Purview e OM, è possibile aggiungere definizioni, termini correlati ed etichette di classificazione.

- È possibile aggiungere ulteriori livelli di gerarchia, dividendo i *glossary terms* in categorie e sottocategorie.
- È possibile associare uno o più *owner* al termine.



- Un asset di Atlan può avere le seguenti tab.
  - *Overview*: è possibile aggiungere descrizioni, *glossary terms*, classificazioni e *owners*.
  - *Columns*: elenco delle colonne e il loro tipo. Anche qui è possibile aggiungere descrizioni, classificazioni, *glossary terms* ed etichette sulla sua sensibilità.
  - *Lineage*: vista navigabile del lineage tabellare e colonnare.
  - *Related Assets*: eventuali asset correlati.

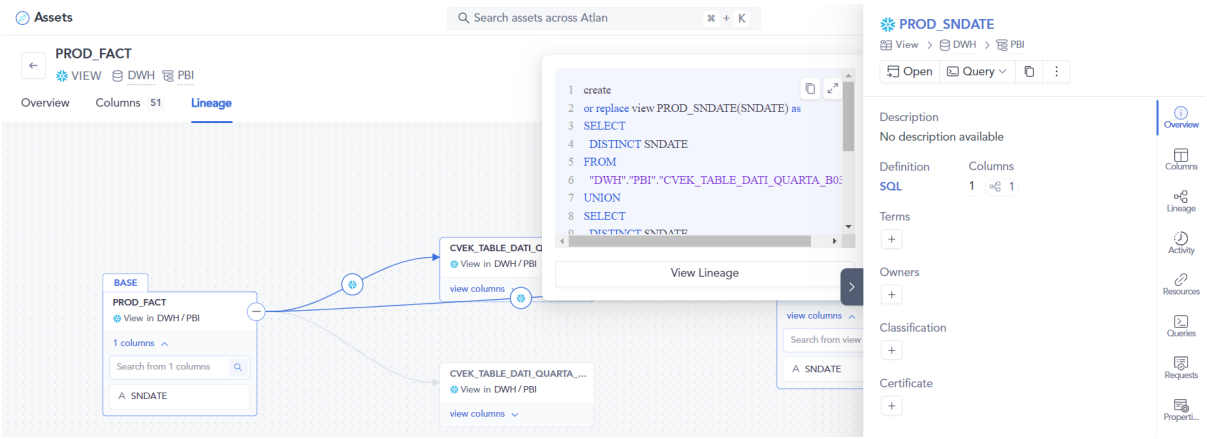
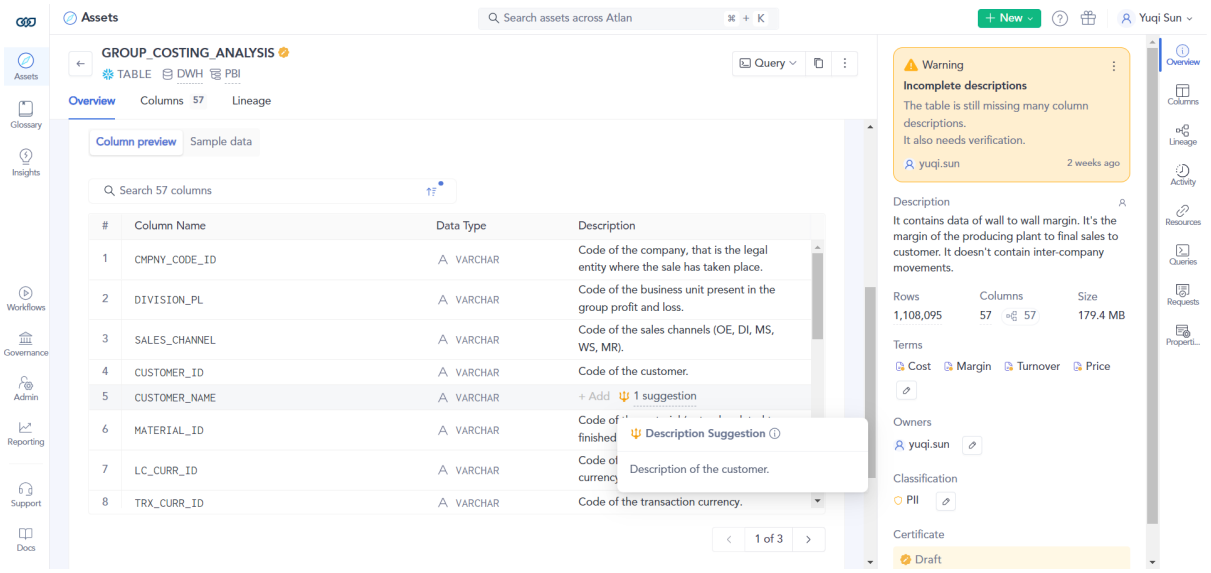
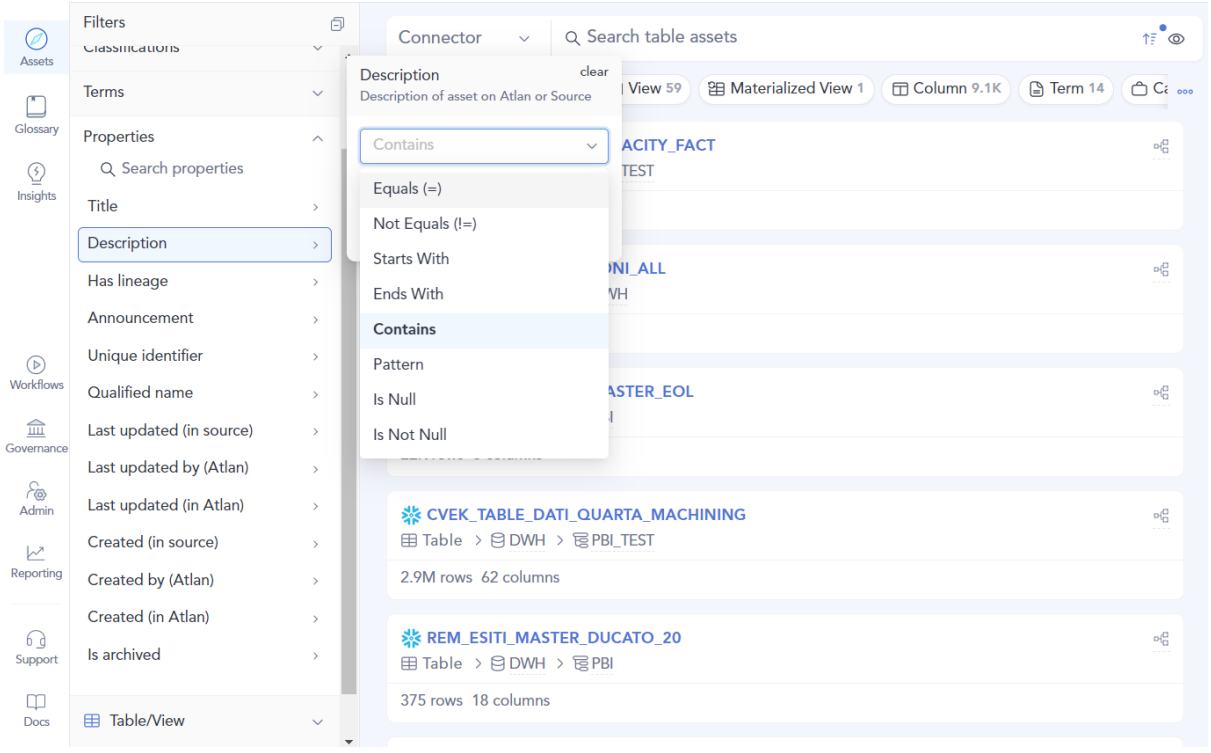


Figura 5.17: Lineage della colonna `SNDATE` e relativa query.

- Atlan si presenta uno strumento molto ricco dal punto di vista dell'arricchimento del data catalog.
  - Oltre alle descrizioni, esiste un ulteriore campo *ReadMe* dove è possibile dettagliare ulteriormente l'asset, aggiungendo tabelle, allegati, link ecc.
  - Esiste un motore di suggerimenti che consiglia la descrizione da aggiungere all'asset sulla base del suo nome.
  - Esiste il concetto di *certificazione* per identificare lo stato corrente di lavoro dell'asset. Può assumere i seguenti valori: "verificato", "bozza", "deprecato".
  - È possibile aggiungere nuovi campi personalizzabili dall'utente.
  - Se abilitato durante la configurazione dei *workflow*, è possibile interrogare direttamente la sorgente. È anche presente un campione di dati.



- È presente una barra di ricerca con cui è possibile cercare gli asset per parole chiave. I risultati possono essere ulteriormente filtrati per sorgente, *glossary terms*, certificato ecc; in alternativa si possono applicare ulteriori operatori che lavorano sulle proprietà stesse degli asset.



- È disponibile un'estensione Chrome con cui è possibile accedere ad Atlan direttamente dalle sorgenti supportate.

- Durante il suo uso non sono state evidenziate problematiche particolari.

5.4 Confronti ed esperienze

5.4.1 Granularità degli asset

Purview, OM e Atlan modellano i propri asset con livelli diversi di granularità.

Per esempio, quando si effettua una ricerca, i tre data catalog cercano corrispondenze in qualunque possibile campo e attributo di un asset. Tuttavia:

- OM ritorna solo tabelle e dashboard;
- i risultati di Purview includono anche dataset e report;
- quelli di Atlan comprendono anche viste, colonne, misure, pagine dei report.

Su Atlan quindi ogni singolo elemento è un asset a sé stante, cercabile dalla barra di ricerca e interagibile dall'utente, mentre OM ha la granularità più alta; Purview, invece, si pone a metà tra i due. Questa differenza è particolarmente evidente quando si analizza il lineage proposto dai tre data catalog.

Prendiamo il dataset `Group Costing Analysis Official` come tabella da usare per confrontare il lineage tra Snowflake e PowerBI prodotto dai tre data catalog. Dei tre strumenti, Atlan estrae il lineage più completo ed esaustivo, modellando ogni singolo componente: tabella Snowflake; tabella, dataset, report e pagina PowerBI. Purview offre invece un lineage più ad alto livello, con meno dettagli ma ugualmente corretto; rimane lo svantaggio che attualmente Purview non crea questo lineage in modo automatico ma è necessario avvalersi di uno script.

OM risulta lo strumento meno compatibile con le attuali richieste della Bonfiglioli, che lavora principalmente con report, mentre OM rappresenta solo le dashboard come asset distinti: per visualizzare il lineage della figura 5.20 è stata appositamente creata una dashboard nello stesso workspace dove sono presenti i report e il dataset "Group Costing Analysis Official".

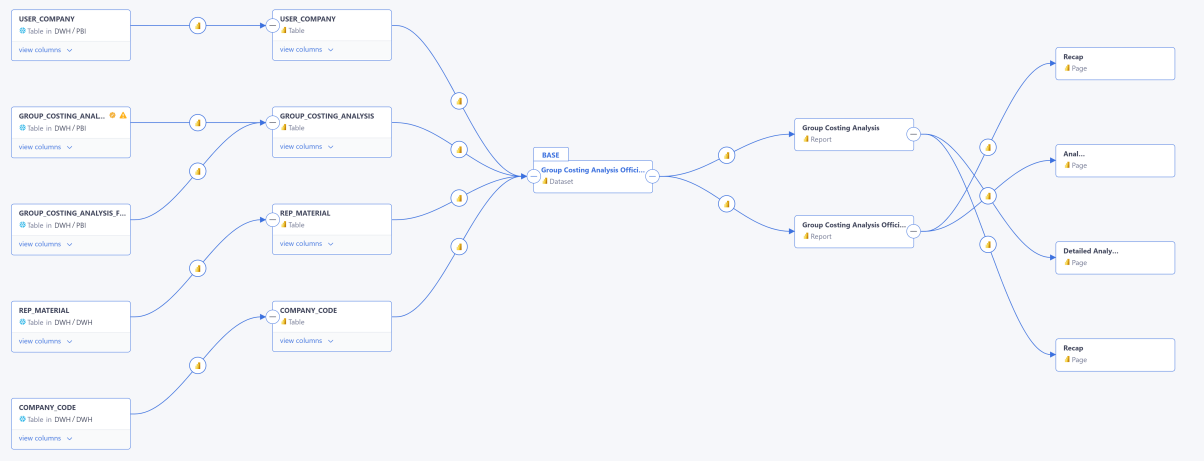


Figura 5.18: Atlan: lineage del dataset `Group Costing Analysis Official`.

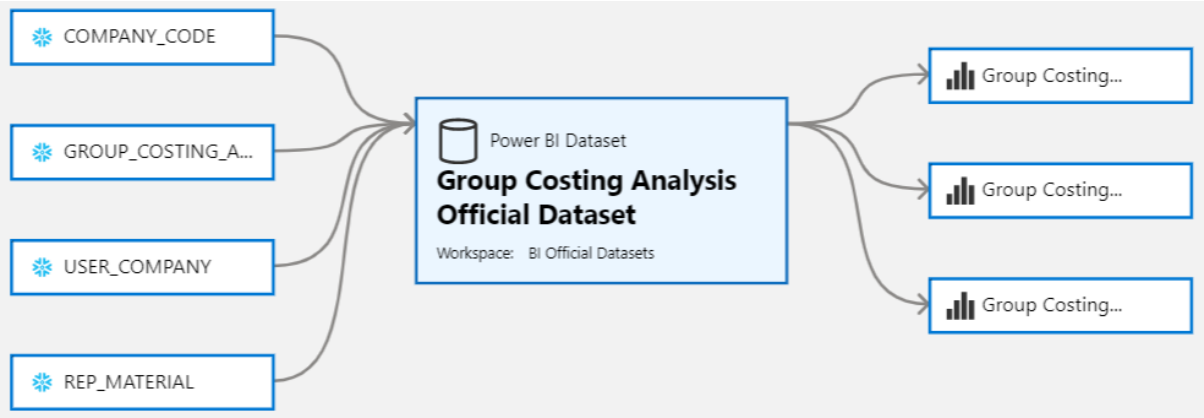


Figura 5.19: Purview: lineage del dataset `Group Costing Analysis Official Dataset`.

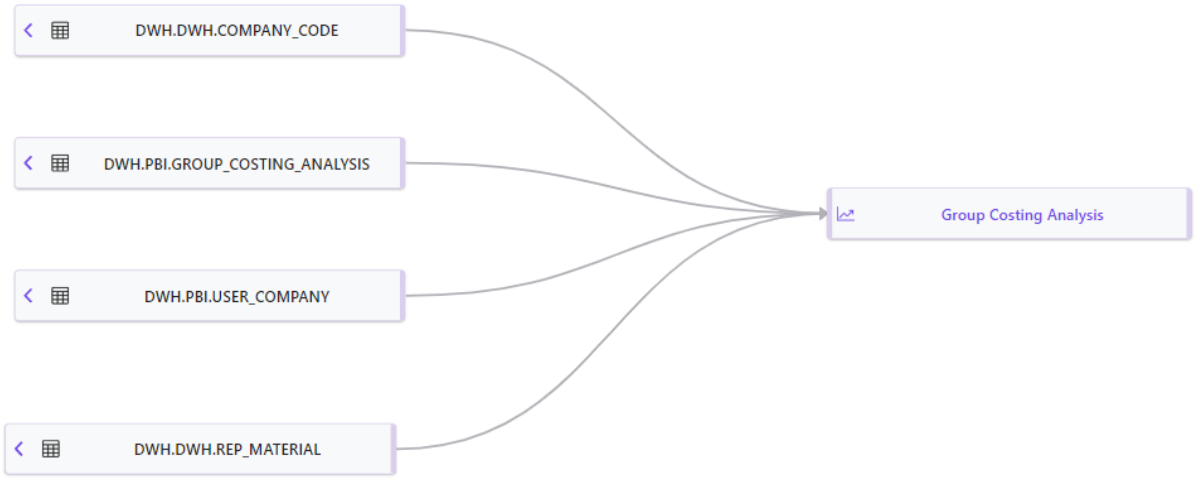


Figura 5.20: OpenMetadata: lineage della dashboard `Group Costing Analysis Official Dataset`.

5.4.2 Arricchimento del data catalog

In generale, i tre data catalog presentano funzionalità simili in termini di *business glossary*. Le differenze principali riguardo l'arricchimento degli asset e possono essere classificate in due aspetti.

Funzionalità aggiuntive

Atlan e OM propongono funzionalità che Purview non ha.

- OM è l'unico data catalog a offrire *data profiling* in maniera esaustiva dei propri asset.

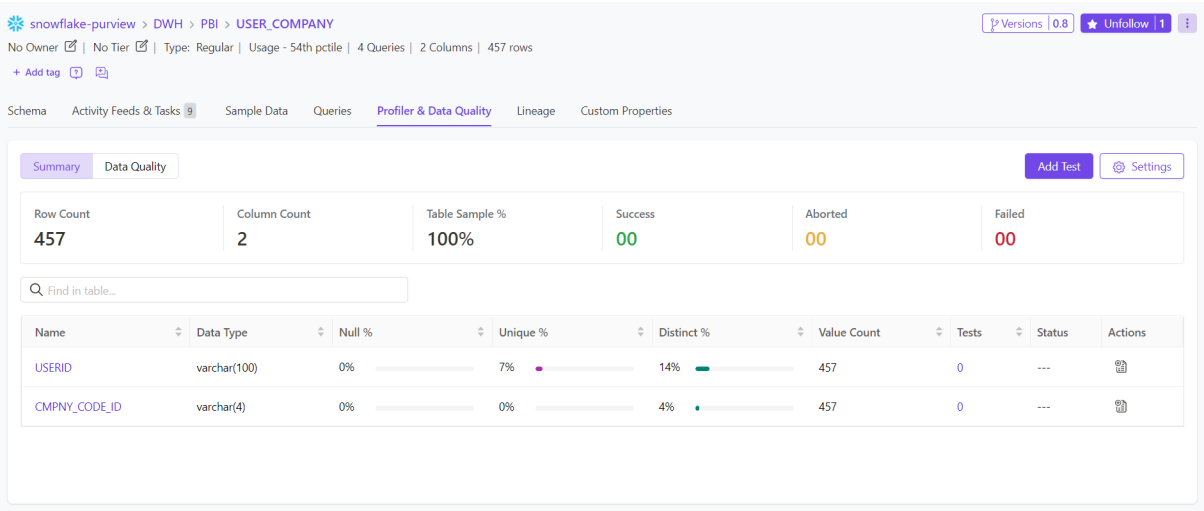


Figura 5.21: OpenMetadata: *data profiling* della tabella `DWH.PBI.USER_COMPANY`.

- Sia OM sia Atlan permettono di creare campi *custom* con cui l'utente può ulteriormente personalizzare i propri asset.
- Sia OM sia Atlan hanno diverse funzionalità minori che supportano la collaborazione tra utenti, come la possibilità di: lasciare commenti; creare richieste o task per suggerire cambiamenti da fare su un asset; visualizzare l'*activity* di un asset o seguire un *asset*; effettuare annunci per notificare modifiche importanti ecc.
- Atlan provvede un motore di suggerimenti per aiutare l'utente nell'arricchimento degli asset.

Presentazione

Tutti e tre i data catalog hanno descrizioni che permettono di formattare il testo, aggiungere allegati, immagini ecc.

Tuttavia, su Purview l'unico campo descrizione presente è tagliata nella tab dello *Schema*: anche se è comunque possibile leggerla portando il cursore sopra il testo, o in alternativa, cliccando sul nome della colonna per aprire una tab laterale, dove è possibile visualizzare per intero i dettagli della colonna, tutti questi passaggi aggiuntivi possono rendere meno piacevole agli utenti la navigazione del data catalog.

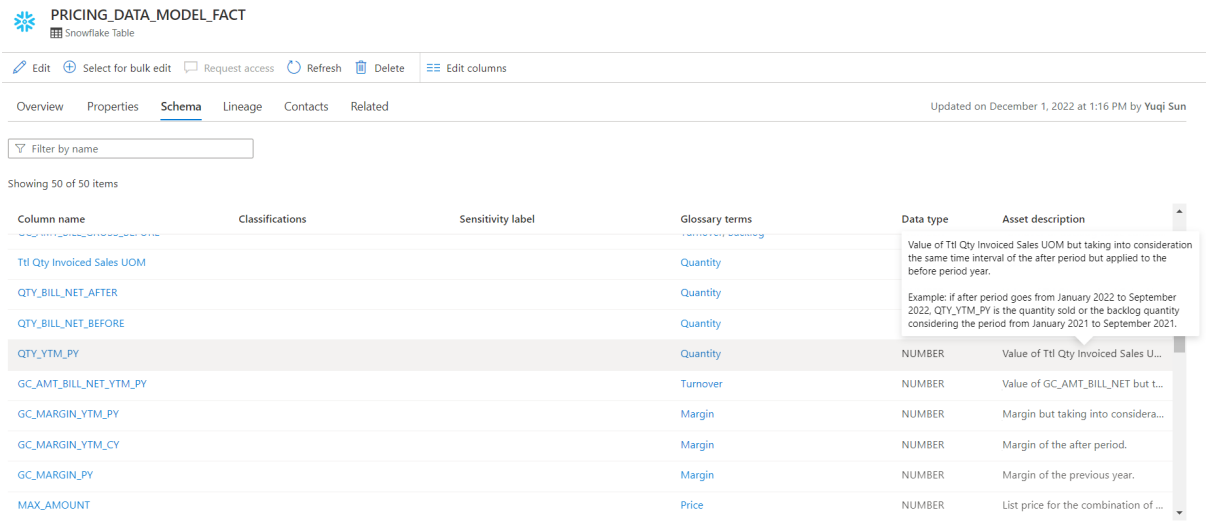


Figura 5.22: Descrizione delle colonne della tabella `PRICING_DATA_MODEL_FACT`.

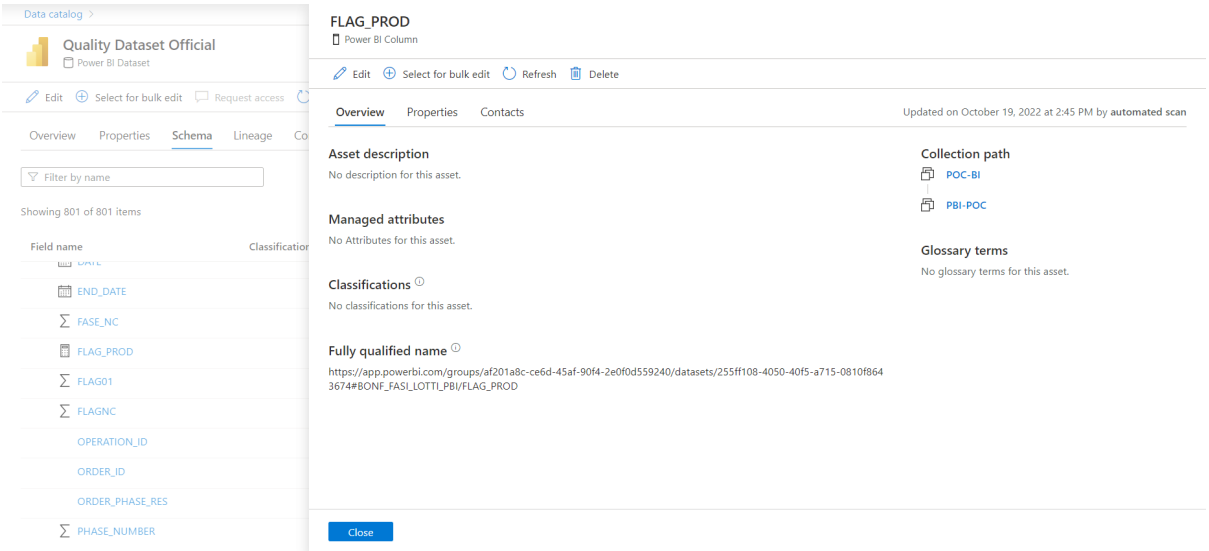


Figura 5.23: Tab laterale con i dettagli di una colonna PowerBI.

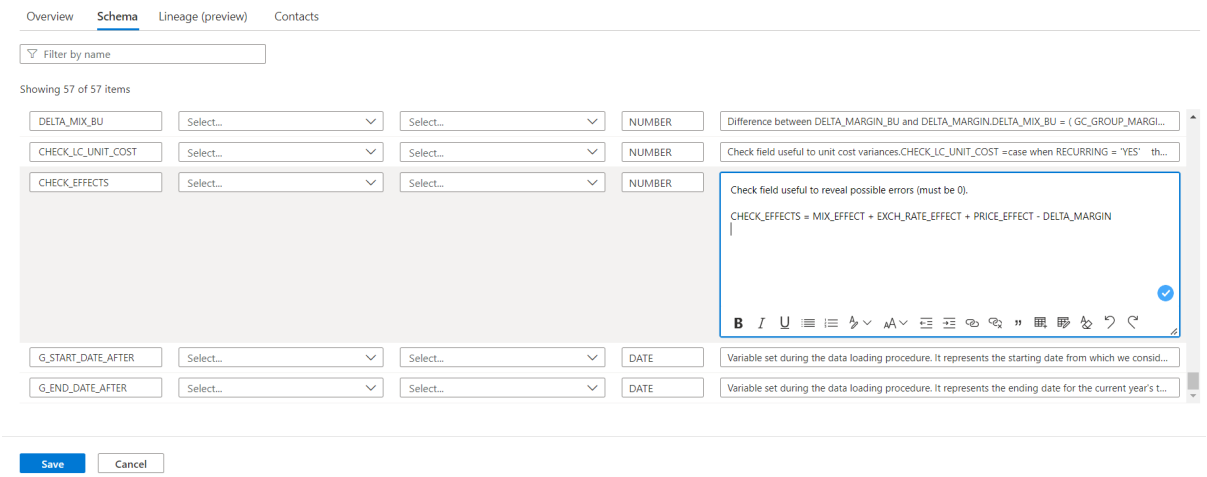


Figura 5.24: Il campo descrizione di Purview.

Anche OM ha un unico campo descrizione, ma a differenza di Purview, il campo è responsivo e non sono necessarie ulteriori azioni da parte degli utenti per visualizzare il testo.

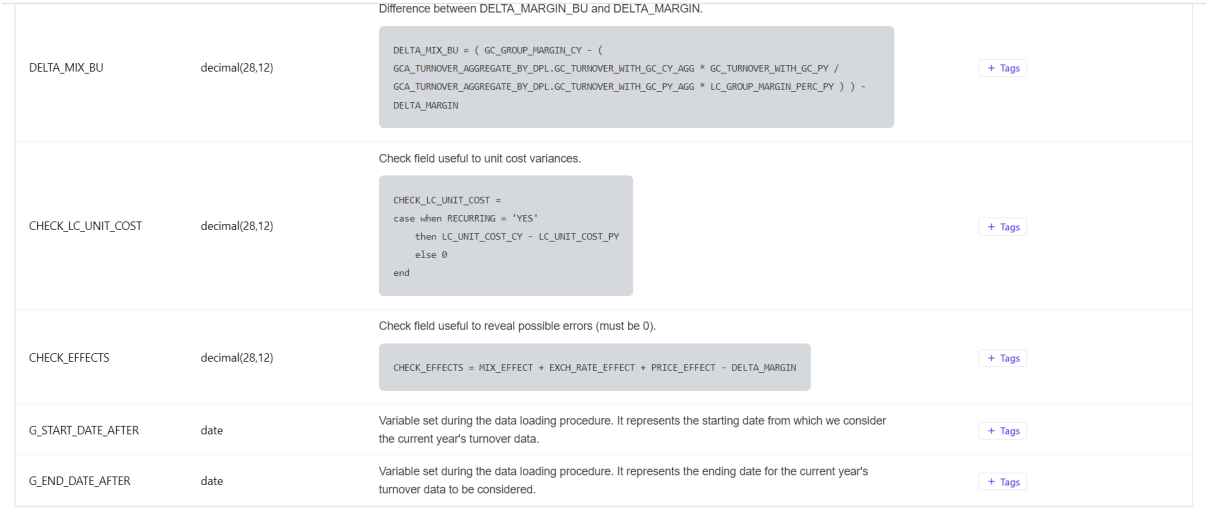


Figura 5.25: Descrizione delle colonne della tabella `GROUP_COSTING_ANALYSIS`.

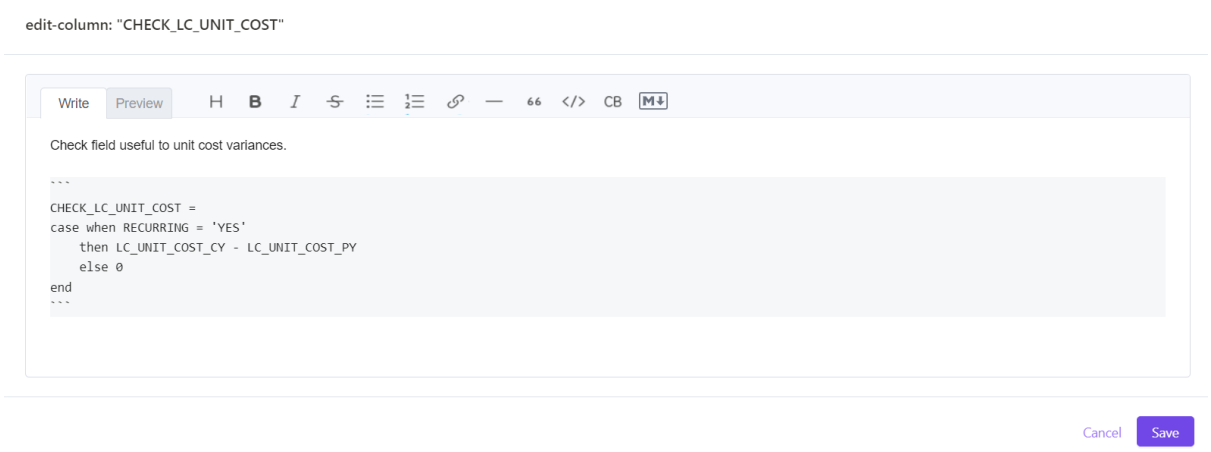


Figura 5.26: Il campo descrizione di OpenMetadata.

Atlan, invece, ha diverse modalità per arricchire un asset: oltre al campo descrizione, è possibile inserire ancor più dettagli nel campo *ReadMe*, che come OM e Purview permette di allegare contenuti multimediali. È anche disponibile un ulteriore tab *Resources*, dove è possibile aggiungere altri link che aiutano a contestualizzare meglio l'asset.

Come su Purview, anche in Atlan è necessario cliccare su una colonna per aprire il pannello laterale dove l'utente può navigare tutte le sue proprietà.

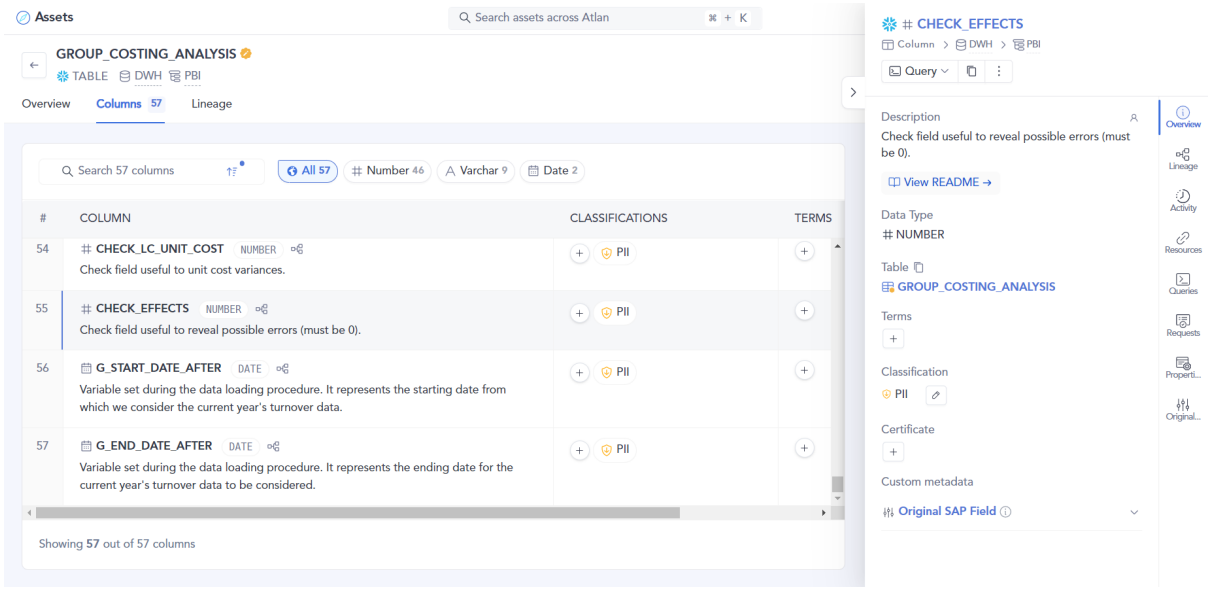


Figura 5.27: Descrizione delle colonne della tabella `GROUP_COSTING_ANALYSIS`. A lato, i dettagli della colonna `CHECK_EFFECTS`.

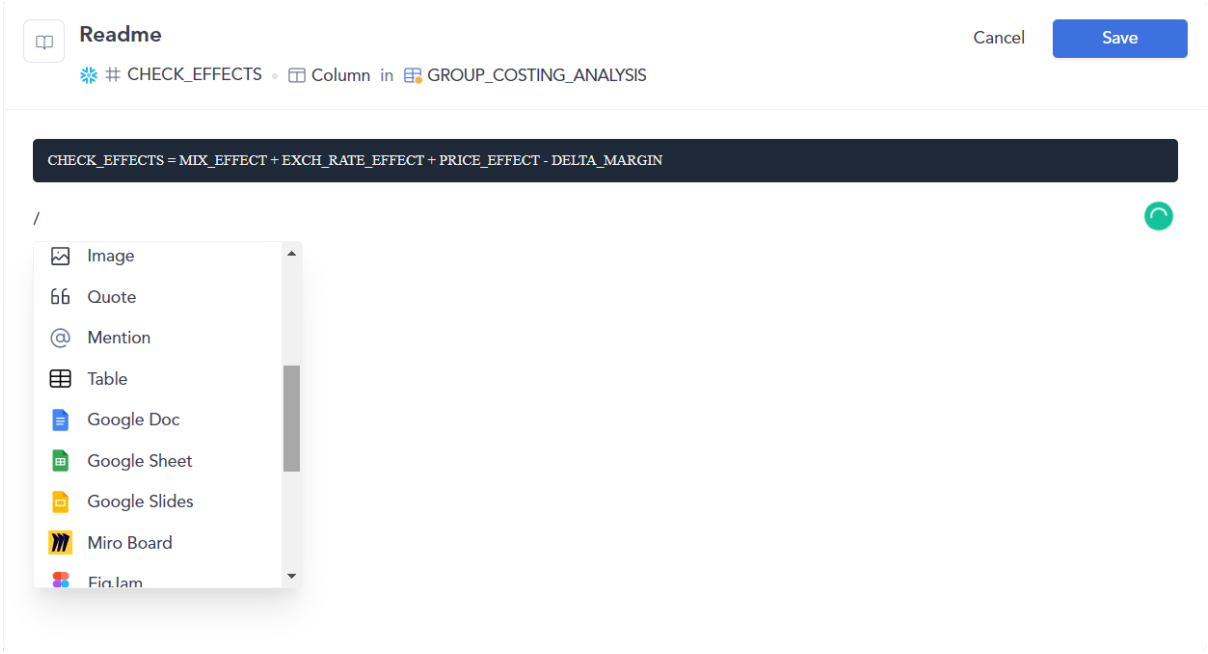


Figura 5.28: Il campo *ReadMe* di Atlan.

5.4.3 Costi

- OM, essendo open-source, non richiede costi di licenza.
- Purview richiede un costo fisso di $310 al mese, a cui si aggiunge un costo variabile basato sulle risorse impiegate per effettuare *scan* di sorgenti esterne all'ambiente Microsoft: ogni core costa $0.63, con una capacità minima di 32 core.
- Su Atlan un utenza *editor* costa $200 al mese, mentre un utenza *member* costa $25 al mese.

5.4.4 User testing

Man mano che venivano testate le principali funzionalità di interesse, la Bonfiglioli ha provato a coinvolgere due gruppi di utenti che lavorano con i report di PowerBI, in modo che potessero esprimere la loro opinione e provare Purview, OpenMetadata e Atlan.

Gruppo I

Il primo gruppo era composto da dipendenti che lavorano con il report "Group Costing Analysis". Durante la presentazione di Purview e OpenMetadata e delle loro funzionalità,

avvenuta a metà del tirocinio, i dipendenti non hanno mostrato particolare interesse a voler testare i due strumenti; tuttavia hanno espresso come un data catalog che presentava delle semplici descrizioni ad alto livello sarebbe stato poco funzionale: ciò di cui avevano veramente bisogno erano descrizioni tecniche e dettagliate dei vari campi, informazioni che attualmente solo il reparto di *business intelligence* poteva fornire.

Il gruppo coinvolto, infatti, aveva già familiarità con i dati con cui lavoravano ma non aveva una conoscenza approfondita dell'origine di alcuni campi, non avendo partecipato alla costruzione del dataset stesso. Il data catalog, per essere utile a loro, deve diventare un punto di riferimento per consultare formule e *query*, con annessa spiegazione.

Gruppo II

Il gruppo II è stato coinvolto verso la fine del tirocinio e comprendeva dipendenti che lavorano con il report "Quality". Questo gruppo si è dimostrato più interessato a testare Purview, OpenMetadata e Atlan, ma come il gruppo I, hanno fatto notare come il loro vero bisogno fossero le descrizioni più tecniche. Da questo punto di vista, Atlan e OpenMetadata risultano gli strumenti migliori, in quanto permettono di visualizzare testi lunghi e complessi.

Durante l'incontro con il gruppo II, si è anche discusso sul livello di dettaglio che era effettivamente necessario mostrare: mentre il lineage tra Snowflake e PowerBI può sicuramente beneficiare il team di BI, è nato il dubbio se queste informazioni fossero invece superflue per gli utenti che lavorano o desiderano creare report in autonomia e se in realtà sarebbe bastata solo la spiegazione del dataset PowerBI. Su questo aspetto, nella sua versione attuale, OM è inadeguato, mentre i metadati estratti da Atlan sono a un livello così dettagliato da poter rendere difficile la navigazione del data catalog per utenti principianti. Nonostante una navigazione meno intuitiva, Purview può risultare lo strumento più adatto se si forma il personale a seguire istruzioni precise per visualizzare le informazioni desiderate.

	Microsoft Purview	**Atlan**	**OpenMetadata**
Configurazione	Cloud (SaaS)	Cloud (SaaS)	On Premise con configurazione manuale
Sorgenti supportate	Azure Storage, Snowflake, Power BI	Snowflake, dbt, Power BI, Tableau	Snowflake, dbt, Power BI, Tableau
Creazione del *business glossary*	Manuale	Manuale	Manuale
Data Lineage	Semi-manuale e Completo	Automatico e Completo	Automatico ma Limitato
Scan automatico delle sorgenti	Presente	Presente	Presente
Data Profiling	Non presente	Non presente	Presente
Governance	Presente	Presente	Presente
Arricchimento del data catalog	Limitata	Ottimale	Molto buona
Facilità d'accesso	SSO integrata con Bonfiglioli tenant	SSO configurabile con MSAD	SSO configurabile con MSAD
Facilità d'uso		Estensione Chrome	
Usage analysis	Non presente	Non presente	Presente
Prezzo	\$310/mese + costo variabile per scan	Utenza editor: \$200/mese; utenza member: \$25/mese per persona	Open source

5.5 Apache Atlas e Spline

A seguire si descrivono i test effettuati su Apache Atlas e Spline.

La loro attrattiva principale è che sono entrambi progetti open-source che non comportano un costo aggiuntivo per l'azienda. Nessuno dei due, però, supporta i software espressi nei requisiti e il loro inserimento all'interno dell'attuale stack tecnologico dell'azienda avrebbe richiesto un costo di operazione e manutenzione che la Bonfiglioli non era disposta a spendere, preferendo soluzioni più complete. Tuttavia sono stati comunque effettuati degli esperimenti per analizzare il loro comportamento nei confronti di Azure Storage e Spark, in quanto sono due sorgenti che non sono supportate dagli altri tre data catalog in oggetto di questa tesi.

5.5.1 Atlas

In mancanza di una integrazione nativa con Azure Storage, il modo più diretto per provare Atlas è interagire con le API offerte.

Dopo aver disposto il container Docker, nella versione di Atlas `2.3.0`, sono stati implementati due piccoli moduli in script bash che supportano l'inserimento di asset nel data catalog a partire da file JSON di log di Azure Storage.

- File di log: il log è composto da tre sezioni: *add* indica i blocchi blob che sono stati aggiunti in una partizione, *remove* i blocchi rimossi da una partizione, mentre *commitInfo* elenca i dettagli sull'operazione effettuata, come modalità di scrittura, numero di file modificati, numero di byte e righe ecc. Benché poco rappresentativo dell'effettiva natura del log, che è una semplice lista di quali file sono stati aggiunti o eliminati dallo storage, per i scopi del test si è deciso che *add* e *remove* sono rispettivamente gli output e gli input di un'operazione, le cui specifiche sono elecante in *commitInfo*.

- Modulo A: effettua due richieste `POST` a `http://localhost:21000/api/atlas/types` per aggiungere due nuovi tipi Atlas, definiti tramite file JSON. Il nuovo tipo *adls_gen2_blob_block* modella un blocco blob di ADLS Gen2, mentre *adls_gen2_blob_process* rappresenta l'operazione avvenuta tra due o più blocchi.

- Modulo B: è composto da uno script orchestratore `main.sh` e da tre script minori, uno per ciascuna operazione possibile: `addInputEntity.sh` per i blocchi *remove*, `addOutputEntity.sh` per i blocchi *add*, `addProcessEntity.sh` per *commitInfo*. Il `main.sh`

effettua il parsing del log e, per ciascuna sezione, richiama lo script corretto, che esegue l'effettiva richiesta POST ad Apache Atlas.

```
{
  "enumTypes": null,
  "structTypes": null,
  "traitTypes": null,
  "classTypes": [
    {
      "typeName": "adls_gen2_blob_block",
      "typeDescription": "Atlas entity-type representing a blob in an Azure Data Lake Storage Gen2",
      "typeVersion": "1.0",
      "attributeDefinitions": [
        {
          "name": "url",
          "dataTypeName": "string",
          "multiplicity": "required",
          "isComposite": false,
          "isUnique": false,
          "isIndexable": true,
          "reverseAttributeName": null,
          "qualifiedName": "https://purviewpocmldl.blob.core.windows.net/purviewpoc@name"
        },{
          ....
        }],
        "hierarchicalMetaTypeName": "org.apache.atlas.typesystem.types.ClassType",
        "superTypes": [
          "DataSet"
          ]
    }
  ]
}
```

Listing 5.1: Frazione della definizione del Type che modella un blocco blob di Azure ADLS Gen2 in formato JSON.

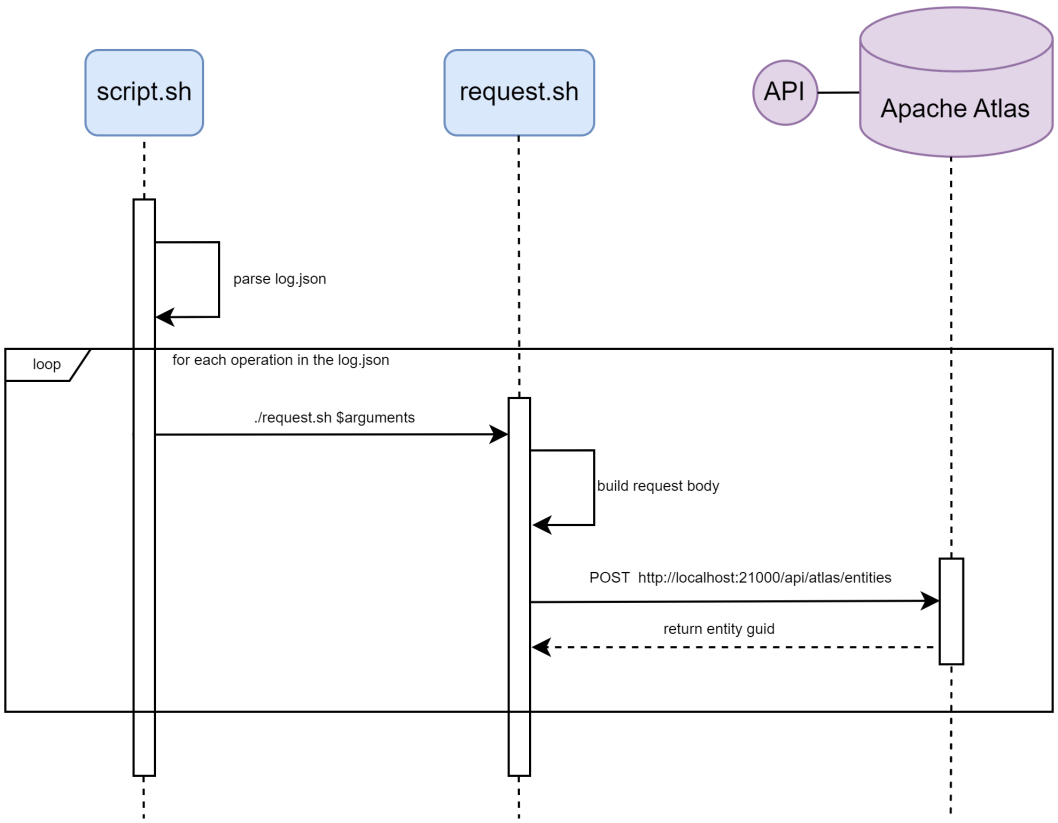


Figura 5.29: Diagramma di sequenza del modulo B.

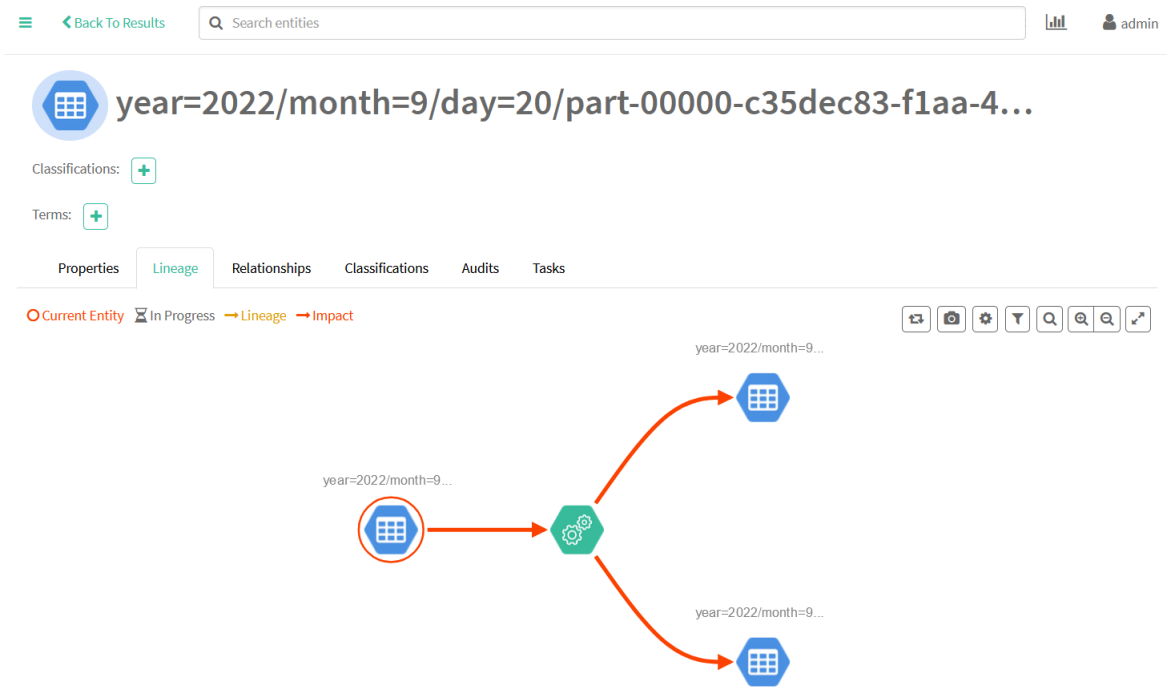


Figura 5.30: Esempio di lineage con un input e due output.

Benché il lineage costruito non abbia valore informativo e non sia indicativo delle effettive operazioni che avvengono su ADLS Gen2, già da questo piccolo esperimento era evidente che per trasformare Apache Atlas in uno strumento adeguato alle necessità della Bonfiglioli, era richiesto un grande sforzo di configurazione, soprattutto in termini di:

- inserimento dei tipi Atlas per ciascuna delle sorgenti per modellare ogni suo componente, quali tabelle, colonne, viste, dataset, report ecc.;
- sviluppo di connettori specifici per estrarre dalle sorgenti i metadati e popolare Atlas.

5.5.2 Spline

Anche per Spline sono stati usati container Docker per mettere su l'infrastruttura necessaria.

Per riuscire a catturare il lineage, basta configurare lo Spark job aggiungendo come `queryExecutionListeners` lo Spline Agent: durante l'esecuzione del job, l'agente invierà il lineage del Job al server e si potrà poi usare il Web UI per visualizzare e interagire con il lineage.

```
spark = SparkSession.builder \
        .appName("Test1") \
        .config("spark.sql.queryExecutionListeners", "za.co.absa.spline
    .harvester.listener.SplineQueryExecutionListener") \
        .config("spark.spline.lineageDispatcher.http.producer.url", "
    http://host.docker.internal:8080/producer") \
        .config("spark.sql.analyzer.failAmbiguousSelfJoin", "false") \
        .getOrCreate()
```

Listing 5.2: La Spark Session configurata con lo Spline Agent.

Dopo aver confermato la costruzione del lineage di uno Spark job che effettuava una semplice operazione di join tra due tabelle Snowflake, non sono stati fatti ulteriori esperimenti, in quanto Spline manca di tutte quelle funzionalità che sono comunemente presenti in un data catalog, come *business glossary*, ricerca e arricchimento degli asset.

Capitolo 6

Conclusione

Il motivo principale per cui la Bonfiglioli aveva deciso di introdurre un data catalog era quella di supportare i dipendenti nella *self-service BI*.

Attualmente i dati dell'azienda sono separati in silos su diverse applicazioni. Anche se grazie a dbt sono stati creati molti modelli di questi dati, cercare il modello desiderato richiede tempo; inoltre l'attuale data warehouse in uso è frutto di un lungo lavoro, con dietro molta storia, e di conseguenza esistono definizioni diverse in tabelle differenti, a volte anche in conflitto tra loro. Tutto ciò causa vari ostacoli ai dipendenti dell'azienda: da una parte, gli utenti consumatori non hanno idea di dove cercare i dati né hanno a disposizione una documentazione aggiornata dove consultare il contesto in cui questi operano; dall'altra parte, chi sviluppa e mantiene l'infrastruttura dati spende molto tempo a risolvere eventuali problemi e a supportare gli utenti consumatori.

L'obiettivo della tesi era quindi quello di testare diversi data catalog e selezionarne uno che fosse in linea con i requisiti dell'azienda: il data catalog sarebbe stato dapprima inserito nel team di BI e condiviso con un numero ristretto di utenti finali provenienti da specifici dipartimenti, e solo in seguito esteso a tutto il resto dell'azienda, con il fine ultimo di sviluppare ed estendere le pratiche di SSBI. I data catalog confrontanti sono stati Apache Atlas, Spline, Microsoft Purview, OpenMetadata e Atlan e per ciascuno di loro, ove possibile, sono stati effettuati il caricamento dei metadati dalle sorgenti supportate e l'esplorazione delle funzionalità offerte, in particolare analizzando il lineage e le possibilità di arricchimento degli asset. Dai vari esperimenti eseguiti durante il tirocinio e condivisi con il reparto di BI, le conclusioni sono state le seguenti:

- sia Atlas sia Spline non sono adatti, in quanto richiedono grandi sforzi di messa in opera e non supportano né Snowflake né PowerBI, le principali sorgenti richieste dalla Bonfiglioli;

- da un punto di vista della *user experience*, Purview è quello che è sembrato meno *user friendly*, ma supporta le principali sorgenti e funzionalità richieste;
- Atlan è il data catalog giudicato migliore in termini sia di usabilità dell'interfaccia sia delle funzionalità offerte, ma ha un costo piuttosto alto in termini di licenze;
- OpenMetadata è open-source, con una interfaccia non troppo dissimile da Atlan, ma nella sua versione attuale ha diversi limiti che lo rendono non completamente compatibile con gli attuali processi della Bonfiglioli.

Un aspetto che è importante da specificare è stato il poco coinvolgimento degli utenti finali: il tirocinio è stato svolto in un ambiente a stretto contatto con il team di BI e non ci sono state molte possibilità di confronto con gli utenti non tecnici. Ciò ha causato, per esempio, la valutazione del livello di granularità offerto dai tre data catalog solo nella fase finale del tirocinio, in quanto la capacità di usare software avanzati può variare di molto all'interno del personale, e un'analisi dei data catalog basata principalmente sui requisiti espressi dal reparto di BI, senza un'esame approfondito degli effettivi bisogni e desideri degli utenti non IT.

6.1 Adozione

A conclusione del tirocinio, la Bonfiglioli non ha scelto un data catalog da adottare, una decisione che, a detta del tutor aziendale, sarebbe stata discussa nell'anno seguente. Dopo averli contattati nel 2023 e anche nella prima metà del 2024, i referenti aziendali hanno detto che non c'erano stati ulteriori sviluppi, citando "la mancanza di tempo", "la revisione delle priorità aziendali" e "uno sviluppo della SSBI poco esteso" come motivi per il mancato investimento.

In caso la Bonfiglioli desideri riprendere l'argomento, è consigliato valutare più attentamente i bisogni degli utenti, affinché il data catalog scelto sia in grado di risolvere i problemi riscontrati nella quotidianità lavorativa e non correre il rischio che il nuovo strumento venga percepito come inutile. Più in generale, quando si sceglie un data catalog, è importante considerare che:

- esistono sia prodotti commerciali, il cui costo deve quindi rientrare entro il budget aziendale, sia prodotti open-source, che possono richiedere costi di installazione e manutenzione più o meno onerosi;

- lo strumento deve integrarsi nell'attuale infrastruttura in uso ed è quindi necessario analizzare bene quali sono le sorgenti dati, i strumenti di reportistica ed elaborazione supportati;
- la facilità d'uso dell'interfaccia deve essere adeguata ai vari livelli di abilità dei dipendenti;
- fornitori diversi provvedono funzionalità diverse e devono essere esaminati sulla base degli effettivi bisogni aziendali; alcuni esempi possono essere:
  - lineage più o meno complessi e dettagliati;
  - presenza o meno di indicatori di qualità dei dati;
  - arricchimento degli asset con diversi livelli di esaustività;
  - funzionalità che favoriscono la collaborazione tra utenti, come la possibilità di lasciare commenti, recensioni ecc.

Bibliografia

[1] R. Abraham, J. Schneider, and J. vom Brocke. Data governance: A conceptual framework, structured review, and research agenda. *International Journal of Information Management*, 49:424–438, 2019. ISSN 0268-4012. doi: https://doi.org/10.1016/j.ijinfomgt.2019.07.008. URL `https://www.sciencedirect.com/science/article/pii/S0268401219300787`.

[2] absaoss. Spline. URL `https://absaoss.github.io/spline/`.

[3] P. Alpar and M. Schulz. Self-service business intelligence. *Business & Information Systems Engineering*, 58:151–155, 2016.

[4] Apache. Apache atlas. URL `https://atlas.apache.org`.

[5] Atlan. Atlan. URL `https://atlan.com/`.

[6] O. Benfeldt, J. S. Persson, and S. Madsen. Data governance as a collective action problem. *Information Systems Frontiers*, 22:299–313, 2020.

[7] W. Bolton. *Programmable Logic Controllers*. Newnes, 2015.

[8] Bonfiglioli. La nostra storia. URL `https://www.bonfiglioli.com/italy/it/Il-Gruppo/la-nostra-storia`.

[9] P. Brous, M. Janssen, and R. Vilminko-Heikkinen. Coordinating decision-making in data management activities: A systematic review of data governance principles. In H. J. Scholl, O. Glassey, M. Janssen, B. Klievink, I. Lindgren, P. Parycek, E. Tambouris, M. A. Wimmer, T. Janowski, and D. Sá Soares, editors, *Electronic Government*, pages 115–125, Cham, 2016. Springer International Publishing. ISBN 978-3-319-44421-5.

[10] M. Center. Mes - manufacturing execution system. URL `http://mescenter.org/en/articles/108-mes-manufacturing-execution-system`.

[11] W. Eckerson. Business-driven bi: Using new technologies to foster self-service access to insights. *Tabeau Software*, 2012.

[12] GitHub. Github issue: Expand powerbi lineage, . URL https://github.com/open-metadata/OpenMetadata/issues/8946.

[13] GitHub. Github issue: Adding a child term to a parent term in glossary produces error, . URL https://github.com/open-metadata/OpenMetadata/issues/9083.

[14] GitHub. Github issue: Parser does not parse ; . URL https://github.com/open-metadata/OpenMetadata/issues/8778.

[15] GitHub. Github issue: Powerbi group not ingested because of limit of top=100, . URL https://github.com/open-metadata/OpenMetadata/issues/8944.

[16] M. Golfarelli and S. Rizzi. *Data Warehouse. Teoria e pratica della progettazione*. McGraw-Hill Education, 2006.

[17] M. Herschel, R. Diestelkämper, and H. Ben Lahmar. A survey on provenance: What for? what form? what from? *The VLDB Journal*, 26(6):881–906, dec 2017. ISSN 1066-8888. doi: 10.1007/s00778-017-0486-1. URL https://doi.org/10.1007/s00778-017-0486-1.

[18] IBM. What is a data catalog? URL https://www.ibm.com/topics/data-catalog.

[19] C. Imhoff and C. White. Self-service business intelligence. *Empowering Users to Generate Insights, TDWI Best practices report, TWDI, Renton, WA*, 2011.

[20] D. International. *DAMA-DMBOK: Data Management Body of Knowledge (2nd Edition)*. Technics Publications, LLC, Denville, NJ, USA, 2017. ISBN 1634622340.

[21] C. Labadie. *Essays on Data Democratization and Protection in the Data-driven Enterprise*. Doctoral Thesis, University of Lausanne, 2021.

[22] C. Labadie, C. Legner, M. Eurich, and M. Fadler. Fair enough? enhancing the usage of enterprise data with data catalogs. In *2020 IEEE 22nd Conference on Business Informatics (CBI)*, volume 1, pages 201–210, 2020. doi: 10.1109/CBI49978.2020.00029.

[23] C. Lennerholt, J. Laere, and E. Söderström. Implementation challenges of self service business intelligence: A literature review. 01 2018. doi: 10.24251/HICSS.2018.631.

[24] Microsoft. Introduction to azure storage, . URL https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction.

[25] Microsoft. Integration runtime in azure data factory, . URL https://learn.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime.

[26] Microsoft. Powerbi, . URL https://powerbi.microsoft.com/en-us/.

[27] Microsoft. Microsoft purview, . URL https://azure.microsoft.com/en-us/products/purview.

[28] O. Olesen-Bagneux. *The Enterprise Data Catalog*. " O'Reilly Media, Inc.", 2023.

[29] OpenMetadata. metadata.py, . URL https://github.com/open-metadata/OpenMetadata/blob/d08c9288015c0808e81ffc3891640b4840a11679/ingestion/src/metadata/ingestion/source/dashboard/powerbi/metadata.py.

[30] OpenMetadata. Openmetadata, . URL https://open-metadata.org/.

[31] D. Petrik, A. Untermann, and H. Baars. Functional requirements for enterprise data catalogs: A systematic literature review. In *Software Business*, pages 3–18, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-53227-6.

[32] Salesforce. Crm 101: What is crm? URL https://www.salesforce.com/crm/what-is-crm/.

[33] SAP. Abap programming for sap, . URL https://learning.sap-press.com/abap.

[34] SAP. Sap data services, . URL https://www.sap.com/products/technology-platform/data-services.html.

[35] SAP. Sap data services - designer guide, . URL https://help.sap.com/doc/PRODUCTION/5b775a8b23ce4bc49182298ddcb3b6e9/4.2.8/en-US/ds_42_designer_en.pdf.

[36] SAP. What is erp?, . URL https://www.sap.com/products/erp/what-is-erp.html.

[37] A. Sen. Metadata management: past, present and future. *Decis. Support Syst.*, 37 (1):151–173, apr 2004. ISSN 0167-9236. doi: 10.1016/S0167-9236(02)00208-7. URL https://doi.org/10.1016/S0167-9236(02)00208-7.

[38] S. Shanmugam and G. Seshadri. Aspects of data cataloguing for enterprise data platforms. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 134–139, 2016. doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.52.

[39] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance techniques. 2005. URL `https://api.semanticscholar.org/CorpusID:15201370`.

[40] Snowflake. Key concepts and architecture of snowflake. URL `https://docs.snowflake.com/en/user-guide/intro-key-concepts`.

[41] A. Spark. Apache spark. URL `https://spark.apache.org/docs/latest/index.html`.

[42] W. Strasser. Microsoft purview and apache atlas apis. URL `https://workingondata.wordpress.com/2023/03/26/microsoft-purview-and-the-apache-atlas-api/`.

[43] P. Strengholt. *Data Management at Scale*. "O'Reilly Media, Inc.", 2023.

[44] S. Sulaiman, J. Gómez, and J. Kurzhöfer. Business intelligence systems optimization to enable better self-service business users. 1049:35–46, 01 2013.

[45] Tableau. Tableau. URL `https://www.tableau.com/`.

[46] S. Viglas. Data provenance and trust. *Data Science Journal*, 12:GRDI58–GRDI64, 08 2013. doi: 10.2481/dsj.GRDI-010.

[47] A. Young and K. M. McConkey. Data governance and data quality: Is it on your agenda?. 2012. URL `https://api.semanticscholar.org/CorpusID:168011214`.