Alma Mater Studiorum · Università di Bologna

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Sviluppo di una data platform innovativa per un'azienda leader nel settore farmaceutico

Tesi di laurea in:
Business Intelligence

Relatore
Stefano Rizzi

Presentata da
Martin Marcolini

IV Sessione di Laurea
Anno Accademico 2023/2024

Introduzione

Nell'era digitale, i dati rappresentano una risorsa strategica per le aziende, influenzando direttamente il processo decisionale e le performance di business. La crescente complessità delle architetture IT e la necessità di integrare dati provenienti da fonti eterogenee rendono fondamentale l'adozione di soluzioni avanzate per la gestione e l'analisi delle informazioni. In questo contesto, le moderne Data Platform svolgono un ruolo chiave, consentendo alle organizzazioni di consolidare, orchestrare e valorizzare i dati in modo strutturato ed efficiente.

L'evoluzione delle Data Platform ha seguito un percorso che ha portato dal Data Warehouse tradizionale, focalizzato sulla qualità e l'integrazione del dato, ai più recenti modelli Data Lake e Data Lakehouse, che combinano scalabilità e governance avanzata.

Oggi, le Data Platform non si limitano più alla semplice archiviazione dei dati, ma fungono da ecosistemi integrati per l'acquisizione, trasformazione e analisi delle informazioni, offrendo un supporto strategico al processo decisionale aziendale. La presente tesi si inserisce in questo scenario e nasce dall'esperienza maturata durante il mio tirocinio presso Iconsulting, azienda leader nel settore della consulenza IT, specializzata in soluzioni di Business Intelligence e Data Management. Il progetto a cui ho contribuito è stato sviluppato per una multinazionale leader nel settore farmaceutico e si è focalizzato sulla realizzazione di una moderna Data Platform con l'obiettivo di ottimizzare l'integrazione e l'utilizzo dei dati aziendali, in particolare nell'ambito della Sales Excellence. L'implementazione di questa soluzione ha consentito di armonizzare e consolidare i dati provenienti dalle diverse istanze locali del CRM in un modello analitico centralizzato, migliorando la coerenza e l'affidabilità delle informazioni e abilitando analisi avanzate per supportare il processo decisionale aziendale.

Il lavoro di tesi è articolato in tre capitoli, ciascuno volto a fornire una visione strutturata del contesto teorico, delle tecnologie adottate e dello sviluppo pratico del progetto.

Capitolo 1 Nel primo capitolo vengono introdotti i concetti fondamentali legati all'evoluzione delle Data Platform, analizzando le architetture tradizionali come Data Warehouse, Data Lake e Data Lakehouse. Successivamente, viene presentata l'architettura di una Data Platform moderna, descrivendone i principali componenti e funzionalità.

Capitolo 2 Nel secondo capitolo vengono illustrate le tecnologie e gli strumenti fondamentali che hanno caratterizzato ogni fase del progetto di tesi. In particolare, vengono analizzate le soluzioni adottate per la gestione dell'intero ciclo di vita del dato, dalla sua acquisizione fino alla fruizione da parte degli utenti finali.

Capitolo 3 Il terzo capitolo entra nel merito del progetto realizzato, descrivendo lo sviluppo dello Use Case CRM. Dopo un'analisi iniziale del modello dati e delle sorgenti, vengono illustrate le fasi di implementazione, articolate in Data Ingestion e Data Preparation. Successivamente, viene descritta la predisposizione del flusso, che automatizza i processi di acquisizione e trasformazione del dato. Infine, il capitolo si conclude con la fase di Data Consumption, in cui i dati vengono resi disponibili per analisi e reportistica.

Indice

Capitolo 1

Data Platform

Una Data Platform è un'infrastruttura tecnologica progettata per raccogliere, gestire, elaborare e distribuire dati provenienti da una varietà di fonti, come database, sistemi legacy, applicazioni cloud e fonti esterne. Il suo scopo principale è garantire la disponibilità di informazioni affidabili e tempestive, che possono essere utilizzate per supportare processi decisionali strategici e operativi.

L'importanza di una Data Platform risiede nella sua capacità di centralizzare i dati e fornire un unico punto di accesso alle informazioni aziendali. Questo consente alle organizzazioni di superare le sfide legate alla gestione di grandi volumi di dati frammentati e di ottenere una visione coerente e unificata. Oltre alla centralizzazione, una Data Platform offre capacità di trasformazione e arricchimento dei dati per garantire la loro qualità, assicurando che le informazioni siano sempre aggiornate e pronte per l'uso. Inoltre, una Data Platform è in grado di supportare molteplici carichi di lavoro, che spaziano dalla reportistica tradizionale alle analisi più avanzate, come l'analisi predittiva e il machine learning. La flessibilità e la scalabilità di una Data Platform consentono alle aziende di adattarsi rapidamente alle mutevoli esigenze del business e di integrare nuove fonti di dati senza compromettere la stabilità dell'infrastruttura.

Questa piattaforma rappresenta una componente fondamentale per le organizzazioni che desiderano implementare strategie data-driven, facilitando la collaborazione tra i vari team e promuovendo un utilizzo efficace delle informazioni. In un contesto di mercato in continua evoluzione, avere una solida Data Platform permette alle aziende di mantenere la competitività e di innovare nei

propri processi.

Nel corso di questo capitolo, verranno esplorati in dettaglio i diversi aspetti delle Data Platform, partendo dalla loro evoluzione storica fino alle caratteristiche distintive delle piattaforme moderne, con un'attenzione particolare ai processi chiave che le rendono efficienti e scalabili.

1.1 Evoluzione delle Data Platform

L'evoluzione delle Data Platform riflette i continui progressi della tecnologia e l'adattamento delle aziende alle nuove sfide imposte dalla gestione di dati complessi e in costante crescita. Dalle prime soluzioni di archiviazione e analisi, come i tradizionali data warehouse, fino alle moderne infrastrutture cloud-native, il percorso evolutivo delle Data Platform ha trasformato radicalmente il modo in cui le organizzazioni raccolgono, elaborano e sfruttano i dati.

Prima di addentrarci in scenari specifici, tuttavia, è opportuno definire quali siano i principali obiettivi di tali sistemi, soffermandoci su quelli più rilevanti per gli aspetti di memorizzazione e interrogazione dei dati:

- **Qualità**: avere dati certificati, informazioni accurate e un'unica versione della verità;
- **Completezza**: utilizzare tutti i dati a disposizione, sfruttando appieno il patrimonio informativo;
- **Self-service**: prevedere reportistica istituzionale (più o meno interattiva) e dare agli utenti finali la possibilità di approfondire in autonomia le analisi;
- **Flessibilità**: evolvere e realizzare nuovi use case di analisi con semplicità e velocità, indirizzando requisiti sempre più stringenti di time to market.

Le prime implementazioni di Data Platform erano spesso rigide e monolitiche, concepite per gestire dati strutturati e supportare reportistica e analisi di base. Con l'aumentare del volume e della varietà dei dati, le aziende hanno cercato soluzioni più flessibili e scalabili, portando all'introduzione dei data lake e, successivamente, dei modelli ibridi come il data lakehouse. Questi sviluppi hanno permesso di unire i punti di forza delle architetture tradizionali con le nuove esigenze di flessibilità, scalabilità e gestione di dati eterogenei.

Oggi, le moderne Data Platform integrano funzionalità avanzate come la gestione automatizzata dei processi ETL, la gestione della data governance e il supporto alle tecnologie di machine learning e intelligenza artificiale. Queste piattaforme cloud-native non solo rispondono alle esigenze di archiviazione e analisi dei dati, ma sono progettate per facilitare decisioni data-driven, migliorando l'agilità aziendale e promuovendo l'innovazione.

Nelle sezioni seguenti verrà esplorata la trasformazione delle Data Platform, partendo dai data warehouse tradizionali fino alle moderne soluzioni cloud-native, analizzando i vantaggi, le limitazioni e le implicazioni per le aziende nel panorama competitivo attuale.

1.1.1 Data Warehouse

La prima generazione di Data Platform, nota come *Data Warehouse* (DW), è stata sviluppata per integrare i dati provenienti dai sistemi transazionali, con l'obiettivo di supportare le attività di Business Intelligence e fornire una base solida per le analisi aziendali. Il DW raccoglie, integra e conserva i dati provenienti da diverse fonti aziendali, consentendo agli utenti di accedere a informazioni storiche e aggregate in un formato standardizzato e coerente. Questo tipo di sistema è stato definito da Inmon come:

Definizione 1. *Una raccolta di dati orientata ai soggetti, integrata, variabile nel tempo e non volatile, creata per supportare il processo decisionale [7].*

Questa definizione mette in evidenza quattro caratteristiche fondamentali del DW:

- **Orientato ai soggetti**: i dati sono organizzati intorno a entità aziendali chiave, come clienti, prodotti e vendite, invece di seguire una struttura operativa.
- **Integrato**: il DW risolve problemi di incoerenza nei dati provenienti da sistemi diversi, integrandoli in un formato uniforme.
- **Variabile nel tempo**: il DW conserva dati storici, il che permette analisi di lungo periodo.
- **Non volatile**: una volta caricati, i dati nel DW non vengono modificati, ma solo aggiornati periodicamente con nuovi dati.

Un elemento essenziale del DW è il *processo di ETL* [6], che permette di estrarre dati dalle sorgenti operative, trasformarli per garantirne l'integrità e la qualità, e caricarli nel DW. Questo processo, rappresentato nella Figura 1.1, è cruciale per garantire che i dati siano affidabili, consistenti e pronti per l'analisi.

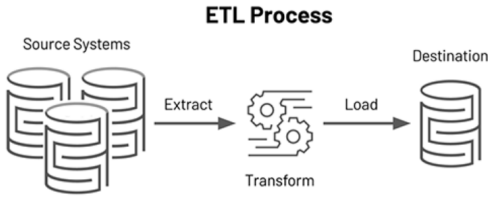


Figura 1.1: Processo di ETL [6]

Le fasi principali di un processo ETL sono:

- **Estrazione**: raccolta dei dati dalle sorgenti operative.
- **Trasformazione**: normalizzazione, arricchimento e ristrutturazione dei dati.
- **Caricamento**: memorizzazione dei dati trasformati nel DW.

Il DW è progettato utilizzando il *modello multidimensionale* [6], mostrato in Figura 1.2, che organizza i dati attorno a fatti e dimensioni. I fatti rappresentano eventi aziendali misurabili (ad esempio, vendite), mentre le dimensioni descrivono il contesto dei fatti (ad esempio, tempo, prodotto, localizzazione). Questo modello consente agli utenti di esplorare i dati secondo diverse prospettive e offre un'interfaccia intuitiva per l'analisi.

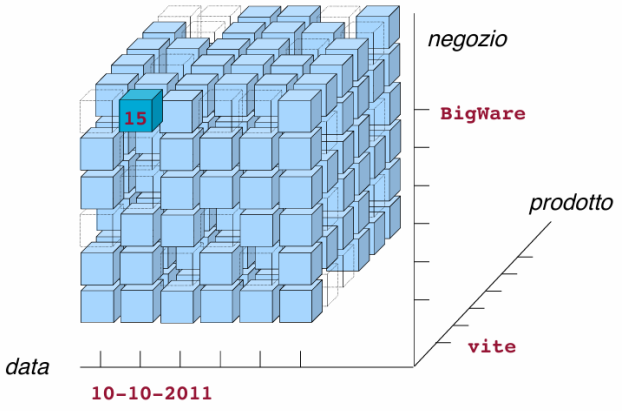


Figura 1.2: Esempio di fatto

Uno strumento fondamentale in fase di progettazione di un DW è il *Dimensional Fact Model (DFM)* [6], di cui un esempio è mostrato in Figura 1.3, che rappresenta graficamente i fatti, le dimensioni e le gerarchie. Il DFM facilita la documentazione e la comunicazione della struttura del DW, fornendo una visione chiara delle relazioni tra i dati.

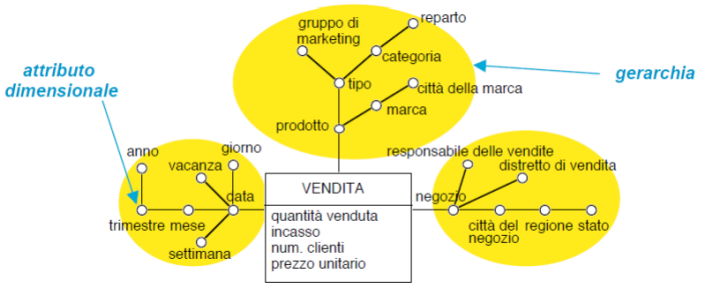


Figura 1.3: Esempio di schema di fatto che modella le vendite

La scelta dell'architettura di un DW è cruciale per soddisfare le esigenze aziendali in termini di gestione, analisi e accesso ai dati.

Esistono tre tipologie di architetture [18]:

- **Corporate Information Factory (CIF) di Inmon**

  Questa architettura, rappresentata in Figura 1.4, adotta un approccio *hub-and-spoke* in cui l'Enterprise Data Warehouse (EDW) funge da repository centrale.

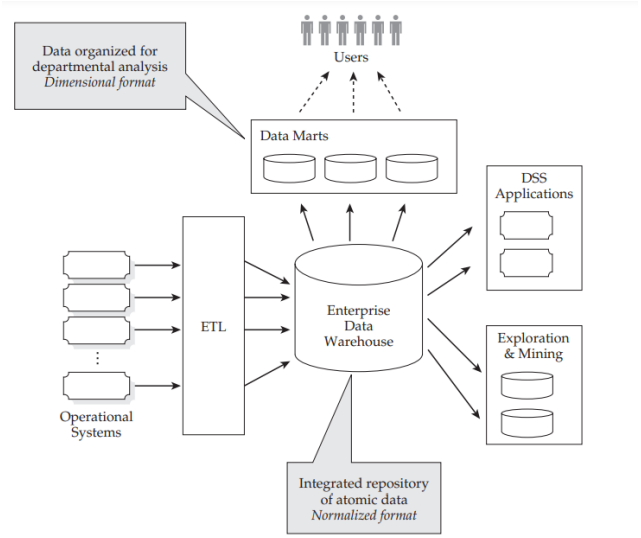


Figura 1.4: Architettura di Inmon [18]

I dati provengono da diverse fonti operative, come database, fogli di calcolo o sistemi gerarchici, e vengono trasformati e integrati attraverso un processo di ETL. Questo processo serve a raccogliere i dati, correggerli e organizzarli in modo coerente. A differenza di altri sistemi, l'EDW non viene usato direttamente per analisi, ma fornisce informazioni a piccoli archivi, chiamati *Data Mart*, che sono specifici per singoli reparti o scopi analitici. Questo approccio permette di mantenere i dati integrati e di prepararli per analisi più rapide e mirate.

- **Architettura a Bus di Kimball**

  L'architettura di Kimball, definita anche *architettura a bus* e mostrata in Figura 1.5, si basa sulla progettazione dimensionale per organizzare i dati in modo semplice e intuitivo. I dati provengono da sistemi operativi separati e vengono consolidati e caricati nel DW attraverso un processo ETL.

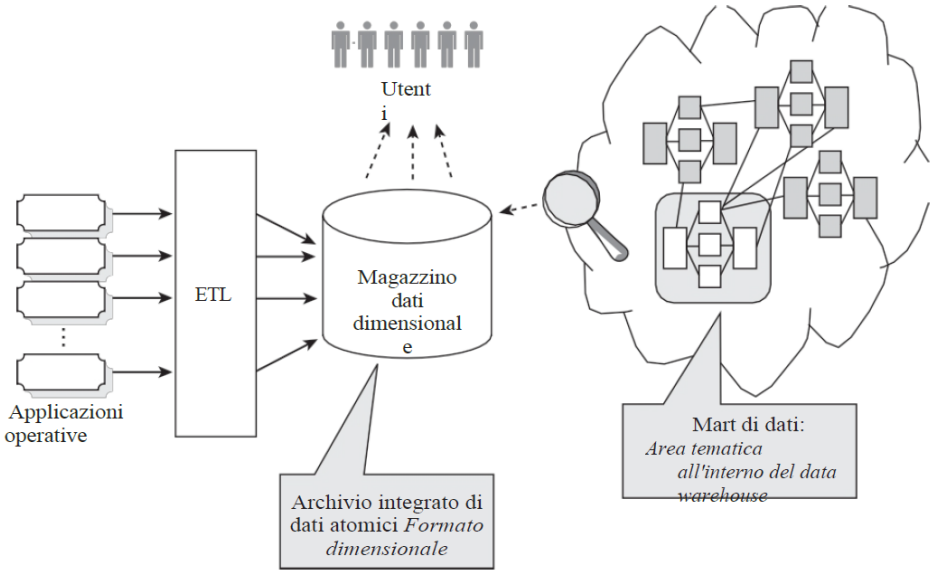


Figura 1.5: Architettura di Kimball [18]

A differenza dell'architettura di Inmon, il DW Dimensionale permette l'accesso diretto ai dati per scopi analitici, eliminando la necessità di Data Mart separati. In questa architettura, i Data Mart sono semplicemente viste tematiche interne al DW, pensate per soddisfare esigenze analitiche specifiche, come quelle di un dipartimento aziendale. Questo approccio

rende il DW più semplice da utilizzare per l'analisi aziendale e offre una maggiore flessibilità nell'accesso ai dati integrati.

- **Architettura Data Mart Stand-Alone**

  In questa architettura, mostrata in Figura 1.6, i dati provengono da diverse sorgenti operative aziendali, come sistemi transazionali, database o file esterni. Successivamente vengono elaborati attraverso un processo di ETL, che ha il compito di raccoglierli, correggerli e integrarli per prepararli all'analisi. A differenza di architetture centralizzate, i dati non vengono consolidati in un unico Data Warehouse, ma caricati direttamente nei **data mart**, piccoli archivi analitici progettati per supportare specifiche aree tematiche, come vendite o marketing.

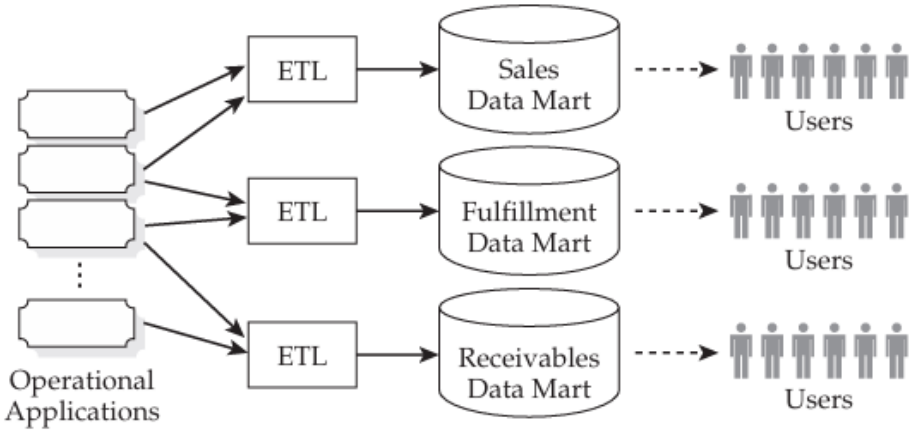


Figura 1.6: Architettura Data-Marts Stand Alone [18]

  I data mart stand-alone sono economici e rapidi da implementare, poiché non richiedono un'infrastruttura centralizzata. Tuttavia, ogni data mart opera in modo indipendente, utilizzando tecnologie e modelli potenzialmente diversi. Questa frammentazione può rendere difficile confrontare i dati tra data mart e soddisfare nuove esigenze analitiche, oltre a incrementare i costi di manutenzione. Questa architettura è utile per progetti limitati e specifici, ma comporta inefficienze nel lungo termine, soprattutto in presenza di più data mart che non condividono definizioni e regole coerenti.

Dopo aver analizzato le caratteristiche e le architetture principali dei DW, è importante considerare i vantaggi che questi sistemi offrono, così come le loro limitazioni. Il DW offre vari vantaggi:

- **Integrazione e qualità dei dati**: i processi ETL garantiscono che i dati siano puliti e integrati, rendendo il DW una fonte affidabile di informazioni.

- **Supporto decisionale**: la natura storica del DW permette analisi di lungo periodo e comparazioni temporali, supportando decisioni strategiche.

Nonostante i benefici, ci sono alcune limitazioni:

- **Alto costo di implementazione e manutenzione**: la progettazione di un DW è complessa e richiede investimenti significativi.

- **Difficoltà nella gestione di dati non strutturati**: i DW sono ottimizzati per i dati strutturati, ma mostrano limitazioni nella gestione di dati non strutturati provenienti da social media, streaming o file di log.

1.1.2 Data Lake

Con l'evoluzione delle Data Platform, il concetto di *Data Lake* è emerso come risposta alla crescente esigenza di archiviare e analizzare enormi volumi di dati eterogenei. Un Data Lake è un repository centralizzato che permette di conservare grandi quantità di dati nel loro formato originale, siano essi strutturati, semi-strutturati o non strutturati. Grazie alla sua flessibilità e scalabilità, il Data Lake è ideale per supportare le moderne esigenze di analisi dei Big Data e del Machine Learning, fornendo un'alternativa o un complemento ai tradizionali DW [11].

Nell'architettura di un Data Lake, i dati vengono acquisiti e conservati in forma grezza, senza strutturazione preventiva. Questa caratteristica, nota come *schema on read*, consente di interpretare e strutturare i dati solo al momento dell'analisi, a differenza del modello dei DW, in cui i dati vengono trasformati e strutturati al momento dell'ingresso. Questa differenza offre una maggiore flessibilità per future analisi su dati non strutturati. La capacità dei Data Lake di accogliere dati in formato nativo rende possibile l'utilizzo di sorgenti come dati da sensori IoT, file multimediali e contenuti dai social media, soddisfacendo il bisogno di una piattaforma aperta e scalabile [8].

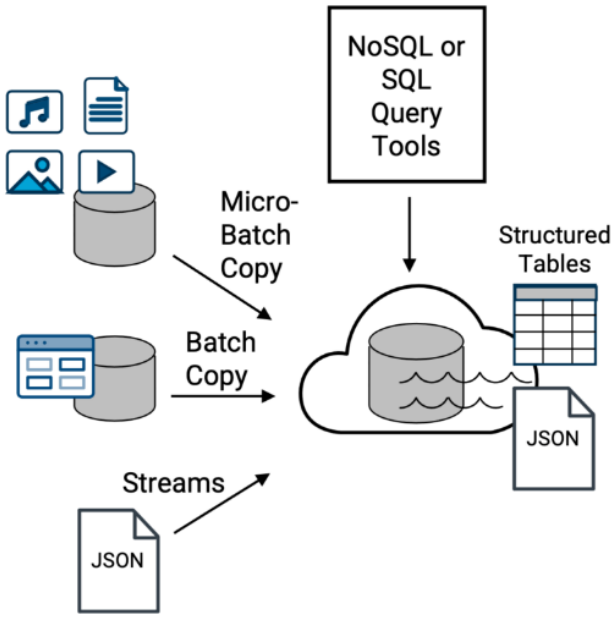


Figura 1.7: Data Lake [11]

L'architettura di un Data Lake, mostrata in Figura 1.7, si basa su una struttura centralizzata che consente di archiviare dati eterogenei nel loro formato originario, permettendo una gestione flessibile e scalabile.

- **Ingestion dei dati**: i dati possono essere caricati tramite batch, micro-batch o streaming in tempo reale. Questo approccio supporta varie sorgenti, come dati IoT, social media e file multimediali, rispondendo alle esigenze di una piattaforma aperta e scalabile [11].
- **Archiviazione e organizzazione**: i dati sono archiviati su file system distribuiti, spesso nel cloud, e possono essere mantenuti in formati come JSON o CSV. Questa gestione a strati facilita l'accesso e la manipolazione dei dati.
- **Accesso e query**: gli utenti possono interrogare i dati tramite strumenti SQL e NoSQL, senza spostare i dati. I dati grezzi possono essere trasformati in tabelle strutturate o conservati nel loro formato originale, permettendo analisi avanzate con strumenti come Apache Spark e piattaforme di BI.

Oltre a flessibilità e scalabilità, il Data Lake offre il vantaggio di poter integrare nuove fonti di dati senza richiedere modifiche strutturali o configurazioni

complesse. Questa caratteristica permette alle organizzazioni di adattarsi rapidamente a nuove esigenze di business e di generare insight avanzati da dati eterogenei. Inoltre, grazie all'utilizzo di file system distribuiti nel cloud, il Data Lake è una soluzione economicamente vantaggiosa per archiviare grandi volumi di dati, risultando più conveniente rispetto ai DW tradizionali [11, 8].

Tuttavia, l'approccio del Data Lake presenta anche delle sfide. L'archiviazione di dati grezzi e non strutturati può compromettere la qualità e l'integrità dei dati, portando alla cosiddetta *"palude di dati"* (data swamp) se non viene gestita adeguatamente. La governance dei dati diventa quindi complessa, specialmente in contesti che richiedono elevati standard di qualità, sicurezza e conformità. Inoltre, la natura *"write once"* dei file system distribuiti limita la possibilità di aggiornare direttamente i dati, aumentando la complessità e il rischio di duplicazioni [3]. Infine, la mancanza di una struttura e di controlli centralizzati rende anche più difficile implementare politiche di sicurezza e di accesso, aumentando i rischi per la conformità, particolarmente nei settori regolamentati [11].

1.1.3 Data Lakehouse

La terza generazione delle Data Platform è il *Data Lakehouse* [5]. Questa architettura unisce le caratteristiche chiave dei DW e dei Data Lake, rispondendo alle crescenti esigenze delle organizzazioni di gestire grandi volumi di dati eterogenei in modo scalabile e flessibile.

I tradizionali DW, pur garantendo elevati standard di qualità, governance e prestazioni analitiche, sono rigidi e costosi. Basandosi sullo schema-on-write, richiedono che i dati siano strutturati e modellati prima del loro caricamento, rendendoli inadatti a nuove fonti di dati non strutturati, come immagini, video e log. Inoltre, l'aumento del volume dei dati fa crescere esponenzialmente i costi infrastrutturali, rendendo questa architettura meno sostenibile. D'altra parte, i Data Lake offrono flessibilità e scalabilità, permettendo di archiviare dati grezzi di qualsiasi tipo in formati aperti. Tuttavia, questa libertà si traduce spesso in problematiche di governance e qualità. Senza adeguati controlli, i Data Lake rischiano di trasformarsi in data swamp, dove le informazioni diventano inutilizzabili a causa di scarsa organizzazione e metadati inadeguati. Il Data Lakehouse nasce per colmare questo divario, unendo la governance e la qualità dei dati tipiche dei DW con la scalabilità e la flessibilità dei Data Lake.

Basandosi su formati di dati aperti come Apache Parquet e ORC, garantisce interoperabilità e riduce il rischio di dipendenza dai fornitori. Inoltre, funzionalità avanzate come transazioni ACID, caching e indicizzazione migliorano la coerenza e l'efficienza, rendendo il Data Lakehouse ideale per applicazioni moderne come il machine learning e le analisi predittive.

L'architettura del Data Lakehouse, mostrata in Figura 1.8, integra le migliori funzionalità di Data Lake e DW. Questa architettura permette di gestire i dati in modo efficiente, garantendo un'elevata qualità e flessibilità operativa.

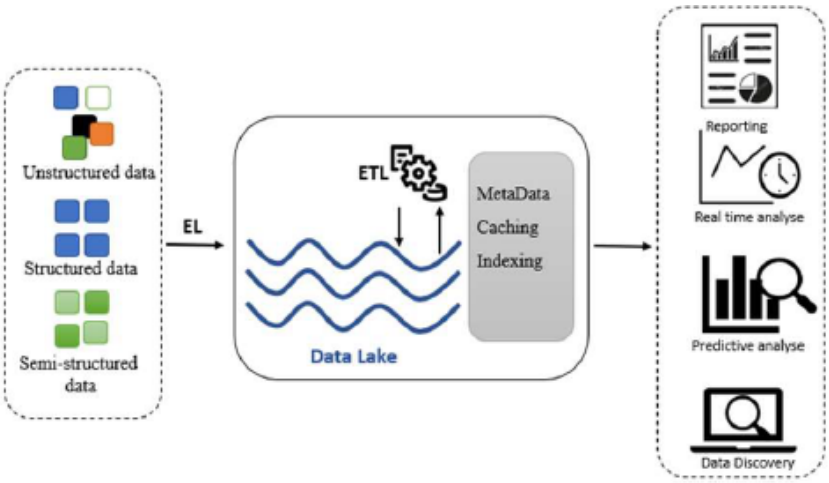


Figura 1.8: Data Lakehouse [5]

I dati provenienti da diverse fonti, strutturati, semi-strutturati e non strutturati, vengono integrati nel sistema senza trasformazioni iniziali, preservandone l'integrità originale. Questa caratteristica permette di archiviare e gestire una varietà di dati, inclusi quelli prodotti da sensori IoT, social media e database transazionali.

Il livello centrale dell'architettura è il Data Lake, che funge da deposito scalabile per tutti i dati. Qui, i dati vengono arricchiti da un sistema di metadati avanzati che consente di catalogarli, organizzarli e accedervi in modo efficiente. Inoltre, funzionalità come il caching e l'indicizzazione ottimizzano le prestazioni delle query, rendendo possibile l'accesso a informazioni in tempo reale o quasi.

Un elemento chiave del Data Lakehouse è il suo motore di elaborazione, che combina capacità di calcolo in batch e in streaming. Utilizzando tecnologie come *Apache Spark*, questo livello consente di eseguire analisi complesse, machine learning e trasformazioni dei dati su larga scala.

L'output del Data Lakehouse è altrettanto versatile: supporta la generazione di reportistica tradizionale basata su SQL, analisi predittive avanzate e visualizzazioni in tempo reale. Inoltre, l'architettura del Data Lakehouse è progettata per sfruttare al meglio l'ambiente cloud, separando le risorse di archiviazione e calcolo. Questo approccio offre un'elevata scalabilità e un modello economico flessibile, consentendo alle organizzazioni di adattare le risorse in base alle loro esigenze operative.

Per garantire affidabilità, scalabilità e funzionalità avanzate, il Data Lakehouse si avvale di tecnologie di archiviazione e gestione dati di ultima generazione. Tra queste:

- **Delta Lake** [4]: supporta transazioni ACID, caching e gestione efficiente dei dati su cloud object store come Amazon S3.
- **Apache Iceberg** [1]: fornisce ottimizzazioni per grandi tabelle analitiche e versioning dettagliato, facilitando l'elaborazione dei dati incrementali.
- **Apache Hudi** [2]: ideale per gestire flussi di dati storici e incrementali, migliorando l'efficienza delle query su larga scala.

Infine il Data Lakehouse offre i seguenti vantaggi:

- **Scalabilità e flessibilità**: capacità di gestire enormi volumi di dati di qualsiasi tipo senza sacrificare le prestazioni.
- **Unificazione dei dati**: eliminazione dei silos informativi grazie a una piattaforma unica per la gestione e l'analisi dei dati.
- **Supporto ad analisi avanzate**: accesso diretto ai dati per applicazioni di machine learning, intelligenza artificiale e analisi predittive, con compatibilità garantita grazie all'utilizzo di formati di dati aperti.

Nonostante i suoi numerosi vantaggi, il Data Lakehouse presenta alcune sfide e limitazioni:

- **Complessità tecnica**: configurazioni avanzate che richiedono competenze specifiche nella gestione di metadati, caching e transazioni ACID.
- **Costi iniziali elevati**: l'implementazione richiede investimenti significativi in infrastrutture e risorse umane, con tempi di integrazione più lunghi rispetto alle architetture tradizionali.

1.2 Architettura di una Data Platform moderna

Negli ultimi anni, le esigenze di business e i progressi tecnologici hanno trasformato radicalmente il panorama delle Data Platform, portando alla nascita di architetture sempre più integrate e flessibili. Se le soluzioni tradizionali come i Data Warehouse e i Data Lake hanno dominato il passato, l'evoluzione delle tecnologie cloud-native, l'esplosione dei Big Data e l'emergere di applicazioni avanzate come il machine learning hanno spinto le organizzazioni verso una nuova generazione di Data Platform. Questa soluzione si distingue per la capacità di combinare scalabilità, flessibilità e governance avanzata in un'unica architettura, progettata per gestire l'intero ciclo di vita dei dati: dalla raccolta all'analisi e al consumo finale. Un esempio concreto di questa evoluzione è rappresentato dall'architettura sviluppata da *Iconsulting*, mostrata in Figura 1.9, che identifica le componenti fondamentali di una Data Platform moderna ed efficace. Grazie a questa struttura, le organizzazioni possono ottimizzare l'utilizzo dei dati, democratizzandone l'accesso e accelerando il processo decisionale. La chiave del successo di questa architettura risiede nella modularità, che permette a ciascuna componente di svolgere un ruolo chiave per garantire la governance, l'efficienza e la sicurezza dei dati.

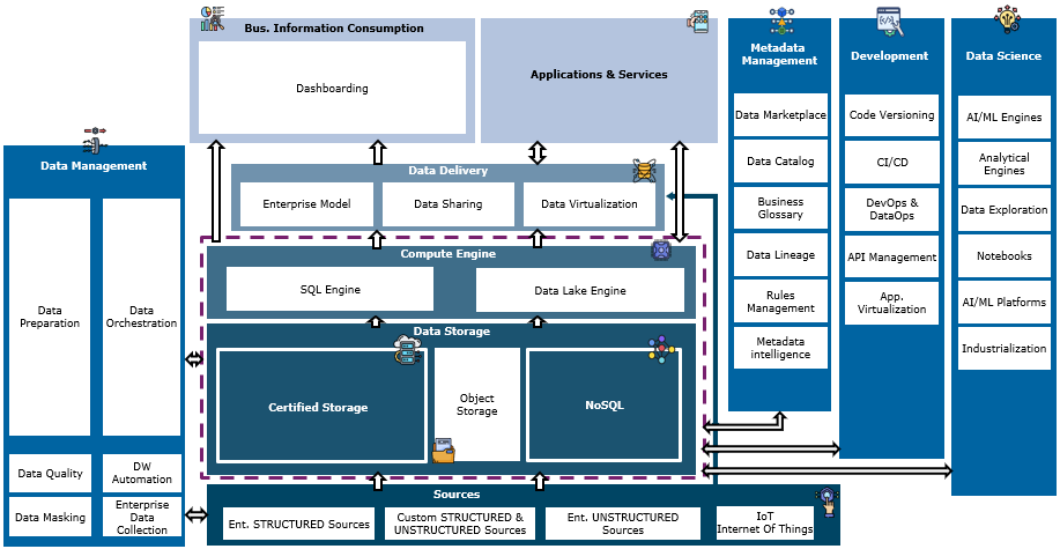


Figura 1.9: Architettura Data Platform

Di seguito, una panoramica delle principali aree funzionali di una Modern

Data Platform:

- **Data Storage**: comprende soluzioni per lo storage certificato (RDBMS), lo storage oggetti e i database NoSQL. Questi sistemi garantiscono scalabilità e prestazioni elevate per dati strutturati e non.
- **Compute Engine**: fornisce la potenza di calcolo necessaria per elaborare i dati. Include motori SQL e altre soluzioni scalabili per supportare l'analisi dei dati e l'esecuzione di algoritmi complessi.
- **Data Delivery**: si occupa della condivisione e virtualizzazione dei dati, permettendo agli utenti di accedere rapidamente a informazioni pronte per l'uso analitico, senza dipendere dai team IT.
- **Business Information Consumption**: fornisce strumenti per la fruizione dei dati attraverso dashboard, report e applicazioni analitiche self-service, consentendo agli utenti di accedere facilmente alle informazioni necessarie per le decisioni strategiche.
- **Applications & Services**: integra piattaforme per lo sviluppo di applicazioni web e strumenti per il Corporate Performance Management (CPM). Questi elementi migliorano la produttività aziendale e l'efficienza operativa.
- **Data Management**: include strumenti e processi per la raccolta, integrazione, trasformazione e pulizia dei dati. Offre funzionalità come ETL, streaming, gestione delle API, orchestrazione dei flussi e automazione del Data Warehouse, assicurando la qualità e la protezione dei dati.
- **Metadata Management**: supporta l'organizzazione e la gestione centralizzata dei metadati, ovvero tutte le informazioni che descrivono i dati aziendali, le loro origini e le trasformazioni subite.
- **Development**: comprende soluzioni per il versionamento del codice, Continuous Integration (CI/CD) e il DevOps/DataOps, favorendo l'automazione e la collaborazione tra i team di sviluppo.
- **Data Science**: offrono un ecosistema completo per l'esplorazione, analisi e industrializzazione di modelli di machine learning e intelligenza artificiale. Include strumenti come AI/ML engines, notebook interattivi e piattaforme di analisi avanzata.

1.2.1 Data Storage

Il livello di *Data Storage* rappresenta il cuore dell'architettura di una Data Platform moderna, fornendo l'infrastruttura essenziale per l'archiviazione, la gestione e la distribuzione dei dati. È il punto di convergenza in cui i dati, provenienti da fonti eterogenee, vengono consolidati per supportare operazioni aziendali e analisi strategiche. Questo livello non si limita a fungere da repository passivo, ma si adatta alle diverse esigenze aziendali, assicurando una gestione efficace dei dati in base al loro formato, utilizzo e profondità storica.

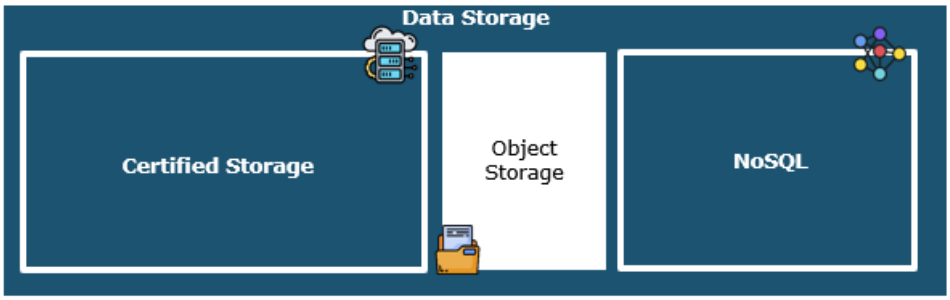


Figura 1.10: Data Storage

Per soddisfare le esigenze aziendali il Data Storage, come mostrato in Figura 1.10, integra una gamma di tecnologie progettate per affrontare sfide specifiche legate alla scalabilità, alle prestazioni e alla flessibilità.

Il **Certified Storage** rappresenta l'infrastruttura principale per la gestione di dati strutturati, garantendo sicurezza, integrità e affidabilità attraverso tecnologie avanzate. Tra queste, i **Relational DBMS (RDBMS)** sono una delle soluzioni più consolidate, e, grazie al supporto delle proprietà ACID (Atomicity, Consistency, Isolation, Durability), sono particolarmente adatti per operazioni transazionali e analisi su grandi volumi di dati.

L'**Object Storage** è una soluzione altamente scalabile, progettata per gestire dati eterogenei. Supporta sia il layer raw, che contiene dati grezzi in formati aperti come JSON, CSV o XML, sia il layer raffinato, che archivia dati elaborati in formati come Parquet o ORC. La sua capacità di scalare orizzontalmente lo rende particolarmente adatto per gestire volumi di dati storici e applicazioni avanzate, come analisi big data.

Infine i **NoSQL DB** offrono flessibilità per gestire dati non strutturati o semi-strutturati e supportano scenari che richiedono scalabilità elevata e modelli di dati dinamici.

1.2.2 Compute Engine

Il *Compute Engine* costituisce un componente essenziale nelle Data Platform moderne, garantendo efficienza e scalabilità nell'elaborazione di grandi volumi di dati strutturati, semi-strutturati e non strutturati.

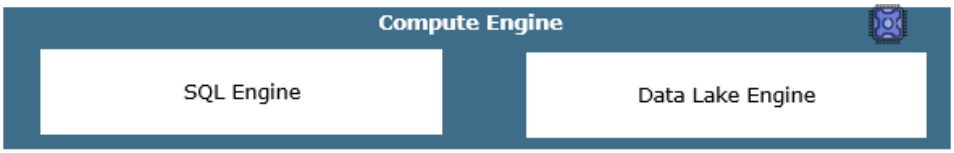


Figura 1.11: Compute Engine

Il Compute Engine, come mostrato in Figura 1.11, integra principalmente due tipi di motori, ciascuno progettato per rispondere a esigenze specifiche di elaborazione e analisi.

Da una parte troviamo gli **SQL Engine**, motori altamente ottimizzati per interpretare ed eseguire istruzioni SQL, consentendo la gestione e l'interrogazione dei dati in contesti relazionali.

Dall'altra parte abbiamo invece i **Data Lake Engine**, che sono progettati specificamente per i grandi volumi di dati eterogenei tipici dei Data Lake. Questi motori separano lo storage dalla computazione, offrendo maggiore flessibilità e scalabilità. Supportano il calcolo distribuito, permettendo l'elaborazione di dati strutturati e non strutturati, con la possibilità di utilizzare interrogazioni SQL-like.

1.2.3 Data Delivery

Il *Data Delivery* è un componente della Data Platform responsabile di rendere i dati accessibili agli utenti finali in modo semplice, sicuro e trasparente. Situato tra i livelli di calcolo e consumo, il Data Delivery semplifica l'accesso ai dati garantendo che le informazioni siano fruibili in modo efficace per supportare decisioni strategiche.

Figura 1.12: Data Delivery

Il Data Delivery, come mostrato in Figura 1.12, si basa su un insieme di approcci progettati per ottimizzare la distribuzione e la fruizione dei dati.

L'**Enterprise Model**, ad esempio, introduce un modello semantico unificato che facilita l'accesso ai dati provenienti da fonti eterogenee. Questo metodo permette agli utenti finali di consultare e analizzare i dati senza preoccuparsi dei dettagli tecnici sottostanti.

Il **Data Sharing** permette una condivisione sicura e controllata delle informazioni, sia all'interno sia all'esterno dell'organizzazione, grazie a protocolli di accesso aperti e meccanismi di profilazione avanzati.

Infine, la **Data Virtualization**, consente di integrare i dati da sorgenti diverse senza duplicarli fisicamente, migliorando le prestazioni con tecniche di caching e garantendo un controllo centralizzato sugli accessi e sulla sicurezza.

1.2.4 Business Information Consumption

Il *Business Information Consumption* rappresenta il punto di accesso principale alle informazioni per gli utenti aziendali. Questo livello è progettato per trasformare i dati in insight fruibili attraverso report, dashboard e strumenti di analisi interattiva.

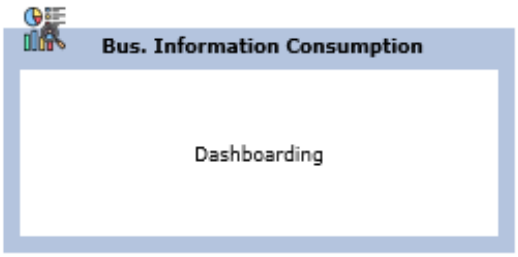


Figura 1.13: Business Information Consumption

Il Business Information Consumption, mostrato inFigura 1.13, si basa su un insieme di strumenti che semplificano l'accesso e l'analisi dei dati per gli utenti finali. L'**Enterprise Reporting Hub** offre una piattaforma centralizzata per la gestione e la distribuzione efficace dei report aziendali, garantendo un accesso semplice e strutturato alle informazioni rilevanti. Lo **Static Reporting**, invece, permette la creazione di report statici in formati facilmente condivisibili come PDF o Excel.

Infine, il **Self-Service Data Visualization** rende gli utenti autonomi nella creazione e personalizzazione di report e dashboard grazie a strumenti intuitivi,

riducendo la dipendenza dal reparto IT e consentendo personalizzazioni rapide e adattabili alle proprie necessità analitiche.

1.2.5 Applications & Services

Il livello *Application & Services* include strumenti e infrastrutture progettati per sviluppare, integrare e gestire applicazioni che supportano i processi aziendali e le decisioni strategiche. Questo livello dell'architettura sfrutta i dati raffinati ed elaborati dai livelli sottostanti per creare soluzioni personalizzate, ideate per assistere le organizzazioni nelle attività operative e analitiche.

Figura 1.14: Applications and Services

Il livello Application & Services, mostrato in Figura 1.14, è caratterizzato da una serie di soluzioni chiave che ottimizzano l'accesso ai dati e migliorano l'efficacia dei processi aziendali. Le **Web App Platforms** permettono la creazione di applicazioni web intuitive, facilitando agli utenti l'accesso ai dati strutturati e consentendo alle aziende di modernizzare rapidamente i propri processi e di sviluppare soluzioni personalizzate senza la necessità di competenze tecniche avanzate. Infine, le **CPM Platforms** (Corporate Performance Management) rappresentano strumenti strategici che traducono i dati in piani operativi concreti, supportando sia la pianificazione aziendale sia il monitoraggio continuo delle performance.

1.2.6 Data Management

Il *Data Management* supporta l'intero processo di acquisizione, integrazione e trasformazione dei dati grezzi provenienti da fonti eterogenee. Questo livello non si limita alla raccolta ma svolge un ruolo chiave nell'orchestrazione dei flussi, nell'arricchimento delle informazioni, nella gestione della qualità e nella protezione dei dati, rendendoli pronti per l'utilizzo nei livelli superiori dell'architettura.

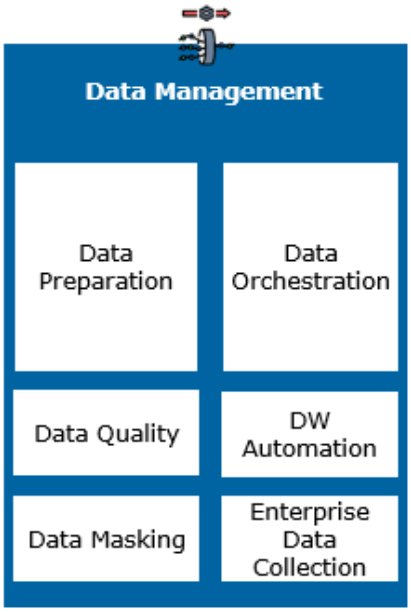


Figura 1.15: Data Management

Il Data Management, come mostrato in Figura 1.15, integra una vasta gamma di strumenti e processi che lavorano in modo coordinato per garantire una gestione efficace dei dati. In particolare nell'ambito della **Data Preparation** rientrano strumenti essenziali quali i processi di **Bulk/Batch ETL**, fondamentali per caricare e trasformare grandi volumi di dati in modalità pianificata o basata su eventi specifici. Parallelamente, l'elaborazione in **streaming** consente di gestire flussi di dati in tempo quasi reale, utilizzando code di messaggi per separare i produttori dai consumatori e garantire un flusso costante e rapido di informazioni. Le **API REST**, invece, forniscono un'interfaccia standard per l'accesso e l'integrazione dei dati, semplificando l'interoperabilità tra sistemi diversi e facilitando l'esposizione dei dati verso applicazioni esterne.

Inoltre un aspetto fondamentale del Data Management è la **Data Orchestration**, che consente di orchestrare i flussi dati all'interno di una pipeline garantendo che le dipendenze tra i processi vengano rispettate. L'automazione del Data Warehouse semplifica ulteriormente la creazione e modifica delle pipeline, riducendo il bisogno di competenze tecniche avanzate e accelerando i tempi di implementazione.

Oltre alla trasformazione e orchestrazione, il Data Management si occupa della raccolta e arricchimento di dati non gestiti dai sistemi tradizionali, attraverso strumenti di **Enterprise Data Collection**, che ampliano il patrimonio

informativo aziendale. La qualità del dato viene garantita da processi avanzati di **Data Quality**, che identificano, correggono e monitorano eventuali anomalie, assicurando che i dati siano sempre accurati e affidabil. Infine, la protezione delle informazioni sensibili è garantita dalle tecniche di **Data Masking**, che identificano automaticamente i dati critici e applicano misure di offuscamento per assicurare conformità alle normative e sicurezza delle informazioni.

1.2.7 Metadata Management

Il *Metadata Management* è un elemento centrale delle Data Platform moderne, progettato per organizzare, tracciare e valorizzare i metadati aziendali. Operando trasversalmente a tutti i livelli dell'architettura offre agli utenti una visione chiara e approfondita dei dati aziendali, dalle origini fino al consumo finale.

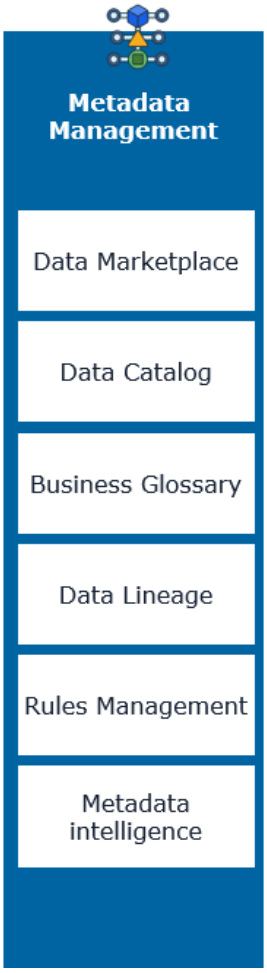


Figura 1.16: Metadata Management

Il Metadata Management, come mostrato in Figura 1.16, comprende diversi strumenti strategici che permettono alle aziende di sfruttare al massimo il valore dei metadati. Tra questi, il **Data Marketplace** offre agli utenti un'esperienza interattiva e intuitiva, simile a quella di uno shop online, facilitando la ricerca, l'accesso e la richiesta di dati rilevanti, migliorando così l'accessibilità e l'efficienza nell'utilizzo delle informazioni aziendali.

Al centro di questo ecosistema si colloca il **Data Catalog**, il cuore pulsante della gestione dei metadati. Questo strumento agisce come un inventario completo degli asset aziendali, dotato di funzionalità avanzate per la ricerca, il linking e l'organizzazione delle informazioni. Grazie all'utilizzo di tecniche di machine learning, automatizza attività complesse come la discovery dei dati, l'ingestion automatica e la creazione di relazioni semantiche.

Parallelamente, il **Business Glossary** facilita la comunicazione e la governance attraverso la definizione e la standardizzazione dei termini aziendali. Riducendo ambiguità e interpretazioni divergenti, assicura che tutti gli utenti condividano una comprensione comune del significato e del contesto dei dati aziendali.

Il **Data Lineage**, invece, fornisce una rappresentazione dettagliata e trasparente del ciclo di vita dei dati, mostrando chiaramente il percorso dal punto di origine fino alle destinazioni finali, comprese tutte le trasformazioni intermedie.

Il **Rules Management** permette di definire e monitorare l'applicazione uniforme delle regole aziendali, assicurando che le politiche e le procedure di governance vengano rispettate in modo coerente.

Infine, il **Metadata Intelligence** valorizza ulteriormente i metadati sfruttando algoritmi avanzati di apprendimento automatico per identificare insight nascosti e relazioni utili fra le informazioni aziendali. Questa capacità permette di estrarre conoscenza preziosa che può migliorare significativamente la gestione complessiva delle risorse informative e le decisioni aziendali.

1.2.8 Devolopment

Il livello di *Development* supporta lo sviluppo e la gestione delle applicazioni attraverso processi automatizzati ed efficienti, garantendo scalabilità e sicurezza. Situato in una posizione chiave nell'architettura, facilita l'integrazione tra i vari livelli e ottimizza il ciclo di sviluppo e rilascio delle soluzioni.

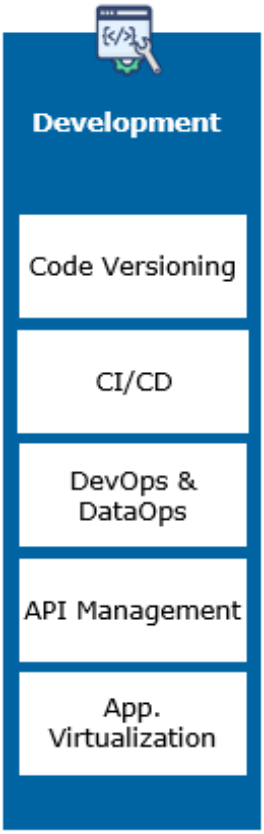


Figura 1.17: Devolopment

Come mostrato in Figura 1.17, questo livello fornisce una serie di strumenti essenziali per gestire in modo fluido e controllato i processi di sviluppo e distribuzione delle soluzioni.

Il **Code Versioning**, assicura una gestione avanzata delle versioni del codice in tutte le fasi del progetto. In stretta collaborazione con questo sistema, i tool di **Continuous Integration** (CI) e **Continuous Deployment** (CD) automatizzano e velocizzano i processi di rilascio, testando e integrando ogni modifica al codice in modo continuo, riducendo così gli errori e migliorando significativamente il time-to-market delle soluzioni.

Altrettanto importanti sono le metodologie **DevOps** e **DataOps**, che operano in sinergia con il versionamento e le pipeline di sviluppo. Mentre DevOps ottimizza il ciclo di vita del software, migliorando la collaborazione tra team e accelerando il rilascio delle applicazioni, DataOps estende tali principi ai processi analitici, garantendo monitoraggio e controllo approfonditi lungo tutto il ciclo di vita del dato, dalla fase di sperimentazione fino alla messa in produzione.

Un ulteriore pilastro del livello Development è l'**API Management**, che fornisce un'interfaccia standardizzata per gestire, esporre e controllare l'accesso alle API.

Infine, l'**Application Virtualization** semplifica e velocizza la distribuzione delle applicazioni tramite l'utilizzo di container, che assicurano una scalabilità trasparente, migliorano la sicurezza delle applicazioni distribuite e favoriscono una governance più solida a livello aziendale.

1.2.9 Data Science

La *Data Science* rappresenta uno dei livelli più innovativi e strategici dell'architettura, progettato per estrarre valore dai dati attraverso l'uso di algoritmi avanzati, modelli predittivi e soluzioni di intelligenza artificiale (AI) e machine learning (ML).

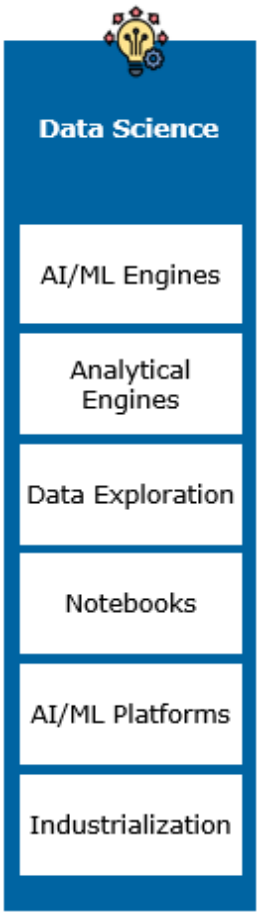


Figura 1.18: Data Science

Questo livello, come mostrato in Figura 1.18, integra una serie di strumenti e tecnologie pensate per coprire l'intero ciclo di vita dei modelli analitici e di intelligenza artificiale.

Alla base di questa architettura si trovano gli **AI/ML Engines**, motori che forniscono un'infrastruttura pronta all'uso per l'implementazione di algoritmi avanzati di machine learning e intelligenza artificiale.

Per supportare le analisi e l'esplorazione dei dati, gli **Analytical Engines** mettono a disposizione le capacità computazionali necessarie per gestire grandi volumi di informazioni e processi analitici complessi. Questi motori possono essere condivisi con altri livelli della Data Platform o dedicati esclusivamente agli ambienti di Data Science, assicurando così le prestazioni richieste per l'elaborazione avanzata dei dati.

Nella fase iniziale del processo di Data Science, la **Data Exploration** riveste un ruolo essenziale, permettendo di analizzare le caratteristiche dei dataset, individuare correlazioni tra variabili e riconoscere pattern significativi.

Un elemento chiave di questo livello è rappresentato dai **Notebooks**, ambienti web-based che combinano codice eseguibile con contenuti testuali, facilitando la prototipazione e l'analisi esplorativa. La loro semplicità d'uso e la possibilità di garantire la riproducibilità dei risultati li rendono strumenti fondamentali per Data Scientists e sviluppatori.

Le **AI/ML Platforms** accelerano ulteriormente il ciclo di sviluppo, offrendo un framework completo per il deployment e la gestione dei modelli di intelligenza artificiale. Grazie a soluzioni preconfigurate e scalabili, queste piattaforme semplificano l'intero processo, riducendo il time-to-market e permettendo alle aziende di implementare rapidamente soluzioni avanzate.

Infine, l'**Industrialization** rappresenta la fase conclusiva, in cui i modelli sviluppati vengono integrati nelle pipeline di produzione seguendo le best practice della Data Engineering. Questo passaggio assicura che le soluzioni siano scalabili, affidabili e facilmente manutenibili, garantendo alle organizzazioni un valore costante nel tempo dai propri modelli di AI e ML.

Capitolo 2

Stack Tecnologico

Come discusso nel capitolo precedente, le Data Platform moderne rappresentano un pilastro fondamentale per le organizzazioni che intendono sfruttare il potenziale dei dati in un contesto in continua evoluzione. La quantità di dati prodotti oggi cresce in modo esponenziale, richiedendo alle aziende non solo di archiviare e gestire informazioni, ma anche di ricavare valore attraverso processi avanzati di analisi e integrazione. In questo scenario, l'architettura tecnologica svolge un ruolo cruciale, consentendo di affrontare sfide legate alla scalabilità, alla flessibilità e alla complessità dei dati.

In questo capitolo, viene presentato lo stack tecnologico sviluppato per il progetto in collaborazione con il cliente. Tale stack non si limita a soddisfare le necessità di gestione e analisi dei dati, ma è stato concepito per rispondere a requisiti specifici del dominio aziendale, integrando i diversi livelli tecnologici necessari per costruire una soluzione robusta e scalabile. L'obiettivo è descrivere le componenti fondamentali di questa infrastruttura, mostrando come esse si integrino per supportare l'intero ciclo di vita dei dati, dalla raccolta iniziale fino al consumo finale.

L'analisi proposta pone particolare attenzione alle logiche architetturali e ai principi generali che hanno guidato la progettazione, offrendo una visione d'insieme delle tecnologie. Questo approccio permette di evidenziare come lo stack tecnologico sia stato personalizzato per rispondere alle esigenze del cliente, fornendo una piattaforma che combina efficienza operativa e flessibilità strategica.

2.1 Data Platform

Per affrontare le sfide legate alla gestione di grandi volumi di dati eterogenei e garantire flessibilità e scalabilità, è stata scelta una Cloud Data Platform. Questo approccio consente di astrarre le logiche tecniche alla base della piattaforma, offrendo agli utenti accesso diretto ai servizi senza doversi preoccupare delle complessità implementative.

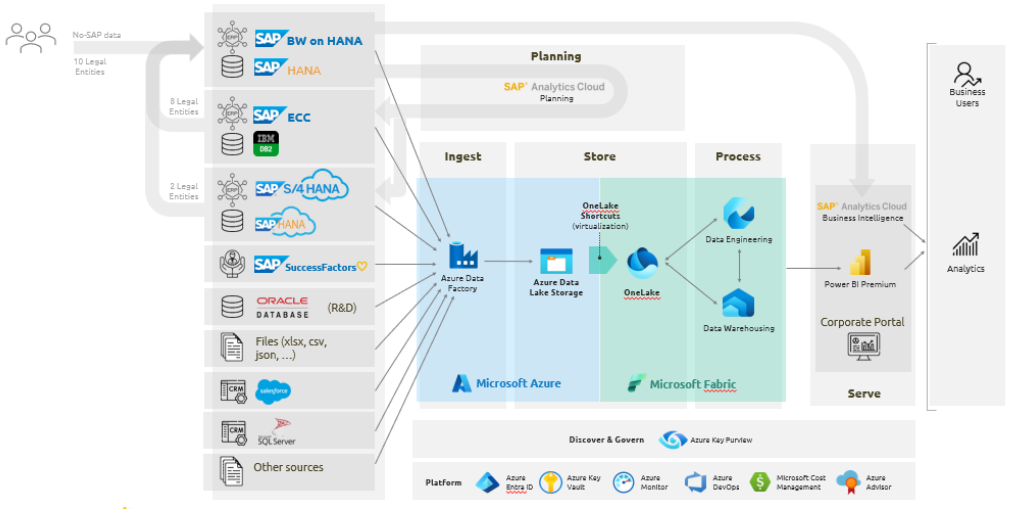


Figura 2.1: Cloud Data Platform

L'architettura della Cloud Data Platform, mostrata in figura 2.1, segue un'organizzazione standard, suddivisa in quattro fasi principali:

- **Ingestion**: questa fase si occupa dell'acquisizione dei dati da sorgenti eterogenee, che possono includere database aziendali, file di log, applicazioni SaaS e dispositivi IoT. I dati vengono raccolti tramite pipeline che operano sia in modalità batch che in tempo reale, garantendo la costante alimentazione della piattaforma.

- **Storage**: una volta acquisiti, i dati vengono archiviati in un sistema centralizzato che garantisce accessibilità, durabilità e sicurezza. Questo livello consente di conservare i dati sia nel loro formato originale (raw data) sia in versioni trasformate e organizzate, facilitando analisi ed elaborazioni successive.

- **Processing**: in questa fase, i dati vengono elaborati e trasformati per renderli utili e pronti per l'analisi. Le operazioni comprendono la pu-

lizia e l'arricchimento dei dati, l'aggregazione per costruire set analitici più specifici e l'applicazione di logiche avanzate per supportare analisi descrittive, diagnostiche o predittive.

- **Serve**: infine, i dati elaborati vengono resi accessibili attraverso strumenti di visualizzazione, reportistica o API. Questa fase consente agli utenti finali di consumare le informazioni in modo interattivo, tramite dashboard o applicazioni integrate.

I Cloud provider utilizzati per la realizzazione del progetto sono *Microsoft Azure* e *Microsoft Fabric*, che offrono un'ampia gamma di strumenti per la gestione e l'analisi dei dati. Di seguito ne vengono descritte le principali caratteristiche.

2.1.1 Microsoft Azure

Microsoft Azure [15] è una piattaforma di cloud computing che fornisce una vasta gamma di servizi per supportare organizzazioni di ogni dimensione nella gestione delle loro applicazioni e infrastrutture. La sua architettura è stata progettata per garantire scalabilità, resilienza e flessibilità, rendendola una delle piattaforme cloud più utilizzate al mondo.

L'architettura di Azure si basa sulla virtualizzazione, che consente di astrarre le risorse fisiche, come server e storage, in unità virtuali che gli utenti possono gestire facilmente. Questo approccio permette alle aziende di configurare ambienti personalizzati, adattandoli dinamicamente alle loro esigenze. Ad esempio, un'organizzazione può implementare un cluster di macchine virtuali per un carico di lavoro temporaneo e poi ridimensionarlo una volta terminata l'elaborazione.

La piattaforma offre una vasta gamma di servizi, tra i più noti vi sono le *Azure Virtual Machines*, che permettono di eseguire applicazioni su sistemi operativi Windows o Linux, e il *Blob Storage*, ideale per archiviare grandi quantità di dati non strutturati. I servizi di rete includono le reti virtuali e i gateway VPN, progettati per garantire una connessione sicura tra risorse on-premise e risorse nel cloud. Inoltre, Azure offre strumenti avanzati per il machine learning e l'intelligenza artificiale, consentendo alle organizzazioni di sviluppare modelli predittivi e soluzioni innovative.

Azure supporta tre principali modelli di servizio: IaaS (Infrastructure as a Service), che offre infrastruttura virtualizzata; PaaS (Platform as a Service),

che fornisce piattaforme preconfigurate per lo sviluppo; e SaaS (Software as a Service), che consente l'uso diretto di applicazioni cloud.

La sicurezza viene garantita da strumenti come *Azure Security Center*, che monitora e protegge applicazioni e dati, e *Azure Active Directory*, che gestisce in modo sicuro ed efficiente le identità degli utenti e l'accesso alle risorse.

Grazie alla sua architettura innovativa, Microsoft Azure si distingue per la capacità di supportare scenari complessi, garantendo affidabilità e prestazioni elevate. La piattaforma rappresenta una soluzione versatile per organizzazioni che desiderano sfruttare il cloud per innovare e ottimizzare i propri processi.

2.1.2 Microsoft Fabric

Microsoft Fabric [16] è una piattaforma dati e analisi end-to-end progettata per rispondere alle esigenze aziendali moderne, dove la gestione dei dati e l'analisi giocano un ruolo cruciale. Questa soluzione unificata elimina la necessità di integrare servizi provenienti da fornitori diversi, offrendo un approccio semplificato e coeso che migliora la produttività e l'efficienza.

Uno degli aspetti distintivi di Microsoft Fabric è la sua capacità di integrare molteplici funzionalità in un unico ecosistema, eliminando la frammentazione tipica delle soluzioni tradizionali. Fabric, come mostrato in Figura 2.2, include i seguenti componenti principali:

- **Data Factory**: strumento per la movimentazione e trasformazione dei dati.
- **Data Engineering**: soluzione per l'elaborazione e l'ottimizzazione dei dati.
- **Data Warehouse**: archiviazione strutturata per l'analisi su larga scala.
- **Data Science**: funzionalità avanzate per lo sviluppo di modelli di intelligenza artificiale.
- **Real-Time Intelligence**: strumenti per il routing degli eventi in tempo reale.
- **Power BI**: creazione di report interattivi e dashboard.

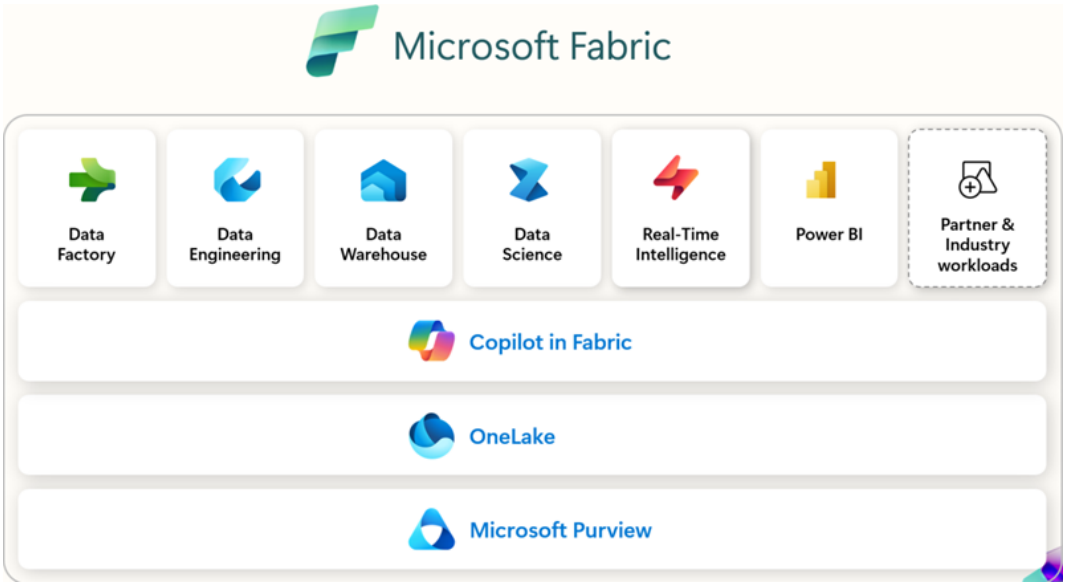


Figura 2.2: Architettura Microsoft Fabric [16]

Grazie al modello SaaS (Software as a Service), Fabric consente di accedere a una piattaforma integrata con un'interfaccia intuitiva e funzionalità avanzate, semplificando le operazioni analitiche e riducendo la complessità tecnica. Fabric introduce *OneLake*, un data lake unificato che centralizza l'archiviazione dei dati, permettendo una gestione semplificata e una condivisione efficace delle informazioni. Questa soluzione elimina la necessità di utilizzare database separati, garantendo una transizione fluida dai dati grezzi a insight utili.

Un altro punto di forza di Fabric è l'integrazione nativa di funzionalità di intelligenza artificiale. Grazie al supporto di *Copilot* e di *Azure AI Foundry*, gli utenti possono sfruttare strumenti avanzati per automatizzare attività e sviluppare modelli AI senza richiedere complessi processi di integrazione manuale.

La piattaforma garantisce una gestione centralizzata dei dati grazie all'integrazione con *Purview*, che applica automaticamente politiche di sicurezza e autorizzazioni in tutti i servizi, assicurando una governance robusta. Fabric non si limita a fornire strumenti per l'analisi, ma accelera l'intero ciclo di vita dei dati. Lo stack infrastrutturale integrato con intelligenza artificiale facilita il percorso dai dati grezzi alle analisi avanzate, offrendo vantaggi come:

- **Analisi integrata e completa**: consente di ottenere insight dettagliati dai dati senza dover utilizzare strumenti separati.

- **Accesso semplice e riutilizzo degli asset**: gli asset, come report e modelli di dati, possono essere facilmente condivisi e riutilizzati all'interno dell'organizzazione.

- **Archiviazione unificata dei dati nel loro stato originale**: i dati rimangono centralizzati e accessibili senza duplicazioni o spostamenti inutili.

2.2 Ingestion

L'ingestion rappresenta una delle fasi chiave all'interno di una moderna Data Platform. Questo processo si occupa di acquisire i dati da diverse fonti eterogenee, come database relazionali, applicazioni SaaS, file di log e dispositivi IoT, per renderli disponibili per successive elaborazioni e analisi.

In questa fase è risultato fondamentale l'utilizzo di *Azure Data Factory*, uno strumento di *Microsoft Azure* dedicato alla gestione dell'acquisizione dei dati. Di seguito, ne vengono descritte le principali caratteristiche.

2.2.1 Azure Data Factory

Azure Data Factory (ADF) [9] è un servizio cloud-based di ETL e integrazione che permette di creare dei flussi data-driven per orchestrare lo spostamento e la trasformazione dei dati su larga scala. ADF consente di gestire il trasferimento e la trasformazione dei dati tra diversi archivi e risorse di calcolo. È possibile progettare e pianificare workflows, chiamati Pipelines, che permettono di acquisire dati da una vasta gamma di sorgenti e trasformarli in un formato più facilmente processabile.

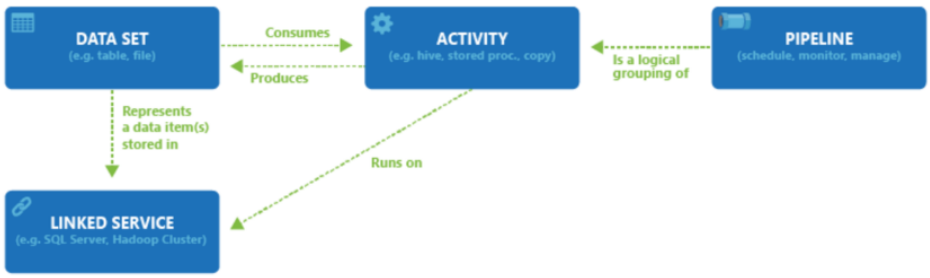


Figura 2.3: Componenti Data Factory [9]

La Figura 2.3 illustra le principali componenti di ADF, che sono descritte più nel dettaglio di seguito:

- **Pipeline**: rappresentano un raggruppamento di attività che insieme svolgono un task specifico. Le attività all'interno di una pipeline possono essere eseguite in sequenza o in parallelo, a seconda delle necessità.
- **Activity**: identifica i singoli passi computazionali che vengono eseguiti all'interno di una Pipeline. ADF supporta tre tipi principali di attività: attività di movimentazione dei dati (data movement), attività di trasformazione dei dati (data transformation) e attività di controllo (control activities).
- **Dataset**: rappresentano le strutture dati all'interno dei data store e fanno riferimento ai dati che si desidera utilizzare come input o output nelle attività della pipeline.
- **Linked Service**: sono simili alle connection strings e definiscono le informazioni di connessione necessarie affinché Data Factory possa accedere a risorse esterne.
- **Integration Runtime** (IR): funge da bridge tra le Activities e i Linked Services. L'IR è l'ambiente di esecuzione utilizzato per eseguire le Activities.
- **Trigger**: determina quando una Pipeline deve essere eseguita. I Trigger possono essere schedulati, periodici o attivati da eventi specifici.

ADF viene tipicamente impiegato per task di ETL/ELT, Data Integration, Data Analytics, Ingestion di dati verso un Data Warehouse e sincronizzazioni periodiche.

2.3 Storage

I sistemi tradizionali di archiviazione dei dati non sono sempre in grado di gestire efficacemente la quantità e la varietà di dati generati oggi. Per rispondere a queste esigenze, è necessario adottare tecnologie avanzate che consentano una gestione scalabile e flessibile dei dati. In questo contesto, risultano fondamentali soluzioni come *Azure Data Lake Storage* di Microsoft Azure e *OneLake* di

Microsoft Fabric. Questi strumenti, concepiti per l'archiviazione e la gestione di grandi volumi di dati eterogenei, saranno descritti nei paragrafi successivi, evidenziandone le principali caratteristiche e i benefici nell'ambito delle moderne Data Platform.

2.3.1 Azure Data Lake Storage

Azure Data Lake Storage (ADLS) [10] è una soluzione innovativa e completa offerta da Microsoft, progettata per soddisfare le esigenze di archiviazione e analisi di dati su larga scala. Questa piattaforma combina le funzionalità di un file system avanzato con una solida infrastruttura di archiviazione, offrendo alle organizzazioni una base flessibile e performante per gestire dati eterogenei. ADLS Gen2 rappresenta un'evoluzione significativa, basandosi sulle funzionalità di archiviazione BLOB di Azure e ottimizzandola per carichi di lavoro di analisi intensivi. Grazie a questa integrazione, gli utenti possono beneficiare di prestazioni elevate, capacità di gestione del ciclo di vita dei dati e una sicurezza avanzata, il tutto supportato dalla robustezza dell'infrastruttura Azure.

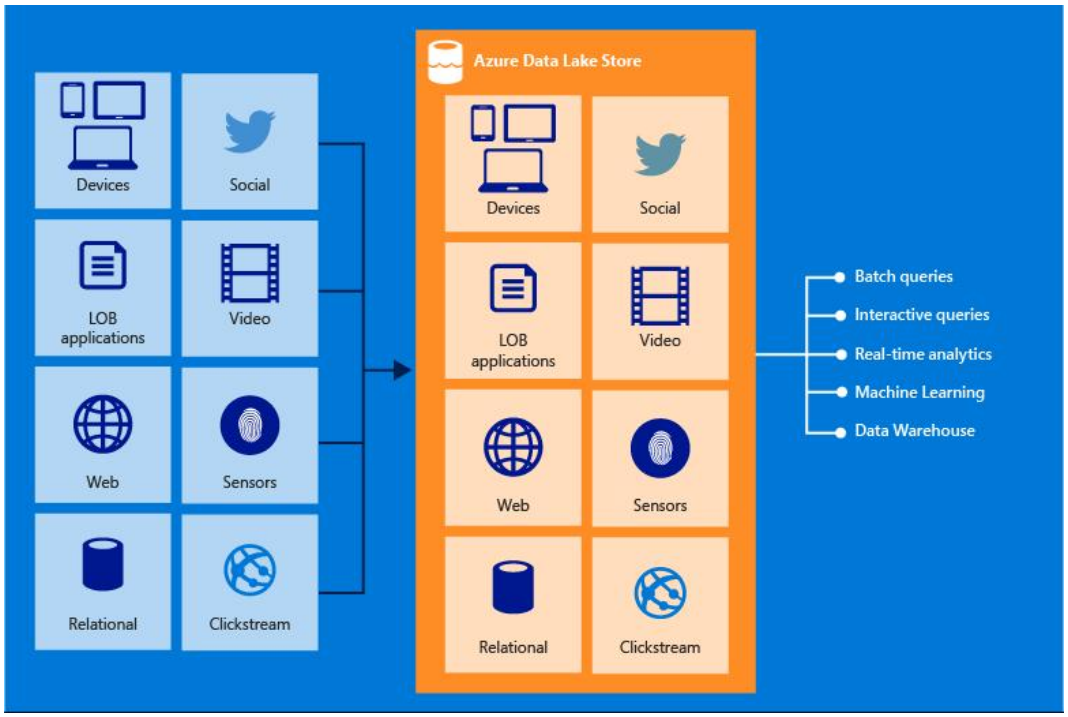


Figura 2.4: Azure Data Lake Storage

ADLS, come mostrato in Figura 2.4, è progettato per gestire dati di qualsiasi dimensione, dai piccoli file fino agli exabyte, offrendo supporto sia per

soluzioni in tempo reale che per processi batch. Una delle sue caratteristiche chiave è la compatibilità con *Hadoop Distributed File System* (HDFS), un file system scalabile e distribuito progettato per archiviare e gestire grandi volumi di dati su cluster di macchine. Questa compatibilità consente di sfruttare l'efficienza e la capacità di HDFS, permettendo un accesso diretto e ottimizzato ai dati archiviati. Inoltre, l'utilizzo di formati ottimizzati come Parquet garantisce compressione ed efficienza, migliorando la gestione e l'elaborazione dei dati.

La sicurezza è garantita da un sistema di autorizzazioni granulari basato su standard POSIX e ACL, che consente di definire con precisione i permessi di accesso ai dati, sia a livello di directory che di singolo file. Inoltre, i dati vengono crittografati utilizzando chiavi di cifratura che possono essere gestite direttamente dal cliente o da Microsoft, garantendo un elevato livello di protezione contro accessi non autorizzati.

L'organizzazione dei dati in una gerarchia di directory simile a un file system semplifica l'accesso e riduce il consumo di risorse computazionali, abbassando tempi e costi. La replica dei dati tramite Archiviazione BLOB garantisce ridondanza locale o geografica, assicurando continuità operativa anche in situazioni di emergenza.

2.3.2 Fabric OneLake

Fabric OneLake [14] è una piattaforma unificata e logica per la gestione dei dati analitici di un'intera organizzazione. Analogamente a OneDrive per i documenti, OneLake è progettato per essere il punto centrale per tutti i dati analitici, eliminando la necessità di creare e gestire più data lake separati per diversi gruppi aziendali. OneLake semplifica la collaborazione, superando le complessità legate alla gestione di più data lake, e migliora l'efficienza operativa riducendo l'overhead amministrativo. Inoltre, i dati possono essere utilizzati da più motori analitici senza la necessità di copie o spostamenti, eliminando i silos di dati e consentendo analisi più rapide e integrate.

La piattaforma introduce un modello di governance predefinito che permette di collaborare in sicurezza tra diverse parti dell'organizzazione, evitando colli di bottiglia amministrativi. La Figura 2.5 illustra come OneLake consenta l'organizzazione dei dati in diversi workspaces, ognuno dei quali è destinato a specifici ambiti operativi, mantenendo una sicurezza e governance unificate.

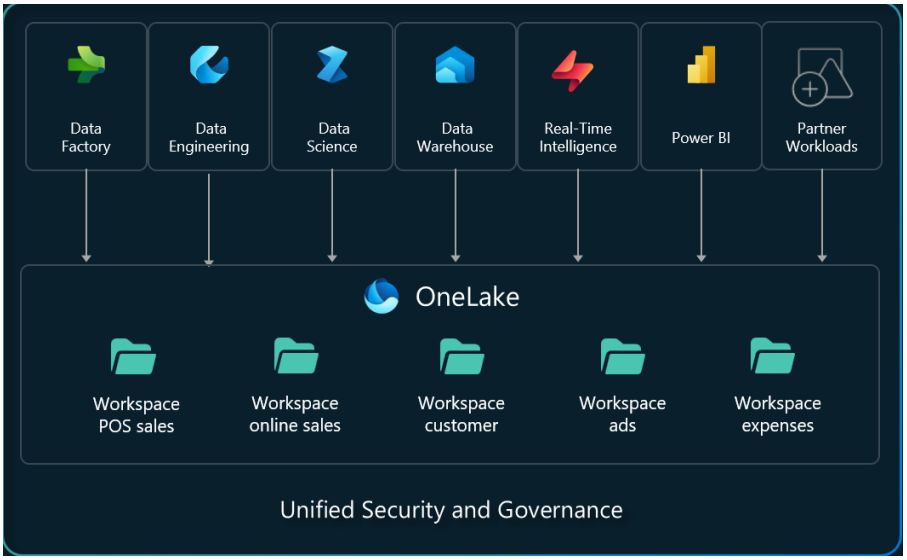


Figura 2.5: Organizzazione Workspaces Fabric One Lake [14]

Fabric OneLake è costruito su *Azure Data Lake Storage Gen2*, garantendo supporto per qualsiasi tipo di file, sia strutturato che non strutturato. Tutti i dati analitici, inclusi quelli provenienti da lakehouse e warehouse, sono archiviati in formato Delta Parquet, un formato aperto che consente la massima interoperabilità tra diversi strumenti e motori analitici. La Figura 2.6 illustra come ogni workspace rappresenta un'unità organizzativa distinta, che raccoglie dati strutturati in sottodirectory per garantire un'integrazione efficiente e un accesso semplificato.

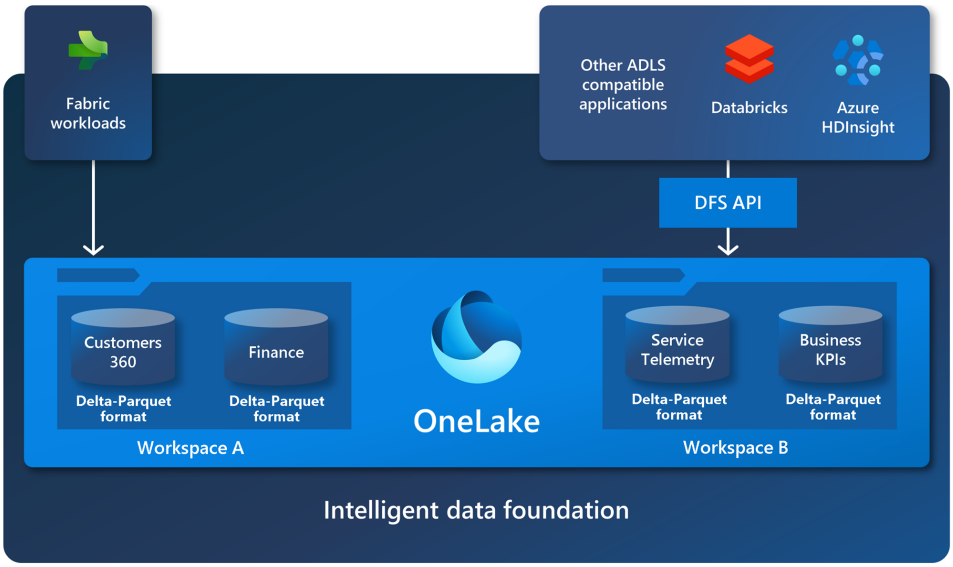


Figura 2.6: Architettura dei workspaces in Fabric OneLake [14]

Una funzionalità distintiva di OneLake è rappresentata dagli *shortcuts*, che permettono di condividere dati tra diversi domini e gruppi aziendali senza spostamenti o duplicazioni. Gli shortcuts creano riferimenti a dati archiviati in altre posizioni, sia all'interno di OneLake che su piattaforme esterne come ADLS, S3 o Dataverse. Questo approccio semplifica la collaborazione e consente di combinare dati da diverse fonti in un unico prodotto virtuale. Fabric OneLake è progettato per massimizzare l'utilizzo di una singola copia dei dati con più motori analitici.

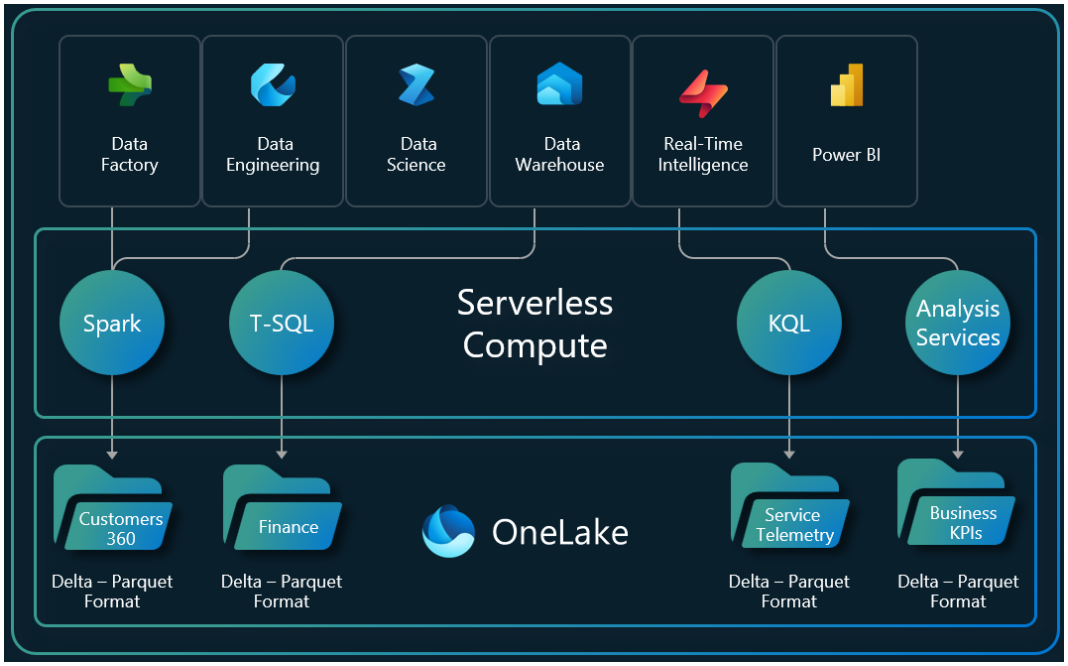


Figura 2.7: Esempio che mostra il caricamento dei dati in Fabric OneLake [14]

Grazie al formato Delta Parquet, i dati possono essere utilizzati simultaneamente da motori diversi, come T-SQL, Apache Spark o Analysis Services, senza necessità di conversioni o copie aggiuntive. Come mostrato nella Figura 2.7, questa architettura consente a ogni team di scegliere il motore più adatto alle proprie esigenze, migliorando l'efficienza dei processi analitici. OneLake offre un'esperienza intuitiva grazie al file explorer per Windows, che consente di navigare, caricare e gestire i dati con la stessa semplicità di OneDrive.

2.4 Processing

Questa fase include tutti gli strumenti che consentono di trasformare e ottimizzare i dati, garantendo che siano pronti per essere interrogati e analizzati in modo efficace. Attraverso operazioni come la pulizia, l'arricchimento e l'aggregazione, i dati vengono preparati per supportare analisi descrittive, diagnostiche e predittive. Per realizzare queste operazioni in modo efficiente, si utilizzano strumenti di Fabric, come *Data Engineering* e *Data Warehousing*, progettati per gestire flussi complessi e creare set analitici pronti per il consumo. Nel seguito di questo sottocapitolo, verranno esaminate le loro caratteristiche mettendo in evidenza il ruolo che ricoprono nell'intero processo.

2.4.1 Fabric Data Engineering

Il *Data Engineering* [12] consente di progettare, costruire e mantenere infrastrutture e sistemi che permettono alle organizzazioni di raccogliere, archiviare, elaborare e analizzare grandi volumi di dati.

La Figura 2.8 illustra le componenti principali della piattaforma di Data Engineering, che consentono di gestire e trasformare i dati in modo efficiente e scalabile.

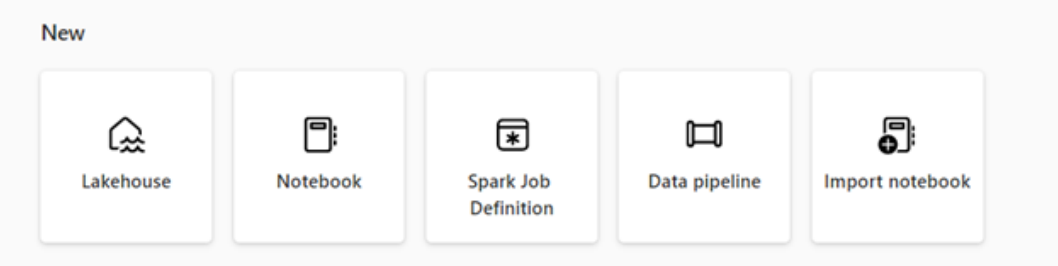


Figura 2.8: Componenti del Data Engineering [12]

- **Lakehouse**: sono architetture che consentono alle organizzazioni di conservare e gestire dati strutturati e non strutturati in modo centralizzato, utilizzando vari strumenti e framework per processare e analizzare tali dati. Questi strumenti e framework possono includere query e analisi basate su SQL, nonché tecniche avanzate di machine learning e altre forme di analisi.

- **Data Pipeline**: sono una sequenza di attività che raccolgono, elaborano e trasformano i dati dalla loro forma grezza a un formato utilizzabile per

l'analisi e il processo decisionale. Costituiscono una componente fondamentale del data engineering, poiché offrono un modo affidabile, scalabile ed efficiente per spostare i dati dalla loro sorgente alla destinazione finale.

- **Apache Spark Job Definition**: rappresentano un insieme di istruzioni che definiscono come eseguire un job su un cluster Spark. Includono informazioni come le sorgenti dati di input e output, le trasformazioni da applicare e le impostazioni di configurazione per l'applicazione Spark. Questi job permettono di inviare elaborazioni batch o streaming a un cluster Spark, consentendo l'applicazione di logiche di trasformazione avanzate ai dati presenti nel Lakehouse. Gli Spark job si dimostrano particolarmente efficaci per gestire grandi volumi di dati e supportare analisi complesse in modo scalabile e distribuito.

- **Notebook**: sono strumenti potenti e flessibili che offrono un ambiente computazionale, progettato per integrare codice eseguibile, visualizzazioni, equazioni e testo descrittivo all'interno dello stesso spazio di lavoro. Questo approccio consente agli utenti di scrivere, eseguire e condividere codice in linguaggi come Python, R e Scala in modo semplice e dinamico. Grazie alla loro natura interattiva, i Notebook rappresentano una soluzione ideale per l'ingestion, la preparazione, l'analisi e molte altre attività legate alla gestione dei dati.

2.4.2 Fabric Data Warehousing

Il *Data Warehousing* [13] è uno strumento di Microsoft Fabric particolarmente efficace nel supportare l'archiviazione e il trattamento dei dati in modo scalabile. Uno degli aspetti più rilevanti è la sua integrazione con Power BI, che consente di analizzare i dati e creare report aggiornati in tempo reale, semplificando le attività analitiche. Fabric Data Warehouse non è un data warehouse aziendale tradizionale; si tratta di un lake warehouse che supporta due principali elementi di warehousing: il Data Warehouse di Fabric e l'endpoint di analisi SQL.

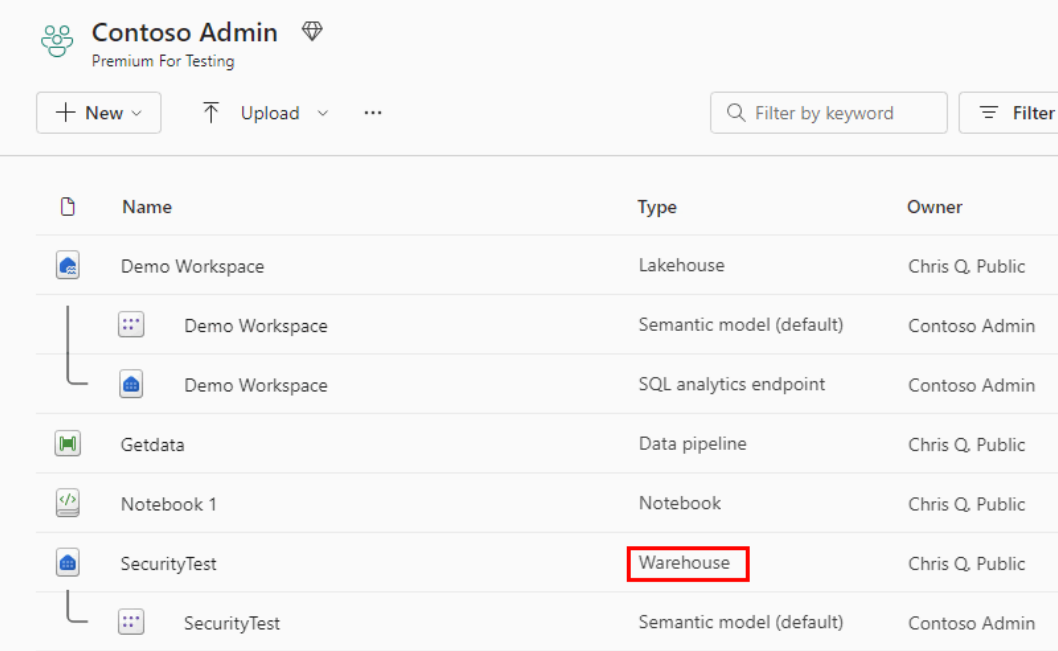


Figura 2.9: Data Warehouse di Fabric [13]

La Figura 2.9 illustra come il *Data Warehouse* di Fabric sia etichettato come "Warehouse" nella colonna Tipo all'interno del workspace di Fabric. Quando è necessaria la piena potenza transazionale, con il supporto a query DDL e DML, questa rappresenta la soluzione più semplice e rapida. Il popolamento del warehouse può avvenire tramite diversi metodi di inserimento dati, come COPY INTO (T-SQL), Pipelines oppure opzioni di inserimento tra database come CREATE TABLE AS SELECT (CTAS), INSERT..SELECT o SELECT INTO.

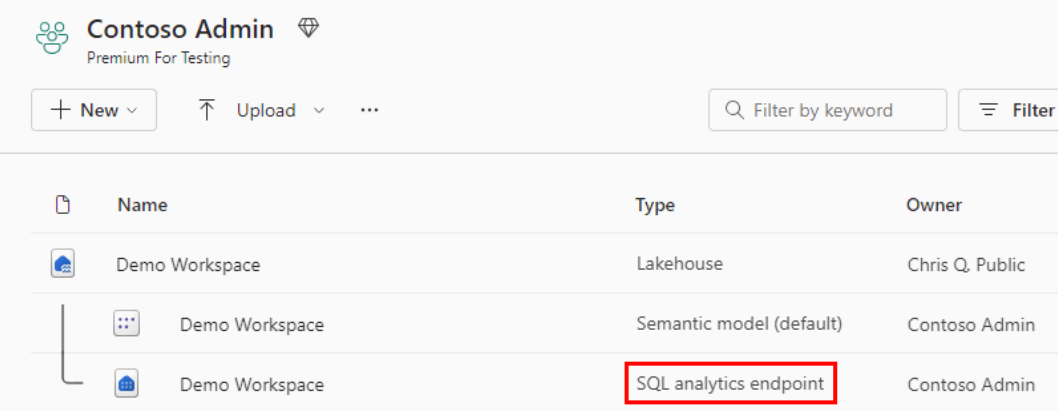


Figura 2.10: Endpoint di analisi SQL del Lakehouse [13]

L'*endpoint di analisi SQL* è una funzionalità generata automaticamente per ogni Lakehouse all'interno del workspace. Esso consente di passare dalla vista "Lake" alla vista "SQL" dello stesso Lakehouse, offrendo la possibilità di creare viste, funzioni, stored procedure e applicare politiche di sicurezza SQL (Figura 2.10). Con questo strumento, i comandi T-SQL possono definire ed eseguire query sugli oggetti dati senza modificare direttamente il contenuto.

Il Data Warehousing offre numerosi vantaggi tra cui:

- **Supporto al formato Delta-parquet**: i dati sono memorizzati in formato Delta-Parquet, garantendo transazioni ACID. Grazie a questa struttura, il Fabric Data Warehouse consente di evitare la duplicazione dei dati, offrendo interoperabilità con altri carichi di lavoro presenti nell'ambiente Fabric.

- **Query cross-database**: il sistema consente di eseguire interrogazioni su più fonti di dati senza duplicazione, accelerando il processo decisionale.

- **Ingestion e trasformazione dei dati**: il sistema permette di gestire facilmente dati su larga scala, utilizzando strumenti come pipeline, flussi di dati, query tra database o comandi avanzati come COPY INTO.

- **Gestione autonoma dei carichi di lavoro**: Il motore di query distribuito elimina la necessità di configurazioni manuali, offrendo prestazioni ottimali in qualsiasi condizione.

- **Scalabilità immediata**: la separazione tra archiviazione e calcolo consente di scalare le risorse in modo quasi istantaneo, adattandosi alle esigenze aziendali.

- **Integrazione semantica con Power BI**: attraverso la modalità Direct Lake, i report sono sempre aggiornati e pronti per essere analizzati, riducendo i tempi necessari per ottenere insight.

2.5 Serve

Questa fase è dedicata alla distribuzione e fruizione dei dati elaborati, con l'obiettivo di fornire agli utenti strumenti efficaci per analizzare e interpretare le informazioni. Attraverso questo livello, i dati vengono trasformati in insight fruibili per supportare decisioni strategiche e operative.

In questa fase, fondamentale è stato l'utilizzo di uno strumento come *Microsoft Power BI*. Questo strumento consente di creare report interattivi e dashboard, offrendo un'interfaccia intuitiva che rende i dati immediatamente accessibili e comprensibili. Nel corso di questo capitolo, verranno esplorate le sue funzionalità principali.

2.5.1 Power BI

Power BI [17] è uno strumento essenziale per l'analisi e la visualizzazione dei dati, che si distingue per la sua capacità di trasformare i dati in informazioni dettagliate, interattive e visivamente accattivanti.

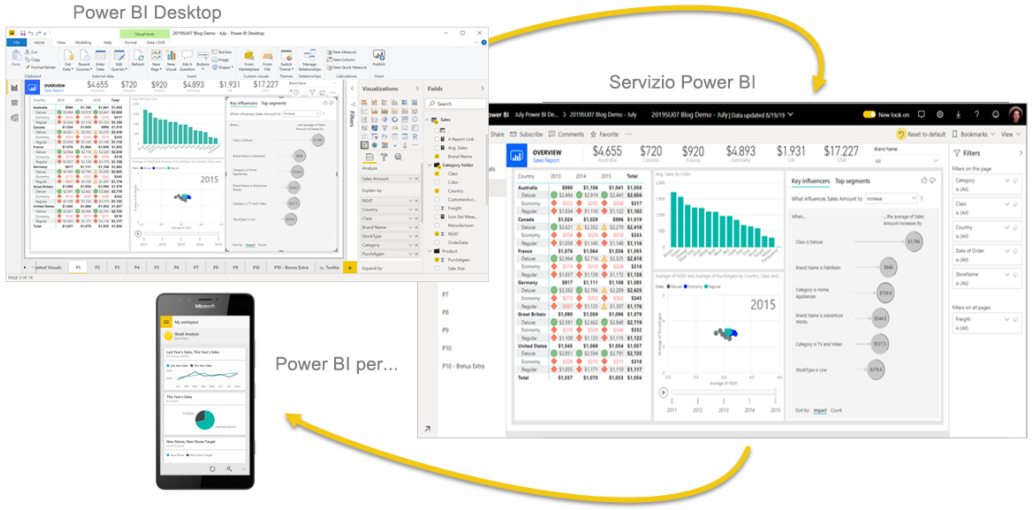


Figura 2.11: Componenti di Power BI [13]

Power BI, come mostrato in Figura 2.11, si compone di tre elementi principali che lavorano in sinergia per supportare i diversi ruoli aziendali e semplificare i flussi di lavoro:

- **Power BI Desktop**: utilizzato per creare report complessi e modelli di dati, fornendo uno strumento avanzato per la progettazione e l'elaborazione delle informazioni.

- **Power BI Service**: una soluzione SaaS che consente di condividere e collaborare sui contenuti, semplificando la distribuzione e l'accesso ai report all'interno dell'organizzazione.

- **Power BI Mobile**: fornisce accesso in tempo reale ai dati da qualsiasi luogo, garantendo la massima flessibilità per consultare report e dashboard su dispositivi Windows, iOS e Android.

Un elemento fondamentale che amplia la versatilità di Power BI è il *Server di report*, una soluzione progettata per l'utilizzo in ambienti locali. Questo server permette di pubblicare e distribuire report creati in Power BI Desktop dietro il firewall aziendale, garantendo un controllo completo sulla sicurezza dei dati. I report possono essere visualizzati in diversi modi, come browser Web, dispositivi mobili o tramite messaggi di posta elettronica.

Il flusso di lavoro tipico di Power BI inizia con la connessione alle origini dati tramite Power BI Desktop, dove si progettano i report. Successivamente, i report vengono pubblicati nel servizio Power BI, da cui possono essere distribuiti agli utenti finali.

Un ulteriore strumento messo a disposizione è *Power BI Report Builder*, utilizzato per la creazione di report impaginati da condividere nel servizio Power BI. I report impaginati sono progettati per essere stampati o condivisi, ideali per operazioni come la generazione di fatture o la gestione di trascrizioni, come mostrato in Figura 2.12.

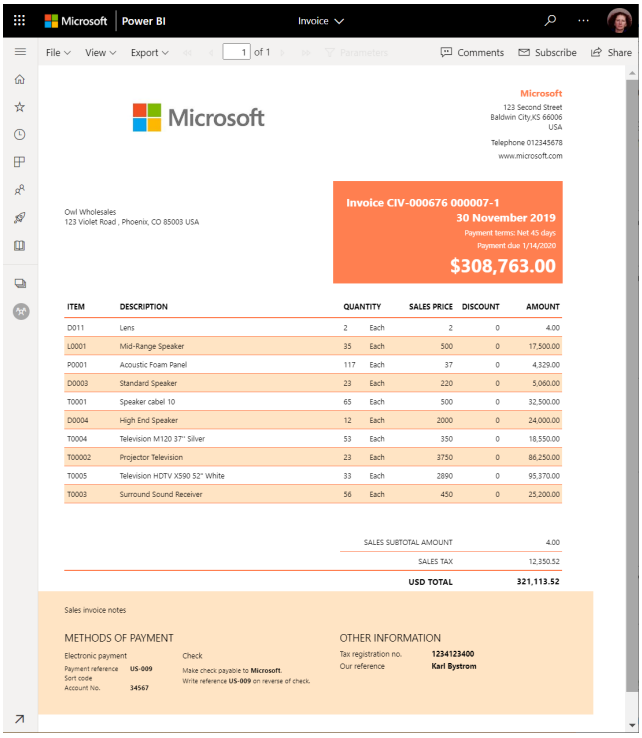


Figura 2.12: Report impaginati [13]

Un'altra funzionalità chiave è la *pipeline di distribuzione*, che permette di testare contenuti come report e dashboard prima del rilascio agli utenti. Questo strumento garantisce un controllo di qualità e facilita la gestione delle versioni, riducendo i rischi di errori nei dati pubblicati.

Infine, Power BI si integra perfettamente con Microsoft Fabric, sfruttando l'archivio dati OneLake per unificare le analisi su diverse fonti di dati. Questa sinergia permette di gestire e analizzare grandi volumi di dati in modo efficiente. Gli utenti possono anche sfruttare le API di Power BI per incorporare report e dashboard in applicazioni personalizzate, rendendolo uno strumento estremamente flessibile.

Capitolo 3

Sviluppo Use Case - CRM

L'obiettivo principale del progetto è guidare l'azienda cliente verso un approccio sempre più data-driven, mettendo i dati al centro dei processi decisionali. Questo implica il coinvolgimento delle diverse aree di business e la valorizzazione dei sistemi di reportistica e analytics esistenti, evolvendoli in una Data Platform di nuova generazione.

Nel contesto attuale, i dati aziendali devono essere utilizzabili non solo per supportare analisi interne, ma anche per alimentare processi esterni, rendendo essenziale la disponibilità di un'unica versione certificata e condivisa del dato, accessibile in modo interoperabile da tutti i reparti aziendali. Per rispondere a questa esigenza, la nuova Data Platform è progettata per superare i limiti delle architetture tradizionali, abilitare nuove capability e supportare l'implementazione di use case innovativi.

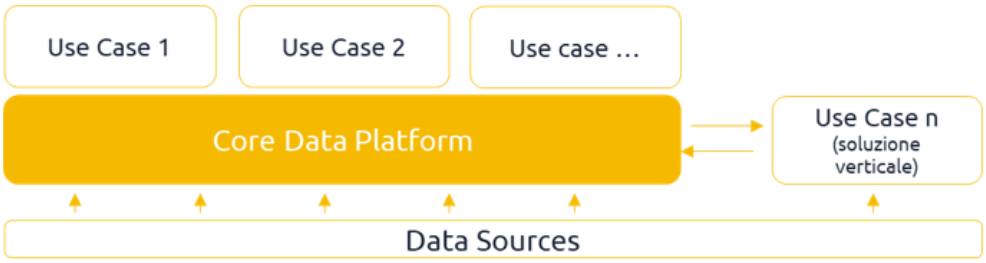


Figura 3.1: Rappresentazione schematica della Data Platform come Data Hub

Questa piattaforma fungerà da Data Hub centrale, come illustrato in Figura 3.1, integrando dati eterogenei in un ecosistema connesso e interoperabile. La gestione centralizzata dei sistemi favorirà analisi trasversali a livello azien-

dale, ottimizzando l'efficienza operativa e migliorando la qualità delle decisioni strategiche.

Il progetto di tesi si inserisce nello sviluppo dello Use Case relativo al CRM, che rientra nel contesto dell'area aziendale di *Global Marketing* del Cliente. Lo Use Case ha l'obiettivo di creare un modello aziendale unificato per il **Sales Excellence**, ottimizzando i processi di vendita attraverso il monitoraggio e l'analisi di dati chiave. Utilizzando i dati provenienti dalle diverse istanze locali del CRM, questi vengono acquisiti, armonizzati e consolidati in un modello analitico centralizzato. Questo approccio consente di definire un set iniziale di KPI aziendali fondamentali, fornendo le basi per ulteriori analisi e miglioramenti futuri.

In seguito ad un'approfondita conoscenza del patrimonio informativo dell'area aziendale in questione, il team di Iconsulting ha progettato un modello dati unificato, indispensabile per garantire l'integrazione e la coerenza dei dati provenienti da fonti diverse.

Il mio contributo si è focalizzato inizialmente nello studio e nell'analisi del modello dati, con l'obiettivo di comprendere la struttura esistente, identificare le dimensioni e i fatti di analisi rilevanti. Successivamente le mie attività si sono focalizzate sull'implementazione di un processo che permettesse di acquisire, armonizzare e consolidare i dati, seguendo il modello dati definito dal team. Grazie a un approccio strutturato, è stato possibile trasformare i dati grezzi in informazioni strutturate, garantendo un flusso dati coerente e scalabile.

Le attività svolte sono sintetizzabili nelle seguenti fasi:

1. **Analisi**: studio del modello dati definito dal team per il seguente Use Case.

2. **Data Ingestion**: acquisizione dei dati dalle istanze locali del CRM, garantendo un'integrazione fluida e sicura.

3. **Data Preparation**: pulizia e trasformazione dei dati per garantirne coerenza e integrità.

4. **Predisposizione flusso CRM**: sviluppo di un flusso automatizzato per gestire in modo efficiente i processi e le attività descritte nella fase di Data Ingestion e Data Preparation.

5. **Data Consumption**: predisposizione dei dati per le analisi, rendendoli disponibili per la definizione e il monitoraggio dei KPI aziendali.

Nei paragrafi successivi verranno illustrate nel dettaglio le attività svolte in ciascuna fase, evidenziando il contributo specifico del lavoro di tesi nello sviluppo di questo Use Case.

3.1 Analisi

La fase di analisi si è concentrata sullo studio e sulla comprensione del modello dati definito dal team di Iconsulting per lo Use Case relativo al CRM. Questo modello costituisce il pilastro dell'intero progetto, assicurando l'allineamento e l'uniformità dei dati provenienti dalle diverse fonti aziendali. Il suo principale punto di forza è la capacità di supportare analisi su più livelli: sia a livello locale, paese per paese, che su scala globale, grazie all'armonizzazione delle informazioni raccolte dalle varie istanze del CRM. Questo approccio è cruciale per identificare dimensioni e fatti rilevanti, necessari al monitoraggio dei KPI aziendali e alla realizzazione di analisi strategiche ed operative.

Successivamente, verrà mostrato il modello logico progettato dal team di Iconsulting (Figura 3.2), seguito dalla descrizione delle dimensioni principali e dei fatti rilevanti.

3.1.1 Modello Dati

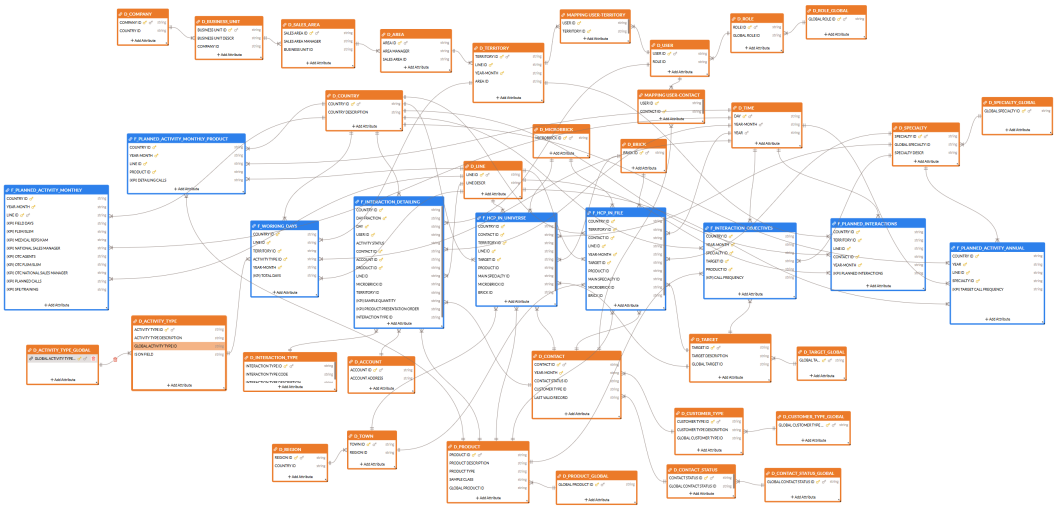


Figura 3.2: Modello Dati Logico sviluppato da Iconsulting

Di seguito sono descritte le principali dimensioni:

- Country: questa dimensione rappresenta le nazioni dalle quali provengono i dati del CRM. Le country sono identificate attraverso il codice ISO della nazione e derivano dai master data del sistema BW. Questa dimensione permette di segmentare i dati per area geografica di alto livello.

- Time: fornisce il contesto temporale necessario per analizzare i dati su base giornaliera, mensile o annuale.

- Line: identifica la linea di vendita di riferimento, come ad esempio Pharma, Reumatologia o Gastroenterologia. È essenziale per segmentare le attività in base alla specializzazione dell'area medica.

- Territory: una dimensione cruciale che definisce il territorio di competenza dei *sales rep*, ovvero i rappresentanti che si occupano delle vendite. Non rappresenta confini geografici, bensì aree di responsabilità definite dal sistema CRM stesso. Questa dimensione determina la visibilità degli utenti sui medici (Contact) e sugli enti (Account).

- Area, Sales Area, Business Unit e Company: dimensioni che definiscono la gerarchia del territorio, ognuna di queste rappresenta una determinata area di responsabilità. La Company rappresenta le principali sedi aziendali del gruppo, ciascuna delle quali include diverse Business Unit. Queste, a loro volta, sono suddivise in più Sales Area, che definiscono specifiche zone commerciali. All'interno di ogni Sales Area si trovano diverse aree, ognuna delle quali è composta da più territori distinti di resposabilità dei sales rep.

- User: identifica i sales rep, ossia i rappresentanti responsabili della promozione e sponsorizzazione dei prodotti aziendali.

- Region e Town: raccolgono informazioni geografiche dettagliate rispettivamente sulle regioni e sulle città.

- Specialty: rappresenta le specializzazioni dei medici, come la loro area di competenza professionale.

- Contact: rappresenta i medici e i clienti a cui l'azienda promuove e vende i propri prodotti. Per ciascun medico, vengono raccolte informazioni dettagliate che ne consentono una corretta identificazione, tra cui: ID

univoco, linea di appartenenza, specializzazione, indirizzo e altre informazioni rilevanti. I medici visitati sono suddivisi tra quelli "in Target" (assegnati ad un sales rep) e "non in Target" (non assegnati).

- Brick e Microbrick: descrivono la suddivisione geografica del settore farmaceutico. Il Brick rappresenta una porzione geografica predefinita, mentre il Microbrick è una suddivisione ancora più granulare.

- Target: rappresenta il potenziale strategico attribuito a un medico, in relazione alla sua capacità di influenzare le vendite di un farmaco. Questo potenziale viene classificato in categorie (A, B e C) e si basa su criteri come la probabilità che il medico prescriva il farmaco e il suo impatto atteso sulle vendite complessive.

- Activity Type: identifica il tipo di attività svolta dagli utenti del CRM, come per esempio: meeting, sessioni di formazione, coaching e attività sul campo.

- Interaction Type: identifica il tipo di interazione tra gli utenti del CRM e i clienti, definendo il canale o la modalità con cui vengono promossi i prodotti aziendali.

- Product: contiene tutte le informazioni relative ai prodotti presentati dai sales rep ai clienti nel corso delle loro attività promozionali.

Successivamente verranno descritti i fatti principali definiti nel modello:

- HCP in File: identifica il numero di medici (HCP) assegnati a ciascun rappresentante all'interno del target specifico. Gli HCP sono storicizzati su base mensile, con uno "snapshot" che cattura la situazione registrata nell'ultimo giorno di ogni mese. Questa struttura consente di monitorare l'evoluzione dell'assegnazione dei medici nel tempo, garantendo una visione cronologica delle modifiche e dei trend.

- HCP in Universe: rappresenta il totale degli HCP visibili ai sales rep, includendo sia quelli all'interno del target che quelli al di fuori di esso. Questo dato, noto come HCP in Universe, non è storicizzato, poiché i modelli riflettono sempre lo stato attuale senza tenere conto di informazioni storiche. A differenza degli HCP in File, che sono calcolati in base al target e al potenziale del medico, gli HCP in Universe forniscono una

panoramica completa e aggiornata di tutti i medici, offrendo un quadro immediato della portata totale disponibile per i sales rep.

- Working Days: identifica il numero di giornate lavorative dei sales rep dell'azienda, includendo sia le attività svolte direttamente sul territorio, come le interazioni dirette con i medici, sia quelle svolte al di fuori, ad esempio la partecipazione a congressi, le attività di formazione o i giorni di ferie.

- Interaction Detailing: fornisce una panoramica dettagliata delle interazioni tra un sales rep e i suoi clienti, includendo non solo la tipologia di interazione avvenuta (ad esempio, incontri in presenza, email o chiamate), ma anche il dettaglio dei prodotti presentati durante ogni incontro e il relativo ordine di esposizione. Questo consente di valutare il peso di ciascuna attività promozionale, poiché i prodotti introdotti per primi tendono ad avere un impatto maggiore rispetto a quelli presentati successivamente.

- Interaction Objectives: definisce il numero di visite che un medico dovrebbe ricevere in un determinato periodo (ad esempio, mensilmente), basandosi su obiettivi specifici relativi alla specialità e al target del medico. Questi dati sono utilizzati per misurare la percentuale di raggiungimento degli obiettivi, fornendo un quadro dettagliato delle performance e dell'efficienza della strategia commerciale.

3.2 Data Ingestion

La fase di **Data Ingestion** ha rappresentato un passaggio fondamentale nello sviluppo del progetto di tesi, permettendo di integrare in maniera fluida e sicura i dati provenienti dalle diverse istanze locali del CRM. Questo processo ha seguito una fase preliminare di studio e analisi approfondita del modello dati definito dal team, con l'obiettivo di centralizzare tutte le sorgenti dati all'interno della Data Platform, rendendole pronte per le trasformazioni future che saranno affrontate nella fase successiva.

Per garantire un'acquisizione efficiente e scalabile, è stato adottato ADF come strumento principale per lo sviluppo del processo di Ingestion. Tuttavia, per ottimizzare le prestazioni in specifici scenari di acquisizione, si è reso ne-

cessario l'utilizzo di Microsoft Fabric, che ha permesso di migliorare l'efficienza del processo riducendo i tempi di importazione dei dati.

I successivi sotto-paragrafi approfondiranno nel dettaglio le sorgenti dati coinvolte nel processo, il flusso di acquisizione del dato e l'ottimizzazione del processo per i dati provenienti da API, illustrando le scelte architetturali adottate e le soluzioni implementate per garantire affidabilità, scalabilità ed efficienza.

3.2.1 Sorgenti Dati

L'integrazione dei dati provenienti dalle diverse istanze del CRM ha richiesto un'attenta analisi delle sorgenti e delle loro peculiarità tecniche. Queste fonti si caratterizzano per eterogeneità sia tecnologica che infrastrutturale, richiedendo approcci su misura per garantire una gestione centralizzata e sicura dei dati.

La maggior parte delle informazioni proviene dai CRM delle nazioni non custom, identificate in questo modo poichè non presentano particolari peculiarità e condividono una struttura dati pressoché uniforme. Questi sistemi, basati su database **SQL Server**, riguardano paesi come: *Svizzera, Repubblica Ceca, Francia, Italia, Polonia, Romania, Russia, Slovacchia e Tunisia.*

Una fonte rilevante è rappresentata dai dati forniti da un'azienda esterna, suddivisi tra le soluzioni **Salesforce** e **Align**. Queste piattaforme sono state utilizzate per gestire l'integrazione del CRM relativo a un prodotto acquisito da un competitor. Grazie ad apposite API, è stato possibile garantire un processo di acquisizione efficiente e aggiornamenti in tempo reale.

Infine, le nazioni con CRM custom, ovvero *Germania, Spagna, Portogallo e Messico*, presentano peculiarità specifiche che richiedono un approccio su misura. La Germania utilizza un'istanza CRM Aidea, basata su una soluzione SQL Server, ma con caratteristiche tecniche proprie che la differenziano dalle altre. Per Spagna, Portogallo e Messico, invece, i dati vengono forniti tramite **connessione SFTP** dai server dell'azienda cliente. Questo metodo, sebbene più tradizionale, garantisce un trasferimento sicuro e controllato verso la Data Platform.

Queste diverse modalità di acquisizione riflettono la varietà delle infrastrutture locali, ma si integrano efficacemente per centralizzare tutte le informazioni in un unico sistema, pronte per le successive trasformazioni.

3.2.2 Processo di acquisizione del dato

Dopo aver analizzato la struttura e le caratteristiche delle principali fonti, si è reso necessario definire un flusso di acquisizione che garantisse un'integrazione efficiente e scalabile delle informazioni provenienti dalle diverse istanze locali.

Come illustrato nel capitolo precedente, ogni istanza del CRM presenta sorgenti dati con specifiche infrastrutturali e tecnologiche diverse. Per questo motivo, è stato essenziale sviluppare un flusso di acquisizione automatizzato in grado di centralizzare tutte le risorse e renderle disponibili per le successive fasi di elaborazione e trasformazione. Per il processo di ingestion, è stato adottato ADF come strumento principale, affiancato da Microsoft Fabric per migliorare le prestazioni in specifici scenari di acquisizione, ottimizzando l'efficienza del processo e riducendo i tempi di importazione dei dati.

Il flusso è strutturato attorno a una pipeline di orchestrazione, responsabile della gestione complessiva del processo di acquisizione e monitoraggio dei dati. Per ogni istanza locale del CRM, viene eseguita una pipeline parametrica che importa le risorse specifiche e le trasferisce nel formato Parquet su Azure Data Lake Gen 2. Infine, l'intero processo è ottimizzato e coordinato da Microsoft Fabric, garantendo un'esecuzione efficiente e affidabile. Nei prossimi paragrafi verranno analizzati nel dettaglio i vari step del processo.

Configurazione Linked Service e Dataset parametrici

Il primo step del processo è la configurazione delle connessioni alle sorgenti dati tramite i **Linked Services**. Questi rappresentano il punto di accesso a un determinato sistema sorgente e forniscono un'interfaccia sicura e standardizzata per connettere ADF alle diverse tipologie di sorgenti dati, come descritto nel Capitolo 2.2.1.

A seconda del tipo di sorgente, è possibile utilizzare connettori specifici. Nel mio caso, i Linked Services sono stati configurati principalmente per connettersi a istanze di SQL Server, che rappresentano la base dati principale del CRM. Inoltre, per le sorgenti basate su file strutturati, come CSV e JSON, sono stati utilizzati connettori SFTP per gestire l'acquisizione e il trasferimento sicuro delle risorse.

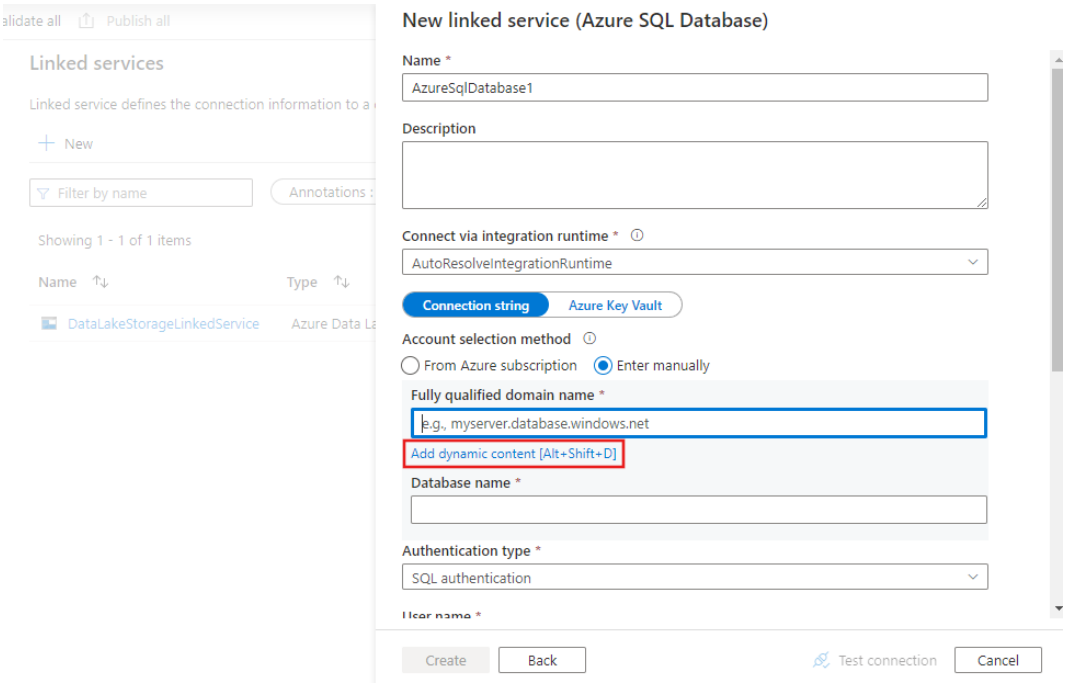


Figura 3.3: Esempio Configurazione di un Linked Service su ADF

In fase di configurazione, come mostrato in Figura 3.3, è necessario definire alcuni parametri fondamentali come: il server che ospita la sorgente dati, il nome de Database ed il metodo di autenticazione richiesto per l'accesso.

Il passo successivo è la configurazione dei **Dataset**, che identificano la sorgente dati alla quale il Linked Service crea la connessione.

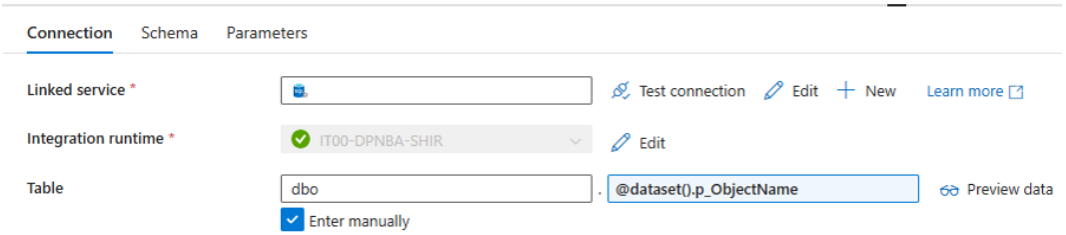


Figura 3.4: Esempio Configurazione Dataset su ADF

Come mostrato in Figura 3.4, un Dataset viene definito specificando parametri fondamentali come il connettore utilizzato (ad esempio, SQL Server) e il nome della risorsa da importare. La configurazione parametrica del Dataset consente di adattare dinamicamente il flusso di acquisizione in base al valore del parametro passato, permettendo di importare una specifica risorsa dalla sorgente dati.

Implementazione del Flusso di Ingestion

Una volta configurati i Linked Services e i Dataset, il passo successivo consiste nell'implementazione del flusso di Ingestion, che consente di trasferire i dati dalle sorgenti alla Data Platform in modo strutturato ed efficiente.

Il primo step di questa fase ha previsto la progettazione e l'implementazione della pipeline parametrica ***{COUNTRY}_FULL***. Questa pipeline utilizza due parametri principali:

- Nome della risorsa: definisce quale tabella o file specifico deve essere acquisito.
- Country: identifica l'istanza locale del CRM da cui prendere la risorsa.

La configurazione parametrica consente di riutilizzare la pipeline per estrarre tutte le risorse necessarie da una specifica sorgente dati, adattandola in base ai valori dei parametri definiti, come il nome della risorsa e la country.

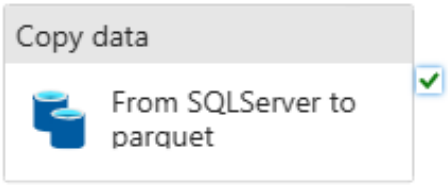


Figura 3.5: PAR_{COUNTRY}_FULL

La Figura 3.5 mostra l'elemento principale di questa pipeline: una **Copy Activity**. Questa activity è responsabile dell'estrazione della risorsa (nel caso specifico illustrato, una tabella) dalla sorgente specificata e del suo salvataggio in formato Parquet. Questo formato è stato scelto per la sua efficienza in termini di compressione e capacità di elaborazione.

Successivamente, per ogni istanza locale del CRM, è stata implementata una pipeline di orchestrazione denominata ***ORC_01_{COUNTRY}_All***. Questa pipeline, permette di organizzare un flusso composto da molteplici activity, configurate per essere eseguite sia in modo sequenziale che in parallelo. Ogni activity è progettata per completare il processo di acquisizione di una determinata risorsa (tabella o file) per una specifica istanza del CRM, invocando la pipeline parametrica descritta in precedenza e passando i parametri relativi alla risorsa da importare e alla country di riferimento.

Questa struttura modulare permette di adattare facilmente il flusso alle esigenze di ciascuna istanza del CRM, consentendo di poter aggiungere risorse da importare in Data Platform in modo efficiente.

Infine, per coordinare il processo di acquisizione e monitorarne lo stato, è stata implementata una pipeline di orchestrazione di secondo livello denominata ***ORC_02_{COUNTRY}_AllWithLog***. Questa pipeline, illustrata in Figura 3.6, si compone di diverse activity ciascuna con una funzione specifica. La prima activity registra nella tabella di log un record che indica l'avvio del processo, facilitando il tracciamento dell'esecuzione. Successivamente, un'activity di tipo **Lookup** recupera l'identificativo della sessione corrente, che viene salvato in una variabile per associarlo al record relativo al flusso in corso. In seguito, la pipeline richiama quella di orchestrazione di primo livello descritta precedentemente per gestire l'acquisizione delle risorse relative a una determinata istanza del CRM. Al termine dell'esecuzione, l'esito viene registrato: in caso di successo, un'activity aggiorna il record nella tabella di log per indicare il completamento positivo del processo, mentre in caso di errore un'altra activity segnala che si è verificato un errore nel flusso.

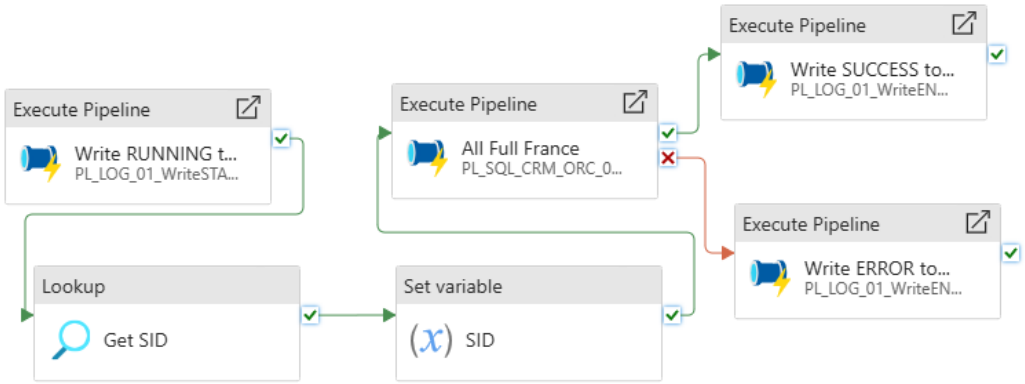


Figura 3.6: ORC_02_{COUNTRY}_AllWithLog

Questa pipeline svolge un ruolo essenziale nel garantire un monitoraggio efficace e centralizzato del processo di integrazione delle risorse nella Data Platform. Inoltre, sarà parte integrante del flusso di esecuzione complessivo del CRM, implementato su Microsoft Fabric, la cui architettura e funzionamento saranno approfonditi nei capitoli successivi.

3.2.3 Ottimizzazione del processo di acquisizione dati da API

L'intero flusso di ingestion non è stato sviluppato esclusivamente mediante pipeline su ADF. Sebbene l'architettura iniziale prevedesse l'utilizzo di questo strumento per tutte le operazioni di acquisizione dati, si è reso necessario adottare un approccio differente per alcuni casi specifici. In particolare, per la gestione dei dati provenienti da API, l'utilizzo di Data Factory ha mostrato significative inefficienze in termini di prestazioni, rendendo l'esecuzione delle pipeline estremamente lenta e non sostenibile per le necessità operative del progetto.

Il contesto di applicazione riguarda il CRM legato all'acquisizione di un prodotto da un competitor. Come descritto nei capitoli precedenti, il sistema prevede due principali categorie di sorgenti dati: una basata su un database tradizionale, che segue il modello di ingestion già illustrato, e l'altra costituita da API, per le quali è stato necessario adottare un approccio alternativo.

Dopo un'attenta analisi con il team tecnico, è stato deciso di sostituire ADF con Microsoft Fabric per l'acquisizione dei dati da API. Questa scelta ha permesso di migliorare significativamente le prestazioni, riducendo i tempi di esecuzione e garantendo un'integrazione più efficiente all'interno della Data Platform. In particolare, si è deciso di sfruttare uno degli strumenti più potenti messi a disposizione da Microsoft Fabric: i **Notebook**. I Notebook rappresentano un ambiente di sviluppo interattivo che permette di eseguire codice Python e Spark in modo scalabile ed efficiente, consentendo di interagire direttamente con le API.

Per supportare questo approccio, sono state sviluppate due librerie Python principali:

- **getTablesFields**: consente di estrarre i campi di una tabella specificata come input, costruendo query mirate per l'estrazione dei dati.

- **getTableSchema**: permette di ottenere lo schema della tabella, utile per creare il DataFrame corrispondente destinato a essere salvato nel livello bronze della Data Platform.

Queste librerie hanno reso il processo di ingestion parametrico e flessibile. In base alla risorsa specificata, l'obiettivo era determinare il relativo schema e i campi, per poi costruire dinamicamente la query necessaria a ottenere i dati richiesti.

Inizialmente è stato sviluppato il Notebook **Full**, ovvero un notebook parametrico progettato per importare tutti i dati della risorsa specificata. Una delle principali difficoltà incontrate durante lo sviluppo riguardava il limite tecnico imposto dalle API, che restituivano un massimo di 1000 record per chiamata. Per superare questa limitazione, è stato implementato un meccanismo di paginazione basato sul valore *next_page* restituito dall'API. Questo metodo consente di iterare automaticamente sulle richieste fino al recupero completo dell'intero dataset, garantendo un'acquisizione efficiente e senza perdita di dati.

Il recupero dei dati è gestito dalla funzione *fetch_data*, che si occupa dell'interazione con l'API, della gestione della paginazione e della raccolta dei record. Internamente, questa funzione utilizza *make_request*, un metodo dedicato all'invio delle richieste API, che accetta i parametri necessari per ottenere i dati e restituisce i risultati in formato JSON. Grazie a questa struttura modulare, il recupero dei dati avviene in modo incrementale, assicurando efficienza e affidabilità.

Dopo la fase di estrazione, i dati vengono trasformati e caricati in un **DataFrame Spark**, il quale prima della scrittura viene arricchito con metadati aggiuntivi, tra cui il timestamp dell'ultima modifica (*TAG_LAST_UPDATE*) e l'origine dei dati (*TAG_DATA_SOURCE*). Questi metadati migliorano la tracciabilità e la governance dei dati, permettendo un controllo più accurato sull'intero processo.

Per migliorare la gestione di grandi volumi di dati, i record non vengono caricati in un unico DataFrame, ma suddivisi in piccoli **batch**, ciascuno contenente circa un decimo del totale. Questo approccio evita il sovraccarico della memoria e garantisce una maggiore stabilità del sistema. Ogni batch viene scritto singolarmente in una **Delta Table**, utilizzando la funzione di *append*. Il frammento di codice riportato di seguito illustra l'approccio adottato:

```

fields = getTablesFields(table_name)
schema = getTableSchema(table_name)

# Fetches all data from the specified table, handling pagination to ensure full retrieval of records

def fetch_data(table_name, session_id, base_url):
    rows = []
    page_offset = 0
    page_size = 1000 # Set the page size for pagination requests
    headers = {
        'Accept': '*/*',
        'Authorization': session_id,
        'Content-Type': 'application/x-www-form-urlencoded'
    }

    # Define a function to make the request and handle pagination details
    def make_request(url):
        response = requests.post(url, headers=headers, data=f"q=SELECT {fields} FROM {table_name} PAGESIZE {page_size} PAGEOFFSET {page_offset}")
        response_data = response.json()
        return response_data['data'], response_data['responseDetails']

    # First request to initialize and get pagination details
    initial_url = f"{base_url}/query"
    data, response_details = make_request(initial_url)
    rows.extend(data)
    next_page_url = f"{base_url}/query/" + response_details['next_page'].split('query/')[1] if 'next_page' in response_details else None

    # Loop to fetch subsequent pages
    while next_page_url:
        data, response_details = make_request(next_page_url)
        rows.extend(data)
        next_page_url = f"{base_url}/query/" + response_details['next_page'].split('query/')[1] if 'next_page' in response_details else None

    return rows

all_rows = fetch_data(table_name, session_id, base_url)

# Function to append data to the Delta table
def create_brz_table(rows):
    df = spark.createDataFrame(rows, schema=schema)
    # add tags columns
    df = df.withColumn("TAG_LAST_UPDATE", current_timestamp())
    df = df.withColumn("TAG_DATA_SOURCE", lit(""))
    df.write.mode("append").option("mergeSchema", "true").format("delta").save(f"Tables/{brz_table_name}")

# Overwrite the Delta table schema if data is available
if(len(all_rows)>0):
```

```
    empty_df = spark.createDataFrame([], schema)
    empty_df.write.mode("overwrite").option("overwriteSchema",
    ↪  "true").format("delta").save(f"Tables/{brz_table_name}")

    # Split data into chunks to make writing more efficient
    num_partitions = 10
    chunk_size = math.ceil(len(all_rows) / num_partitions)
    for i in range(num_partitions):
        subset = all_rows[i * chunk_size: (i + 1) * chunk_size]
        create_brz_table(subset)
```

Successivamente, è stato sviluppato il **Notebook Delta**, progettato per ottimizzare l'acquisizione dei dati aggiornati o modificati rispetto all'ultima estrazione. Questo notebook sfrutta una data di riferimento, prelevata dalla tabella di log, per filtrare solo i record nuovi o aggiornati, evitando di ricaricare l'intero dataset a ogni esecuzione. Una volta ottenuti i dati aggiornati, il processo prevede la loro integrazione all'interno della Delta Table corrispondente nel **Lakehouse** di Microsoft Fabric. L'aggiornamento avviene attraverso un'operazione di **merge**, che consente di modificare direttamente i record esistenti e inserire quelli nuovi, garantendo consistenza e integrità del dataset.

Poiché il Notebook Delta condivide molte funzionalità con il Notebook Full, il codice riportato di seguito mostra esclusivamente la parte relativa all'estrazione dei dati aggiornati basata sulla data di riferimento e alla loro successiva integrazione nella Delta Table. Le altre funzionalità, come la gestione della paginazione e l'interazione con l'API, seguono lo stesso approccio già descritto nel Notebook Full.

```
df = spark.read.format("delta").load("stringconnections")
df.createOrReplaceTempView("execution_logs_view")
result_df = spark.sql(f"""
    SELECT FLOW_NAME, MAX(START_DATE) AS MOST_RECENT_START_DATE
    FROM execution_logs_view
    WHERE FLOW_NAME = '{pipeline_name}' AND EXECUTION_STATUS = 'SUCCESS'
    GROUP BY FLOW_NAME
""")

most_recent_start_date = result_df.first()["MOST_RECENT_START_DATE"]
date_part = str(most_recent_start_date)[:10]
time_part = str(most_recent_start_date)[11:19]
start_date = f"{date_part}T{time_part}.000Z"

def make_request(url):
    response = requests.post(url, headers=headers, data=f"q=SELECT {fields} FROM {table_name}
    ↪  WHERE created_date__v > '{start_date}' AND modified_date__v > '{start_date}' PAGESIZE
    ↪  {page_size} PAGEOFFSET {page_offset}")
```

```
    response_data = response.json()
    #print(response_data)
    return response_data['data'], response_data['responseDetails']

try:
    # get existing bronze table (to be updated)
    target = DeltaTable.forPath(spark, DELTA_TABLE_PATH)
    print(f"Total row table brz before delta: {target.toDF().count()}")
    # create merge condition
    merge_condition = "tgt.id = src.id"

    # perform merge
    target.alias("tgt").merge(source_df.alias("src"),
    ↪  merge_condition).whenMatchedUpdateAll().whenNotMatchedInsertAll().execute()
    result = target.toDF()
    row_count = result.count()
```

Una volta completato lo sviluppo dei Notebook, è stato necessario implementare un sistema di orchestrazione per garantire un'esecuzione automatizzata e monitorata del processo di acuisizione del dato da API. Per supportare l'intero flusso, sono state implementate due pipeline principali, ciascuna con un ruolo specifico.

La pipeline di orchestrazione di primo livello, ***ORC_01_A2B_DeltaFromAPI***, è stata sviluppata per gestire in parallelo l'esecuzione di tutti i Notebook Delta necessari all'aggiornamento o integrazione delle tabelle. Ogni notebook viene avviato con i parametri specifici per la tabella da processare.

Per coordinare l'esecuzione della pipeline descritta precedentemente e garantire un controllo completo sull'intero processo, è stata progettata la pipeline di orchestrazione ***ORC_02_A2B_DeltaFromAPI***. Il flusso di esecuzione della pipeline, mostrato in Figura 3.7, si articola in una sequenza di activity progettate per monitorare lo stato di avanzamento e coordinare l'intero processo. Queste activity comprendono la registrazione dello stato corrente del flusso in una tabella di log e l'esecuzione della pipeline di orchestrazione di primo livello descritta precedentemente.

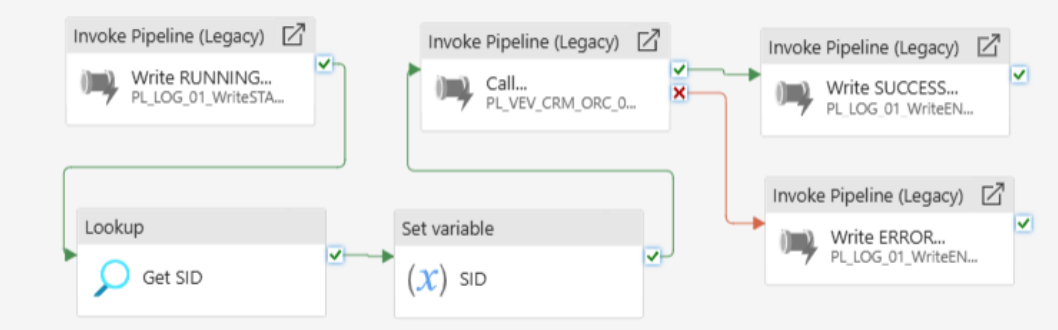


Figura 3.7: ORC_02_A2B_DeltaFromAPI

Questo approccio garantisce un monitoraggio continuo e una gestione strutturata del processo, assicurando che eventuali errori siano prontamente identificati e che lo stato finale venga correttamente registrato al termine dell'esecuzione.

In conclusione, l'integrazione di uno strumento come Microsoft Fabric ha introdotto un modello di gestione dei dati differente rispetto a quello adottato per le sorgenti basate su database relazionali. Mentre nel flusso definto su ADF i dati vengono prima salvati nel formato Parquet su Azure Data Lake Gen 2, per poi essere trasformati in Delta Table attraverso un processo che verrà descritto nel capitolo successivo, in questo scenario i dati acquisiti tramite API vengono direttamente salvati in Delta Table sul Lakehouse, rendendoli immediatamente disponibili per le successive fasi di trasformazione ed analisi.

Nei capitoli successivi verrà illustrato nel dettaglio il processo di trasformazione e le strategie di armonizzazione adottate per garantire la qualità e la coerenza delle informazioni integrate nella Data Platform.

3.3 Data Preparation

Una volta integrate nella Data Platform tutte le risorse provenienti dalle sorgenti delle istanze locali del CRM, il passo successivo consiste nella fase di **Data Preparation**, che ha previsto la trasformazione, armonizzazione e consolidamento dei dati acquisiti. L'obiettivo principale è stato quello di riprodurre le dimensioni e i fatti di analisi definiti nel modello logico sviluppato dal team di Iconsulting, elementi fondamentali per la costruzione di un modello analitico centralizzato. Quest'ultimo, una volta strutturato, è stato reso disponibile a un'azienda esterna incaricata della realizzazione dei report per il cliente. Que-

sto processo si è rivelato cruciale per garantire la coerenza e la qualità del dato, ponendo le basi per la definizione e il monitoraggio dei KPI aziendali, assicurando così un sistema di analisi affidabile e scalabile.

Per strutturare il flusso di trasformazione del dato, le attività di Data Preparation sono state sviluppate seguendo l'**Architettura Medallion**, un modello che prevede un approccio a tre livelli per la gestione e la trasformazione dei dati:

- **Bronze Layer**: raccolta e trasformazione delle risorse acquisite (mediante ADF) in Delta Table all'interno del **Lakehouse** di Microsoft Fabric.
- **Silver Layer**: pulizia, armonizzazione e trasformazione dei dati, con la definizione delle dimensioni e dei fatti necessari per la costruzione del modello analitico.
- **Gold Layer**: consolidamento dei dati trasformati e materializzazione delle viste ottimizzate, che leggono dalle dimensioni e dai fatti Silver, le quali vengono archiviate sul **Warehouse** di Microsoft Fabric.

Fondamentale per lo sviluppo dell'intero processo di Data Preparation è stato l'utilizzo di **Microsoft Fabric**, che ha fornito un ambiente integrato per la gestione e trasformazione dei dati. Fabric è stato impiegato per la memorizzazione e l'archiviazione dei dati all'interno del Lakehouse e del Warehouse, mentre strumenti come Notebook e Pipeline hanno permesso di automatizzare e ottimizzare le operazioni, garantendo efficienza e scalabilità all'intero workflow.

Nei paragrafi successivi verranno illustrate nel dettaglio le attività svolte in ciascun layer, evidenziando le strategie adottate per garantire un processo di trasformazione efficace e scalabile.

3.3.1 Bronze Layer

La fase iniziale del processo di Data Preparation prevede il trasferimento e la gestione delle risorse importate tramite ADF all'interno di ADLS Gen2, sotto forma di Delta Table nel Lakehouse di Microsoft Fabric. Questa operazione è fondamentale per garantire la disponibilità dei dati in un formato ottimizzato, sui quali poi verranno applicate operazioni di pulizia, trasformazione e armonizzazione.

Inizialmente è stato necessario definire un riferimento alle risorse salvate su ADLS Gen2, operazione realizzata attraverso la creazione di uno Shortcut su Fabric. Grazie all'integrazione nativa tra ADF e Microsoft Fabric, è possibile far comunicare direttamente i due strumenti. In questo caso specifico, utilizzando il nome della connection precedentemente definita su ADF, è stato possibile accedere alle risorse archiviate su ADLS Gen2 direttamente da Fabric, evitando così la necessità di duplicarle o archiviarle nuovamente.

Una volta ottenuto il riferimento alle risorse, il passo successivo è stato la loro trasformazione da file Parquet a Delta Table all'interno del Lakehouse di Fabric. Questa operazione è stata eseguita mediante un Notebook Python parametrico, denominato ***PAR_F2B_FullFromParquet***, che riceve in input il codice dell'istanza locale del CRM, permettendo di importare esclusivamente i dati relativi alla specifica istanza nel Lakehouse.

Per individuare i file da trasformare, il notebook accede direttamente al container di ADLS Gen2, che contiene tutte le risorse associate all'istanza selezionata. Successivamente, come mostrato dal codice sottostante, ciascun file viene caricato in un DataFrame Spark, trasformato in Delta Table e archiviato all'interno del Lakehouse.

Per garantire un'organizzazione chiara e strutturata, è stata applicata una convenzione di nomenclatura standardizzata, che associa a ogni tabella un nome univoco. Questo nome riflette il livello dell'architettura Medallion (Bronze, Silver o Gold), il tipo di sorgente, il codice del paese corrispondente all'istanza locale del CRM (Country Code) e il nome della risorsa.

```
# Define the path to the folder
path = {stringconnection}


folder = mssparkutils.fs.ls(path)

# Get the list of files in the folder
files = mssparkutils.fs.ls(folder.path)
folder_country_code = os.path.basename(subfolder.path)[:2]

if folder_country_code != country_code:
    continue

# Loop through each file in the subfolder
for file in files:

    # Read the parquet file into a DataFrame
    df = spark.read.format("parquet").load(file.path)

```

```
    # Get the base name of the file to use as the table name
    table_name = f"BRZ_SQLCRM_{country_code}_{os.path.basename(file.path).replace('.parquet',
    ↪  '')}"
    print("Doing: " + table_name)

    # Write DataFrame to a Delta table named after the file
    df.write.mode("overwrite").option("overwriteSchema",
    ↪  "true").format("delta").save("Tables/" + table_name)
```

Tuttavia, per garantire che l'intero processo di trasferimento e trasformazione delle risorse avvenisse in modo efficiente e scalabile, è stata sviluppata una pipeline di orchestrazione chiamata ***ORC_01_F2B_ADFAndNotebook***.

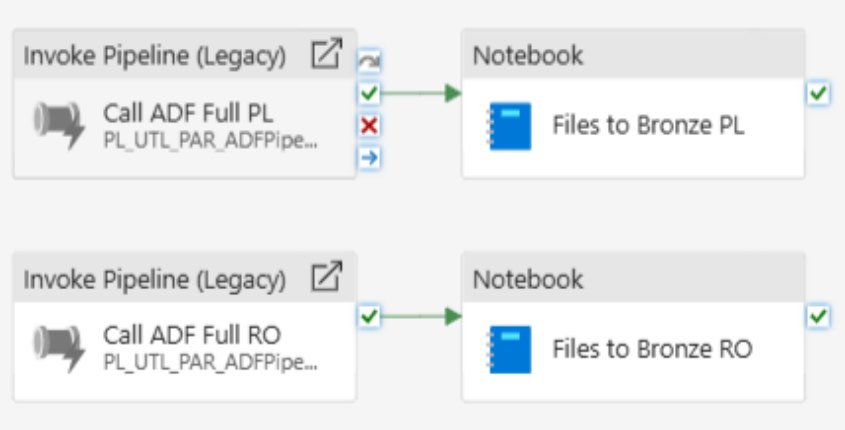


Figura 3.8: ORC_01_F2B_ADFAndNotebook

Come illustrato nella Figura 3.8, la pipeline presenta una sequenza di activity ben definita per ogni istanza locale del CRM. Il primo step consiste nell'invocare la pipeline di Ingestion su ADF, che ha il compito di acquisire tutte le risorse dalla sorgente dati e archiviarle su ADLS Gen2. Una volta completata questa fase, la pipeline attiva il notebook parametrico descritto in precedenza, il quale esegue la trasformazione dei file Parquet in Delta Table e della loro archiviazione nel Lakehouse di Fabric. Un aspetto fondamentale di questo processo è che il trasferimento e la trasformazione delle risorse avvengono in parallelo per ogni istanza locale del CRM, consentendo di ridurre significativamente i tempi di elaborazione. Questo approccio garantisce che i dati siano sempre aggiornati e immediatamente disponibili per le successive fasi di armonizzazione e modellazione.

3.3.2 Silver Layer

Dopo aver consolidato i dati grezzi nel Layer Bronze, il passo successivo nella Data Preparation è il **Layer Silver**, che si occupa della pulizia, trasformazione e armonizzazione delle informazioni. L'obiettivo di questa fase è strutturare i dati in dimensioni e fatti coerenti con il modello logico definto dal team, garantendo che ogni entità sia definita in modo uniforme tra le diverse istanze del CRM.

Le attività svolte in questa fase comprendono:

- Rimozione delle anomalie e dei dati duplicati, assicurando la qualità delle informazioni.
- Applicazione di regole di trasformazione per la costruzione delle dimensioni e dei fatti definiti nel modello.
- Armonizzazione delle entità tra i diversi paesi, per ottenere una visione globale e uniforme dei dati.
- Storicizzazione del dato, consentendo di tracciare le variazioni nel tempo.

Un elemento chiave in questa fase è stato lo sviluppo di Notebooks python parametrizzati, progettati per generare le dimensioni e i fatti definiti nel modello logico.

Ogni notebook è strutturato per implementare la versione locale di una specifica dimensione o fatto, adattandosi alle peculiarità delle singole istanze del CRM. L'esecuzione avviene in modo parametrico, basandosi sul codice dell'istanza locale del CRM passato come input. Questo parametro guida l'applicazione di logiche di trasformazione specifiche, permettendo di personalizzare join, regole di filtraggio e aggregazioni in funzione delle caratteristiche dei dati disponibili per ciascuna istanza.

Solo dopo che le versioni locali delle dimensioni e dei fatti sono state sviluppate e validate, queste vengono consolidate in una struttura globale armonizzata. Questo processo, realizzato attraverso un notebook di **UNION**, consente di unificare i dati delle diverse istanze locali del CRM in una unica Delta Table. In questo modo si garantisce un modello dati coerente e uniforme, eliminando eventuali differenze strutturali e assicurando un'unica versione del dato.

Nei prossimi paragrafi verrà descritto il processo di definizione delle dimensioni, utilizzando come esempio la dimensione **Line**, e successivamente quello relativo ai fatti, prendendo come riferimento **Working Days**.

Definizione delle dimensioni - Esempio Line

Il processo di definizione e sviluppo delle dimensioni prevede una serie di passaggi fondamentali volti a garantire coerenza, qualità e uniformità del dato tra le diverse istanze del CRM. Per illustrare nel dettaglio le varie fasi che compongono questo processo, verrà preso in esame il caso della dimensione ***Line***, sviluppata per l'istanza del CRM relativa a un prodotto acquisito dall'azienda Cliente. La dimensione Line identifica la linea aziendale di riferimento, ovvero la suddivisione delle attività in base alla specializzazione dell'area medica, come cardiologia, oncologia o medicina generale.

Inizialmente è stato necessario condurre un'analisi preliminare delle tabelle Bronze, relative alla specifica istanza del CRM, affinchè si possano individuare le "fonti" dati utili alla costruzione della dimensione. Prima di proseguire con la descrizione delle tabelle bronze, ci tengo a precisare che per motivi di privacy non potrò citare il nome effettivo della tabelle ma bensì farò riferimento a variabili rappresentative.

Dalla documentazione aziendale è emerso che le informazioni relative alle linee di business sono contenute nella tabella ***BRZ_TABLE_LINE***, che include gli identificativi delle linee, la loro descrizione e i periodi di validità. Inoltre, considerando che questo CRM raccoglie informazioni provenienti da più paesi, è necessario che ogni istanza della dimensione Line contenga un identificativo chiaro della nazione di appartenenza, evitando ambiguità nella gestione e nell'aggregazione delle informazioni. In seguito a questa analisi si è reso opportuno integrare anche la tabella ***BRZ_TABLE_COUNTRY***, che fornisce i dettagli relativi alle diverse nazioni.

Una volta individuate le tabelle di riferimento, è stata sviluppata la query per la costruzione della dimensione su un ambiente di test come **SQLServer**. L'elaborazione ha seguito un approccio strutturato, garantendo la corretta gestione dei dati attraverso più passaggi. Il primo step ha riguardato la definizione della chiave primaria, costruita concatenando il codice della country, l'ID della Line e l'ultimo giorno del mese di riferimento. Questo approccio ha permesso di ottenere una chiave univoca in grado di identificare in modo pre-

ciso ogni linea di business nel tempo, garantendo una corretta gestione della storicizzazione dei dati.

L'informazione relativa all'*ultimo giorno del mese di riferimento* non proviene direttamente dalle tabelle Bronze, ma è stata aggiunta in fase di modellazione della dimensione. Questo valore viene inserito per tenere traccia del momento esatto in cui la dimensione è stata costruita, consentendo di ottenere una "fotografia" mensile dei dati. L'obiettivo è quello di modellare le informazioni in modo tale da rappresentare uno stato preciso delle linee di business alla chiusura del mese.

La gestione di questa informazione è affidata a una funzione all'interno del Notebook che si occupa dell'esecuzione della query e della generazione della dimensione. Tale funzione definisce dinamicamente la data di riferimento, attribuendo a ogni record il valore dell'ultimo giorno del mese corrente, assicurando così la corretta strutturazione della dimensione nel Lakehouse.

Successivamente, per garantire la qualità e l'affidabilità dei dati, sono state applicate delle regole di validazione, tra cui l'esclusione delle linee prive di riferimenti validi e la selezione delle sole linee attive nel periodo di riferimento. Questo è stato realizzato utilizzando un filtro sulle date di validità, confrontando l'ultimo giorno del mese con i valori di *start_date_aln* ed *end_date_aln*, in modo da considerare solo le linee attualmente valide nel periodo di riferimento. noltre, per garantire la completezza e l'integrità del dato, è stato implementato un meccanismo di gestione delle anomalie, sfruttando la funzione *COALESCE*. Questa funzione permette di sostituire eventuali valori nulli con un valore di default, evitando la perdita di informazioni.

```
SELECT
DISTINCT
COALESCE(c.iso_code__aln, '*') AS COUNTRY,
CONCAT(c.iso_code__aln,'-', f.id,'_','{LAST_DAY_OF_MONTH}') AS PK_LINE,
CONCAT(c.iso_code__aln, '-',f.id) AS LINE_ID,
CONCAT(c.iso_code__aln, '-',f.id) AS LINE_CODE,
f.name AS LINE_DESCRIPTION,
'{LAST_DAY_OF_MONTH}' AS LAST_DAY_OF_MONTH,
f.TAG_LAST_UPDATE AS TAG_LAST_UPDATE,
f.TAG_DATA_SOURCE AS TAG_DATA_SOURCE
FROM (SELECT id, RIGHT(name__v, LENGTH(name__v) - INSTR(name__v, '_'))as
    ↪ name, start_date__aln, end_date__aln, country__aln,
    ↪ TAG_LAST_UPDATE_ON_NBA_UTC, TAG_DATA_SOURCE
```

```
                FROM {ORIGIN}.{BRZ_TABLE_NAME_GAL_4}
                WHERE parent_field_force__aln is not null) f
LEFT JOIN {ORIGIN}.{BRZ_TABLE_NAME_GAL_3} c on f.country__aln = c.id
WHERE CAST('{LAST_DAY_OF_MONTH}' AS DATE) BETWEEN
    COALESCE(f.start_date__aln, TO_DATE('1900-01-01', 'yyyy-MM-dd'))
    AND COALESCE(f.end_date__aln, TO_DATE('9999-12-31', 'yyyy-MM-dd'))
```

Per garantire flessibilità ed efficienza nella costruzione delle dimensioni, ogni notebook parametrizzato dedicato al loro sviluppo è stato progettato con due parametri fondamentali. Il primo è il codice dell'istanza locale del CRM, che determina per quale country deve essere sviluppata la dimensione. Il secondo è il parametro ***FULL***, che consente di specificare il volume di dati da elaborare in base alle esigenze di aggiornamento.

L'introduzione di questo secondo parametro ha un impatto significativo sull'ottimizzazione del flusso, permettendo di scegliere tra due modalità di esecuzione. Se impostato a TRUE, il notebook esegue il processing completo, elaborando tutti i dati disponibili nelle tabelle Bronze e ricostruendo interamente la dimensione su un range temporale più ampio, ovvero dal 2021 in poi. Al contrario, se impostato a FALSE, l'elaborazione viene limitata agli ultimi tre mesi, consentendo di ottenere una versione aggiornata della dimensione senza dover rieseguire l'intero calcolo. Questa logica consente di ridurre significativamente i tempi di elaborazione e il consumo di risorse computazionali, evitando operazioni ridondanti su dati già processati.

Successivamente si passa all'implementazione del notebook, che ha il compito di elaborare i dati provenienti dalle tabelle Bronze e trasformarli in una dimensione strutturata e storicizzata. Uno degli aspetti centrali di questo processo è la storicizzazione dei dati, che consente di ottenere una "fotografia" mensile della dimensione, riflettendo lo stato delle linee di business in un determinato periodo.

Dopo aver impostato la logica di storicizzazione, il passaggio successivo consiste nel replicare all'interno del notebook la query SQL precedentemente validata su SQL Server.

Una volta eseguita la query e ottenuto il risultato sotto forma di DataFrame , il notebook procede con la scrittura dei dati in Delta Table. Questa tabella rappresenta la versione consolidata della dimensione Line, in cui sono state applicate tutte le trasformazioni, le regole di validazione e la storicizzazione necessarie per garantire coerenza e qualità del dato. La dimensione così

ottenuta è pronta per essere armonizzata con le altre istanze locali del CRM attraverso il processo di UNION, che consentirà di ottenere una visione unica e standardizzata della Line.

```
if COUNTRY == 'GALJYS':
    for day in last_days:
        LAST_DAY_OF_MONTH = day.strftime('%Y-%m-%d')
        query = f"""
            SELECT
DISTINCT
COALESCE(c.iso_code__aln, '*') AS COUNTRY,
CONCAT(c.iso_code__aln,'-', f.id,'_','{LAST_DAY_OF_MONTH}') AS PK_LINE,
CONCAT(c.iso_code__aln, '-',f.id) AS LINE_ID,
CONCAT(c.iso_code__aln, '-',f.id) AS LINE_CODE,
f.name AS LINE_DESCRIPTION,
'{LAST_DAY_OF_MONTH}' AS LAST_DAY_OF_MONTH,
f.TAG_LAST_UPDATE AS TAG_LAST_UPDATE,
f.TAG_DATA_SOURCE AS TAG_DATA_SOURCE
FROM (SELECT id, RIGHT(name__v, LENGTH(name__v) - INSTR(name__v, '_'))as
↪  name, start_date__aln, end_date__aln, country__aln,
↪  TAG_LAST_UPDATE_ON_NBA_UTC, TAG_DATA_SOURCE
                FROM {ORIGIN}.{BRZ_TABLE_NAME_GAL_4}
                WHERE parent_field_force__aln is not null) f
LEFT JOIN {ORIGIN}.{BRZ_TABLE_NAME_GAL_3} c on f.country__aln = c.id
WHERE CAST('{LAST_DAY_OF_MONTH}' AS DATE) BETWEEN
    COALESCE(f.start_date__aln, TO_DATE('1900-01-01', 'yyyy-MM-dd'))
    AND COALESCE(f.end_date__aln, TO_DATE('9999-12-31', 'yyyy-MM-dd'))
            """
        line = spark.sql(query)
        line.write \
                .format("delta") \
                .mode("overwrite") \
                .option("partitionOverwriteMode", "dynamic") \
                .partitionBy("LAST_DAY_OF_MONTH") \
                .save("Tables/" + SLV_TABLE_NAME)
```

Una volta implementata e validata la versione locale di una dimensione, il passo successivo consiste nell'integrare i dati a livello globale tramite l'implementazione o l'aggiornamento del notebook di UNION. Questo processo consente di consolidare tutte le versioni locali della dimensione in una struttura unificata, garantendo coerenza e uniformità tra le diverse istanze del CRM.

In particolare, ogni volta che viene sviluppata una nuova versione locale della dimensione, il notebook viene aggiornato, aggiungendo alla query SQL un nuovo blocco di codice che permette di integrare i dati della nuova istanza nella versione globale della dimensione. Questo aggiornamento avviene estendendo la query con un'operazione di UNION, come mostrato nel codice seguente:

```
UNION
SELECT
COUNTRY,
PK_LINE,
LINE_ID,
LINE_CODE,
LINE_DESCRIPTION,
LAST_DAY_OF_MONTH,
TAG_LAST_UPDATE,
TAG_DATA_SOURCE
FROM {ORIGIN}.{GALJYS_TABLE_NAME}
```

Il processo di scrittura della tabella consolidata avviene con le stesse modalità descritte in precedenza, utilizzando l'istruzione write di Spark per salvare i dati nel Lakehouse di Fabric in formato Delta Table.

In conclusione, l'approccio adottato per l'implementazione della dimensione Line è stato applicato in modo sistematico allo sviluppo di tutte le altre dimensioni definite nel modello logico. Questo ha permesso di creare un modello dati centralizzato e armonizzato, garantendo coerenza e uniformità tra le diverse istanze del CRM. Grazie a questa struttura, i dati sono ora pronti per essere utilizzati nelle successive fasi di materializzazione e analisi, supportando in modo efficace il processo decisionale aziendale.

Definizione dei fatti - Esempio Working Days

Per illustrare il processo di definizione di un fatto, verrà preso in esame il caso di ***Working Days*** per il CRM Francia. Questo fatto ha l'obiettivo di identificare il numero di giornate lavorative registrate dai sales rep dell'azienda, includendo sia le attività svolte direttamente sul territorio, come le interazioni con i medici, sia quelle effettuate al di fuori del territorio, come la partecipazione a congressi, attività di formazione o giorni di ferie.

Inizialmente è stato necessario svolgere un'attenta analisi preliminare volta a comprendere la metrica da calcolare e a identificare le misure chiave necessarie per rappresentare correttamente il fatto. In questo caso, le due misure principali da ricavare sono il numero totale di ore lavorative per ciascun rappresentante e il numero di giornate lavorative, ottenuto convertendo il totale delle ore in giorni sulla base di una giornata standard di 8 ore (480 minuti).

Successivamente è stato necessario verificare quali dimensioni risultassero collegate al fatto, consultando il modello logico definito dal team. Il fatto Working Days è stato associato alle seguenti dimensioni:

- Line: identifica la linea aziendale di riferimento del rappresentante.
- Territory: associa ogni rappresentante alla propria area di competenza.
- Activity Type: identifica la tipologia di attività svolta.

Dopo aver individuato le dimensioni collegate, il passo successivo è stato l'identificazione delle tabelle Bronze da cui ricavare le informazioni necessarie. In particolare, i dati relativi alle attività svolte dai rappresentati sono contenuti nella tabella ***BRZ_TABLE_ACTIVITY***, che memorizza informazioni dettagliate come tempistiche ed il tipo di interazione avvenuta con il cliente. Inoltre, per garantire la corretta associazione ai territori di competenza e alle tipologie di attività svolte, è stato necessario integrare i dati con le tabelle ***BRZ_TABLE_TERRITORY*** e ***BRZ_TABLE_ACTIVITY_TYPE***.

Una volta completata questa analisi, si è passati alla scrittura della query (illustrata nel codice sottostante) su SQL Server, necessaria per verificare la correttezza delle associazioni e dei calcoli. La costruzione del fatto ha seguito lo stesso approccio descritto per le dimensioni, iniziando con l'estrazione dei dati dalle tabelle Bronze, applicando le stesse verifiche per garantire la qualità del dato. In particolare, sono stati implementati controlli sulla validità temporale dei record, filtrando i dati in base alle date di riferimento.

Successivamente, è stato necessario ricostruire il legame tra il fatto e le dimensioni Silver, definendo una catena di join per integrare correttamente le informazioni dalle tabelle Bronze con le entità consolidate nel livello Silver. L'associazione è stata garantita utilizzando le chiavi primarie delle dimensioni Silver, ricostruite dinamicamente dai dati estratti dalle Bronze.

```
1 SELECT
2         COALESCE(li.PK_LINE,'*') AS KEY_LINE,
```

```
3         COALESCE(ti.PK_TERRITORY,'*') AS KEY_TERRITORY,
4         COALESCE(ac.PK_ACTIVITY_TYPE,'*') AS KEY_ACTIVITY,
5         b.LAST_DAY_OF_MONTH,
6         SUM(b.ScheduledDurationMinutes/60) AS TOTAL_HOURS,
7         SUM(CAST(b.ScheduledDurationMinutes AS FLOAT))/480 AS TOTAL_DAYS,
8         b.TAG_LAST_UPDATE,
9         b.TAG_DATA_SOURCE
10        FROM (
11                SELECT
12                ac.TBActivityTypeId,
13                t.TBTerritoryId,
14                t.LineId,
15                a.ScheduledStart,
16                LAST_DAY(a.ScheduledStart) AS LAST_DAY_OF_MONTH,
17                a.ScheduledDurationMinutes,
18                a.TAG_LAST_UPDATE,
19                a.TAG_DATA_SOURCE
20                FROM {ORIGIN}.{ACT_TABLE_NAME} a
21                inner join  (select * from {ORIGIN}.{MANAGER_TABLE_NAME}
                  ↪  where Entity='territory' and d_date is null)  m ON
                  ↪  a.OwningUser = m.SystemUserId
22                inner join  {ORIGIN}.{TERRITORY_TABLE_NAME} t ON
                  ↪  a.New_TerritoryID = UPPER(t.TBTerritoryId)
23                inner join  {ORIGIN}.{ACTIVITY_TYPE_TABLE_NAME} ac ON
                  ↪  a.New_tbActivityTypeId = UPPER (ac.TBActivityTypeId)
24                inner join  {ORIGIN}.{LINE_TABLE_NAME} l ON t.LineId =
                  ↪  l.LineId
25                WHERE a.StateCode = 1 and a.DeletionStateCode <> 2 and
                  ↪  YEAR(a.ScheduledStart) = 2024
26                and CAST(ScheduledStart AS STRING) BETWEEN CAST(m.ValidFrom
                  ↪  AS STRING) AND COALESCE(CAST(m.ValidTo AS STRING),
                  ↪  CAST(ScheduledStart AS STRING))
27        ) AS b
28        LEFT JOIN {ORIGIN}.{LINE_TABLE_NAME_SLV} li ON CONCAT('{COUNTRY}',
          ↪  '-', b.LineId, '_', LAST_DAY(CAST(b.ScheduledStart AS DATE))) =
          ↪  li.PK_LINE
29        LEFT JOIN {ORIGIN}.{TERRITORY_TABLE_NAME_SLV} ti ON
          ↪  CONCAT('{COUNTRY}', '-', b.TBTerritoryId, '_',
          ↪  COALESCE(b.LineId, '*'), '_', LAST_DAY(CAST(b.ScheduledStart AS
          ↪  DATE))) = ti.PK_TERRITORY
30        LEFT JOIN {ORIGIN}.{ACTIVITYTYPE_TABLE_NAME} ac ON
          ↪  CONCAT('{COUNTRY}', '-', b.TBActivityTypeId, '_',
          ↪  LAST_DAY(CAST(b.ScheduledStart AS DATE))) = ac.PK_ACTIVITY_TYPE
```

```
31          GROUP BY
32                  li.PK_LINE,
33                  ti.PK_TERRITORY,
34                  ti.TERRITORY_DESCRIPTION,
35                  ac.PK_ACTIVITY_TYPE,
36                  b.LAST_DAY_OF_MONTH,
37                  b.TAG_LAST_UPDATE,
38                  b.TAG_DATA_SOURCE
```

Definita la query, si è poi proceduto con il processo di quadratura del dato, volto a verificare che le metriche calcolate corrispondessero ai dati riportati nei report forniti dal cliente. Le immagini seguenti mostrano il confronto tra i risultati ottenuti dalla query e quelli riportati nel report di riferimento, nei quali, per motivi di privacy, non viene visualizzato il nome completo dei sales rep.

	TERRITORY_DESCRIPTION		TOTAL_DAYS
1	1001	Angelique	254
2	1003	Karine	256
3	1005	Anis	254
4	1006	Yasmine	254
5	1007	Touatia	256
6	1008	Mireille	248
7	1010	Georges	246
8	1013	Christelle	253
9	1018	Yolande	253

Figura 3.9: Risultato della query

1001	Angelique	254.0
1003	Karine	256.0
1005	Anis	254.0
1006	Yasmine	254.0
1007	Touatia	256.0
1008	Mireille	248.0
1010	Georges	246.0
1013	Christelle	253.0
1018	Yolande	253.0

Figura 3.10: Risultato del Report

Una volta completata la validazione della query attraverso il processo di quadratura, questa è stata implementata all'interno di un notebook dedicato, adottando lo stesso approccio utilizzato per le dimensioni. Il notebook esegue la query, elabora i dati necessari per il calcolo del fatto e, una volta ottenuto il risultato, procede con la scrittura della tabella in formato Delta Table nel Lakehouse di Fabric.

Dopo l'implementazione e la validazione della versione locale del fatto, il processo di armonizzazione tra le diverse istanze del CRM è stato eseguito tramite il notebook di UNION, adottando lo stesso approccio descritto in precedenza per le dimensioni.

In conclusione, l'approccio adottato per l'implementazione del fatto Working Days è stato applicato in modo sistematico allo sviluppo di tutti gli altri fatti definiti nel modello logico.

3.3.3 Gold Layer

Una volta definite le dimensioni e i fatti nel Layer Silver, garantendone qualità, storicizzazione e armonizzazione, il passo successivo nella Data Preparation è il **Layer Gold**.

In questa fase, l'obiettivo è ottenere una versione definitiva delle entità del modello, migliorandone l'accessibilità e le prestazioni per le analisi.

Il processo prevede inizialmente la creazione di viste ottimizzate che filtrano i dati secondo criteri specifici, attingendo alle dimensioni e ai fatti consolidati nel Layer Silver. Successivamente, queste viste vengono materializzate in tabelle, andando ad implementare la versione definitiva delle entità che compongono il modello analitico. Per garantire un equilibrio tra flessibilità e prestazioni, è stata adottata una strategia a doppio livello, con un layer di viste per l'applicazione di filtri e un layer di tabelle materializzate per velocizzare l'accesso ai dati e ottimizzare le interrogazioni.

Per illustrare il processo, si prende come esempio la dimensione **Line**. La prima operazione consiste nella creazione di una vista sulla dimensione globale Silver, selezionando solo i dati a partire dal 2024, al fine di ottimizzare le interrogazioni e ridurre il volume dei dati da analizzare. La query SQL per la definizione della vista è la seguente:

```
1  CREATE VIEW GLD_SQLCRM_D_Line_V
2  AS
3      SELECT
4          COUNTRY AS KEY COUNTRY                                    ,
5          PK_LINE AS PK LINE,
6          LINE_ID AS LINE ID,
7          LINE_CODE AS LINE CODE,
8          LINE_DESCRIPTION AS LINE DESCRIPTION,
9          LAST_DAY_OF_MONTH AS LAST DAY OF MONTH,
10         TAG_LAST_UPDATE AS TAG LAST UPDATE,
11         TAG_DATA_SOURCE as TAG DATA SOURCE
12       FROM SLV_SQLMDM_D_Line
13       WHERE YEAR(LAST_DAY_OF_MONTH) >= 2024
```

Successivamente, per garantire un accesso rapido ai dati la vista viene materializzata in una tabella, creando così la versione definitiva della dimensione. La query SQL per la materializzazione della vista è la seguente:

```
1 CREATE TABLE GLD_SQLCRM_D_Line
2 AS SELECT * FROM GLD_SQLCRM_D_Line_V
```

Lo stesso approccio descritto per la dimensione Line è stato applicato a tutte le altre dimensioni e fatti del modello logico.

In conclusione, il Layer Gold rappresenta la fase finale della Data Preparation, in cui le informazioni consolidate nel Layer Silver vengono trasformate per generare la versione definitiva del modello analitico centralizzato. Per garantire che il modello rimanga sempre aggiornato e coerente, è essenziale lo sviluppo di un flusso automatizzato che integri e gestisca in modo efficiente tutte le fasi di Data Integration e Data Preparation. Nel capitolo successivo verrà illustrata la predisposizione di questo flusso, approfondendo le pipeline e i processi implementati.

3.4 Predisposizione Flusso CRM

Un aspetto fondamentale del progetto è stata la predisposizione di un flusso automatizzato per garantire che i dati del CRM fossero sempre aggiornati e affidabili. Questa attività è stata trasversale a tutte le fasi del progetto, poiché ogni nuova componente sviluppata doveva essere integrata nel flusso per assicurare continuità operativa e coerenza dei dati.

L'adozione di Microsoft Fabric come orchestratore dei flussi ha permesso di gestire in modo scalabile e strutturato tutte le operazioni necessarie per aggiornare il modello analitico centralizzato. Fabric consente di integrare diverse tecnologie e strumenti per l'elaborazione dei dati, migliorando la scalabilità, la gestione dei processi e la loro automazione.

La pipeline di orchestrazione ***All-CRM-SyncAPI***, illustrata in Figura 3.8, è responsabile della gestione completa del flusso di aggiornamento dei dati del CRM. Questa pipeline è composta da una serie di activity che monitorano lo stato di avanzamento del processo. Ad ogni esecuzione viene assegnato un nuovo session ID (SID), che viene tracciato nella tabella di log. La tabella registra il progresso di ogni SID, aggiornando lo stato da 'running' a 'success' o 'failed'. In caso di errore, il sistema invia automaticamente una notifica email tramite Outlook, mentre in caso di completamento con successo, viene avviato

il refresh del modello su Power BI per garantire che le analisi aziendali siano sempre basate su dati aggiornati.

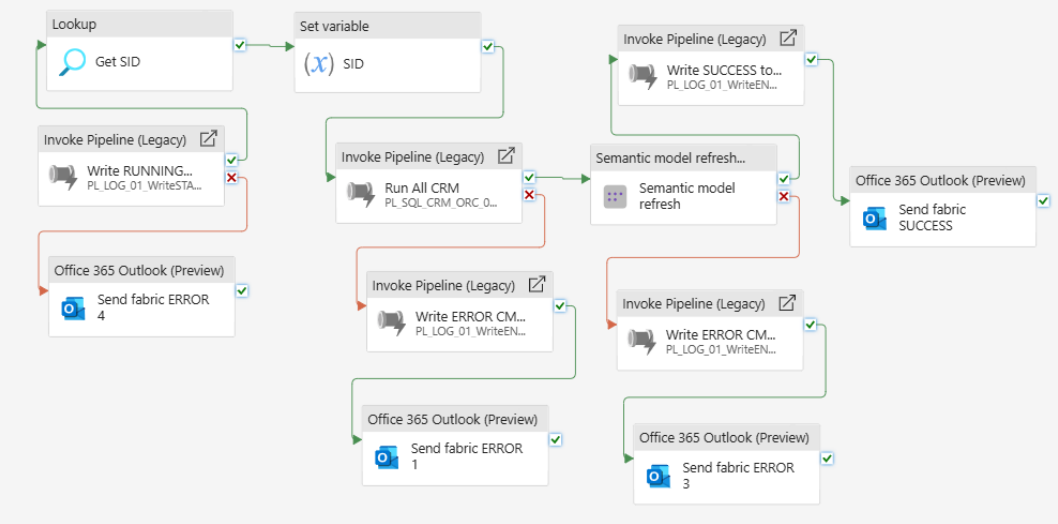


Figura 3.11: Pipeline All-CRM-SyncAPI

Questa pipeline rappresenta il punto di partenza per l'esecuzione dell'intero flusso di Data Ingestion e Data Preparation. In particolare l'activity **Run All CRM**, invoca la pipeline di orchestrazione ***F2G_All*** la quale coordina le diverse fasi del flusso, dall'ingestion dei dati fino alla loro trasformazione e armonizzazione nei livelli Bronze, Silver e Gold. Per fornire una spiegazione più dettagliata e chiara, questa pipeline viene suddivisa in tre parti principali:

- **Data Ingestion**: processo di acquisizione dei dati delle varie istanze del CRM e caricamento nel Layer Bronze.

- **Bronze to Silver**: trasformazione e armonizzazione dei dati nel Layer Silver, suddivisa in gestione di dimensioni e fatti.

- **Silver to Gold**: creazione della versione definitiva delle entità del modello nel Layer Gold.

Ogni fase verrà analizzata nei capitoli successivi, evidenziando il ruolo delle pipeline di dettaglio che gestiscono le trasformazioni specifiche per le dimensioni e i fatti del modello.

3.4.1 Data Ingestion

La prima parte della pipeline F2G_All, illustrata in Figura 3.12, è dedicata alla gestione completa della Data Ingestion. Questa fase ha il compito di acquisire i dati dalle diverse istanze del CRM e caricarli nel Layer Bronze, preparandoli per le successive trasformazioni.

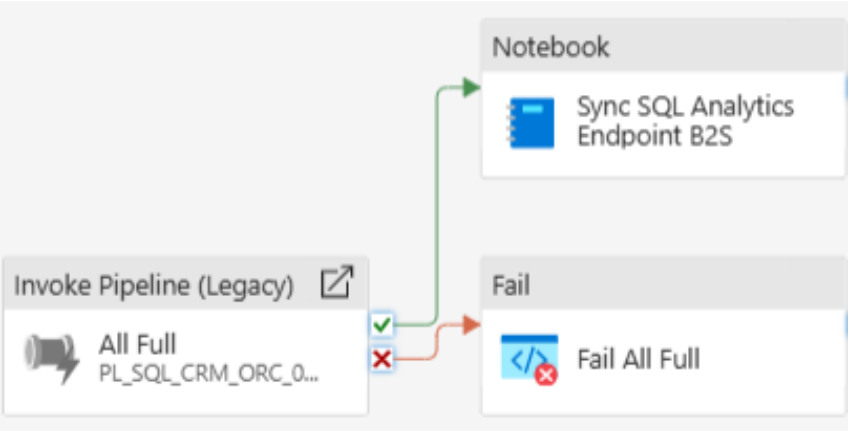


Figura 3.12: Prima parte pipeline F2G_All - Data Ingestion

In particolare, l'activity ***All Full*** svolge un ruolo centrale, poiché si occupa dell'invocazione della pipeline ***ORC_01_F2B_ADFAndNotebook***, responsabile dell'acquisizione delle risorse dalle varie sorgenti dell'istanze del CRM e della loro trasformazione in Delta Table nel Lakehouse di Fabric come spiegato nel capitolo .

3.4.2 Bronze to Silver

Successviamente una volta completato il processo di ingestion, la pipeline F2G_All prosegue con la trasformazione e armonizzazione dei dati presenti nel Layer Bronze con l'obiettivo di definire o aggiornare i fatti e le dimensioni del modello. Il flusso è stato progettato per aggiornare prima le dimensioni e successivamente elaborare i fatti. Questa scelta non è casuale, poiché i fatti dipendono direttamente dalle dimensioni a cui sono collegati.

Elaborazione delle dimensioni

In Figura 3.13, viene illustrata la parte della pipeline di orchestrazione F2G_All responsabile dell'elaborazione delle dimensioni.

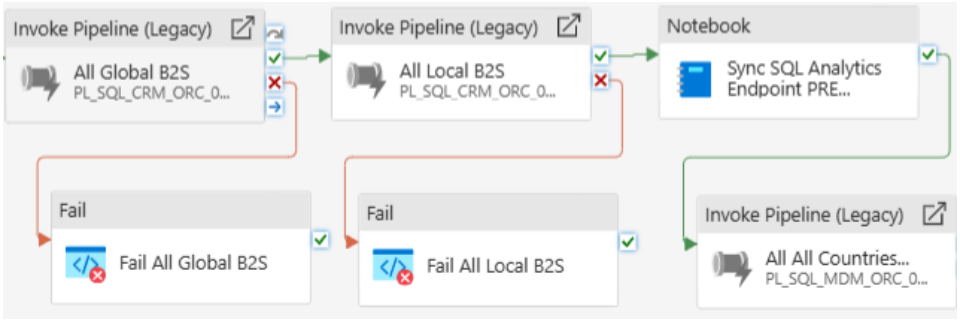


Figura 3.13: Seconda parte pipeline F2G_All - Elaborazione dimensioni

La prima activity invoca la pipeline di orchestrazione ***ALLGlobalB2S***, illustrata in Figura 3.14, responsabile dell'elaborazione delle dimensioni globali. Queste dimensioni contengono informazioni a livello corporate, garantendo l'uniformità dei dati tra le diverse istanze del CRM. Le dimensioni in questione sono: Role, CustomerType, ContactStatus, Specialty, ActivityType, InteractionType e Target. L'elaborazione avviene in parallelo e, per ciascuna dimensione, viene eseguito un notebook dedicato che filtra i dati da un file CSV e li archivia in una Delta Table.

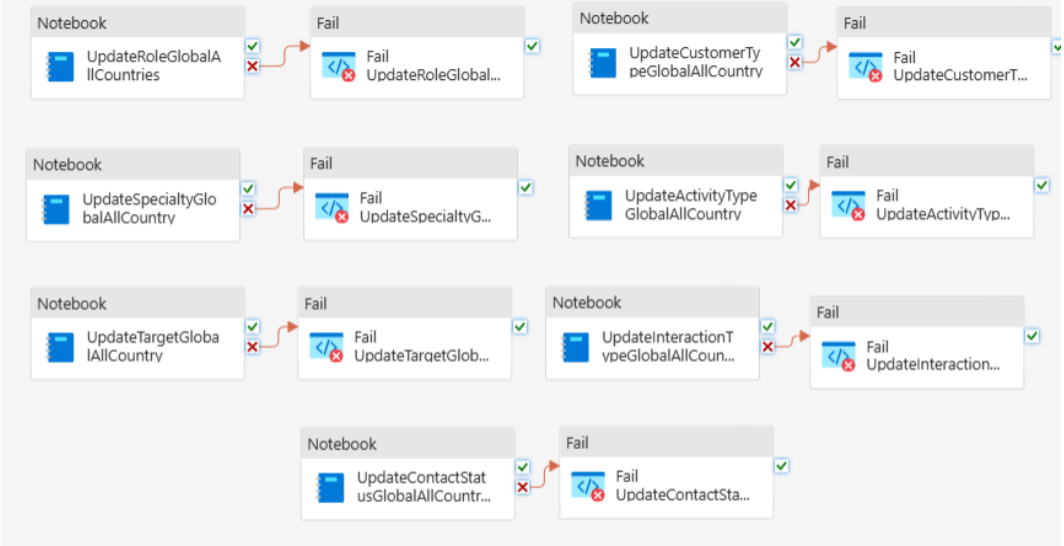


Figura 3.14: Pipeline AllGlobalB2S

Successivamente, proseguendo nell'analisi del flusso descritto in Figura 3.13, viene invocata la pipeline di orchestrazione ***AllLocalB2S*** responsabile dell'elaborazione di tutte le dimensioni del modello per ogni istanza del CRM in versione locale. Questa pipeline è strutturata in modo tale che per ogni istanza del CRM sia presente un'activity dedicata all'elaborazione delle dimensioni.

Inoltre presenta un parametro *Full*, che specifica se è richiesta o meno una ricostruzione completa dello storico, oppure un refresh solo degli ultimi tre mesi. Per ottimizzare le risorse e migliorare l'efficienza dell'elaborazione, le istanze sono state suddivise in gruppi di tre, elaborate in parallelo, mentre all'interno di ciascun gruppo le istanze vengono processate in sequenza. Per motivi di spazio, in Figura 3.15 è stata riportata solo una parte della pipeline, mostrando la logica di elaborazione per un sottoinsieme delle istanze del CRM.

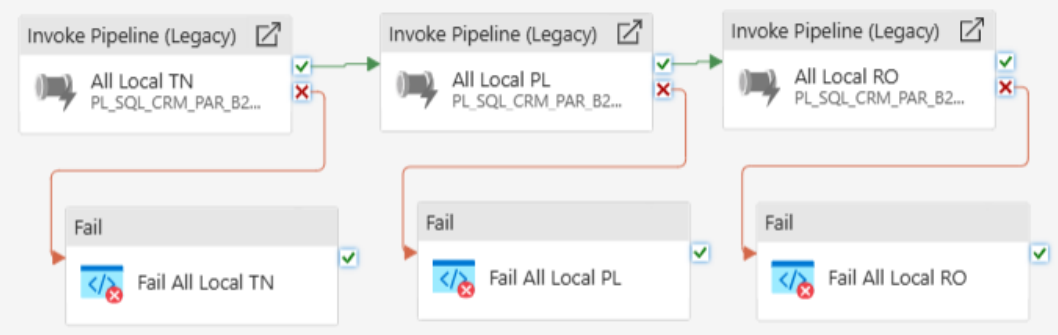


Figura 3.15: Pipeline AllLocal

Successivamente, ciascuna di queste activity invoca la pipeline parametrica ***AllLocalNotebook***, incaricata dell'elaborazione di tutte le dimensioni del modello per una specifica istanza.

Questa pipeline esegue un notebook dedicato per ogni dimensione, seguendo il processo descritto nel capitolo 3.3.2, in cui è stato preso in esame il caso della dimensione Line. Inoltre, riceve dalla pipeline chiamante il valore del parametro Full, determinando se eseguire una ricostruzione completa o un aggiornamento parziale. Presenta anche un ulteriore parametro che specifica per quale istanza del CRM devono essere elaborate le dimensioni. Per motivi di spazio, in Figura 3.16, è riportata solo una parte della pipeline, focalizzata su un sottoinsieme di notebook responsabili dell'elaborazione di alcune dimensioni. Questa pipeline tiene anche conto delle eventuali dipendenze tra le dimensioni, come la gerarchia della geography mostrata in figura.

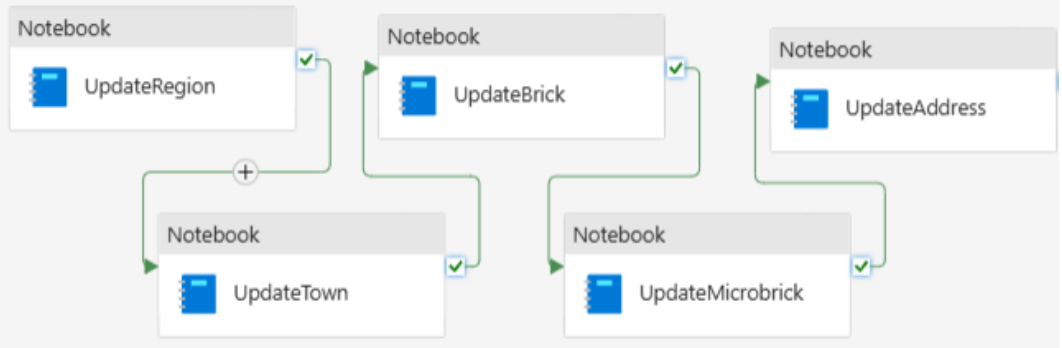


Figura 3.16: Pipeline AllLocalNotebook

Una volta completata l'elaborazione delle dimensioni in versione locale si procede con il processo di armonizzazione. Seguendo il flusso descrittto in Figura 3.13, viene invocata la pipeline ***AllAllCountries***.

Questa pipeline, mostrata in Figura 3.17, esegue un notebook specifico per ciascuna dimensione, incaricato di effettuare l'operazione di UNION tra tutte le versioni locali della dimensione corrispondente, seguendo la logica descritta nel capitolo 3.3.2, con riferimento all'entità Line. Per motivi di spazio, la figura mostra solo una porzione della pipeline, evidenziando alcuni dei notebook coinvolti nel processo.

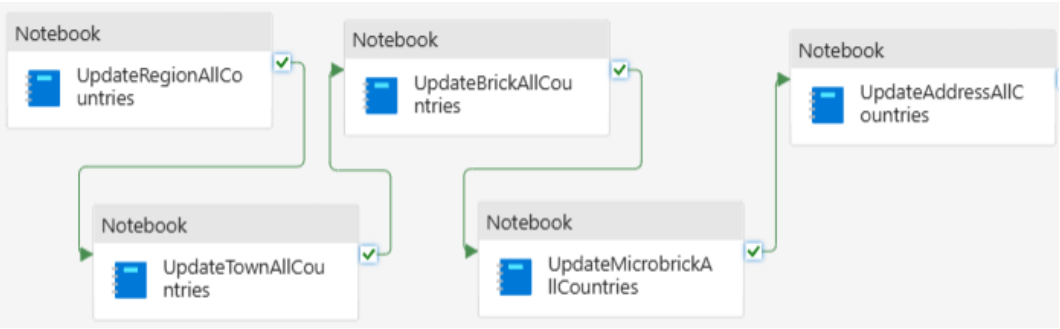


Figura 3.17: Pipeline AllAllCountries

Elaborazione dei fatti

Una volta completata l'elaborazione delle dimensioni, il flusso della pipeline F2G_All prosegue con la trasformazione e l'armonizzazione dei fatti, come illustrato in Figura 3.18.

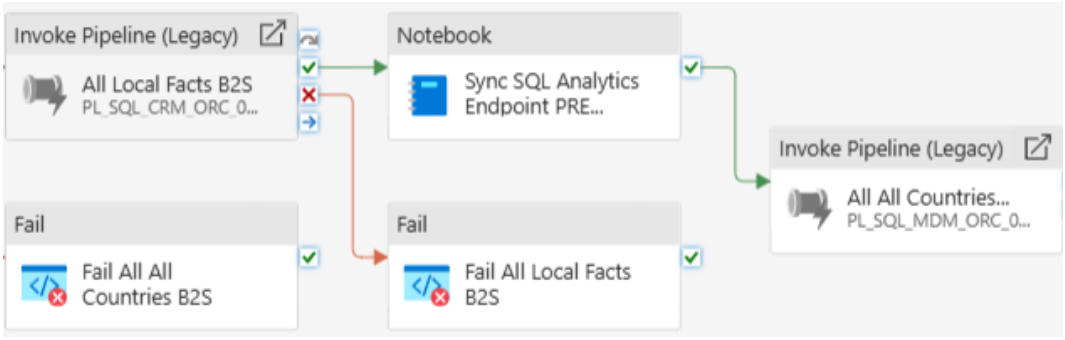


Figura 3.18: Seconda parte pipeline F2G_All - Elaborazione fatti

A tale scopo, viene invocata la pipeline di orchestrazione ***AllLocalFactsB2S***, mostrata in Figura 3.19, che si occupa dell'elaborazione dei fatti per ciascuna istanza del CRM. La struttura di questa pipeline è analoga a quella descritta per le dimensioni. Ogni istanza del CRM ha un'activity dedicata all'elaborazione dei fatti e, per ottimizzare le prestazioni, le istanze sono suddivise in gruppi di tre. Ogni gruppo viene elaborato in parallelo, mentre all'interno di ciascun gruppo le istanze vengono processate in sequenza.

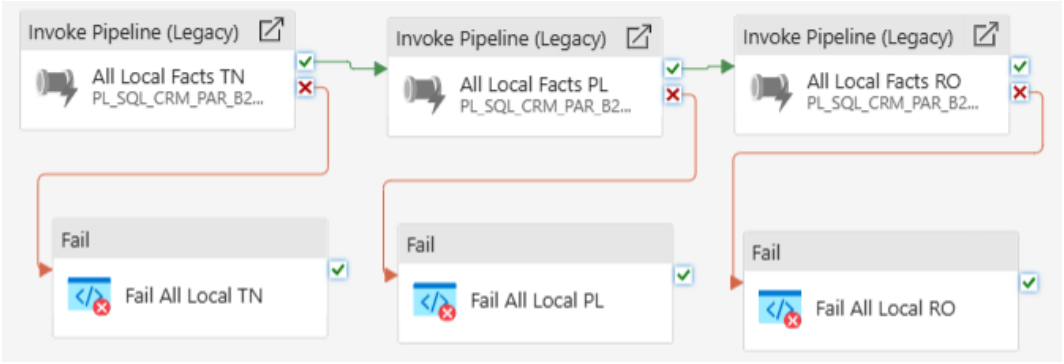


Figura 3.19: Pipeline AllFacts

Successivamente, ciascuna activity della pipeline invoca la ***AllFactsNotebook***, incaricata dell'elaborazione di tutti i fatti del modello per una specifica istanza. Come mostrato in Figura 3.20, questa pipeline esegue un notebook dedicato per ogni fatto, seguendo il processo descritto nel capitolo 3.3.2, prendendo come riferimento il caso di Working Day.

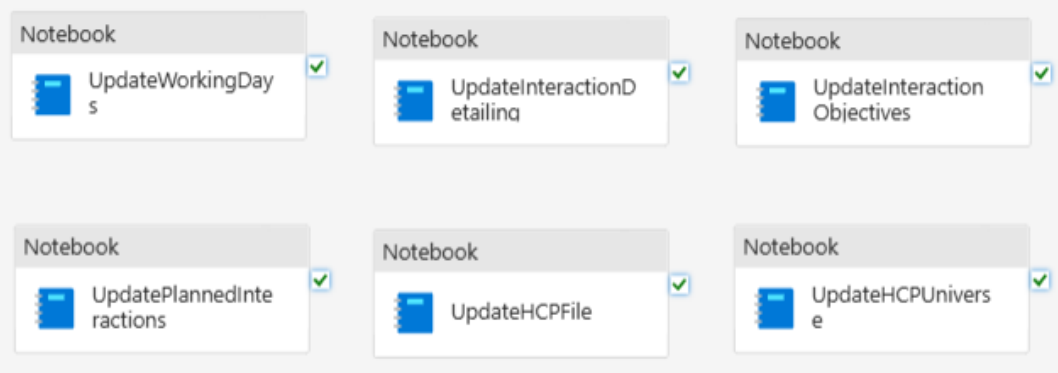


Figura 3.20: Pipeline AllFactsNotebook

Una volta completata l'elaborazione dei fatti per ciascuna istanza del CRM, si procede con il processo di armonizzazione delle loro versioni locali. A tal proposito seguendo il flusso descritto in Figura 3.18, viene invocata la pipeline ***AllAllCountriesFacts***, illustrata in Figura 3.21, che si occupa di eseguire un'operazione di UNION tra tutte le versioni locali dei fatti, seguendo lo stesso principio adottato per le dimensioni.

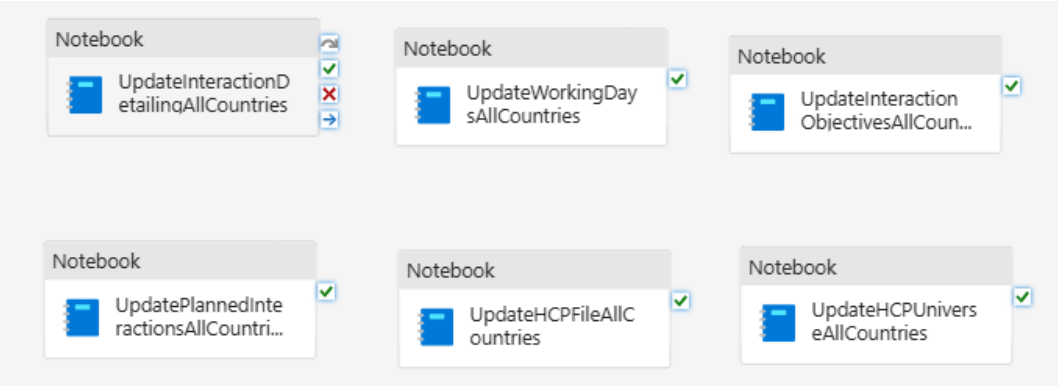


Figura 3.21: Pipeline AllAllCountriesFacts

3.4.3 Silver to Gold

Dopo aver definito una prima versione delle dimensioni e dei fatti del modello attraverso il processo di trasformazione e armonizzazione, il flusso della pipeline F2G_All, illustrata in Figura 3.22, prosegue con la transizione al Layer Gold. In questa fase, le dimensioni e i fatti vengono consolidati nella loro versione definitiva, come spiegato nel capitolo 3.3.3, dando forma al modello analitico centralizzato.

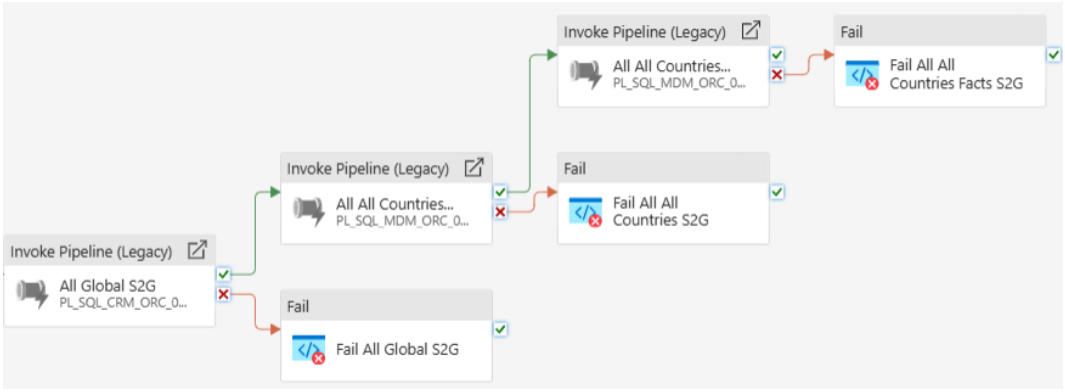


Figura 3.22: Terza parte pipeline F2G_All - Silver to Gold

Inizialmente è stato necessario predisporre la materializzazione delle dimensioni globali attraverso la pipeline ***AllGlobalS2G***, illustrata in Figura 3.23. Questa pipeline, per ogni dimensione, esegue una Stored Procedure che si occupa di materializzare la vista che legge dalla dimensione Silver, creando la corrispondente tabella materializzata nel layer Gold.

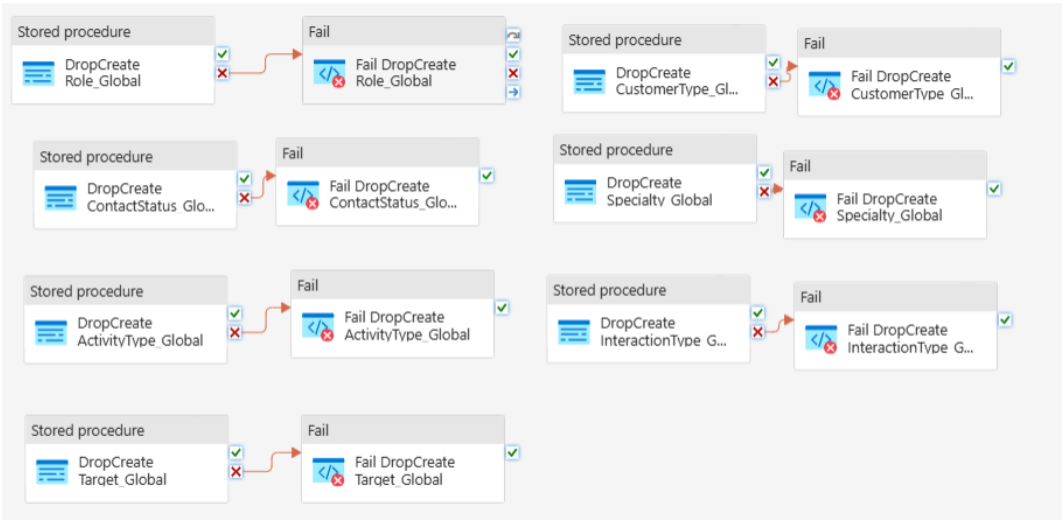


Figura 3.23: Pipeline AllGlobalS2G

Successivamente, il flusso prosegue con la materializzazione delle dimensioni armonizzate, a seguito del processo di UNION, invocando la pipeline ***AllAllCountriesS2G***, illustrata in Figura 3.24. Questa pipeline esegue, per ciascuna dimensione armonizzata, una Stored Procedure che, come spiegato in precedenza, materializza i dati nel Layer Gold, consolidando la versione definitiva della dimensione e contribuendo alla costruzione del modello analitico

centralizzato. Questo meccanismo consente di intercettare eventuali anomalie durante l'elaborazione del dato: se un errore viene rilevato, la materializzazione non avviene, il flusso lo segnala e si può intervenire tempestivamente senza impatti sul business.

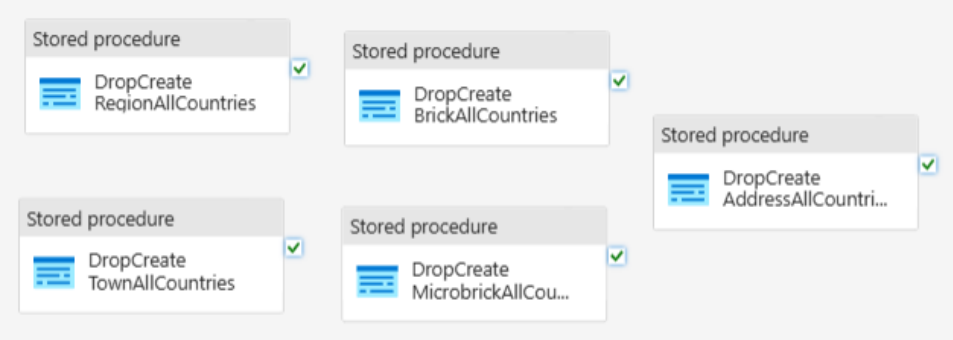


Figura 3.24: Pipeline AllAllCountriesS2G

Infine, una volta completata la materializzazione delle dimensioni, il flusso prosegue con quella dei fatti tramite la pipeline ***AllAllCountriesFactsS2G***, illustrata in Figura 3.25. Questa esegue, per ogni fatto, una Stored Procedure che materializza la vista che legge dalla versione Silver, creando la corrispondente tabella nel Layer Gold, seguendo lo stesso principio applicato per le dimensioni.

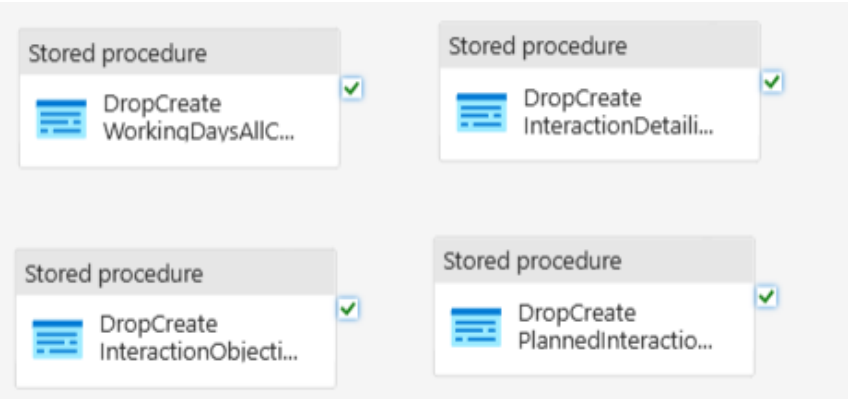


Figura 3.25: Pipeline AllAllCountriesFactsS2G

In conclusione, l'implementazione di un sistema di orchestrazione automatizzato ha reso la gestione del flusso dati efficiente e scalabile, dall'acquisizione iniziale alla definizione del modello analitico centralizzato. L'integrazione di pipeline e notebook in Microsoft Fabric ha garantito coerenza, affidabilità e

aggiornamenti continui, riducendo al minimo gli interventi manuali. Il flusso implementato ha trasformato i dati grezzi delle diverse istanze del CRM in entità strutturate, armonizzandoli per fornire un'unica versione del dato.

3.5 Data Consumption

Dopo aver completato la fase di Data Preparation, il modello analitico è ora pronto per essere integrato in Power BI, dove verranno sviluppati e testati i primi report prototipali da presentare al cliente. In questa fase, il focus si sposta sulla fruizione del dato, con l'obiettivo di mostrare come le informazioni strutturate nel Layer Gold vengano utilizzate per supportare le analisi aziendali e il processo decisionale.

Per garantire chiarezza e leggibilità, verrà illustrata la struttura del modello analitico, prendendo come riferimento il fatto **Working Days**.

In Figura 3.26 viene illustrata una sezione del modello che descrive il fatto in esame, evidenziando le dimensioni chiave che lo compongono.

Grazie alla struttura del modello, è possibile effettuare analisi sia a livello locale, per singolo paese, sia a livello globale, confrontando più regioni. Questo approccio consente di ottenere insight dettagliati sulle giornate lavorative della forza vendita, analizzando le differenze tra le attività sul campo, come le visite dirette ai medici, e quelle fuori dal territorio, come partecipazioni a congressi o corsi di formazione.

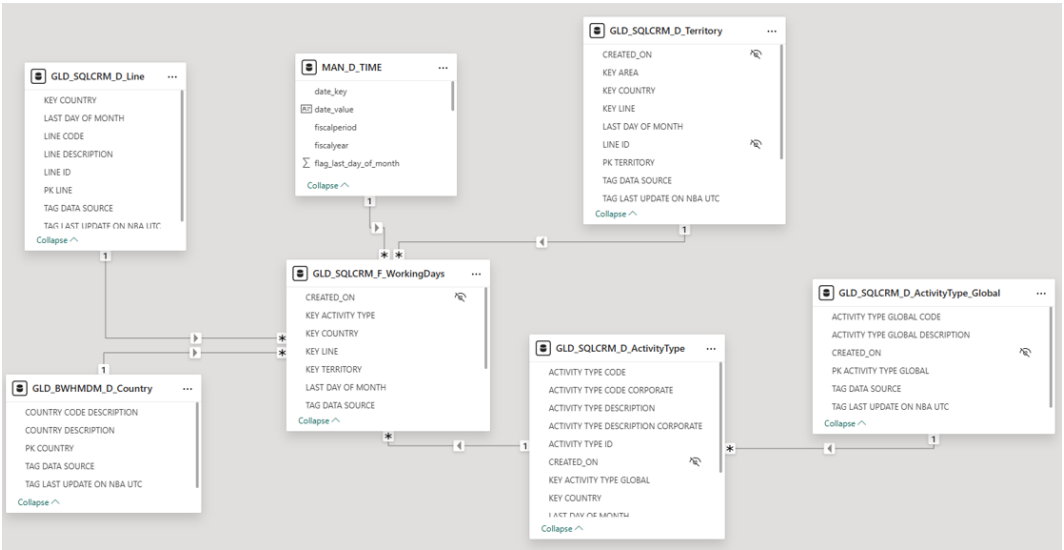
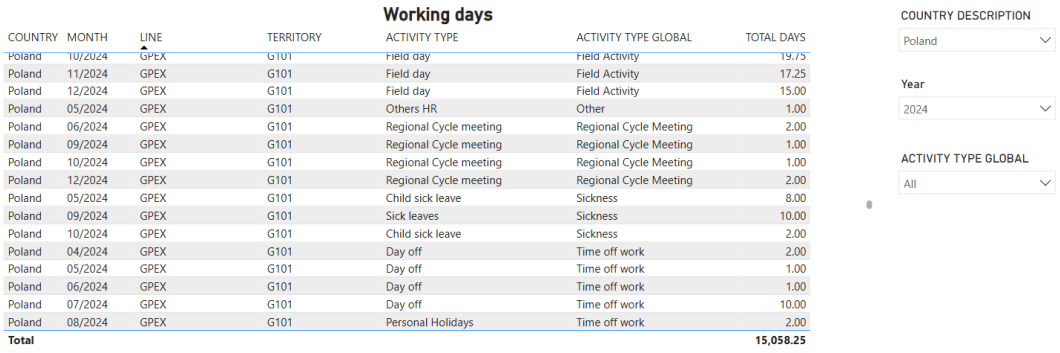


Figura 3.26: Struttura Modello Analitico - Working Days

Successivamente, sulla base del seguente modello, è stato necessario sviluppare un report prototipale, illustrato in Figura 3.27, con l'obiettivo di mostrare al cliente che la metrica calcolata rispettasse le sue aspettative. Dal punto di vista tecnico, il report non ha richiesto l'implementazione di complesse operazioni DAX, in quanto la misura *TOTAL_DAYS* viene calcolata direttamente a back-end, adattandosi dinamicamente ai filtri applicati. In base all'aggregazione selezionata, il valore della metrica varia, consentendo un'analisi più approfondita del dato in diverse prospettive.

Per migliorare la navigabilità del report e offrire una maggiore interattività, sono stati integrati filtri dinamici, che consentono di segmentare i dati in base al tipo di attività, all'anno di riferimento e al paese selezionato, permettendo così un'analisi dettagliata e flessibile.



Working days

COUNTRY	MONTH	LINE	TERRITORY	ACTIVITY TYPE	ACTIVITY TYPE GLOBAL	TOTAL DAYS
Poland	10/2024	GPEX	G101	Field day	Field Activity	19.75
Poland	11/2024	GPEX	G101	Field day	Field Activity	17.25
Poland	12/2024	GPEX	G101	Field day	Field Activity	15.00
Poland	05/2024	GPEX	G101	Others HR	Other	1.00
Poland	06/2024	GPEX	G101	Regional Cycle meeting	Regional Cycle Meeting	2.00
Poland	09/2024	GPEX	G101	Regional Cycle meeting	Regional Cycle Meeting	1.00
Poland	10/2024	GPEX	G101	Regional Cycle meeting	Regional Cycle Meeting	1.00
Poland	12/2024	GPEX	G101	Regional Cycle meeting	Regional Cycle Meeting	2.00
Poland	05/2024	GPEX	G101	Child sick leave	Sickness	8.00
Poland	09/2024	GPEX	G101	Sick leaves	Sickness	10.00
Poland	10/2024	GPEX	G101	Child sick leave	Sickness	2.00
Poland	04/2024	GPEX	G101	Day off	Time off work	2.00
Poland	05/2024	GPEX	G101	Day off	Time off work	1.00
Poland	06/2024	GPEX	G101	Day off	Time off work	1.00
Poland	07/2024	GPEX	G101	Day off	Time off work	10.00
Poland	08/2024	GPEX	G101	Personal Holidays	Time off work	2.00
Total						**15,058.25**

Figura 3.27: Report Working Days

In conclusione, l'utilizzo di Power BI in questa fase ha avuto un ruolo strategico nel processo di validazione del modello. In base agli accordi presi con il cliente, la nostra attività si è focalizzata esclusivamente sulla costruzione del modello analitico centralizzato, mentre lo sviluppo della reportistica finale sarà gestito da un'azienda esterna. Tuttavia, la realizzazione di report prototipali si è rivelata essenziale per verificare la correttezza del modello risultante, permettendo al cliente di esplorare le metriche chiave e confermare che l'elaborazione dei dati rispondesse ai requisiti richiesti.

Conclusioni

L'attività di progetto descritta in questa tesi si è conclusa con successo, segnando un avanzamento significativo nell'evoluzione della Data Platform aziendale. L'obiettivo principale è stato la realizzazione di un modello analitico centralizzato per il Sales Excellence, un sistema pensato per ottimizzare i processi di vendita attraverso il monitoraggio e l'analisi di dati chiave. Questo modello fornisce una visione unificata delle performance aziendali, consentendo analisi sia a livello locale che globale e offrendo insight strategici al top management per supportare il processo decisionale. L'integrazione dei dati provenienti dalle diverse istanze locali del CRM ha permesso di armonizzare e consolidare le informazioni in un unico modello, supportato da un flusso automatizzato che ne garantisce l'aggiornamento continuo. L'adozione di tecnologie avanzate come Azure Data Factory e Microsoft Fabric ha assicurato scalabilità e gestione efficiente dei dati, permettendo all'azienda di ottenere insight accurati in tempo reale e riducendo la necessità di interventi manuali. Un valore aggiunto fondamentale della Data Platform è la capacità di migliorare la governance e la qualità dei dati, rendendoli accessibili e interpretabili per tutti gli stakeholder. Il consolidamento delle informazioni ha eliminato la frammentazione tra le diverse fonti, creando un unico punto di riferimento per l'analisi delle performance.

Il progetto continuerà a evolversi nei prossimi anni, con il completamento previsto entro il 2026. Sono stati già identificati nuovi Use Case da integrare nella Data Platform, ampliando la capacità di analisi e reporting. In particolare, per lo Use Case descritto, è prevista l'integrazione dei dati provenienti dal CRM di Cina e Stati Uniti, ampliando la copertura globale del modello analitico e consentendo una gestione più accurata delle performance aziendali su scala internazionale.

Bibliografia

[1] Apache iceberg. https://iceberg.apache.org.

[2] Hello from apache hudi — apache hudi. https://hudi.apache.org.

[3] Amazon Web Services. What is a data lake? https://aws.amazon.com/what-is/data-lake.

[4] M. Armbrust, A. Or, C. Torres, S. Yurchenko, M. Zhu, R. Xin, P. Saraf, S. Meng, S. Ulanov, X. Li, G. Kannan, S. Paranjpye, S. Liaw, G. Suryanarayana, and S. Bhagwan. Delta lake: high-performance acid table storage over cloud object stores. *Proceedings of the VLDB Endowment*, August 2020.

[5] Mohssine Bentaib, Abdelaaziz Ettaoufik, Abderrahim Tragha, and Mohamed Azzouazi. Storage structures in the era of big data: From data warehouse to lakehouse. *Journal of Theoretical and Applied Information Technology*, March 2024.

[6] Matteo Golfarelli and Stefano Rizzi. *Data Warehouse: Teoria e pratica della progettazione*. McGraw-Hill, Milano, Italia, seconda edizione edition, 2006. Cap 1. par. 1.4 e 1.5, Cap 5.

[7] William H. Inmon. *Building the Data Warehouse*. Wiley, New York, 1992. (cit. a p. 31).

[8] Marco Mantovani. Data lake, data warehouse, data lakehouse: il punto della situazione. https://www.iconsulting.biz/highlight/data-lake, 2022.

[9] Microsoft. Azure data factory documentation. https://learn.microsoft.com/en-us/azure/data-factory.

[10] Microsoft. Azure data lake storage documentation. https://learn.microsoft.com/en-us/training/modules/introduction-to-azure-data-lake-storage/.

[11] Microsoft. Che cos'è un data lake? https://azure.microsoft.com/it-it/resources/cloud-computing-dictionary/what-is-a-data-lake.

[12] Microsoft. Fabric data engineering documentation. https://learn.microsoft.com/en-us/fabric/data-engineering/data-engineering-overview.

[13] Microsoft. Fabric data warehousing documentation. https://learn.microsoft.com/it-it/fabric/data-warehouse/data-warehousing.

[14] Microsoft. Fabric onelake documentation. https://learn.microsoft.com/en-us/fabric/onelake/onelake-overview.

[15] Microsoft. Microsoft azure documentation. https://learn.microsoft.com/en-us/azure.

[16] Microsoft. Microsoft fabric documentation. https://learn.microsoft.com/it-it/fabric/get-started/microsoft-fabric-overview.

[17] Microsoft. Power bi documentation. https://learn.microsoft.com/it-it/power-bi/fundamentals/power-bi-overview.

[18] Iconsulting S.p.A. *Star Schema Fundamentals*. 2020. Architetture del Data Warehouse (cap. 2).

Ringraziamenti

Siamo arrivati alla fine di questo grande percorso ed è arrivato il momento di ringraziare tutte le persone che mi sono state vicine, che hanno sempre creduto in me e che mi hanno permesso di raggiungere questo grande traguardo.

Un sincero grazie al Prof. Stefano Rizzi e a tutto il mio team in Iconsulting, in particolare alla mia tutor Chiara Forlani. Il vostro supporto è stato essenziale per la realizzazione di questa tesi, offrendomi sempre disponibilità e aiuto per chiarire ogni mio dubbio o incertezza.

In questi ringraziamenti non posso non citare coloro che mi hanno donato la vita, ovvero i miei genitori.

A mio babbo Claudio, perché definirti solo un padre sarebbe riduttivo. Sei stato, e sei tuttora, il mio più grande migliore amico. Noi legati da una passione immensa e tu sai bene quale. Noi, con quel rapporto speciale che ci porta a parlare per ore, finché la mamma non si arrabbia... perché sì, spesso cenate alle dieci per colpa nostra.

A mia mamma Pia con la quale condivido il mio carattere, sì perché alla fine siamo uguali io e te sia negli aspetti positivi che negativi. Solo nell'ultimo periodo ho capito quanto siamo simili, perché se prendiamo un impegno non esistono mezze misure o tutto o niente. Siamo disposti a dormire anche poche ore a notte pur di raggiungere i nostri obiettivi.

A voi che mi avete sostenuto in questo percorso e avete trasmesso i valori che oggi mi rendono la persona che sono.

Un grazie di cuore a mio fratello Matteo, punto di riferimento costante, e a mia cognata Elisa, che per me è molto più di una cognata, una seconda sorella. Un pensiero speciale va alle mie nipotine, Aurora e Ginevra, la mia vera forza, capaci di regalarmi sorrisi anche nei momenti più difficili.

Un ringraziamento speciale a mia zia Carla, forse la persona più importante della mia vita dopo i miei genitori. Mi ha visto crescere e maturare, accompagnandomi con affetto e incoraggiamento fino al raggiungimento di questo grande traguardo.

Un ringraziamento speciale ai miei amici di sempre, che mi sono stati accanto nel bene e nel male. Il vostro supporto è stato essenziale per affrontare e portare a termine questo percorso, soprattutto negli ultimi mesi così intensi, in cui anche una semplice uscita con voi mi permetteva di staccare e ricaricare le energie.

Grazie alla mia dolce metà, Gaia, la mia migliore amica, confidente, morosa e maestra. Grazie per avermi supportato e sopportato, soprattutto in questi ultimi mesi, quando la fatica e le difficoltà mi facevano spesso innervosire e abbattere. Sei stata la mia guida, la mia ancora nei momenti più complessi. Questo traguardo è anche il tuo, perché senza di te, nulla sarebbe stato lo stesso.

Rileggendo questi ringraziamenti, mi accorgo che manca qualcuno... Ah sì, proprio tu, Martin. È il momento di fermarti un attimo . Lo so, non è nella tua indole, ma voltati indietro e renditi conto dello straordinario percorso che hai portato a termne inziato quasi sei anni fa. Un grazie profondo a me stesso, per non aver mai mollato, come in campo, anche nello studio, lavorando sodo per raggiungere i miei obiettivi. Oggi si chiude un capitolo della mia vita, ma sono pronto ad affrontare i prossimi con la stessa tenacia e grinta che mi hanno sempre contraddistinto.