POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica

Tesi di Laurea

Sistema di monitoraggio applicato ai processi ETL: sviluppo di una soluzione per l'integrazione e la visualizzazione dei log di esecuzione

Relatore
Prof. Daniele Apiletti
Tutor Aziendale
Dott. Filippo Balla

Candidato
Lisa Gamarro

Anno Accademico 2024-2025

Sommario

Nel contesto aziendale moderno, il monitoraggio dei processi ETL (Extract, Transform, Load) è un elemento cruciale per garantire contemporaneamente qualità, affidabilità e prestazioni dei flussi di integrazione dei dati.
La presente tesi, svolta in collaborazione con Mediamente Consulting srl, affronta il problema del monitoraggio delle esecuzioni di tali flussi implementati in SSIS (SQL Server Integration Services) e propone un sistema di analisi e visualizzazione basato su un'architettura dimensionale e una dashboard interattiva in Power BI.
L'analisi parte dallo studio del catalogo SSISDB, evidenziandone le principali limitazioni tra cui la complessità nell'interrogazione diretta, la frammentazione dell'informazione disponibile, l'assenza di una reportistica efficace. Per rispondere a queste criticità, è stato progettato un modello dati ottimizzato per la raccolta e l'analisi delle informazioni sulle esecuzioni, con l'obiettivo di migliorarne l'accessibilità e la fruibilità per gli utenti.
Il processo di sviluppo ha seguito un approccio strutturato che viene articolato in più fasi. In primo luogo, è stato condotto uno studio approfondito dei progetti analizzati, al fine di identificare i principali KPI da monitorare e i requisiti specifici per l'analisi delle esecuzioni. Successivamente, è stato definito un modello dati dimensionale che è stato popolato mediante un flusso ETL in SSIS, realizzato in base alle esigenze specifiche. Infine, il sistema è stato integrato con Power BI, permettendo la visualizzazione dei principali indicatori di performance e l'individuazione tempestiva di anomalie nelle esecuzioni dei pacchetti SSIS.
I risultati ottenuti dimostrano come un approccio metodico al monitoraggio possa migliorare significativamente la gestione operativa dei processi ETL, fornendo agli utenti uno strumento più efficace per l'analisi e la diagnostica degli errori.

Indice

Elenco delle tabelle 5

Elenco delle figure 6

1 Introduzione 7

2 Data Warehouse 11
2.1 Data Mart e Database relazionali . . . 12
2.1.1 Data Mart . . . 12
2.1.2 Database relazionale . . . 12
2.1.3 Interazione tra le tre strutture dati . . . 12
2.1.4 OLAP vs OLTP . . . 13
2.2 Integrare un Data Warehouse in un contesto aziendale . . . 14
2.3 Architetture dei Data Warehouse . . . 16
2.3.1 Star Schema . . . 16
2.3.2 Snowflake Schema . . . 16

3 I processi ETL 19
3.1 Le fasi del processo . . . 19
3.1.1 Estrazione dei dati . . . 20
3.1.2 Trasformazione dei dati . . . 20
3.1.3 Caricamento dei dati . . . 20
3.2 Vantaggi e limitazioni del processo ETL . . . 21
3.3 Confronto tra ETL e E-LT . . . 21
3.4 Framework aziendale . . . 22
3.4.1 L0 - Staging Area . . . 23
3.4.2 L1 - Operational Data Storage . . . 24
3.4.3 L2 - Publication Data Layer . . . 26
3.4.4 Tabelle di metadati . . . 26
3.5 Strumenti per l'ETL . . . 26
3.5.1 SQL Server Integration Services (SSIS) . . . 29

4 Monitoraggio 33
4.1 Fasi operative del Monitoraggio . . . 34
4.2 Aspetti complessi del Monitoraggio . . . 34
4.3 Monitoraggio nei flussi ETL . . . 35
4.3.1 Studi sul Monitoraggio . . . 36
4.4 Monitoraggio in SSIS . . . 37
4.4.1 Logging e SSISDB . . . 38
4.4.2 Event Handler . . . 40
4.4.3 Punti di forza e limiti del Monitoraggio in SSIS . . . 41

5 Progettazione del sistema di monitoraggio 43
5.1 I progetti . . . 43
5.2 Identificazione KPI da monitorare . . . 46
5.3 Scelta degli strumenti . . . 47
5.4 Definizione del modello . . . 48
5.5 Flusso SSIS per le due tabelle di catalogo . . . 51
5.6 Dashboard . . . 56

6 Conclusione 59

A Tabelle di log 65

B Esempi di query 69
B.1 Query di DLT per la vista executions . . . 69
B.2 Query di ODS per la vista executions . . . 70
B.3 Query di OUT . . . 71
B.4 Query di creazione e popolamento della dimensione PACCHETTO . . . 72
B.5 Query di MERGE per la tabella dei fatti . . . 73

C Categorie e pattern di errori e warning 75
C.1 Categorie di errore . . . 75
C.2 Categorie di warning . . . 77

Elenco delle tabelle

2.1 Confronto tra OLTP e OLAP 15
3.1 Confronto tra ETL ed E-LT. 22

Elenco delle figure

2.1 Architettura di un Data Warehouse 13
2.2 Snowflake schema vs Star schema 17
3.1 Processo ETL 19
3.2 Struttura del flusso di integrazione di Mediamente Consulting 23
3.3 Quadrante magico Gartner Software di integrazione 2024 27
5.1 Schema a stella del modello progettato 48
5.2 Flusso di controllo pacchetto 000_MAIN 52
5.3 Flusso di controllo pacchetto 010_L0_MOVIMENTI 52
5.4 Flusso di controllo pacchetto 100_STG_DLT 53
5.5 Flusso di dati per tabella DLT_EVENT_MESSAGES 54
5.6 Flusso di controllo pacchetto 200_OK_ODS 54
5.7 Flusso di controllo pacchetto 002_MAIN_L2 55
5.8 Flusso di controllo pacchetto 030_L2_ANAGRAFICHE 56
5.9 Dashboard relativa al cliente 1 58
5.10 Dashboard relativa al cliente 2 58

Capitolo 1

Introduzione

Nell'era della trasformazione digitale, la gestione e l'analisi dei dati aziendali sono diventate attività strategiche per qualsiasi organizzazione. La crescente complessità dei sistemi informativi, unita all'esigenza di integrare fonti di dati eterogenee, ha portato alla diffusione di **Data Warehouse** e processi di **ETL (Extract, Transform, Load)**, strumenti essenziali per supportare le decisioni aziendali basate su dati affidabili e consolidati. Tuttavia, con l'aumento della quantità di dati gestiti e della complessità dei flussi di integrazione, diventa sempre più importante garantire **un monitoraggio efficace dei processi ETL** per identificare errori, ottimizzare le prestazioni ed evitare interruzioni operative.
Il monitoraggio dei processi ETL rappresenta quindi un'attività cruciale per assicurare la **qualità, la tracciabilità e l'efficienza dei flussi di integrazione**. Senza un adeguato sistema di controllo, il rischio di ritardi nei processi, dati incoerenti o mancati aggiornamenti delle informazioni può compromettere le analisi aziendali e influenzare negativamente le decisioni strategiche. Per questo motivo, la possibilità di tracciare e analizzare in modo strutturato l'esecuzione dei pacchetti ETL diventa un elemento chiave per il successo di un'infrastruttura di data integration.

Il presente lavoro di tesi nasce dall'esperienza maturata durante un tirocinio presso Mediamente Consulting, azienda specializzata in Business Intelligence e Data Integration. L'obiettivo principale è stato la **progettazione e implementazione di un sistema di monitoraggio per i flussi ETL sviluppati in SQL Server Integration Services (SSIS)**. Per raggiungere questo scopo, è stato sviluppato un modello dati a supporto del monitoraggio, implementato un processo ETL per la raccolta e l'organizzazione dei dati di log e realizzata una dashboard interattiva su Power BI per la visualizzazione e l'analisi dei dati raccolti. La soluzione proposta consente di identificare eventuali errori, tracciare i tempi di esecuzione, individuare colli di bottiglia e valutare l'affidabilità dei flussi ETL.

Il lavoro è articolato in più capitoli, ciascuno dei quali approfondisce un aspetto specifico della progettazione e realizzazione del sistema di monitoraggio.
I primi tre capitoli a seguire sono principalmente teorici, presentano una revisione dello

stato dell'arte riguardante tutte le tematiche e gli strumenti utilizzati poi per la realizzazione del progetto in sè, mentre i successivi si concentrano più sull'aspetto pratico del progetto, descrivendo i dettagli dell'imlementazione del modello e del sistema di monitoraggio, oltre che i risultati ottenuti.

Il capitolo 2 introduce il concetto di **Data Warehouse**, descrivendone la struttura e il ruolo all'interno di un'organizzazione. Viene fatta una distinzione tra le varie strutture dati attualmente esistenti e viene illustrato il confronto tra differenti modelli di gestione dei dati. Infine, viene evidenziata l'importanza dell'integrazione di un Data Warehouse in un contesto aziendale e vengono presentate le principali architetture utilizzate.

Nel terzo capitolo vengono approfondite le **fasi del processo ETL**, ovvero l'estrazione, la trasformazione e il caricamento dei dati. Vengono poi analizzati i vantaggi e le limitazioni di questo approccio, confrontandolo con la metodologia E-LT (Extract, Load, Transform), alternativa che si basa su una logica diversa di elaborazione dei dati. Infine, viene illustrato il **framework aziendale** adottato da Mediamente Consulting, basato su tre livelli. In chiusura, vengono introdotti gli strumenti principali per l'ETL, con un focus particolare su quello utilizzato nella trattazione successiva, ovvero **SQL Server Integration Services (SSIS)**.

Il capitolo 4 affronta il tema del **monitoraggio nei flussi ETL**, spiegandone l'importanza e le sfide operative. Vengono presentate le tecniche di logging e il catalogo SSISDB, che costituisce la fonte principale dei dati di monitoraggio nel progetto. Vengono inoltre illustrati i metodi di gestione degli eventi in SSIS, ed infine si analizzano i punti di forza e i limiti delle soluzioni di monitoraggio offerte da SSIS, individuando le criticità affrontate nel progetto.

Nel capitolo 5 viene descritto nel dettaglio il **sistema di monitoraggio implementato**. Dopo un'introduzione ai progetti aziendali su cui si basa il lavoro, viene effettuata un'identificazione dei KPI utili per il monitoraggio. Successivamente, viene presentato il modello dati a stella, destinazione del flusso ETL, e viene descritto il processo di trasformazione e caricamento che permette di popolare la struttura del modello di monitoraggio. Viene poi introdotta la **dashboard Power BI** sviluppata per la rappresentazione grafica dei dati di monitoraggio. Viene illustrata la struttura della dashboard, le metriche chiave visualizzate e le funzionalità implementate per l'interazione con gli utenti.

L'ultimo capitolo raccoglie infine le **considerazioni finali** sul lavoro svolto, evidenziando i principali risultati ottenuti e le possibili evoluzioni future del sistema di monitoraggio.

Questo lavoro di tesi dimostra come un approccio strutturato al monitoraggio dei processi ETL possa migliorare significativamente la qualità e l'efficienza dei flussi di integrazione dati in un contesto aziendale. L'integrazione tra SSIS, un modello dati ottimizzato e strumenti di Business Intelligence come Power BI consente di ottenere una visione chiara e dettagliata delle performance dei processi ETL, facilitando l'individuazione di problemi e l'ottimizzazione dei flussi.

Attraverso la progettazione e implementazione del sistema descritto, si offre una soluzione pratica ed efficace per il controllo e l'analisi delle esecuzioni SSIS, con l'obiettivo di supportare le aziende nella gestione consapevole e strategica delle proprie operazioni di data integration.

Capitolo 2

Data Warehouse

Il termine **Data Warehouse** (DWH) si riferisce a un sistema centralizzato di raccolta e archiviazione di dati provenienti da diverse fonti aziendali, è progettato per supportare l'analisi e il processo decisionale in un contesto aziendale. A differenza di un tradizionale database operativo definito anche *transazionale*, che è ottimizzato per le operazioni quotidiane di un'organizzazione come l'elaborazione di transazioni, un Data Warehouse è destinato a memorizzare grandi volumi di dati storici e a consentire l'analisi delle informazioni nel tempo. Il principale obiettivo di un data warehouse è fornire un'unica fonte affidabile di dati consolidati che possano essere utilizzati per generare report analitici, eseguire query complesse e supportare il processo di Business Intelligence (BI).
I Data Warehouse si contraddistinguono per alcune caratteristiche fondamentali che li differenziano dai database operativi tradizionali. Queste sono state definite da William H. Hinmon [9], che descrive il DWH come una struttura dati:

1. **Orientata ai soggetti**: i dati raccolti sono focalizzati sugli aspetti rilevanti per i soggetti dell'organizzazione, e le informazioni vengono strutturate in modo da rispondere alle esigenze decisionali degli utenti piuttosto che a quelle operative dei singoli processi;

2. **Integrata**: i dati vengono raccolti a partire da una molteplicità di fonti eterogenee sia aziendali che esterne e possono essere di natura diversa: file di log, database operativi, sistemi esterni, API e persino sorgenti non strutturate. Tutte queste informazioni vengono rappresentate in un formato uniforme, affinché possano consentire una visione complessiva e coerente delle informazioni aziendali;

3. **Variante nel tempo**: il DWH mantiene un ampio orizzonte temporale, memorizzando i dati storici in modo da supportare analisi e reportistiche che considerano l'evoluzione nel tempo;

4. **Non volatile**: Una volta che i dati sono caricati nel data warehouse, non vengono modificati, ma solo letti. Questo fa in modo che i dati siano consistenti nel tempo e non vengano cancellati o alterati da operazioni quotidiane.

2.1 Data Mart e Database relazionali

I Data Warehouse sono fondamentali all'interno di un'organizzazione ma non sono l'unica modalità di archiviazione dei dati disponibile e utilizzata. É importante dunque esplorare le varie soluzioni possibili in termini di caratteristiche e funzionalità.

2.1.1 Data Mart

Un **Data Mart** è un DWH di tipo dipartimentale, progettato per soddisfare le esigenze di un particolare gruppo di utenti o di un'area funzionale specifica all'interno di un'organizzazione. La struttura utilizzata è la stessa ma, a differenza del Data Warehouse che raccoglie i dati da tutta l'azienda, un data mart è focalizzato su un sottoinsieme di informazioni, come quelle relative a un dipartimento, funzione o processo aziendale specifico (come le vendite, la finanza o il marketing). I Data Mart consentono agli utenti di accedere a dati più pertinenti e di dimensioni più contenute, riducendo i tempi di risposta delle query e semplificando l'analisi.

2.1.2 Database relazionale

Un database è una collezione di dati organizzati, utilizzati per raccogliere, memorizzare e gestire informazioni. Esistono diverse tipologie di database che utilizzano strutture differenti per memorizzare e gestire i dati. Tra le varie, spiccano in particolare i **database relazionali**, uno dei modelli tradizionali usati spesso in molti contesti aziendali.
Si tratta di sistemi progettati per la gestione dei dati operativi quotidiani che vengono immagazzinati in tabelle collegate tra loro tramite chiavi primarie ed esterne, seguendo un modello normalizzato che minimizza la ridondanza e assicura l'integrità dei dati. I database relazionali sono ideali per gestire operazioni transazionali come la registrazione di ordini, pagamenti e altre attività aziendali quotidiane, che richiedono aggiornamenti frequenti e in tempo reale. Tuttavia, questa struttura normalizzata e la gestione delle transazioni non sono adatte per le analisi complesse tipiche di un data warehouse, in quanto le query analitiche potrebbero risultare lente e poco efficient, a causa della necessità di numerose operazioni di join su tabelle altamente normalizzate.

2.1.3 Interazione tra le tre strutture dati

In un'architettura di data warehousing, i vari data warehouse, data mart e database interagiscono in modo complementare per supportare le esigenze di gestione, analisi e accesso ai dati a vari livelli dell'organizzazione.
Il data warehouse rappresenta la componente centrale dell'architettura, in cui vengono centralizzati e memorizzati grandi volumi di dati provenienti da diverse fonti aziendali. Tra le varie sorgenti si collocano i database relazionali, utilizzati per gestire i dati operativi quotidiani e alimentare il data warehouse con i dati più recenti e aggiornati, che vengono successivamente elaborati, aggregati e storicizzati per scopi analitici.
In aggiunta al data warehouse, ci possono essere diversi data mart dipartimentali, progettati per soddisfare le esigenze di specifici dipartimenti o funzioni aziendali, alleggerendo il

carico complessivo della struttura centrale e migliorandone conseguentemente l'efficienza. L'interazione tra database, data warehouse e data mart , raffigurata in figura 2.1, consente di integrare e trasformare i dati operativi in informazioni strategiche, supportando le decisioni aziendali a tutti i livelli.

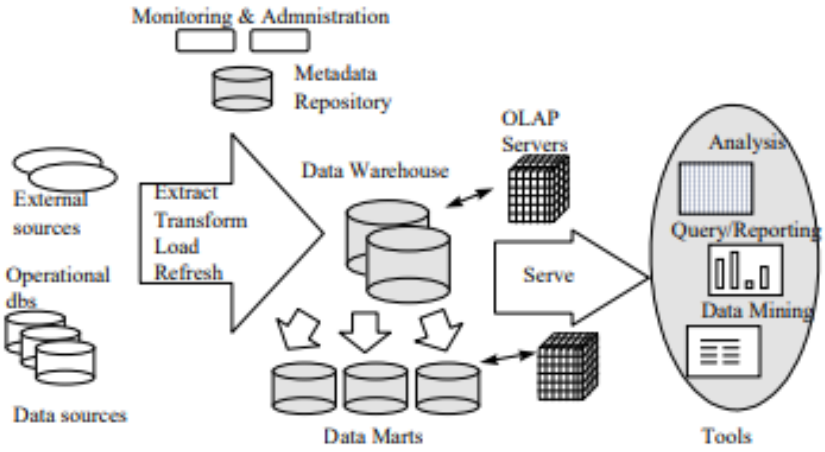

Figura 2.1. Architettura di un Data Warehouse

A partire da database operazionali e altre fonti esterne, avvengono la pulizia, la trasformazione e l'integrazione dei dati, e il loro successivo caricamento all'interno del Data Warehouse. Inoltre, sono previsti meccanismi per l'aggiornamento periodico del data warehouse, al fine di riflettere le modifiche avvenute nelle fonti originali e per l'eventuale archiviazione di dati storici su storage meno performanti.
Oltre al data warehouse principale, possono essere presenti diversi data mart dipartimentali dedicati a specifiche aree aziendali. La gestione e l'archiviazione dei dati avviene attraverso uno o più server di data warehouse, che offrono viste multidimensionali ai vari strumenti di analisi, tra cui strumenti di interrogazione, reportistica, analisi avanzata e data mining. Infine, un repository centralizzato consente la gestione e l'archiviazione dei metadati, supportando strumenti dedicati al monitoraggio e all'amministrazione dell'intero sistema di data warehousing.

2.1.4 OLAP vs OLTP

Nell'ambito dei sistemi di gestione dei dati, è importante distinguere tra **OLAP** (Online Analytical Processing) e **OLTP** (Online Transaction Processing), due approcci con finalità e caratteristiche differenti che rispondono a esigenze molto diverse all'interno di un'organizzazione.

OLTP è progettato per gestire transazioni quotidiane e operazioni in **tempo reale**. I sistemi OLTP sono tipicamente utilizzati in ambienti aziendali dove è necessario registrare

e gestire continuamente transazioni, come acquisti, vendite, pagamenti e altre operazioni di business. Questi sistemi sono ottimizzati per gestire un grande numero di piccole transazioni simultanee e richiedono alta disponibilità, velocità e integrità dei dati. Un database relazionale tradizionale è un esempio di sistema OLTP, poiché gestisce dati altamente normalizzati, riducendo la ridondanza e ottimizzando le operazioni di lettura e scrittura frequenti e rapide. In un sistema OLTP, l'attenzione è focalizzata sulla **coerenza** e sull'**affidabilità** delle operazioni transazionali, garantendo che ogni transazione sia correttamente registrata, anche in caso di guasti.

Al contrario, **OLAP** è focalizzato sull'analisi dei dati e sul **supporto alle decisioni strategiche**. I sistemi OLAP sono utilizzati per eseguire analisi complesse su grandi volumi di dati, permettendo agli utenti di esplorare e aggregare informazioni storiche attraverso dimensioni multiple, come tempo, geografia, prodotto e altre variabili. Un Data Warehouse è tipicamente un sistema OLAP, progettato per eseguire operazioni di lettura su dati storici con una struttura denormalizzata, che permette di velocizzare le query analitiche e ridurre i tempi di risposta. A differenza di un sistema OLTP, un sistema OLAP **non è ottimizzato per operazioni in tempo reale** o per la gestione di transazioni quotidiane. Piuttosto, è studiato per l'analisi di dati aggregati e la generazione di report complessi che supportano le decisioni aziendali a lungo termine. Le operazioni in un sistema OLAP riguardano principalmente la lettura dei dati piuttosto che la loro modifica.

La **differenza principale** tra OLTP e OLAP risiede dunque nell'uso che se ne fa all'interno dell'organizzazione. OLTP è ottimizzato per la gestione di operazioni transazionali quotidiane e la manutenzione dei dati in tempo reale, mentre OLAP è pensato per analizzare grandi volumi di dati e supportare decisioni strategiche complesse. Di conseguenza, anche il tipo di utente varierà a seconda del sistema: OLTP è utilizzato da utenti operativi e transazionali, come operatori di sistema, impiegati, customer service e venditori, mentre OLAP è utilizzato da utenti analitici e decisionali, tra cui analisti, manager e specialisti in BI [22].

Per una visione complessiva, le principali differenze tra i due sistemi sono riassunte nella tabella 2.1.
Risulta quindi chiaro che, proprio per le diversità evidenziate, in molte architetture aziendali moderne i sistemi OLTP e OLAP coesistono, con i dati provenienti da un sistema OLTP che alimentano il data warehouse per l'analisi e la reportistica OLAP.

2.2 Integrare un Data Warehouse in un contesto aziendale

Nei paragrafi precedenti sono state delineate le caratteristiche, il funzionamento e l'utilizzo di un DWH. Tuttavia, è necessario fare un ulteriore passo avanti e approfondire il motivo per cui integrare una struttura di questo tipo risulti davvero indispensabile in un contesto aziendale. La nascita e l'evoluzione di queste architetture risalgono agli anni Novanta, un periodo segnato da un significativo sviluppo nel panorama aziendale, data la

Caratteristica	**OLTP**	**OLAP**
Funzione	Gestione giornaliera	Supporto alle decisioni
Progettazione	Orientata alle applicazioni	Orientata al soggetto
Frequenza	Giornaliera	Sporadica
Dati	Recenti, dettagliati	Storici, riassuntivi, multidimensionali
Accesso	Read/Write	Read
Tipo utenti	Operatori	Manager
# Record acceduti	Decine	Migliaia
Dimensione DB	100 MB - GB	100 GB - TB

Tabella 2.1. Confronto tra OLTP e OLAP

crescita di queste ultime sia in termini di complessità che di espansione, anche a livello globale. L'accelerazione di questa crescita ha spinto le aziende a ricercare soluzioni innovative per mantenere la propria competitività. I sistemi operazionali esistenti garantivano sufficienti informazioni quotidiane ma, nonostante la grande quantità di dati operazionali raccolti, la tecnologia disponibile non era sufficiente a rendere queste informazioni fruibili per l'analisi strategica. La necessità di strumenti per procurarsi tali informazioni portò dunquealla rivoluzione dell'immagazzinamento dei dati, con la conseguente creazione del Data Warehouse [19].
Tuttavia, la transizione verso questa nuova modalità di gestione dei dati non è stata né semplice né rapida. Nonostante i numerosi e decisivi vantaggi che un data warehouse apporta, i dirigenti aziendali nutrivano molte perplessità, in particolare riguardo ai costi, alla percezione della complessità e della durata della realizzazione di un progetto di questo tipo, oltre alla difficoltà di misurare i benefici immediati e al rischio di una gestione inefficace dei dati. Nonostante queste iniziali preoccupazioni, oggi è ampiamente riconosciuta la capacità di un DWH di centralizzare e integrare dati provenienti da diverse fonti, garantendo maggiore coerenza, qualità e affidabilità delle informazioni. Tra i benefici a lungo termine rientra anche la storicizzazione dei dati, tramite cui le aziende possono monitorare le proprie performance nel tempo, supportando analisi predittive e strategiche, oltre ad una migliore scalabilità, sicurezza e possibilità di introdurre l'automazione dei processi di reporting, riducendo i costi operativi e ottimizzando il processo decisionale. Questi vantaggi, uniti alla crescente accessibilità delle tecnologie di data warehousing, hanno reso l'implementazione di questa soluzione un elemento fondamentale per un numero crescente di aziende che puntano ad una gestione dei dati sempre più efficiente e competitiva.

2.3 Architetture dei Data Warehouse

Nel contesto dei DWH, esistono due principali modelli di architettura utilizzati per strutturare i dati: lo **Star Schema** (schema a stella) e lo **Snowflake Schema** (schema a fiocco di neve). Entrambi i modelli organizzano i dati in tabelle centrali di fatti e tabelle di dimensioni, ma differiscono per il modo in cui le tabelle di dimensioni sono strutturate e collegate tra loro.

2.3.1 Star Schema

Lo star schema è il modello più semplice e intuitivo, la cui struttura richiama proprio quella di una stella, in cui la tabella centrale, chiamata **tabella dei fatti**, è circondata da **tabelle di dimensioni** che descrivono gli attributi dei fatti (come il tempo, la geografia, i prodotti, ecc.). La tabella dei fatti, che è quella dominante anche per le sue notevoli dimensioni, racchiude le informazioni rilevanti di un evento di interesse per l'impresa, come possono essere le vendite, le spedizioni, gli acquisti) e contiene delle quantità, chiamate misure, che ne descrivono degli aspetti quantitativi aggregati (numero di unità vendute, prezzo unitario e similari) [5]. Le tabelle di dimensioni sono **denormalizzate**, il che significa che contengono tutte le informazioni necessarie per una determinata dimensione in un'unica tabella, riducendo la necessità di complesse operazioni di join. Questo schema è ottimizzato per velocizzare le query analitiche, poiché la sua struttura semplice consente di accedere rapidamente ai dati e generare report complessi con un **numero limitato di join** tra le tabelle. Tuttavia, la denormalizzazione può comportare una certa ridondanza nei dati e una maggiore necessità di spazio di archiviazione.

2.3.2 Snowflake Schema

Lo Snowflake Schema è una versione più complessa e normalizzata dello Star Schema. In questo modello, le tabelle di dimensioni sono **normalizzate**, cioè suddivise in più tabelle collegate tra loro (da cui il nome), con l'obiettivo di **ridurre la ridondanza** e migliorare l'efficienza nello spazio di archiviazione. Sebbene la normalizzazione riduca la duplicazione dei dati, questa struttura comporta una **maggiore complessità nelle query**, poiché è necessario eseguire più join tra le tabelle per ottenere informazioni complete su una dimensione. Lo snowflake schema è più adatto a situazioni in cui la gestione dello spazio di archiviazione è critica, ma le query potrebbero risultare più lente a causa della necessità di elaborazioni più complesse.

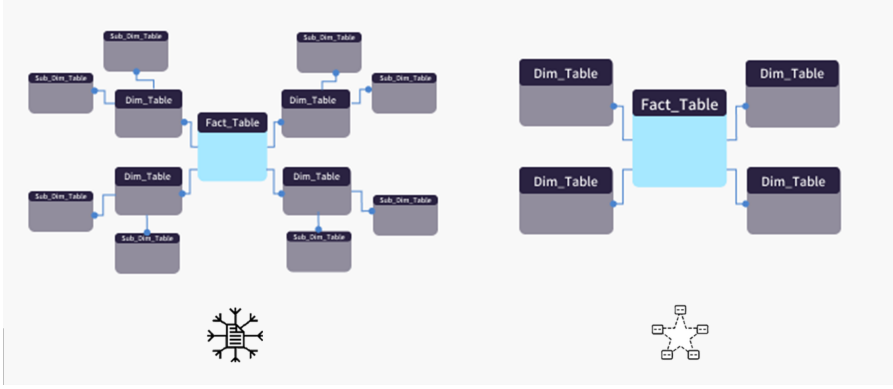

Figura 2.2. Snowflake schema vs Star schema

La scelta tra Star o Snowflake schema dipende dalle specifiche esigenze aziendali: lo Star Schema è preferito per la sua semplicità e velocità nelle query, mentre lo Snowflake Schema è più adatto quando è necessario ottimizzare l'uso dello spazio di archiviazione e ridurre la ridondanza dei dati. Entrambi i modelli sono utilizzati per facilitare l'accesso e l'analisi dei dati in un data warehouse, ma la scelta del modello dipende da trade-off tra prestazioni e gestione dello spazio. Un altro aspetto determinante nella scelta è l'utilizzo di uno specifico strumento di visualizzazione: alcuni strumenti di Business Intelligence e reporting, infatti, prediligono un modello piuttosto che l'altro, influenzando così la decisione progettuale. Esistono strumenti, tra cui Power BI, che sono ottimizzati per lavorare con lo Star Schema, traendo vantaggio da una struttura più denormalizzata. D'altra parte, strumenti più orientati all'analisi SQL diretta, come soluzioni basate su OLAP multidimensionale, possono risultare più efficaci con la minore ridondanza e maggiore efficienza di storage dello Snowflake Schema.

Capitolo 3

I processi ETL

Nel passaggio tra i dati sorgente e il Data Warehouse è indispensabile effettuare delle operazioni in grado di unificare i dati, renderli corerenti ed integrati e fruibili negli step successivi. Per questo è necessario introdurre le pipeline **ETL** (Extract, Transform, Load), che contengono una serie di operazioni strutturate che elaborano i dati in maniera standardizzata, consentendo il passaggio dai dati grezzi ad una nuova versione uniformata e consolidata, di alta qualità e di semplice interpretazione ai fini delle decisioni strategiche. L'ETL è un processo cardine nell'ambito della gestione dei dati, soprattutto nel contesto dei Data Warehouse e dei sistemi di Business Intelligence. Esso consente di raccogliere dati da diverse fonti, trasformarli per renderli utili alle esigenze aziendali e caricarli in un sistema di destinazione per analisi e reportistica. Attraverso le tre fasi che lo costituiscono, l'ETL permette di consolidare dati eterogenei e garantire una visione coerente e unificata dell'informazione aziendale.

3.1 Le fasi del processo

Il processo di ETL consiste di tre fasi sequenziali: l'**estrazione**, la **trasformazione** e il **caricamento**.

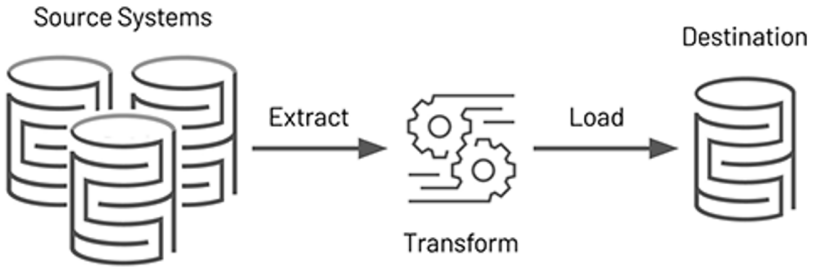

Figura 3.1. Processo ETL

3.1.1 Estrazione dei dati

La prima fase di **estrazione** consiste nel raccogliere i dati da diverse fonti eterogenee. È una fase delicata in cui è cruciale preservare l'integrità e la completezza delle informazioni estratte. Gli strumenti di ETL più moderni supportano diverse modalità di estrazione, che può essere di tipo **statico** o **incrementale**. L'estrazione statica implica il caricamento di un dataset completo dalla sorgente al sistema di destinazione in un'unica operazione. Questo approccio è utilizzato quando si deve trasferire una grande quantità di dati in modo massivo, generalmente *una tantum* (per la migrazione iniziale di dati). L'estrazione incrementale consiste nel trasferire solo i dati nuovi o aggiornati dalla sorgente al sistema di destinazione, riducendo il volume complessivo di dati da gestire. Viene utilizzata per gestire l'aggiornamento periodico del data warehouse e necessita di meccanismi ben precisi per identificare le modifiche.

3.1.2 Trasformazione dei dati

Nella fase di **trasformazione**, i dati vengono trasformati e uniformati per soddisfare i requisiti tecnici e aziendali del sistema di destinazione. Questa fase comprende tutte le attività che consentono di ottenere un dato accurato, completo, coerente e dunque di qualità. Tra gli aspetti standard su cui intervenire si collocano:

- dati duplicati;
- dati mancanti;
- uso non previsto di un campo;
- valori impossibili;
- valori errati dovuti ad errorii battitura;
- inconsistenza tra valori logicamente associati;
- inconsistenza dovuta a convenzioni o formati differenti.

La trasformazione è la fase che ha un impatto maggiore sulla definizione del dato pulito richiesto dalle analisi di business. Questa parte del processo può essere più o meno approfondita nel dettaglio a seconda delle necessità aziendali. Il punto chiave è che può essere adattata su misura in base ai requisiti dell'azienda e, in quanto tale, ha una vasta applicazione in diversi ambiti e a diversi livelli.
É la fase più delicata, poiché bisogna prestare attenzione a requisiti e regole aziendali, ma anche quella con più potenzialità, vista la facoltà di effettuare tutte le trasformazioni desiderate e nel modo preferito.

3.1.3 Caricamento dei dati

La fase finale consiste nel **caricare** i dati trasformati nel sistema di destinazione. A seconda delle necessità, il caricamento può avvenire in **modalità completa** (refresh)

sostituendo tutti i dati esistenti, o **incrementale** (update) aggiornando solo i dati nuovi o modificati. È essenziale garantire che il caricamento avvenga senza compromettere l'integrità dei dati, rendendo i dati pronti per analisi, reportistica e supporto decisionale.

3.2 Vantaggi e limitazioni del processo ETL

Il processo di ETL è ormai una **scelta consolidata** per l'integrazione dei dati in un Data Warehouse. Tra i suoi principali punti di forza vi è la capacità di garantire dati puliti, strutturati e ottimizzati prima del caricamento, migliorando l'efficienza delle analisi e la qualità delle informazioni disponibili per il business. Inoltre, l'ETL permette di applicare regole di trasformazione complesse che garantiscono coerenza ed integrità.
Tuttavia, si tratta di una componente costosa ed impegnativa da integrare all'interno di un'architettura di Data Warehousing, che può impiegare anche un terzo del budget previsto per l'intero progetto. Anche dal punto di vista delle tempistiche, secondo [4], l'implementazione dell'ETL può arrivare ad occupare anche l'80% del tempo di sviluppo dell'architettura nel complesso.
Proprio sulla base dello sforzo significativo da impiegare per la sua integrazione, è necessario individuare e tenere a mente anche le limitazioni associate all'ETL.
Il processo di trasformazione, eseguito prima del caricamento, può risultare oneroso in termini di tempo e risorse computazionali, rallentando l'ingestione dei dati, soprattutto per grandi volumi. Inoltre, la scalabilità dell'ETL può essere limitata rispetto a soluzioni più moderne, ed,infine, l'implementazione e la manutenzione di una pipeline ETL richiedono spesso un'infrastruttura dedicata e investimenti significativi, sia in termini di costi che di competenze tecniche, rendendolo meno adatto ad ambienti con necessità di analisi in tempo reale o altamente dinamiche.

3.3 Confronto tra ETL e E-LT

Il processo di ETL non è l'unica metodologia disponibile per effettuare il caricamento dei dati all'interno di un Data Warehouse. Tra le alternative esistenti, si distingue l'**E-LT** (Extract, Load, Transform), una tecnica in costante sviluppo e sempre più adottata. Comprendere a fondo i principi e le funzionalità di entrambe le metodologie è fondamentale per scegliere la soluzione più adatta al proprio contesto aziendale.
L'ETL, considerato il metodo *classico*, è stato introdotto negli anni Settanta [23] e da allora rappresenta un pilastro delle architetture dei Data Warehouse. L'E-LT, invece, è una metodologia più recente, affermatasi solo intorno al 2010 come valida alternativa all'approccio tradizionale costituito dall'ETL [23]. Pur mantenendo le tre fasi principali, l'E-LT **rivoluziona l'ordine delle operazioni**: la fase di estrazione rimane simile a quella dell'ETL e la novità introdotta consiste nello **scambio** tra lo step di trasformazione e quello di caricamento. I dati vengono infatti caricati direttamente nel DWH nella loro forma grezza, senza subire trasformazioni, le quali vengono apportate successivamente, sfruttando la potenza di calcolo del Data Warehouse stesso.
Questa metodologia, spesso integrata con soluzioni cloud, offre **vantaggi significativi**,

come una **maggiore scalabilità** per gestire grandi volumi di dati in entrata in un secondo momento, costi generalmente più contenuti ed una maggiore flessibilità. Caricando i dati grezzi nel Data Warehouse, infatti, è possibile eseguire le trasformazioni in un qualsiasi momento successivo, senza richiedere ulteriori interventi complessi e semplificando così il processo.
In sintesi, un confronto tra i due è ben sintetizzato in [23] e può essere visualizzato nella tabella a seguire:

Criterio	**ETL**	**E-LT**
Performance	Più lento a causa delle trasformazioni effettuate prima del caricamento	Più veloce grazie al caricamento diretto dei dati grezzi
Scalabilità	Limitata dalla capacità del motore di trasformazione	Scala in base alle capacità del data warehouse
Costo	Costi iniziali di configurazione e manutenzione più elevati	Costi iniziali più bassi, ma potenzialmente costi operativi più elevati in seguito

Tabella 3.1. Confronto tra ETL ed E-LT.

Segue quindi che l'ETL è generalmente consigliato per le organizzazioni che hanno esigenze di integrazione e trasformazione dei dati piuttosto stabili e prevedibili, e che possono investire in motori di trasformazione dedicati. Questo approccio si adatta bene a contesti in cui l'architettura dei dati è consolidata e i processi di trasformazione richiedono un controllo rigido prima che i dati vengano caricati nel Data Warehouse.
Al contrario, l'E-LT risulta più vantaggioso in ambienti moderni, soprattutto in presenza di infrastrutture cloud. Tuttavia in ambienti on-premise, l'E-LT può risultare meno efficiente a causa della rigidità delle infrastrutture tradizionali, che spesso non sono progettate per gestire in modo ottimale carichi di lavoro così dinamici.
La scelta tra ETL ed E-LT va ben ponderata, prendendo in considerazione tutti i fattori sopra menzionati per offrire all'organizzazione la soluzione più adatta alle esigenze specifiche delle infrastrutture a disposizione.

3.4 Framework aziendale

Per framework si intende un insieme strutturato di principi, metodologie, strumenti e best practice progettato per guidare lo sviluppo, l'implementazione e la gestione dei processi all'interno di un'organizzazione. Il suo scopo principale è fornire una **base standardizzata** per affrontare sfide strategiche e operative, migliorando l'efficienza e la scalabilità aziendale. In pratica, un framework aziendale facilita la creazione e la gestione di soluzioni tecnologiche e operative, consenendo alle aziende, attraverso un approccio sistematico e replicabile, di ottimizzare le risorse, ridurre i rischi e affrontare la complessità organizzativa in modo strutturato e sostenibile.

Di seguito viene analizzato nel dettaglio il framework adottato da **Mediamente Consulting**, azienda in cui è stato svolto il tirocinio, per l'integrazione dei dati. Tale framework è articolato in tre livelli correlati tra di loro, come visibile in 3.2.

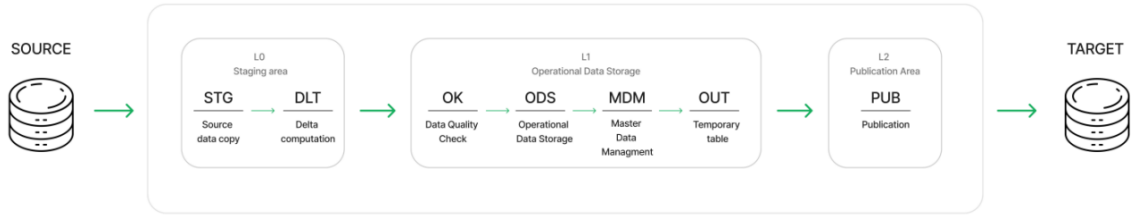

Figura 3.2. Struttura del flusso di integrazione di Mediamente Consulting

3.4.1 L0 - Staging Area

Il livello L0, noto anche come **Staging Area**, rappresenta la **prima fase** del processo ETL all'interno di un Data Warehouse. Il suo scopo principale è importare i dati grezzi provenienti da diverse fonti senza alcuna trasformazione, mediante un inserimento periodico ed indipendente dai livelli successivi.
Il livello L0 è suddiviso in due fasi principali, che generano due tabelle distinte: **STG** (Staging) e **DLT** (Delta).

STG

Nella fase di STG, i dati vengono **estratti e caricati** nella tabella STG, mantenendo il dettaglio minimo e senza modifiche. Vengono aggiunti due campi fondamentali che vengono mantenuti lungo l'intero flusso, assicurando tracciabilità e coerenza nelle operazioni di aggiornamento: **JOBID**, un identificatore numerico generato al momento del caricamento (formato YYYYMMDDHHMMSS), che permette di partizionare i dati in base ai flussi di caricamento, e **INS_TIME**, un timestamp che registra l'istante in cui ogni singolo dato viene effettivamente scritto nella tabella.

DLT

Il secondo step ha poi lo scopo di **identificare le variazioni** rispetto al caricamento precedente per caricarle nella tabella DLT, evitando la duplicazione di dati invariati. La DLT contiene quindi le modifiche avvenute nei dati dall'ultimo caricamento, suddividendole in nuovi record, aggiornamenti e cancellazioni. Per ottenere questa classificazione, vengono confrontati i dati della STG con quelli già presenti nel DWH utilizzando due approcci:

- **Lettura in delta**: se la sorgente possiede un campo che indica l'ultimo aggiornamento, si estraggono solo i dati nuovi o modificati, che vengono direttamente caricati nella DLT senza passare per la STG, e le cancellazioni vengono tracciate attraverso una tabella di log;
- **Lettura in full**: quando la sorgente non fornisce informazioni sulle modifiche, vengono effettuate due operazioni di confronto (query di minus): una tra i dati di oggi e quelli di ieri per individuare nuovi record, e l'altra tra quelli di ieri e oggi per identificare le cancellazioni. In questo caso, un attributo **FLG_NEG** viene utilizzato per distinguere i record aggiunti (0) da quelli eliminati (1).

La DLT contiene solitamente le informazioni più recenti, con una memoria predefinita di 10 caricamenti che può essere eventualmente modificata e adattata in base alle esigenze del cliente. A volte, oltre ai dati a breve termine, può essere necessario mantenere uno storico con tutte le informazioni estratte fino al momento attuale dai sistemi sorgente. Per far fronte a questa esigenza, si può aggiungere al flusso la tabella **DLT_HIS**, compilata allo stesso modo e contemporaneamente con la DLT. Questa tabella avrà delle prestazioni decisamente inferiori rispetto alle altre, ma questo non costituisce un problema in quanto l'accesso ad essa avverrà solo in casi di necessità.

3.4.2 L1 - Operational Data Storage

Il secondo livello, denominato L1 o **Operational Data Storage (ODS) Layer**, è il cuore del framework, in quanto gestisce la trasformazione e il consolidamento dei dati. In questa fase vengono eseguite le operazioni più complesse e computazionalmente onerose, al fine di verificare la qualità, l'integrità e la coerenza delle informazioni prima che i dati vengano caricati del Data Warehouse.
Tra le varie operazioni svolte, si distinguono quelle per garantire la pulizia, la standardizzazione, l'arricchimento e l'applicazione delle regole aziendali, ottenendo tabelle progettate per ottimizzare il caricamento nei livelli successivi del Data Warehouse.
Dopo la fase di L0, il livello L1 si occupa della **storicizzazione** e **armonizzazione** delle informazioni, al fine di uniformare e riconciliare i dati provenienti da sistemi eterogenei, mantenendo un elevato livello di granularità, essendo una copia diretta dei dati di L0. Inoltre, in questa fase, i dati vengono arricchiti con informazioni aggiuntive provenienti dal cliente o da fonti esterne.

OK

Il primo step del processo consiste nella validazione dei dati attraverso la tabella OK: i dati provenienti dalla tabella DLT, considerando solo i record con JOBID più recente, vengono sottoposti a controlli di **Data Quality**. Tra i controlli effettuati rientrano quelli di integrità referenziale, in cui vengono verificate le chiavi esterne attraverso operazioni di lookup per garantire la validità delle relazioni tra le varie tabelle, l'applicazione di regole aziendali, specifiche per individuare dati anomali, valori mancanti o errati, e la validazione dei record, con cui vengono scartati i dati che non rispettano determinati criteri, come la presenza di valori duplicati, negativi o nulli. I record non conformi vengono

inseriti in tabelle di errore, per essere poi rivalutati in futuro ed eventualmente reintegrati.

ODS

L'**Operational Data Storage (ODS)** rappresenta la tabella in cui i dati risultano certificati e storicizzati. Questo livello assicura che le informazioni siano consolidate e validate, offrendo una copia certificata e normalizzata dei dati operativi.
L'aggiornamento della tabella ODS consente di gestire i dati in maniera diversa in base alla loro presenza e variazione:

- **Dati invariati**: se un record della tabella **OK** è già presente nella ODS senza modifiche, non viene eseguita alcuna operazione;

- **Nuovi inserimenti**: se un record della tabella **OK** non esiste nella ODS, viene inserito per la prima volta;

- **Dati aggiornati**: se il record è già presente nella ODS ma con valori differenti, vengono aggiornati solo i campi che nel frattempo sono stati modificati.

Per svolere queste operazioni vengono inseriti due nuovi campi **UPD_TIME** e **JOBID_UPD** (rispettivamente nello stesso formato di INS_TIME e JOBID), per tenere traccia dell'ultimo istante di tempo in cui il record è stato modificato.

MDM

Il **Master Data Management (MDM)** è una fase chiave in cui si effettuano due operazioni principali. Innanzitutto avviene un'ulteriore integrazione dei dati, in cui le tabelle provenienti da più fonti vengono unite per generarne una unica. I dati poi subiscono anche uno step di arricchimento (**Data Enrichment**), in cui le informazioni vengono valorizzate attraverso operazioni di *join* con tabelle dimensionali, che forniscono dettagli descrittivi aggiuntivi. Un dettaglio essenziale in questa fase è la gestione delle chiavi, con particolare attenzione alla creazione di **chiavi surrogate (SK)**. Si tratta di identificatori numerici univoci, utilizzati per ottimizzare le operazioni di *join* e migliorare le prestazioni del sistema. Le **SK** diventano le nuove chiavi primarie delle tabelle dimensionali al posto delle chiavi naturali, che vengolo comunque mantenute come attributi.

OUT

Nella fase finale del livello L1, denominata **OUT**, i dati arricchiti e integrati nelle operazioni precedenti vengono preparati per la pubblicazione. Le tabelle OUT contengono il dato pronto per essere utilizzato nel livello L2. In questa fase, le operazioni principali includono la **denormalizzazione** e l'**esecuzione delle aggregazioni** necessarie per rispondere alle specifiche richieste analitiche.

3.4.3 L2 - Publication Data Layer

Il livello finale del processo è **L2**, denominato **Publication Data Layer**, in cui i dati vengono resi accessibili per scopi analitici e decisionali. In questa fase, le informazioni sono organizzate in **aree tematiche**, generalmente allineate ai processi aziendali, e strutturate in **Data Mart** che contengono dimensioni e fatti significativi per l'utente.

PUB

Le **tabelle PUB**, derivate direttamente dalle OUT di L1, rappresentano il dato consolidato e ottimizzato per strumenti di **Business Intelligence**, di visualizzazione e altre applicazioni aziendali. A differenza delle OUT, che vengono svuotate a ogni ciclo, le PUB vengono storicizzate per supportare il reporting e le analisi continuative.
La struttura del **Data Warehouse** in L2 può seguire differenti schemi, tra cui **lo schema a stella** e **lo schema a fiocco di neve**, trattati nel dettaglio nel Capitolo 2. La scelta del modello dipende dalle esigenze specifiche dell'azienda e dallo strumento di visualizzazione adottato, in modo da ottimizzare le prestazioni.

3.4.4 Tabelle di metadati

Il framework aziendale comprende due tabelle di metadati: FLOW_MANAGER e TABLE_MANAGER, arricchite da campi tecnici per il tracciamento delle operazioni ETL.

La tabella **FLOW_MANAGER** tiene traccia della cronologia e dell'avanzamento delle iterazioni ETL, monitorando lo stato di ogni livello di ingestione per consentirne l'orchestrazione. Ogni record è identificato dal JOBID e da altri campi chiave come IDENTITY, che raggruppa i flussi secondo l'area di lavoro, GROUP, che aggrega tabelle con dipendenze funzionali, e LEVEL, che rappresenta la fase di ETL (L0, L1 o L2). La tabella registra lo stato del processo con un valore numerico (0 per esecuzione completata, 1 per job in corso, -3 per errori e interruzione), oltre ai timestamp di inizio (START_DATE) e fine (END_DATE) dell'operazione.
La tabella **TABLE_MANAGER** è dedicata al monitoraggio delle operazioni di lettura (livello L0) a livello di singola tabella. Ogni record è identificato da JOBID, IDENTITY, GRP_NAME e TABLE_NAME e include campi specifici per tracciare il flusso di dati. Il NUM_ROWS registra il numero di righe lette, consentendo di individuare anomalie o errori nei dati. Infine, il LOAD_DATE memorizza il momento in cui il caricamento della tabella è stato completato.

3.5 Strumenti per l'ETL

Gli strumenti di ETL rappresentano una componente fondamentale nell'ambito della gestione dei dati, e oggi sul mercato esiste una vasta gamma di soluzioni disponibili, ciascuna con funzionalità specifiche, per soddisfare esigenze aziendali diverse.
La posizione di uno strumento software nel mercato è determinata dalla sua diffusione e

dal livello di consolidamento nel settore di riferimento. Un metodo comunemente utilizzato per valutare questa posizione è il **Quadrante Magico di Gartner** (figura 3.3), che classifica le soluzioni di integrazione dei dati in base a due criteri principali: la completezza della visione strategica e la capacità di esecuzione. Il modello suddivide gli strumenti in quattro categorie: *Leader*, ovvero soluzioni che offrono prestazioni eccellenti e sono considerate punti di riferimento nel settore, *Sfidanti*, prodotti con una solida capacità operativa, ma con una visione strategica meno innovativa o ancora in fase di sviluppo, *Visionari*, che propongono innovazioni interessanti e un approccio strategico avanzato, ma che possono avere difficoltà nell'esecuzione o nel consolidarsi sul mercato, e *Attori di nicchia*, nonchè strumenti specializzati che rispondono a esigenze specifiche, ma che presentano limiti in termini di diffusione o capacità di evoluzione. Le soluzioni appartenenti alla categoria dei Leader sono generalmente quelle da cui ci si aspetta le migliori prestazioni e il maggiore impatto nel settore.

Figura 3.3. Quadrante magico Gartner Software di integrazione 2024

Sulla base di quanto appena citato e di una rassegna di alcuni dei prodotti che dominano il mercato secondo [3] e [16], vengono ora introdotti alcuni dei principali software per l'integrazione dati,con una panoramica delle caratteristiche che li contraddistinguono. Tra i più affermati *strumenti commerciali* per l'ETL si trovano:

- **Informatica PowerCenter**, noto per la sua robustezza e scalabilità, è utilizzato soprattutto in contesti enterprise grazie al supporto per processi ETL complessi e all'ampia capacità di gestione dei dati provenienti da diverse fonti. Questo strumento, tuttavia, ha un costo elevato e una curva di apprendimento significativa, caratteristiche che lo rendono più adatto a grandi aziende con budget che a contesti più piccoli in cui la mancanza di risorse adeguate può rappresentare un limite.

- **IBM DataStage** è un'altra soluzione interessante: è un potente strumento adatto per processi di integrazione e trasformazione dei dati da fonti eterogenee. Offre un'interfaccia intuitiva per la gestione delle pipeline e supporta l'integrazione scalabile di grandi volumi di dati, garantendo elevate prestazioni attraverso l'elaborazione parallela [15]. È ideale per aziende che necessitano di collegare sistemi complessi, ma il costo della licenza può rappresentare un limite per organizzazioni più piccole.

- **SQL Server Integration Services (SSIS)** è un altro popolare strumento ETL, parte della suite di Microsoft SQL Server. SSIS è progettato principalmente per processi ETL e offre un'ampia gamma di funzionalità, tra cui trasformazioni di dati complesse, automazione dei processi e integrazione nativa con l'ecosistema Microsoft. È particolarmente adatto per aziende che utilizzano già prodotti Microsoft, ma meno versatile per organizzazioni che lavorano con sistemi diversi.

- **Oracle Data Integrator (ODI)** si distingue per il suo approccio ELT piuttosto che ETL, ottimizzando l'elaborazione dei dati sfruttando direttamente la potenza di calcolo del database. ODI è efficace soprattutto per aziende che utilizzano database Oracle, poiché offre integrazione nativa e alte prestazioni per carichi di lavoro pesanti [15]. Tuttavia, può risultare meno competitivo rispetto ad altri strumenti in ambienti che non utilizzano tecnologie Oracle.

Come strumenti *open source* di maggiore rilevanza figurano invece:

- **Talend**, una piattaforma open-source e più flessibile che offre sia funzionalità ETL che ELT, adattandosi bene a scenari in cui è necessario integrare dati tra sistemi tradizionali e piattaforme cloud. Sebbene Talend sia apprezzato in particolare per il modello freemium, può al contempo mostrare limiti in termini di prestazioni su volumi di dati molto elevati.

- **Pentaho Data Integration (PDI)**, conosciuto anche come **Kettle**, è un'altra soluzione open-source orientata all'ETL che combina facilità d'uso con una buona capacità di gestione dei dati. È ideale per scenari con budget limitati o per piccoli progetti, ma può risultare limitato per implementazioni su larga scala o ambienti con requisiti complessi di trasformazione.

Infine per i contesti *cloud*, strumenti come **AWS Glue** e **Google Cloud Dataflow** rappresentano soluzioni per approcci ELT progettate per il cloud di Amazon e Google rispettivamente. Entrambi sono eccellenti per ambienti cloud-first e si integrano perfettamente con i servizi dei rispettivi ecosistemi, ma possono presentare difficoltà di integrazione con infrastrutture on-premise.

In conclusione, strumenti ETL come SSIS, ODI ed Informatica PowerCenter sono più adatti per scenari on-premise e Data Warehouse tradizionali, mentre soluzioni come Talend e Pentaho si adattano meglio a budget ridotti e necessità più flessibili. Per scenari streaming, infine i servizi cloud come AWS Glue e Google Cloud Dataflow offrono le funzionalità più avanzate. La scelta dello strumento dipende quindi dal budget, dal contesto aziendale, dall'architettura esistente, dai volumi di dati e dai requisiti specifici del progetto.

3.5.1 SQL Server Integration Services (SSIS)

SQL Server Integration Services è una piattaforma di **Microsoft** per la creazione di soluzioni di integrazione e trasformazione di dati a livello aziendale [17]. SSIS usa i servizi di integrazione per risolvere problemi aziendali complessi, tra cui copiare o scaricare file da diverse origini (come file XML, file flat e database relazionali), pulire ed estrarre i dati secondo le esigenze aziendali, caricarli nei data warehouse e gestire oggetti e dati di SQL Server.
La prima versione di SSIS è stata rilasciata con SQL Server 2005, in sostituzione dei Data Transformation Services (DTS) che erano disponibili con SQL Server 7.0 e SQL Server 2000 [11].
Rispetto ad altre piattaforme di integrazione, SSIS si distingue per la sua stretta integrazione con l'ecosistema Microsoft, il che lo rende una scelta privilegiata per aziende che già utilizzano SQL Server o altre tecnologie Microsoft.
Nel contesto aziendale, SSIS si rivela indispensabile per automatizzare la gestione dei dati. Può essere utilizzato per aggiornare quotidianamente Data Warehouse aziendali, sincronizzare i dati tra sistemi differenti, eseguire analisi dei dati preliminari attraverso operazioni di trasformazione e arricchimento. Le sue funzionalità chiave includono un'ampia gamma di attività predefinite, strumenti grafici che offrono un'interfaccia visiva per progettare i flussi ETL e un database del catalogo SSIS per l'archiviazione e la gestione dei pacchetti.
Tra gli altri vantaggi che si riscontrano nell'utilizzare SSIS figurano la facilità di utilizzo trattandosi di uno strumento **low-code**, la possibilità di integrare del codice personalizzato e il basso costo rispetto ad altre piattaforme commerciali che offrono servizi simili [11].
La struttura base all'interno dell'architettura di SSIS è il **pacchetto**, che rappresenta l'unità operativa principale per la progettazione e l'esecuzione dei processi ETL. Un pacchetto è una raccolta di attività, logiche di controllo, connessioni e flussi di lavoro organizzati per estrarre, trasformare e caricare i dati da una o più fonti verso una destinazione. Ogni pacchetto è progettato per svolgere un compito specifico, come la copia dei dati, la trasformazione complessa o l'automazione di processi ricorrenti.

Un pacchetto è composto da tre elementi principali:

- **Control Flow**: definisce la sequenza logica delle attività nel pacchetto. Può includere attività come *Data Flow Task* (per gestire l'elaborazione dei dati), *Execute SQL Task* (per eseguire query SQL), *Script Task* (per eseguire codice personalizzato in C# o VB.NET);

- **Data Flow**: gestisce il movimento e le trasformazioni dei dati e include una o più *sorgenti*, da cui vengono letti i dati (tra le sorgenti supportate OLE DB Source, Flat File Source come CSV o TXT, sorgenti XML e JSON), *trasformazioni*, con cui vengono elaborati i dati (es. Data Conversion per convertire i tipi di dati, Lookup per unire dati da diverse sorgenti basate su una chiave comune, Conditional Split per separare i dati in base a condizioni specifiche, Aggregate per raggruppare e sommare i dati), e una o più *destinazioni*, su cui vengono infine scritti i dati (es. OLE DB Destination, Database relazionali per popolare tabelle di fatto o dimensioni nel DWH, File flat per esportare dati in formato testo);

- **Event Handlers**: permettono di gestire gli eventi (ad esempio errori o completamento) durante l'esecuzione del pacchetto.

Questi pacchetti possono essere configurati con parametri dinamici, programmati per l'esecuzione automatica e monitorati attraverso strumenti di logging, offrendo flessibilità e controllo nei flussi ETL aziendali. SSIS, infatti, offre diversi strumenti avanzati per la gestione delle varie casistiche.
Per quanto riguarda la **gestione degli errori**, SSIS permette di intercettare e gestire eventuali anomalie grazie a diverse funzionalità. L'*Error Output* consente di reindirizzare i dati che non soddisfano determinati requisiti verso tabelle di errore o file di log, facilitando così l'analisi e la risoluzione delle problematiche. Inoltre, la *Retry Logic* offre la possibilità di riprovare automaticamente un'operazione in caso di errori temporanei, riducendo il rischio di interruzioni nei processi ETL. Il sistema di *Logging*, infine, registra eventi ed errori in modo dettagliato, permettendo un monitoraggio costante e una diagnostica efficace.
L'automazione e lo scheduling dei pacchetti ETL, invece, vengono gestiti tramite *SQL Server Agent*, un servizio che consente di pianificare l'esecuzione automatica dei pacchetti a intervalli specifici o in risposta a determinati eventi. Ciò permette di orchestrare e mantenere aggiornati i flussi di dati senza intervento manuale, garantendo la regolarità e la tempestività delle elaborazioni.
Dal punto di vista del monitoraggio e delle **performance**, SSIS offre strumenti avanzati per analizzare le esecuzioni dei pacchetti e ottimizzarne le prestazioni. Tecniche di *Performance Tuning*, come l'uso dell'elaborazione parallela e la creazione di indici nelle tabelle di destinazione, consentono l'ottimizzazione del flusso di dati, migliorandone l'efficienza e riducendo i tempi di esecuzione dei processi.
Infine, **lo sviluppo e il debugging** dei pacchetti ETL avviene attraverso SQL Server Data Tools (SSDT), un ambiente di sviluppo integrato che fornisce un'interfaccia drag-and-drop intuitiva per la progettazione dei flussi di lavoro. SSDT offre anche strumenti avanzati per il debugging, permettendo di eseguire i pacchetti in modalità interattiva,

monitorare i dati in tempo reale e identificare rapidamente eventuali errori.
Tutte queste funzionalità rendono SSIS una soluzione robusta ed efficiente per la gestione dei processi di integrazione dati.

Capitolo 4

Monitoraggio

Il **monitoraggio** rappresenta un elemento fondamentale nella gestione dei dati e nell'ambito della Business Intelligence. Esso comprende un insieme di tecnologie, metodologie e strumenti avanzati finalizzati a garantire la disponibilità, le prestazioni del sistema e la qualità dei servizi. Si tratta di uno strumento indispensabile per supportare decisioni informate e, soprattutto, per identificare e risolvere tempestivamente eventuali complicazioni, evitando che si evolvano in problematiche più gravi.
Il monitoraggio trova applicazione in numerosi contesti, tra cui spiccano i sistemi IT, le infrastrutture cloud e i processi aziendali [14]. Tuttavia, il significato del termine varia a seconda del contesto di riferimento, rendendo complessa l'elaborazione di una definizione generica.
Nel caso del **monitoraggio dei dati**, l'attenzione si concentra sulla **qualità** degli stessi, con particolare riguardo a elementi quali **coerenza**, **accuratezza**, **completezza** e **validità**.
Più frequentemente, il monitoraggio è associato ai **processi**, dove il focus è sull'**analisi dello stato del sistema** e sulla valutazione delle sue **prestazioni**. Ciò avviene attraverso l'osservazione di metriche come i tempi di esecuzione, il throughput e l'utilizzo delle risorse, oltre all'individuazione e alla gestione di eventuali errori operativi.
Per il presente studio, è necessario fornire una definizione specifica al fine di distinguere chiaramente il concetto di monitoraggio da quello di sistema di monitoraggio e di alerting. Secondo [14], si possono delineare le seguenti differenze:

- Il **monitoraggio** è il processo di **osservazione** e **analisi delle variazioni** di stato e del flusso di dati all'interno di un sistema. Questo processo include la registrazione e l'analisi di input quantitativi, ovvero misurazioni numeriche che riflettono lo stato attuale e i cambiamenti recenti, memorizzati sotto forma di metriche. Il monitoraggio ha come obiettivo primario l'**identificazione** di eventuali **guasti** e il supporto alla loro successiva risoluzione. Le tecniche impiegate per il monitoraggio dei sistemi informativi spaziano tra elaborazione in tempo reale, statistica e analisi dei dati.

- Un **sistema di monitoraggio** si riferisce a un **insieme di componenti software** dedicati alla misurazione, raccolta, elaborazione, interpretazione e presentazione dei dati generati durante il monitoraggio.

- L'**alerting**, invece, rappresenta una funzionalità del sistema di monitoraggio che consente di rilevare e **notificare agli operatori** di competenza eventi significativi, indicativi di un cambiamento critico di stato, come può essere il superamento di una soglia predefinita. Le notifiche, definite *alert* (o avvisi), consistono in semplici messaggi che possono essere inviati tramite diversi canali, quali e-mail, SMS, telefonate o sistemi di ticketing. La modaltà più utilizzata è la notifica tramite e-mail, in quanto si tratta di uno strumento veloce, ad ampio spettro e privo di costi aggiuntivi.

4.1 Fasi operative del Monitoraggio

Il monitoraggio è un processo continuo articolato in una serie di fasi cicliche e interdipendenti. Questo ciclo può essere suddiviso in tre principali fasi operative:

1. **Raccolta dei dati**: in questa fase vengono acquisiti i dati relativi alle operazioni del sistema. Tali dati provengono da fonti quali log di sistema, statistiche operative e altre misurazioni effettuate in tempo reale o a intervalli regolari;

2. **Aggregazione e archiviazione dei dati**: i dati raccolti vengono successivamente organizzati e raggruppati in base alle specifiche esigenze operative. Questa fase prevede la memorizzazione dei dati seguendo metriche predefinite, al fine di garantire un'organizzazione coerente e funzionale;

3. **Presentazione dei dati**: l'ultima fase consiste nella produzione di report che sintetizzano le informazioni raccolte. Tali report possono essere presentati sotto forma di serie temporali, dashboard interattive o altre modalità di visualizzazione, per agevolare l'analisi e supportare il processo decisionale.

4.2 Aspetti complessi del Monitoraggio

Il monitoraggio apporta un valore significativo ai sistemi e, sebbene in passato sia stato spesso sottovalutato, sta acquisendo crescente attenzione grazie alla consapevolezza dei benefici che può offrire.
L'obiettivo principale del monitoraggio è fornire una visione quasi in tempo reale dello stato attuale del sistema e delle sue prestazioni recenti, facilitando una diagnosi precoce, il processo decisionale basato su dati concreti e l'automazione. Inoltre, il monitoraggio consente di effettuare analisi predittive relative a futuri livelli di risorse, prestazioni e potenziali complicazioni.
La **rapidità di rilevamento** dei problemi critici è di gran lunga l'obiettivo più importante del monitoraggio [14] ed è la funzione della parte di allerta del sistema. Tuttavia, la sfida principale consiste nel bilanciare due obiettivi contrastanti: **velocità** e **accuratezza**. È fondamentale segnalare tempestivamente le anomalie classificate come problematiche, evitando al contempo la generazione di falsi allarmi, che possono portare gli operatori ad ignorare o a sottovaluutare segnalazioni critiche in momenti successivi.
Oltre alla gestione degli alert, il monitoraggio presenta ulteriori aspetti complessi. Uno dei principali è la **difficoltà nella configurazione iniziale** e nella manutenzione di un

sistema di monitoraggio, che può richiedere risorse significative, sia in termini di tempo e costo, sia di competenze umane. Gli operatori svolgono un ruolo cruciale nella definizione delle soglie iniziali e devono possedere una profonda conoscenza del sistema e delle sue specifiche esigenze. Tale conoscenza è essenziale per evitare di trascurare informazioni che, pur apparendo marginali ad un'analisi superficiale, potrebbero avere un impatto rilevante sul sistema.

4.3 Monitoraggio nei flussi ETL

Il monitoraggio nei flussi ETL svolge un ruolo cruciale nel garantire l'**affidabilità** e la **scalabilità** delle pipeline, fondamentali per l'elaborazione e l'integrazione dei dati. Uno degli obiettivi principali è **prevenire la propagazione di errori** che potrebbero compromettere la qualità dei dati nei sistemi di destinazione. Questo implica tenere sotto controllo diversi aspetti chiave, come le prestazioni delle pipeline, monitorando i tempi necessari per le operazioni di estrazione, trasformazione e caricamento, così da identificare eventuali colli di bottiglia o inefficienze. È altrettanto importante assicurarsi che la qualità dei dati venga mantenuta durante le trasformazioni, verificando la coerenza, l'accuratezza e la completezza delle informazioni. Infine, il monitoraggio si concentra sull'identificazione e sulla gestione di errori e fallimenti nei task ETL, adottando **misure proattive** per garantire la continuità dei processi e ridurre al minimo i tempi di inattività.
Il monitoraggio dei flussi ETL può essere effettuato attraverso diverse modalità, che si sono evolute nel tempo con il progresso delle tecnologie e l'aumento della complessità delle pipeline. I *metodi tradizionali* si basano principalmente su **log statici** e **report** generati manualmente, che richiedono un'analisi dettagliata da parte degli operatori per individuare eventuali problemi o inefficienze. Sebbene questi approcci siano funzionali, sono spesso limitati dalla **mancanza di automazione e di visibilità** in tempo reale. Con l'avvento di *strumenti più moderni*, il monitoraggio si è arricchito di **dashboard interattive** che offrono una panoramica in tempo reale delle pipeline, **notifiche automatiche** per segnalare anomalie o errori, e funzionalità di tracciabilità end-to-end, che permettono di seguire il percorso completo dei dati dalla sorgente alla destinazione. Parallelamente, si sono sviluppati *approcci predittivi* per il rilevamento delle anomalie, grazie all'uso di **algoritmi di machine learning** che identificano schemi insoliti o potenziali problemi prima che si verifichino, migliorando così la robustezza e l'efficienza dei processi.

Al giorno d'oggi, il panorama degli strumenti per il monitoraggio ETL è variegato, comprendendo sia **soluzioni open-source** che **piattaforme commerciali**, e un numero sempre crescente di fornitori di **servizi cloud** offre soluzioni di monitoraggio e sistemi di avviso come parte integrante dell'offerta. Ciascun prodotto è specializzato per fornire supporto in maniera diversa: alcuni si distinguono per l'alto livello di **personalizzazione**, altri per soluzioni complete con **interfacce user-friendly**, spesso arricchite da funzionalità aggiuntive per la gestione dei dati. Molti strumenti sono inoltre in grado di scalare dinamicamente e di integrarsi efficacemente con **piattaforme cloud** e infrastrutture più ampie.
La scelta dello strumento dipende dunque dalle **specifiche esigenze** del progetto, dal

livello di complessità delle pipeline e dal **budget** disponibile, ma l'obiettivo comune rimane quello di garantire un monitoraggio efficace e proattivo, essenziale per mantenere l'affidabilità e la qualità dei flussi ETL.

4.3.1 Studi sul Monitoraggio

La tematica ha guadagnato una crescente attenzione, dando origine a **numerosi studi** che hanno approfondito l'applicazione del monitoraggio in diversi contesti e processi.
In [20] viene sottolineata l'importanza dei **metadati**, identificati come uno dei fattori chiave per il successo di un progetto di data warehousing [27]. A partire da questa premessa, il Data Warehouse oggetto di studio è stato integrato con i metadati disponibili per definire un modello di **metadati ETL**, progettato appositamente per supportare il processo di refresh del Data Warehouse, la sua matutenzione e il monitoraggio dei cicli batch. Il modello di monitoraggio sviluppato è inoltre indipendente dallo strumento di ETL utilizzato, user-friendly e in grado di fornire supporto al team di produzione.
Anche [28] propone un modello ETL basato sui metadati, integrando una **componente LogServices** con la funzionalità di garantire la qualità del processo di ETL integrando log operazionali, di errore, di analisi su informazioni statistiche come tempi e frequenze di esecuzione, oltre che log di cambiamento delle informazioni di sistema.
Un **modello concettuale** di monitoraggio ed alarming per una pipeline generica di dati viene sviluppato in [21]. Attraverso un'**analisi cross-case**, è stato definito un metamodello basato su nodi e connettori in grado di **automatizzare** il monitoraggio in ogni fase del processo. Questo approccio semplifica e **migliora la gestione complessiva**, poiché il carico di analisi non è più concentrato su un'unica persona, ma suddiviso tra più figure specializzate, ciascuna responsabile di una specifica parte della pipeline.
In [18] viene criticato l'uso del logging limitato ai soli casi di **comportamenti inaspettati** del sistema, sottolineando l'importanza di impiegarlo per guidare l'intero processo di ETL. A tal fine, viene proposto un sistema di logging efficiente, integrato in un **Pattern-Oriented-Approach (POA)** per lo sviluppo del sistema di ETL. Per il LogPattern, è stato adottato un modello basato su **grafi**.
Analogamente, in [13] il processo viene suddiviso in più attività mediante un **grafo aciclico diretto (DAG)**. In questo lavoro, l'automazione, lo scheduling ed il monitoraggio vengono implementati in **Apache Airflow**, un'applicazione open source progettata per gestire anche processi ETL in streaming. Attraverso le tabelle di metadati ed i sistemi di logging, possono essere visualizzate informazioni sull'esecuzione del flusso, insieme alle informazioni operazionali.
Un altro strumento caratterizzato da ottime performance, specialmente per quanto riguarda la velocità, é quello proposto in [29]. Questa soluzione a **basso costo**, denominata **Agile ETL**, si distingue per la sua efficacia nello sviluppo e nella manutenzione dei processi ETL. Agile ETL offre un'interfaccia grafica **intuitiva** e un sistema di monitoraggio **integrato**, basato su una tabella centralizzata che raccoglie tutte le informazioni utili sullo stato dei processi.
[1] presenta una rassegna di vari approcci per l'ottimizzazione delle performance di un flusso ETL disponibili come supporto in strumenti commerciali e open source. Si evidenzia come la maggior parte di questi strumenti presenti una significativa **carenza nel**

monitoraggio, in particolare per quanto riguarda l'identificazione dei colli di bottiglia nel processo. Per colmare questa lacuna, viene proposto un framework dotato di una componente denominata **Monitoring Agent**, in grado di monitorare le esecuzioni del flusso ETL, identificare colli di bottiglia ed errori, pianificare le esecuzioni e raccogliere statistiche dettagliate sulle performance.
Infine, diversi studi mostrano l'applicazione del monitoraggio dei flussi ETL in contesti specifici, mettendo in luce i benefici apportati in tali settori.
Un esempio significativo nell'ambito **finanziario** è presentato in [10], in cui viene sottolineata l'importanza di integrare un sistema di monitoraggio nel mondo della finanza per limitare errori ed inconsistenze, particolarmente frequenti a causa della natura **eterogenea** dei dati finanziari, che spesso provengono da fonti multiple con formati differenti, e richiedono l'elaborazione anche in tempo reale. Lo studio fornisce una panoramica delle **best practices** per garantire la qualità dei dati e implementare un monitoraggio robusto ed efficace, oltre a descrivere gli strumenti attualmente disponibili per raggiungere questi obiettivi.
Un altro settore in cui il monitoraggio dei flussi ETL riveste un ruolo cruciale è quello **sanitario**. [2] mostra l'importanza delle pipeline ETL ottimizzate nel supportare le organizzazioni sanitarie nell'utilizzo efficace dei dati per migliorare l'**assistenza ai pazienti** e l'**efficienza operativa**. Lo studio ha preso in esame metriche come il tempo medio di **elaborazione**, il **throughput** e il **tasso di errore** per valutare le performance, mentre per l'analisi delle risorse sono stati monitorati l'utilizzo della CPU e della memoria. Le tecniche di ottimizzazione adottate hanno prodotto significativi miglioramenti nelle prestazioni della pipeline ETL.
Anche [24] propone un'architettura per il monitoraggio della qualità dei dati e logging utilizzando un flusso ETL in un datawarehouse clinico. Tale architettura è strutturata in **blocchi** e include una componente dedicata alla visualizzazione dei risultati, attraverso un'interfaccia personalizzabile per la presentazione delle dashboard. L'efficacia di questa implementazione è stata validata mediante il suo utilizzo in tre diversi progetti di ricerca medica.

4.4 Monitoraggio in SSIS

Il monitoraggio e la gestione dei processi in SSIS sono elementi cruciali per garantire la **stabilità** e la **performance** dei flussi ETL. Nel tempo, il monitoraggio in SQL Server Integration Services ha subito un'evoluzione significativa, passando da approcci manuali e personalizzati a soluzioni **integrate** e sempre più **avanzate**. Tra le funzionalità presenti attualmente in SSIS per questo scopo spiccano: **logging integrato** nel catalogo **SSISDB**, che consente una tracciabilità completa delle esecuzioni, **Event Handler**, dotato di invio mail per rispondere dinamicamente a eventi come errori o avvisi, e un'altra serie di strumenti di **reportistica**, per analizzare e diagnosticare i processi. Inoltre SSIS offre la possibilità di integrazione con piattaforme esterne per ampliare ulteriormente le capacità di gestione e analisi. Di seguito, verranno esaminati in dettaglio questi strumenti e le loro potenzialità.

4.4.1 Logging e SSISDB

Nelle prime versioni di SSIS, il monitoraggio dei flussi ETL richiedeva la configurazione manuale del logging direttamente nei pacchetti, utilizzando provider come file di testo, tabelle SQL Server o Windows Event Log. Questo approccio, seppur funzionale, comportava una gestione **frammentata** e **dispendiosa**. Con il rilascio di SQL Server 2012, l'introduzione del Catalogo SSIS (SSISDB) ha **rivoluzionato** il monitoraggio, offrendo un sistema **centralizzato** per la gestione, la reportistica e il tracciamento delle esecuzioni dei pacchetti.
Il catalogo SSISDB consente di memorizzare automaticamente i dettagli di ogni esecuzione, inclusi i log degli eventi, i parametri utilizzati e lo stato dei pacchetti. Questo sistema di log integrato permette di tracciare eventi a livello di pacchetto o attività, registrando errori, avvisi e informazioni diagnostiche utili per la risoluzione dei problemi. SSISDB supporta inoltre funzionalità avanzate, come la configurazione di parametri di progetto e di pacchetto, la definizione di ambienti per specificare valori runtime personalizzati e l'accesso a report dettagliati per analizzare l'efficacia e le performance dei flussi ETL [17].
Tra gli oggetti archiviati nel catalogo SSISDB sono inclusi, oltre a pacchetti, parametri e ambienti, anche informazioni su progetti e cronologia operativa. Il tutto può essere visualizzato semplicemente eseguendo una query sulle viste nel database SSISDB e gestito chiamando le **stored procedures** nel database SSISDB o usando l'interfaccia utente del catalogo.
Grazie a queste caratteristiche, il catalogo SSISDB non solo semplifica il monitoraggio e la gestione, ma consente anche di **diagnosticare** e **ottimizzare** rapidamente eventuali **criticità** nei processi ETL, rendendo SSIS una soluzione robusta e versatile per l'integrazione dei dati in contesti aziendali complessi.

L'elemento chiave del logging con SSISDB è costituito dall'**esecuzione**. Un'esecuzione è un'istanza di un'esecuzione di un pacchetto SSIS all'interno di SSISDB. Quando un pacchetto viene avviato, viene registrata un'istanza nel catalogo, associata a un identificativo univoco, che permette di tracciare lo stato, la durata, l'esito del processo e tutta una serie di informazioni associate. Ad ogni esecuzione vengono associati uno o più **eventi**, che sono messaggi generati dal motore SSIS durante l'esecuzione del pacchetto. Gli eventi possono includere informazioni di stato, avvisi o errori, e vengono memorizzati nel catalogo per facilitare il debug e l'analisi.
Per analizzare esecuzioni ed eventi, SSISDB fornisce diverse viste e tabelle di sistema. Tra le più rilevanti per il monitoraggio di interesse troviamo:

- **catalog.executions**: questa vista contiene un record per ogni esecuzione di pacchetto SSIS;
- **catalog.operations**: traccia tutte le operazioni eseguite nel catalogo SSISDB, incluse non solo le esecuzioni, ma anche attività amministrative come distribuzioni di pacchetti o modifiche alle configurazioni;
- **catalog.operation_messages**: registra i messaggi di log generati dalle operazioni nel catalogo;

- **catalog.extended_operation_info**: fornisce dettagli aggiuntivi sulle operazioni registrate in catalog.operations, inclusi parametri di esecuzione e altre informazioni contestuali;

- **catalog.event_messages**: questa vista è fondamentale per il monitoraggio delle esecuzioni, poiché registra tutti i messaggi generati dagli eventi di SSIS. Ogni riga rappresenta un evento specifico (ad esempio, inizio di un'attività, completamento, errore o warning) ed è collegata a un'esecuzione tramite l'ID dell'esecuzione;

- **catalog.event_message_context**: fornisce ulteriori dettagli contestuali sugli eventi registrati in catalog.event_messages, come i parametri utilizzati durante l'esecuzione e altri metadati utili per il debugging.

L'interazione tra *catalog.executions* e *catalog.event_messages* è particolarmente importante per l'analisi delle performance e la risoluzione dei problemi. Ogni esecuzione registrata in *catalog.executions* può avere molteplici eventi associati in *catalog.event_messages*, consentendo di esaminare in dettaglio il comportamento di un pacchetto SSIS nel tempo. Analizzando questi dati, è possibile identificare le fasi critiche di un processo ETL, individuare eventuali errori e ottimizzare le prestazioni delle pipeline di dati.
In seguito queste due viste verranno utilizzate ed analizzate nel dettaglio, per cui è necessario capirne a fondo la struttura e i contenuti. In Appendice A, come riportato nella documentazione Microsoft [17], vengono riportati tutti i campi delle due viste per intero. In questa sezione, invece, vengono visualizzate solo le informazioni relative alle colonne utilizzate nella trattazione a seguire.
Per la vista *catalog.executions*, si considerano:

Nome colonna	**Tipo di dati**	**Descrizione**
execution_id	bigint	Identificatore univoco per l'istanza di esecuzione.
folder_name	sysname(nvarchar(128))	Nome della cartella in cui è contenuto il progetto.
project_name	sysname(nvarchar(128))	Nome del progetto.
package_name	nvarchar(260)	Nome del primo pacchetto avviato durante l'esecuzione.
status	int	Stato dell'esecuzione: Creata (1), In esecuzione (2), Operazione annullata (3) Operazione non riuscita (4), In sospeso (5), Terminata in modo inatteso (6), Operazione riuscita (7), Arresto in corso (8) Operazione completata (9).
start_time	datetimeoffset	Ora di avvio dell'istanza di esecuzione.
end_time	datetimeoffset	Ora di fine dell'istanza di esecuzione.

Per quanto concerne la vista *catalog.event_messages*, invece, le informazioni di interesse sono racchiuse nelle colonne a seguire:

Nome colonna	**Tipo di dati**	**Descrizione**
Event_message_ID	bigint	ID univoco del messaggio dell'evento.
Operation_id	bigint	Tipo di operazione.
Message_time	datetimeoffset(7)	Ora di creazione del messaggio.
Message	nvarchar(max)	Testo del messaggio.
Message_source_name	nvarchar(4000)	Componente del pacchetto che rappresenta l'origine del messaggio.
Event_name	nvarchar(1024)	Evento di run-time associato al messaggio.
Execution_path	nvarchar(max)	Percorso completo dal pacchetto padre al punto in cui viene eseguito il componente, incluse le iterazioni.

Le due viste trovano corrispondenza attraverso il campo **execution_id**, presente in *catalog.executions*, che costituisce un sottoinsieme delle istanze del campo operation_id, presente in *catalog.event_messages*.

4.4.2 Event Handler

Le funzionalità di monitoraggio includono l'uso degli Event Handlers, che consentono di eseguire azioni specifiche in risposta a eventi generati da pacchetti e attività in fase di esecuzione. Attraverso la scheda **Gestori eventi** è possibile creare gestori di eventi personalizzati per rispondere a tali eventi, estendere le funzionalità dei pacchetti e semplificarne la gestione in fase di esecuzione. Ad esempio, è possibile configurare un gestore di evento per inviare una notifica via email nel caso in cui un'attività non venga completata correttamente.
In SSIS sono disponibili diversi gestori di eventi, ciascuno associato a eventi specifici che possono verificarsi durante l'esecuzione del pacchetto. Di seguito, i dettagli per ciascuno [17]:

- **OnError**: generato quando si verifica un errore durante l'esecuzione;
- **OnExecStatusChanged**: generato quando un'attività viene sospesa o ripresa durante il debug;
- **OnInformation**: generato durante la convalida e l'esecuzione di un eseguibile per la segnalazione di informazioni;
- **OnPostExecute**: generato da un eseguibile al completamento delle attività;
- **OnPostValidate**: generato dopo la convalida del file eseguibile;

- **OnPreExecute**: generato immediatamente prima dell'esecuzione del file eseguibile;
- **OnPreValidate**: generato all'avvio della convalida del file eseguibile;
- **OnProgress**: generato dopo un avanzamento misurabile dell'esecuzione del file eseguibile;
- **OnQueryCancel**: generato da un eseguibile per determinare se l'esecuzione deve essere arrestata;
- **OnTaskFailed**: generato quando un'attività ha esito negativo;
- **OnVariableValueChanged**:generato quando il valore di una variabile viene modificato;
- **OnWarning**: generato quando viene segnalato un avviso, indicante una criticità di livello inferiore rispetto all'errore.
- **PipelineComponentTime**: generato per ogni componente del flusso di dati, per ogni fase di convalida ed esecuzione. La voce di log specifica il tempo di elaborazione per ogni fase;
- **Diagnostic o DiagnosticEx**: viene registrata una voce di log che fornisce informazioni diagnostiche, ad esempio in corrispondenza di una chiamata a un provider di dati esterno, oppure quando si vogliono trovare i nomi delle colonne nel flusso di dati che contengono errori.

4.4.3 Punti di forza e limiti del Monitoraggio in SSIS

Il monitoraggio in SSIS offre diversi vantaggi, tra cui la sua **integrazione nativa** con SQL Server, che semplifica la gestione dei processi ETL all'interno dell'ecosistema Microsoft. Inoltre, SSIS garantisce una **scalabilità** adeguata per gestire grandi volumi di dati, dà la possibilità di combinare le funzionalità di logging e supporta il tracciamento degli eventi in tempo reale. Tuttavia, il sistema presenta alcune limitazioni che possono rendere complessa l'analisi dettagliata delle esecuzioni e giustificano la necessità di un approccio più strutturato.
Uno dei principali limiti è rappresentato dalla necessità di **interrogare tramite query** il catalogo SSISDB per ottenere le informazioni ricercate, e dalle relative difficoltà che ne possono derivare. Sebbene SSISDB contenga un'ampia quantità di informazioni sulle esecuzioni, infatti, l'accesso ai dati richiede la scrittura di query SQL complesse a causa della frammentazione delle informazioni tra diverse tabelle e viste, che non sono sempre chiaramente collegate tra loro. Ad esempio, per ottenere un quadro completo di un'esecuzione, è spesso necessario unire manualmente più viste, come *catalog.executions*, *catalog.event_messages* e *catalog.operation_messages*, aumentando il rischio di query inefficienti e difficili da manutenere.
Un altro aspetto critico riguarda la struttura delle viste predefinite, che non forniscono un modello relazionale chiaro per analizzare le esecuzioni in modo semplice e immediato. La

frammentazione delle informazioni utili in più tabelle e la mancanza, talvolta, di relazioni esplicite tra le stesse, rende complesso costruire report dettagliati senza un'analisi preliminare approfondita della struttura del database. Questo limita l'efficacia del monitoraggio per gli utenti meno esperti, costringendoli a sviluppare soluzioni personalizzate per aggregare e visualizzare correttamente i dati.
Un ulteriore problema riguarda la **reportistica già esistente**, che risulta spesso **non sufficiente** e inadeguata per soddisfare le esigenze di un monitoraggio avanzato. Sebbene SqlServer Management Studio (SSMS) offra alcune funzionalità di base per visualizzare lo storico delle esecuzioni e gli errori, la mancanza di dashboard interattive e report intuitivi rende difficile individuare rapidamente anomalie e trend critici. Inoltre, le funzionalità di reporting di SSMS non sono facilmente personalizzabili, costringendo gli utenti a ricorrere a strumenti esterni come Power BI per una visualizzazione più efficace dei dati.
Altri limiti includono la **complessità iniziale** nella configurazione e la **necessità di una manutenzione periodica** del database SSISDB. Poiché il catalogo SSISDB accumula un elevato numero di log nel tempo, senza una gestione adeguata può degradare le prestazioni del server SQL. É quindi necessario implementare strategie di pulizia periodica dei dati storici per evitare impatti negativi sulla reattività delle interrogazioni e sull'utilizzo delle risorse.

Per superare queste limitazioni, è possibile adottare un modello di monitoraggio personalizzato che organizzi i dati in modo più strutturato e accessibile. Questo approccio, integrato con strumenti di visualizzazione avanzati come Power BI, consente di migliorare la reportistica e rendere il monitoraggio più intuitivo ed efficace.

Capitolo 5

Progettazione del sistema di monitoraggio

In questo capitolo viene progettato il sistema di monitoraggio nella sua interezza. Vengono identificati i KPI che si vogliono monitorare, determinati sulla base delle problematiche maggiormente riscontrate e delle specifiche esigenze dei clienti. Definiti questi indicatori, si procede alla costruzione del modello dati, adottando un'architettura a stella sulla base delle tabelle di log che erano state identificate come le più pertinenti, insieme ai relativi attributi. Successivamente, viene implementato un flusso SSIS per gestire l'estrazione, la trasformazione e il caricamento dei dati, assicurando il trasferimento delle informazioni di interesse dalle tabelle di log al modello dimensionale. Infine, il sistema viene testato su due distinti progetti di integrazione dati, sviluppati per clienti differenti, al fine di validarne l'efficacia e l'affidabilità.

5.1 I progetti

L'analisi prenderà in esame due progetti preesistenti, utilizzandoli come base di partenza per lo studio e lo sviluppo del sistema di monitoraggio. A tal fine, i flussi ETL attualmente impiegati nelle due aziende saranno esaminati nel dettaglio, al fine di acquisire una comprensione approfondita dei processi di integrazione dati e garantire un'interpretazione chiara delle informazioni di monitoraggio ottenute.

Il primo progetto trattato riguarda un flusso realizzato per un'azienda italiana leader nel settore dei servizi e delle soluzioni digitali.
Il flusso di integrazione è schedulato per essere eseguito più volte nel corso della giornata, con aggiornamenti previsti alle ore 7:00, 8:00, 9:00, 13:30 e 16:00. Questa frequenza fa sì che le tabelle di log analizzate presentino un'**elevata cardinalità**: la tabella *executions* contiene più di un migliaio di record, mentre la tabella *event_messages* raggiunge l'ordine del milione. Questo volume di dati sottolinea la necessità di un sistema strutturato per la raccolta e l'analisi delle informazioni, facilitando interrogazioni efficienti e la visualizzazione dei KPI di interesse.

Il flusso di integrazione dati già implementato in SSIS per questo progetto segue quasi integralmente le fasi previste dal framework aziendale. In particolare, sono stati implementati i tre livelli logici (L0, L1, L2) necessari per la costruzione di un modello dati strutturato, comprendente sia tabelle dei fatti che tabelle dimensionali. All'interno di ciascun livello, ci sono pacchetti per gestire speratamente movimenti e anagrafiche.
La configurazione dei file sorgente consente di omettere la fase di Staging (STG), permettendo di derivare direttamente il contenuto delle tabelle di Delta (DLT) individuando le differenze rispetto ai caricamenti precedenti.

Il flusso di elaborazione ha inizio con l'esecuzione del pacchetto principale **000_MAIN**, il quale gestisce le relazioni tra gli altri pacchetti e coordina l'intera esecuzione del processo ETL. La prima operazione prevista all'interno di questo pacchetto è un'istruzione di tipo *Execute Package Task*, che esegue il pacchetto **UT_SEND_EMAIL**. Quest'ultimo si occupa dell'invio di una notifica via e-mail ai soggetti interessati, informandoli dell'avvio dell'attività. Il testo viene generato tramite l'esecuzione di una query SQL a seconda del contesto, mentre l'invio effettivo avviene attraverso l'esecuzione di uno script precompilato.
A seguito dell'invio della mail, viene eseguita un'operazione di tipo *Expression Task*, finalizzata alla determinazione del **JOBID**, che viene generato e assegnato ad un'apposita variabile e propagato poi lungo l'intero flusso dati.
Successivamente, vengono eseguiti in sequenza i pacchetti responsabili dell'elaborazione dei tre livelli logici L0, L1 e L2, ovvero 000_MAIN_L0, 001_MAIN_L1 e 002_MAIN_L2. Il passaggio da un pacchetto al successivo avviene solo dopo il completamento dello stesso e, in caso di errore, viene nuovamente inviata una notifica e-mail tramite un'operazione di tipo *Execute Package Task*. Il contenuto del messaggio varia in funzione del motivo della segnalazione, così da fornire un'informazione dettagliata sulla natura dell'errore. Al termine dell'intero flusso di elaborazione, viene inviata un'ulteriore e-mail per notificare la conclusione delle attività.

Il primo pacchetto figlio eseguito, e dunque il primo analizzato nel dettaglio, è quello denominato **000_MAIN_L0**. Al suo interno, l'elaborazione avviene in parallelo grazie a un'operazione di tipo *Sequence Container*, che avvia simultaneamente due pacchetti figli: **010_L0_ANAGRAFICHE** e **010_L0_MOVIMENTI**. Ciascuno di essi gestisce il caricamento delle tabelle di Delta per due gruppi di dati distinti, corrispondenti a due sorgenti indipendenti: AS400 e CRM.
Per quanto riguarda le anagrafiche, il calcolo delle DLT viene effettuato con metodologie differenti a seconda del gruppo di riferimento. Nel caso di AS400, il processo avviene tramite un'operazione di tipo *Data Flow Task*, che esegue una query direttamente sulla sorgente dati e inserisce i risultati nella tabella di destinazione predefinita. Per il gruppo CRM, invece, il caricamento delle DLT viene gestito mediante pacchetti dedicati, i quali eseguono il medesimo processo attraverso l'utilizzo combinato di uno script e una stored procedure.
Un approccio analogo viene adottato per il trattamento dei movimenti, garantendo la corretta estrazione e caricamento delle informazioni nelle tabelle di DLT.

Il secondo pacchetto figlio a essere richiamato ed eseguito è **001_MAIN_L1**. Al suo interno, l'elaborazione è suddivisa in due distinti *Sequence Container*, rispettivamente dedicati alla gestione delle anagrafiche e dei movimenti.
Per quanto riguarda le **anagrafiche**, viene eseguita una stored procedure che agisce direttamente sulle tabelle di ODS attraverso un'operazione di *merge* tra i dati già presenti e le nuove informazioni provenienti dalle tabelle di DLT. All'interno dello stesso *Sequence Container*, viene poi eseguito un ulteriore pacchetto responsabile dell'aggiornamento delle tabelle di MDM sempre in *merge*.
Per i **movimenti**, il processo segue lo stesso principio, con la differenza che non è previsto l'aggiornamento delle tabelle MDM. Le tabelle in ODS vengono aggiornate separatamente per i due gruppi di dati, AS400 e CRM, attraverso due pacchetti distinti con query di *merge*.

L'ultimo pacchetto eseguito nel flusso è **002_MAIN_L2**, il quale, analogamente al precedente, prevede due *Sequence Container* dedicati ad anagrafiche e movimenti.
In questa fase conclusiva, i dati vengono riorganizzati per rispondere alle esigenze di visualizzazione e analisi successive. Nello specifico, a partire dalle informazioni di anagrafica, vengono costruite le **tabelle delle dimensioni**, mentre dai movimenti vengono generate le **tabelle dei fatti**. Per la creazione delle dimensioni, vengono eseguiti due pacchetti distinti per i gruppi AS400 e CRM, seguiti da un terzo pacchetto dedicato alla definizione della dimensione CLIENTE tramite le tabelle di MDM. Al termine del processo, le dimensioni risultanti sono: ORDINE, FATTURA, STATUS, STATO_TRATTATIVA, MACRO_STATO_TRATTATIVA, RISORSA, BUSINESSUNIT, TRATTATIVA, TIPO_OPPORTUNITÀ, OPPORTUNITÀ, SALES_TEAM, ACCOUNTS, CLIENTE.
Per quanto riguarda i movimenti, vengono costruite le tabelle dei fatti relative a TRATTATIVE e OPPORTUNITÀ a partire dai dati relativi alla sorgente CRM.

Il secondo progetto in analisi, invece, è stato svolto per un'azienda storica del settore Food & Beverage.
Il flusso di integrazione dati è programmato per essere eseguito a intervalli regolari nel corso della giornata. Le schedulazioni previste avvengono alle ore 02:00, 10:00, 14:00 e 18:00.
Per quanto riguarda le dimensioni delle tabelle di log di interesse, il **volume di dati** gestito è **significativo**. In particolare, la tabella *event_messages* contiene circa 12 milioni di record, mentre la tabella *executions* registra approssimativamente 17.000 occorrenze.
Il flusso di integrazione dati progettato per questa azienda è significativamente **meno articolato** del precedente e non segue integralmente i passaggi previsti dal framework aziendale. In particolare, vengono eseguiti i livelli L0 e L2, effettuando quindi una riorganizzazione delle informazioni contenute nei file sorgente all'interno di un Data Warehouse. Questi file (di tipo *Excel* e *csv*) provengono da un ERP (Enterprise Resource Planning) e contengono un volume significativo di informazioni relative a vendite, acquisti e clienti.

Dal punto di vista implementativo, anche questo flusso è stato interamente realizzato

in SSIS e prevede l'esecuzione sequenziale di **tre pacchetti**:

1. **Populate_Staging_Anagrafiche**
2. **Populate_Staging_Fatti**
3. **Populate_DWH**

In questo contesto, il processo di integrazione dati include esclusivamente i livelli L0 (realizzato mediante i primi due pacchetti di staging) e L2 (popolato tramite Populate_DWH). In tutti e tre i pacchetti, il caricamento avviene in modalità *full replace*: per ciascuna tabella di anagrafiche e movimenti, nonché successivamente per le tabelle di dimensioni e fatti, viene eseguita inizialmente un'operazione di *truncate table* tramite un'*Attività Esegui SQL* e, successivamente, un'*Attività Flusso di Dati* si occupa di copiare i dati dalla sorgente alle tabelle di destinazione previste.
Nel dettaglio, per il livello L0, i dati provenienti dai file sorgente vengono caricati nelle rispettive tabelle di staging. Per il livello L2, invece, il processo prevede il trasferimento delle informazioni dalle tabelle di staging in L0 alle corrispondenti tabelle di dimensione costruite in L2, finalizzando così l'organizzazione dei dati all'interno del Data Warehouse. Vengono così definite tutte le informazioni su VENDITE, ACQUISTI, BUDGET, TRANSAZIONI FINANZIARIE e SCADENZARIO CLIENTI.
Tale approccio è stato scelto in quanto il volume di dati gestito non risulta particolarmente elevato, rendendo questa strategia efficiente e sostenibile.

5.2 Identificazione KPI da monitorare

La scelta dei **KPI (Key Performance Indicators)** adeguati è fondamentale per garantire un monitoraggio efficace dei flussi di integrazione dati. Un buon set di KPI permette di identificare tempestivamente anomalie, colli di bottiglia e problemi operativi, migliorando così l'affidabilità e le prestazioni del sistema ETL. Indicatori ben definiti devono essere chiari, misurabili e pertinenti rispetto agli obiettivi aziendali, consentendo agli utenti di ottenere una visione completa dell'efficacia del processo di integrazione.
Gli indicatori sono stati definiti a partire dalle esigenze operative e strategiche emerse dall'esperienza aziendale: l'analisi approfondita dei processi e delle criticità ripetutamente riscontrate nei progetti ha permesso di individuare i KPI più significativi, contribuendo a sviluppare una soluzione non solo efficace, ma anche flessibile e applicabile non solo ai progetti attuali, ma anche ad altri scenari aziendali con esigenze analoghe.
Gli indicatori selezionati sono:

1. **Numero totale di esecuzioni**, relativo ad un periodo temporale specifico: misura la quantità di esecuzioni dei flussi ETL in un determinato periodo. Questo KPI è essenziale per valutare la frequenza e la regolarità dell'aggiornamento dei dati;
2. **Percentuale di esecuzioni completate con successo**: indica l'affidabilità del processo ETL e consente di valutare eventuali tendenze di instabilità nel tempo;

3. **Tempo medio di esecuzione** per pacchetto: aiuta a monitorare le prestazioni del flusso ETL, individuando eventuali rallentamenti che potrebbero impattare la disponibilità dei dati;

4. **Numero totale di errori e warning**: fornisce una misura della stabilità del processo in una fascia temporale prestabilita, consentendo di identificare le esecuzioni più problematiche e di approfondire le cause sottostanti;

5. **Frequenza delle tipologie degli errori rilevati**: consente di identificare ed isolare le categorie di errori maggiormente problematiche per facilitare l'analisi delle cause ricorrenti e supportare azioni correttive mirate.

Questi KPI coprono sia l'affidabilità che la performance del flusso ETL, permettendo di valutare il corretto funzionamento dei processi, individuare trend negativi e supportare decisioni basate su dati concreti, migliorando così la governance del sistema di integrazione dati.

5.3 Scelta degli strumenti

Per l'integrazione e il monitoraggio dei processi ETL, è stato scelto **SQL Server Integration Services**, una soluzione consolidata e ampiamente utilizzata, anche a livello aziendale, per la gestione dei flussi di dati. SSIS offre strumenti avanzati per l'estrazione, trasformazione e caricamento, supportando automazione, logging e gestione degli errori. Inoltre, la scelta di SSIS è stata dettata dalla necessità di mantenere coerenza con le soluzioni già implementate nei progetti aziendali trattati, rendendo più agevole l'integrazione del nuovo flusso di monitoraggio. Per creare, modificare e gestire il progetto in SSIS, viene utilizzato l'ambiente di sviluppo integrato **Visual Studio**, mentre per lo sviluppo e la gestione dei database è stato adottato **SQL Server Management Studio** (SSMS), utile per attività come la definizione delle tabelle necessarie e la scrittura e l'esecuione di query.

Per la visualizzazione e l'analisi dei dati, la scelta è ricaduta su **Power BI**, uno strumento leader nel settore della Business Intelligence, che permette di trasformare i dati raccolti in dashboard interattive e report dinamici. Grazie alla sua nativa compatibilità con SQL Server e alla sua predisposizione per la gestione di modelli a stella, Power BI consente di connettere il modello sviluppato per il monitoraggio SSIS, trasformando i dati grezzi in KPI e visualizzazioni intuitive ed immediatamente accessibili agli utenti. Le sue capacità di aggiornamento automatico e condivisione efficace all'interno dell'organizzazione migliorano la fruibilità delle informazioni e facilitano il processo decisionale.

L'utilizzo combinato di SSIS per l'integrazione dei dati e Power BI per la visualizzazione consente quindi di sviluppare un sistema di monitoraggio completo ed efficace, offrendo un valore aggiunto in termini di controllo sulle esecuzioni, analisi delle prestazioni e gestione proattiva degli errori.

5.4 Definizione del modello

Nel presente lavoro, il modello dati utilizzato per l'analisi delle esecuzioni dei pacchetti SSIS è stato progettato seguendo un'**architettura a stella**. Questa scelta è stata effettuata perchè ottimale per assecondare le esigenze di analisi e reporting in Power BI, che predilige modelli semplificati e facilmente navigabili. Il modello descritto di seguito, e visibile in figura 5.1, corrisponde a quello generato nelle tabelle di PUB della fase L2 del flusso dati, derivato dalle tabelle di log di SSISDB e illustrato nel dettaglio in seguito. La struttura permette di raccogliere e organizzare in modo efficiente le informazioni relative ad esecuzioni e relativi errori, rendendole pronte per l'analisi e la visualizzazione in Power BI.

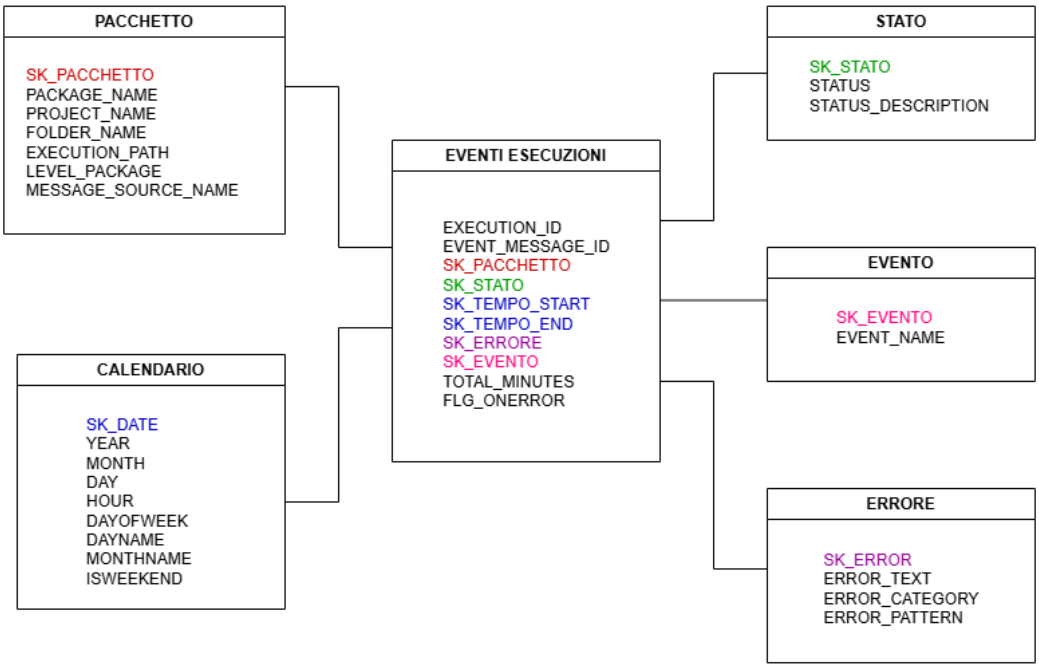

Figura 5.1. Schema a stella del modello progettato

La **dimensione STATO** è una tabella statica, caricata *una tantum*, che associa a ciascuno stato dell'esecuzione un identificativo univoco e una descrizione testuale. I campi contenuti sono dunque:

- SK_STATO (int),
- STATUS (int),

- STATUS_DESCRIPTION (varchar(100)).

La **dimensione EVENTO** è anch'essa una dimensione statica e raccoglie le diverse tipologie di eventi, associate ad una chiave surrogata. I campi contenuti sono:

- SK_EVENTO (int),
- EVENT_NAME (varchar(400)).

La **dimensione CALENDARIO** è una classica tabella calendario che fornisce il dettaglio temporale per anno, mese, giorno, fino al livello dell'ora. Vengono salvate anche informazioni aggiuntive quali il giorno della settimana, sia in formato numerico che testuale, il nome del mese di appartenenza e un campo per distinguere tra infrasettimanale e fine settimana. La chiave viene gestita con un campo derivato nel formato YYYYMMDDHH, che mette insieme data ed ora, permettendo analisi dettagliate su intervalli temporali diversi. Contiene:

- SK_DATE (int),
- YEAR (int),
- MONTH (int),
- DAY (int),
- HOUR (int),
- DAYOFWEEK (int) (1 = Domenica, 7 = Sabato),
- DAYNAME (varchar(20)),
- MONTHNAME (varchar(20)),
- ISWEEKEND (int).

La **dimensione PACCHETTO** descrive le informazioni chiave sui pacchetti SSIS eseguiti, includendo il nome della cartella di appartenenza (folder name), il nome del progetto di riferimento (project name), il nome del pacchetto (package name), il percorso di esecuzione, a partire dal pacchetto di appartenenza, della componente specifica eseguita (execution path), il nome della componente specifica (message source name) e il livello di profondità di tale componente nel flusso (level package). Tutti questi attributi, escluso il nome della componente ed il livello, sono chiavi esterne che permettono di collegare i pacchetti agli eventi delle esecuzioni registrate. I campi risultanti sono quindi:

- SK_PACCHETTO (int),
- PACKAGE_NAME (varchar(400)),
- PROJECT_NAME (varchar(400)),
- FOLDER_NAME (varchar(400)),

- EXECUTION_PATH (varchar(4000)),
- LEVEL_PACKAGE (int),
- MESSAGE_SOURCE_NAME (varchar(400)).

Per l'analisi degli errori viene creata la **dimensione ERRORE**, che categorizza gli eventi di errore e di avviso, ovvero quelli con *event_name* = 'OnError' ed *event_name* = 'OnWarning', in base ai seguenti attributi: il tipo di evento (errore o avviso), il messaggio stesso di errore, la **categoria** a cui viene assegnato e il pattern, che può anche essere considerata una sottocategoria, ovvero la stringa presente all'interno del messaggio che consente l'attribuzione della categoria. Per fare ciò, è stata precedentemente effettuata un'analisi dettagliata di tutti i messaggi riscontrati in entrambi i processi: sono state analizzate le stringhe per individuare delle porzioni ricorrenti e, sulla base di questo, sono stati creati dei cluster affini per tipologia di errore. Maggiori dettagli su questa classificazione sono disponibili in appendice C. Una volta individuati e definiti questi attributi, l'associazione di categoria e pattern specifici è stata effettuata all'interno della query di *insert* con una ricerca per somiglianza. All'interno della dimensione in esame vi è un ulteriore campo (error hash), che costituisce la chiave primaria della tabella: si tratta di una stringa univoca per ogni messaggio di errore. Viene introdotta per facilitare il trasferimento dei dati, vista la lunghezza notevole che può raggiungere il messaggio originale. Anche in questo caso, le informazioni vengono estratte in modo univoco attraverso l'utilizzo di una **tabella temporanea** contenente i messaggi distinti con hash associato, di supporto nel passaggio dalla OUT alla dimensione stessa. Il collegamento alla tabella dei fatti avviene sempre tramite una chiave esterna. I campi contenuti nella dimensione sono dunque:

- SK_ERROR (int),
- ERROR_TEXT (varchar(8000)),
- ERROR_CATEGORY (varchar(50)),
- ERROR_PATTERN (varchar(255)).

La tabella centrale, **EVENTI ESECUZIONI**, rappresenta la **tabella dei fatti** e raccoglie tutti gli eventi significativi associati alle esecuzioni dei pacchetti SSIS. I campi, sia misure che chiavi surrogate, sono di tipo numerico per favorire l'efficienza computazionale. Le misure principali vengono derivate e includono un indicatore di performance, ovvero la durata dell'esecuzione in minuti, e un flag specifico per il monitoraggio del successo dell'esecuzione. Vengono poi inserite le chiavi surrogate (SK) provenienti dalle dimensioni e l'execution_id e l'event_message_id, identificatori dell'evento specifico di un'esecuzione. Al suo interno troviamo quindi:

- EXECUTION_ID (int),
- EVENT_MESSAGE_ID (int),
- SK_PACCHETTO (int),

- SK_STATO (int),
- SK_TEMPO_START (bigint),
- SK_TEMPO_END (bigint),
- SK_ERRORE (int),
- SK_EVENTO (int),
- TOTAL_MINUTES (float),
- FLG_ONERROR (int).

5.5 Flusso SSIS per le due tabelle di catalogo

I progetti dei due clienti vengono gestiti in maniere differenti.
Per quanto riguarda il primo cliente introdotto, la scelta è stata quella di effettuare un caricamento *una tantum* dei dati delle tabelle di log mediante dei file csv che forniscono dunque un'immagine istantanea dei dati. Di conseguenza, non risulta in questo caso utile effettuare tutti gli step di trasformazione previsti dal framework, ma viene eseguito il caricamento in STG e poi direttamente in OUT.
In merito al secondo cliente, invece, si è scelto di costruire un flusso in SSIS in modo tale da consentire un aggiornamento periodico ed efficiente dei dati, con la possibilità di introdurre una schedulazione regolare. Questo è stato reso possibile grazie alla disponibilità della connessione ai gestionali del cliente e, di conseguenza, anche al catalogo SSISDB.

La creazione del flusso in SSIS per il passaggio dei dati dalle tabelle di log selezionate a quelle del modello a stella definito ha seguito, per quanto possibile, il framework aziendale spiegato in precedenza.
Come prima cosa, viene stabilita la connessione al database aziendale usato per contenere il modello di destinazione e al database del cliente in cui sono contenute le informazioni di interesse.
Viene creato il pacchetto padre 000_MAIN, definito come visibile in figura 5.2, che sarà responsabile dell'esecuzione dell'intero flusso. Al suo interno, vengono definiti in maniera appropriata variabili e parametri necessari e viene calcolato il JOBID, fondamentale nella gestione di diversi inserimenti, mediante l'espressione:

$$\begin{aligned} @[User :: JOBID] = ((DT\_I8)\,\mathrm{YEAR}(\mathrm{GETDATE}())) \times 10000 \times 1000000 \\ + \mathrm{MONTH}(\mathrm{GETDATE}()) \times 100000000 \\ + \mathrm{DAY}(\mathrm{GETDATE}()) \times 1000000 \\ + \mathrm{DATEPART}(\text{"}HH\text{"}, \mathrm{GETDATE}()) \times 10000 \\ + \mathrm{DATEPART}(\text{"}MI\text{"}, \mathrm{GETDATE}()) \times 100 \\ + \mathrm{DATEPART}(\text{"}SS\text{"}, \mathrm{GETDATE}()) \end{aligned} \tag{5.1}$$

Vengono a questo punto eseguiti in successione i pacchetti di 000_MAIN_L0, 001_MAIN_L1 e 002_MAIN_L2.

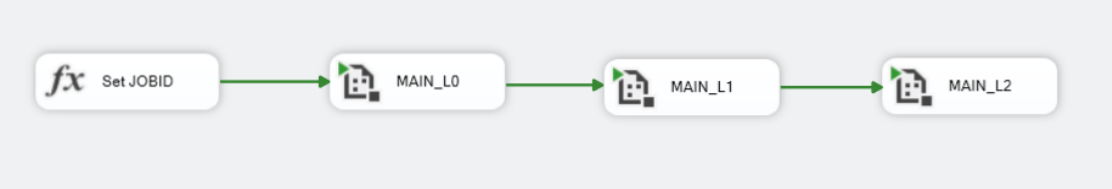

Figura 5.2. Flusso di controllo pacchetto 000_MAIN

Il pacchetto 000_MAIN_L0 prevede l'esecuzione di un singolo sottopacchetto denominato 010_L0_MOVIMENTI; sebbene questo passaggio potrebbe sembrare ridondante, è stato aggiunto per completezza e per poter eventualmente adattare il flusso con ampliamenti successivi. Al suo interno, è prevista la scrittura sulla FLOW_MANAGER, tabella di metadati di supporto, prima e dopo l'esecuzione del pacchetto responsabile delle fasi di STG e DLT, al fine di registrare le informazioni relative al flusso quali livello, stato, orari di inizio e fine (figura 5.3). Il pacchetto di STG e DLT, infine, costituisce il fulcro della

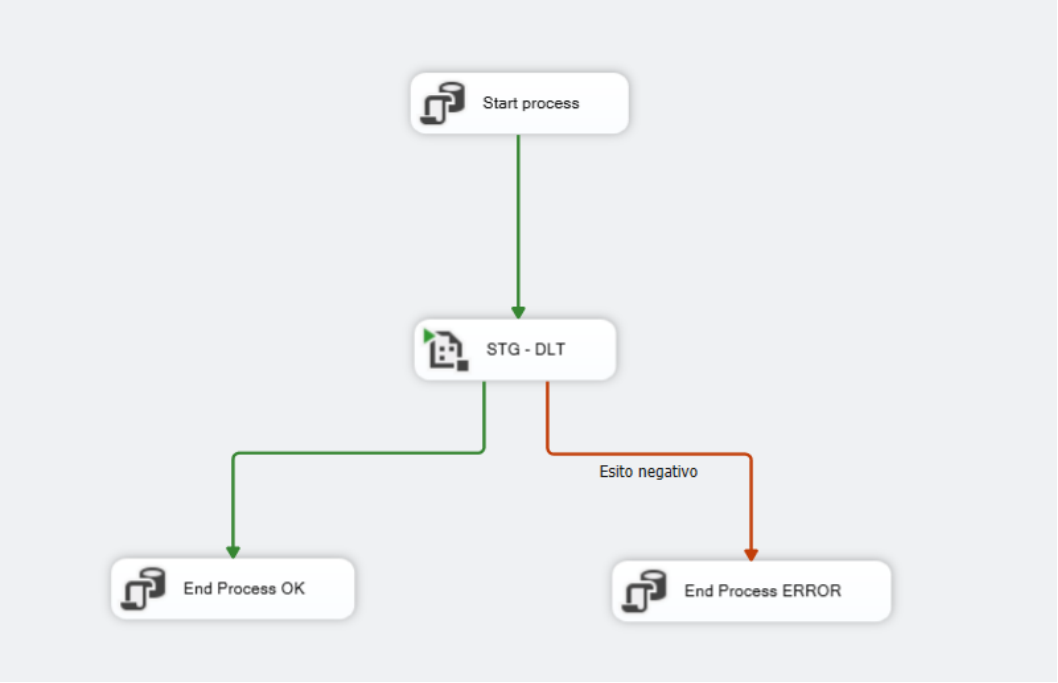

Figura 5.3. Flusso di controllo pacchetto 010_L0_MOVIMENTI

fase di L0: contiene gli step a cui devono essere sottoposti i dati per essere conformi con le caratteristiche previste per definizione dalle tabelle di destinazione. Al suo interno viene effettuata in un primo momento la lettura da sorgente (pssaggio da source a STG, con aggiunta della colonna relativa al JOBID per adattare il formato tabellare), escludendo tutti i record relativi ad *event_name* uguale a *OnPreValidate* e *OnPostValidate* a causa della loro forte incidenza numerica e perchè ritenuti poco utili ai fini dell'analisi, e subito dopo il caricamento delle DLT mediante query di *minus* (che può essere visionata in appendice B), entrambe con un'attività flusso di dati. Una volta completate queste operazioni, le informazioni sulla lettura (numerosità e tempo di inserimento per ogni tabella letta) vengono scritte sulla tabella di metadati TABLE_MANAGER. I dettagli si possono visionare nelle figure 5.4 e 5.5.

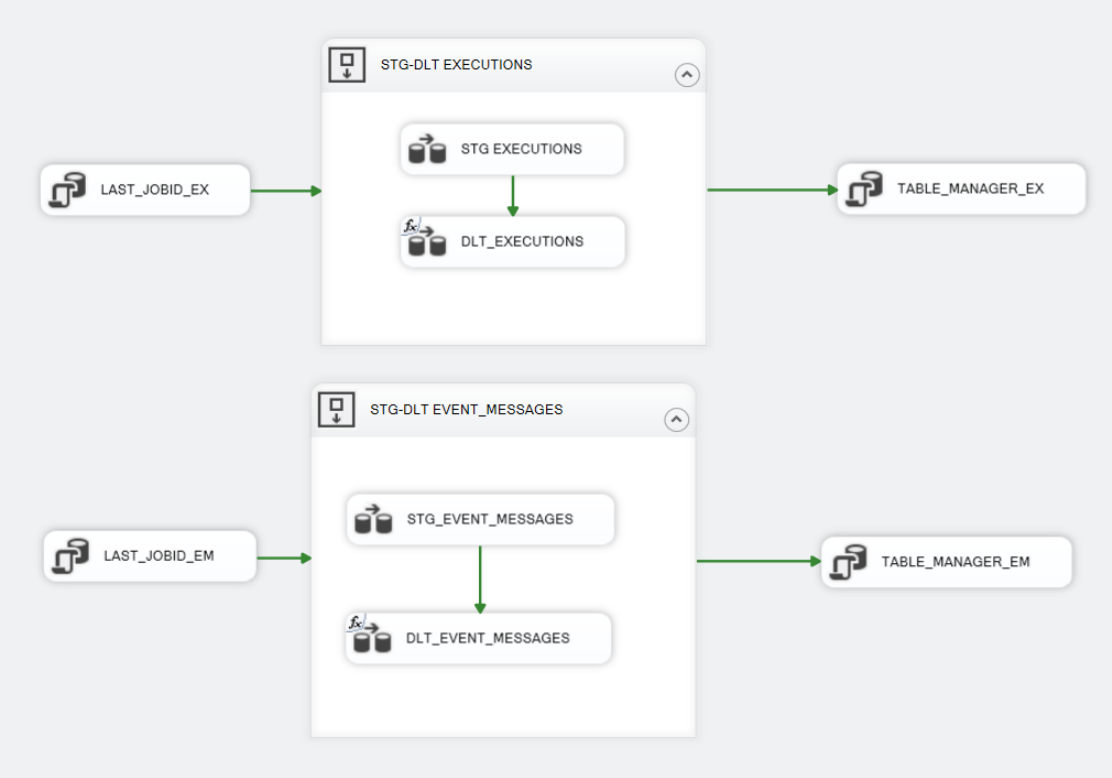

Figura 5.4. Flusso di controllo pacchetto 100_STG_DLT

Successivamente si passa al pacchetto 001_MAIN_L1, in cui si eseguono le trasformazioni che determinano la definizione delle tabelle di OK e ODS. Anche in questo caso vi è il pacchetto di transizione L1_MOVIMENTI al cui interno si richiama nuovamente la tabella FLOW_MANAGER, eseguendo nel mentre le attività di flusso dati e di esegui

Figura 5.5. Flusso di dati per tabella DLT_EVENT_MESSAGES

SQL responsabili del caricamento in OK e ODS rispettivamente. Il tutto è visibile nella figura 5.6 e viene effettuato mediante query definite ad hoc, consultabili in appendice B.

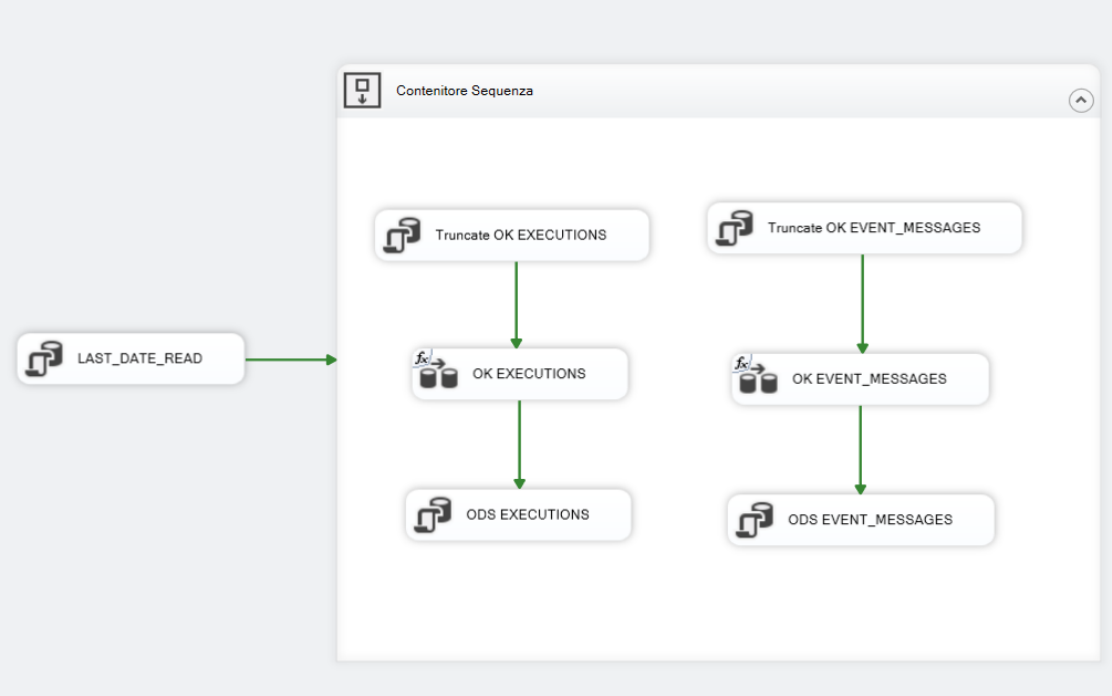

Figura 5.6. Flusso di controllo pacchetto 200_OK_ODS

Per portare a conclusione il processo, viene eseguito infine il pacchetto 002_MAIN_L2. Al suo interno, solo per i dati relativi all'inserimento più recente, viene prima effettuato il passaggio dei dati in OUT, mediante una riorganizzazione dei dati contenuti in entrambe le

tabelle con una query strutturata appositamente (appendice B), con la creazione dei campi aggiuntivi necessari per le analisi. In questo passaggio viene inostre esclusa un'ulteriore tipologia di eventi, ovvero *OnInformation*, per alleggerire gli step successivi e consentire di indirizzare il focus sugli eventi di maggiore interesse. Al termine, in un contenitore sequenza (figura 5.7), vengono eseguiti in successione i pacchetti L2_ANAGRAFICHE e L2_MOVIMENTI, con cui si raggiunge il riempimento della struttura finale definita dal modello nel paragrafo precedente. Mediante attività flusso di dati, gestite nuovamente con query di *merge* apposite (esempio fornito in appendice B), vengono caricate tutte le dimensioni (fatta eccezione per la dimensione STATO, che si ricorda essere statica e quindi inserita *una tantum*) e poi, a partire da queste, la tabella dei fatti. Le dimensioni vengono riempite secondo il seguente criterio: per i nuovi record, nel caso in cui l'identificativo sia già presente nella OUT, tutti gli altri campi vengono aggiornati con le nuove inforazioni (che possono essere cambiate), altrimenti è previsto l'inserimento di una nuova riga. Per quanto riguarda la tabella dei fatti, il meccanismo è lo stesso ma viene effettuato a partire dalla tabella che si crea unendo in *left join* la tabella di OUT con tutte le dimensioni mediante un'associazione di chiavi primarie. Una volta che tutte queste tabelle sono in relazione tra di loro, vengono selezionate ed inserite nella tabella di destinazione solo i campi di interesse. Vengono quindi tralasciate le chiavi primarie in favore delle chiavi surrogate, più adatte per le successive operazioni di *join*. In figura 5.8 il dettaglio sul pacchetto L2_ANAGRAFICHE. Analogamente poi per L2_MOVIMENTI.

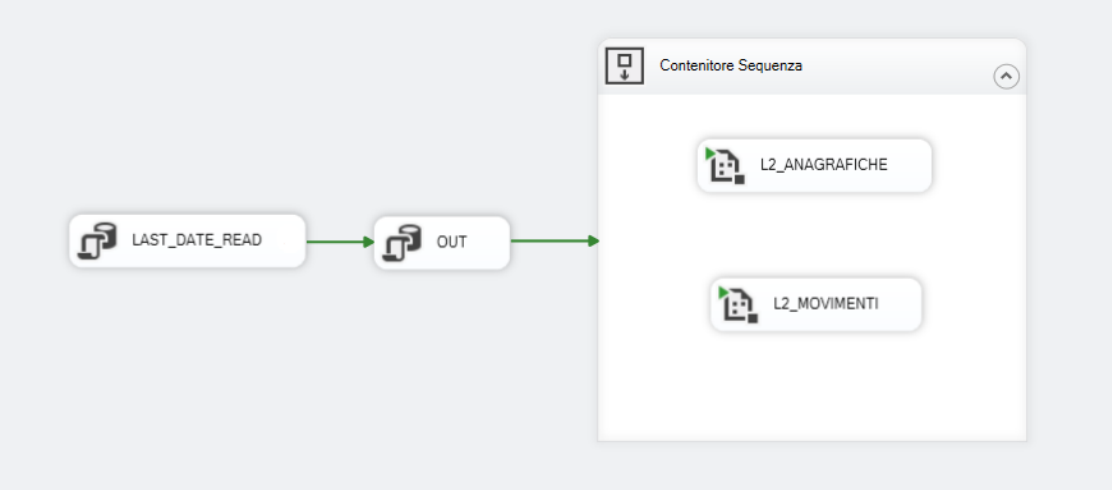

Figura 5.7. Flusso di controllo pacchetto 002_MAIN_L2

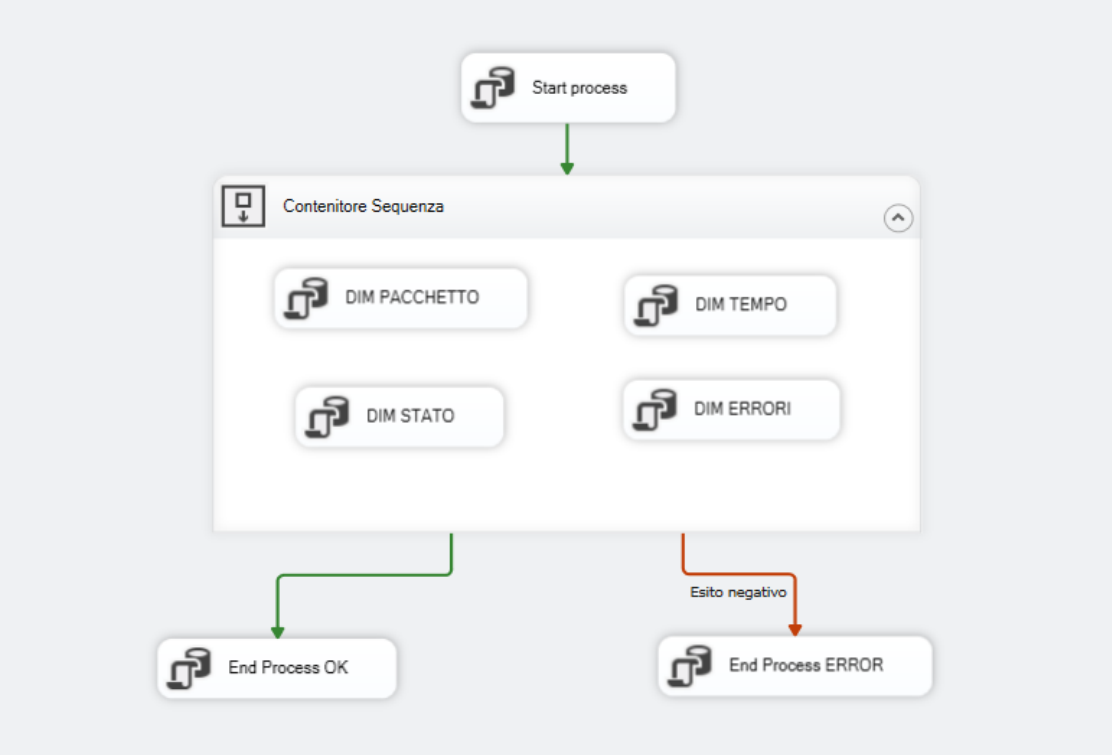

Figura 5.8. Flusso di controllo pacchetto 030_L2_ANAGRAFICHE

5.6 Dashboard

Una volta aggregate le informazioni provenienti dalle due tabelle di log, il modello è stato importato in PowerBI per creare una dashboard dinamica che offre un quadro completo dei dati raccolti e delle informazioni che possono essere ricavate a partire dagli stessi.
Una dashboard è un'interfaccia visiva che **aggrega, organizza e presenta dati chiave** in modo intuitivo e interattivo, consentendo agli utenti di monitorare indicatori di prestazione, analizzare tendenze e prendere decisioni informate. Nel caso in esame, la dashboard in Power BI serve a monitorare i flussi ETL, fornendo una **visione d'insieme delle esecuzioni, dei successi, degli errori e dei tempi di elaborazione**.

La dashboard costruita presenta un'unica pagina, in cui sono racchiuse e riassunte tutte le informazioni principali.
Nella sezione di sinistra sono riportati, in relazione ad uno specifico progetto e ad un periodo di tempo che possono essere impostati dall'utente mediante i filtri in alto a destra, gli indicatori relativi al numero di esecuzioni effettuate e al totale di avvisi (event_name = *OnWarning*) e di errori (event_name = *OnError*) che si sono presentati durante le esecuzioni. Il totale di esecuzioni viene inoltre suddiviso in base all'esito (determinato dallo

stato), in modo tale da mostrare la percentuale di operazioni terminate con successo, non riuscite, annullate o terminate in modo in atteso.
Viene poi rappresentato graficamente anche il tempo medio di esecuzione di ogni singolo pacchetto in secondi.
Un importante contributo della dashboard consiste nel poter individuare velocemente le categorie di errori ed avvisi maggiormente frequenti e, dunque, potenzialmente più impattanti nella non riuscita delle esecuzioni. A questo fine vengono visualizzate, separatamente per errori e avvisi, le occorrenze delle varie categorie precedentemente definite, ordinate dalla più impattante in maniera decrescente.
Infine, per fornire un maggiore dettaglio, è presente anche una tabella in cui vengono visualizzati, per ogni esecuzione, tutti i messaggi di errore e warning che si sono verificati, insieme alla categoria di appartenenza e al pattern contenuto.

La dashbord finale consente, in un'unica schermata, di rispondere in maniera veloce e completa a tutti i KPI che erano stati precedentemente stabiliti, con la possibilità di interagire con i singoli oggetti per avere informazioni più specifiche a seconda delle esigenze.

Il tutto è visibile nelle figure 5.9 e 5.10, relative rispettivamente al primo e al secondo cliente precedentemente menzionati. Si può notare come la diversa struttura dei due flussi corrispondenti e la varietà dei dati trattati influenzino le prestazioni monitorate e come la dashboard consenta, già a colpo d'occhio, di coglierne le differenze.
Tali differenze vengono riscontrate sia nella distribuzione degli errori e dei warning, sia nelle esecuzioni e relative percentuali di successo. Per quanto riguarda il numero di esecuzioni, si vede subito che la seconda dashboard mostra un volume molto più elevato rispetto alla prima, con tasso di successo più alto: nel flusso del primo cliente la percentuale di esecuzioni riuscite è inferiore, suggerendo potenziali criticità nel processo. Anche la distribuzione di errori e warning è visibilmente diversa, non solo in numero assoluto di occorrenze, ma anche le categorie di errore e warning più frequenti variano tra i due casi, indicando che le problematiche principali affrontate dai due clienti sono diverse. Ad esempio, nella seconda dashboard i warning sembrano essere molto più numerosi e legati prevalentemente a colonne non utilizzate mentre gli errori risultano essere riconducibili principalmente alla fase di validazione. Diversamente, nella prima dashboard emergono più errori di connessione e timeout. Infine, anche i tempi di esecuzione medi per pacchetto risultano differenti, suggerendo che i carichi di lavoro o la complessità dei processi possano essere diversi nei due ambienti. Queste differenze evidenziano come le esigenze e le criticità operative possano variare significativamente tra i clienti e, al tempo stesso, come il modello permetta un monitoraggio efficace e dettagliato in entrambe le situazioni.

Figura 5.9. Dashboard relativa al cliente 1

Figura 5.10. Dashboard relativa al cliente 2

Capitolo 6

Conclusione

In questo lavoro è stato sviluppato un sistema per il monitoraggio delle esecuzioni dei flussi ETL, partendo dall'analisi dei prerequisiti e delle tabelle di log disponibili nel catalogo SSISDB. Sono stati identificati i dati rilevanti e i KPI da tracciare, permettendo la definizione di un modello dimensionale strutturato secondo l'approccio a stella. Il popolamento delle tabelle del modello è stato realizzato mediante un flusso SSIS, garantendo un processo automatizzato e coerente di raccolta e gestione delle informazioni. La rielaborazione dei dati ha consentito di analizzare e visualizzare in modo efficace le metriche chiave attraverso una dashboard in Power BI, fornendo un sistema di monitoraggio chiaro ed accessibile.

Nonostante i risultati ottenuti, esistono alcuni limiti e margini di miglioramento che potranno essere esplorati successivamente. Un aspetto critico attualmente è la scarsità di errori registrati, dovuta all'utilizzo dei dati provenienti da flussi che sono in produzione solo dal 2024, che riduce la disponibilità di dati per le analisi. Una potenziale soluzione potrebbe essere l'impiego di dati sintetici per simulare scenari di errore e migliorare la capacità del sistema di gestire situazioni anomale.

Dal punto di vista evolutivo, diversi sviluppi futuri potrebbero arricchire il progetto.
Come prima cosa, il flusso potrebbe essere migliorato implementando direttamente il caricamento delle tabelle di DLT, evitando il passaggio intermedio tramite STG. Attualmente, il processo prevede il trasferimento di tutti i dati in STG e poi l'applicazione della logica di caricamento incrementale in DLT. Tuttavia, poiché i dati contengono un campo temporale affidabile, è possibile filtrare direttamente in fase di acquisizione solo le nuove informazioni, senza dover riprocessare i dati già caricati. Questa modifica ridurrebbe il volume di dati movimentati e il carico di lavoro sulle risorse di elaborazione, migliorando le prestazioni complessive del sistema. Inoltre, eliminando la dipendenza dallo staging, si semplificherebbe l'architettura e si ridurrebbero i tempi di latenza tra l'acquisizione e la disponibilità dei dati aggiornati.
Si può poi anche valutare l'integrazione con gli Event Handler, che potrebbero rivelarsi utili specialmente dal punto di vista della reattività delle risposte ad eventi critici; nel

progetto sviluppato questa funzionalità non è stata implementata in quanto necessiterebbe di approvazione da parte dei clienti.
Un altro sviluppo potrebbe essere l'introduzione di una componente predittiva, che consentirebbe di potenziare l'analisi dei KPI con l'uso di modelli in grado di stimare il verificarsi di errori o di prestazioni anomale dei flussi ETL. L'applicazione di tecniche di machine learning, come modelli di regressione o reti neurali, potrebbe consentire di anticipare i problemi, ottimizzare i processi e migliorare la qualità complessiva delle operazioni.
Infine, il sistema di monitoraggio sviluppato potrebbe avere un impatto ancora più significativo se consentisse l'analisi in tempo reale: attualmente, la dashboard fornisce una visione a posteriori delle esecuzioni, ma un'estensione in tempo reale permetterebbe di avere una visione costantemente aggiornata e dunque di intervenire tempestivamente in caso di criticità. Questo potrebbe essere realizzato sfruttando strumenti di streaming analytics o aggiornamenti frequenti del modello dati, ad esempio al termine di ogni singola esecuzione dei flussi dei clienti.

In conclusione, nel complesso sono già stati ottenuti risultati soddisfacenti che migliorano l'esperienza di monitoraggio e l'aggiunta di questi ulteriori sviluppi, se implementati, potrebbe contribuire nel rendere il sistema ancora più robusto e funzionale, aumentando il valore delle informazioni fornite e migliorando l'efficienza operativa complessiva.

Bibliografia

[1] S. M. F. Ali and R. Wrembel. From conceptual design to performance optimization of etl workflows: current state of research and open problems. *The VLDB Journal*, 26(6):777–801, 2017.

[2] R. Bathani. Optimizing etl pipelines for scalable data lakes in healthcare analytics. *International Journal on Recent and Innovation Trends in Computing and Communication*, 9(10):17–24, 2021.

[3] David Carter. 13 migliori strumenti etl nel 2025. URL https://guru99.com/it/best-etl-tools.html. Ultimo aggiornamento: 3 dicembre 2024.

[4] M. Demarest. The politics of data warehousing. *June, http://www. hevanet. com/demarest/marc/dwpol. html*, 6(03):1998, 1997.

[5] F. La Noce e L. D'Ercole. *Data Warehousing: dal dato all'informazione*. FrancoAngeli, 2001.

[6] S. Chaudhuri e U. Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.

[7] S.H.A. El-Sappagh, A.M.A. Hendawi, and A.H. El Bastawissy. A proposed model for data warehouse etl processes. *Journal of King Saud University-Computer and Information Sciences*, 23(2):91–104, 2011.

[8] C. Imhoff, N. Galemmo, and J. G. Geiger. *Mastering data warehouse design: relational and dimensional techniques*. John Wiley & Sons, 2003.

[9] W.H. Inmon. *Building the data warehouse*. John Wiley & Sons, 2005.

[10] A. Katari. Data quality management in financial etl processes: Techniques and best practices. *International Journal of Science and Research (IJSR)*, 8(4):2498–2501, April 2019.

[11] R. Katragadda, S. S. Tirumala, and D. Nandigam. Etl tools for data warehousing: an empirical study of open source talend studio versus microsoft ssis. 2015.

[12] R. Kimball and J. Caserta. *The data warehouse ETL toolkit*. John Wiley & Sons, 2004.

[13] R. S. Koppula. Etl automation and orchestration with apache airflow. *International Journal of Science and Research (IJSR)*, 9(10):1218–1222, October 2020.

[14] S. Ligus. *Effective monitoring and alerting*. O'Reilly Media, Inc., 2013.

[15] N. Mali and S. Bojewar. A survey of etl tools. *International Journal of Computer Techniques*, 2(5):20–27, 2015.

[16] Alex McFarland. 10 migliori strumenti etl. URL https://www.unite.ai/it/best-etl-tools/. Ultimo aggiornamento: 1 gennaio 2025.

[17] Microsoft. Sql server integration services. URL https://learn.microsoft.com/it-it/sql/integration-services/sql-server-integration-services?view=sql-server-ver16. Ultimo aggiornamento: 2 gennaio 2025.

[18] B. Oliveira, Ó. Oliveira, T. Matos, V. Santos, and O. Belo. An etl pattern log for log configuration and analysis.

[19] P. Ponniah. *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*. John Wiley & Sons, 2001.

[20] N. Rahman, J. Marz, and S. Akhter. An etl metadata model for data warehousing. *Journal of computing and information technology*, 20(2):95–111, 2012.

[21] A. Raj, J. Bosch, H. H. Olsson, and T. J. Wang. Modelling data pipelines. In *2020 46th Euromicro conference on software engineering and advanced applications (SEAA)*, pages 13–20. IEEE, 2020.

[22] G.S. Reddy, R. Srinivasu, M.P.C. Rao, and S.R. Rikkula. Data warehousing, data mining, olap and oltp technologies are essential elements to support decision-making process in industries. *International Journal on Computer Science and Engineering*, 2(9):2865–2873, 2010.

[23] D. Seenivasan. Etl vs elt: Choosing the right approach for your data warehouse. *International Journal for Research Trends and Innovation*, pages 110–122, 2022.

[24] H. Spengler, I. Gatz, F. Kohlmayer, K.A. Kuhn, and F. Prasser. Improving data quality in medical research: a monitoring architecture for clinical and translational data warehouses. In *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, pages 415–420. IEEE, 2020.

[25] W. H. Tok, R. Parida, M. Masson, and X. Ding. *Microsoft SQL Server 2012 Integration Services*. Pearson Education, 2012.

[26] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. Conceptual modeling for etl processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21, 2002.

[27] T. Vetterli, A. Vaduva, and M. Staudt. Metadata standards for data warehousing: open information model vs. common warehouse metadata. *ACM Sigmod Record*, 29 (3):68–75, 2000.

[28] H. Wang and Z. Ye. An etl services framework based on metadata. In *2010 2nd International Workshop on Intelligent Systems and Applications*, pages 1–4. IEEE, 2010.

[29] C. Xavier and F. Moreira. Agile etl. *Procedia Technology*, 9:381–387, 2013.

Appendice A

Tabelle di log

Struttura vista *catalog.executions*:

Nome colonna	**Tipo di dati**	**Descrizione**
execution_id	bigint	Identificatore univoco per l'istanza di esecuzione.
folder_name	sysname(nvarchar(128))	Nome della cartella in cui è contenuto il progetto.
project_name	sysname(nvarchar(128))	Nome del progetto.
package_name	nvarchar(260)	Nome del primo pacchetto avviato durante l'esecuzione.
reference_id	bigint	Ambiente a cui fa riferimento l'istanza di esecuzione.
reference_type	char(1)	Tipo di riferimento: relativo (R) o assoluto (A).
environment_folder_name	nvarchar(128)	Nome della cartella dell'ambiente.
environment_name	nvarchar(128)	Nome dell'ambiente usato nell'esecuzione.
project_lsn	bigint	Versione del progetto per l'istanza di esecuzione.
executed_as_sid	varbinary(85)	SID dell'utente che ha avviato l'esecuzione.
executed_as_name	nvarchar(128)	Nome dell'entità di database che ha avviato l'esecuzione.

use32bitruntime	bit	Se il pacchetto usa runtime a 32 bit (1) o a 64 bit (0).
operation_type	smallint	Tipo di operazione.
created_time	datetimeoffset	Data/ora di inizializzazione dell'istanza.
object_type	smallint	Tipo di oggetto: progetto (20) o pacchetto (30).
object_id	bigint	ID dell'oggetto coinvolto nell'operazione.
status	int	Stato dell'esecuzione: Creata (1), In esecuzione (2), Operazione annullata (3) Operazione non riuscita (4), In sospeso (5), Terminata in modo inatteso (6), Operazione riuscita (7), Arresto in corso (8) Operazione completata (9).
start_time	datetimeoffset	Ora di avvio dell'istanza di esecuzione.
end_time	datetimeoffset	Ora di fine dell'istanza di esecuzione.
caller_sid	varbinary(85)	SID dell'utente autenticato via Windows.
caller_name	nvarchar(128)	Nome dell'account che ha eseguito l'operazione.
process_id	int	ID processo del programma esterno, se applicabile.
stopped_by_sid	varbinary(85)	SID dell'utente che ha arrestato l'esecuzione.
stopped_by_name	nvarchar(128)	Nome dell'utente che ha arrestato l'esecuzione.
dump_id	uniqueidentifier	ID di un dump di esecuzione.
server_name	nvarchar(128)	Informazioni relative al server Windows.
machine_name	nvarchar(128)	Nome del computer del server SQL Server.

worker_agent_id	uniqueidentifier	ID agente di lavoro dello Scale Out Worker.
total_physical_memory_kb	bigint	Memoria fisica totale disponibile in KB.
available_physical_memory_kb	bigint	Memoria fisica disponibile in KB.
total_page_file_kb	bigint	Memoria in pagine totale in KB.
available_page_file_kb	bigint	Memoria in pagine disponibile in KB.
cpu_count	int	Numero di CPU logiche disponibili.
executed_count	int	Numero di esecuzioni dell'operazione.

Struttura vista *catalog.event_messages*:

Nome colonna	**Tipo di dati**	**Descrizione**
Event_message_ID	bigint	ID univoco del messaggio dell'evento.
Operation_id	bigint	Tipo di operazione.
Message_time	datetimeoffset(7)	Ora di creazione del messaggio.
Message_type	smallint	Tipo di messaggio visualizzato.
Message_source_type	smallint	Origine del messaggio.
message	nvarchar(max)	Testo del messaggio.
Extended_info_id	bigint	ID di informazioni aggiuntive correlate.
Package_name	nvarchar(260)	Nome del file del pacchetto.
Event_name	nvarchar(1024)	Evento di run-time associato al messaggio.
Message_source_name	nvarchar(4000)	Componente del pacchetto che rappresenta l'origine del messaggio.
Message_source_id	nvarchar(38)	ID univoco dell'origine del messaggio.
Subcomponent_name	nvarchar(4000)	Componente del flusso di dati che corrisponde all'origine del messaggio.
Package_path	nvarchar(max)	Percorso univoco del componente all'interno del pacchetto.

Execution_path	nvarchar(max)	Percorso completo dal pacchetto padre al punto in cui viene eseguito il componente, incluse le iterazioni.
threadID	int	ID per il thread in esecuzione al momento della registrazione del messaggio.
Message_code	int	Codice associato al messaggio.

Appendice B

Esempi di query

B.1 Query di DLT per la vista executions

```
SELECT UNION_1.execution_id
      , UNION_1.folder_name
      , UNION_1.project_name
      , UNION_1.package_name
      , UNION_1.status
      , UNION_1.start_time
      , UNION_1.end_time
      , UNION_1.FLG_NEG
FROM (
    /* IERI - OGGI */
    SELECT MINUS_NEG.execution_id
         , MINUS_NEG.folder_name
         , MINUS_NEG.project_name
         , MINUS_NEG.package_name
         , MINUS_NEG.status
         , MINUS_NEG.start_time
         , MINUS_NEG.end_time
         , 1 AS FLG_NEG
    FROM (
        SELECT execution_id
             , folder_name
             , project_name
             , package_name
             , status
             , start_time
             , end_time
        FROM L0.STG_EXECUTIONS STG1
        WHERE STG1.JOBID_L0 = " + (DT_WSTR, 20) @[User::LAST_JOBID] + "
        EXCEPT
        SELECT execution_id
             , folder_name
```

```
            , project_name
            , package_name
            , status
            , start_time
            , end_time
        FROM L0.STG_EXECUTIONS  STG2
        WHERE STG2.JOBID_L0 = " + (DT_WSTR, 20) @[$Package::JOBID] + "
    ) MINUS_NEG
    UNION ALL
     /* OGGI - IERI */
    SELECT MINUS_POS.execution_id
          , MINUS_POS.folder_name
          , MINUS_POS.project_name
          , MINUS_POS.package_name
          , MINUS_POS.status
          , MINUS_POS.start_time
          , MINUS_POS.end_time
          , 0 AS FLG_NEG
    FROM (
        SELECT execution_id
              , folder_name
              , project_name
              , package_name
              , status
              , start_time
              , end_time
        FROM L0.STG_EXECUTIONS  STG1
        WHERE STG1.JOBID_L0 = " + (DT_WSTR, 20) @[$Package::JOBID] + "
        EXCEPT
        SELECT execution_id
              , folder_name
              , project_name
              , package_name
              , status
              , start_time
              , end_time
        FROM L0.STG_EXECUTIONS  STG2
        WHERE STG2.JOBID_L0 = " + (DT_WSTR, 20) @[User::LAST_JOBID] + "
    ) MINUS_POS
) UNION_1
```

B.2 Query di ODS per la vista executions

```
MERGE INTO [TIROCINI DWH].L1.ODS_EXECUTIONS AS ods
USING [TIROCINI DWH].L1.OK_EXECUTIONS AS ok
ON (ods.execution_id = ok.execution_id)
WHEN MATCHED THEN
```

```
    UPDATE SET
        execution_id = ok.execution_id,
        folder_name = ok.folder_name,
        project_name = ok.project_name,
        package_name = ok.package_name,
        status = ok.status,
        start_time = ok.start_time,
        end_time = ok.end_time,
        FLG_NEG = ok.FLG_NEG,
        JOBID_UPD = ok.JOBID
WHEN NOT MATCHED THEN
    INSERT (execution_id, folder_name, project_name, package_name,
        status, start_time, end_time, JOBID, FLG_NEG, JOBID_UPD)
    VALUES (ok.execution_id, ok.folder_name, ok.project_name,
        ok.package_name, ok.status, ok.start_time, ok.end_time,
        ok.JOBID, ok.FLG_NEG, ok.JOBID);
```

B.3 Query di OUT

```
INSERT INTO [TIROCINI DWH].L2.OUT (
      execution_id
    , event_message_id
    , folder_name
    , project_name
    , package_name
    , status
    , execution_path
    , levelPackage
    , message_source_name
    , message_time
    , event_name
    , flg_OnError
    , Block_Message
    , Hash_error
    , start_time
    , end_time
    , TotalMinutes
)
select *,
       CAST(substring(Elapsed, 1, 2) as float) * 60
           + CAST(substring(Elapsed, 4, 2) as float)
           + CAST(substring(Elapsed, 7, 2) as float)/60
           + CAST(substring(Elapsed, 10, 3) as float) /60000
          as TotalMinutes
from(
    select execution_id
    , event_message_id
```

```
    , folder_name
    , project_name
    , package_name
    , status
    , execution_path
    , levelPackage
    , message_source_name
    , message_time
    , event_name
    , CASE WHEN event_name = 'OnError' THEN 1 ELSE 0 END
        AS FLG_ERROR
    , Block_Message
    , HASHBYTES('SHA2_256', Block_Message) Hash_error
    , cast(message_time + ' ' + case when event_name = 'OnPreExecute'
    then SUBSTRING(Block_Message, CHARINDEX(',', Block_Message) + 2, 8)
    else '00:00:00' end as datetime) as start_time
    , cast(message_time + ' ' + case when event_name = 'OnPostExecute'
    then SUBSTRING(Block_Message, CHARINDEX(',', Block_Message) + 2, 8)
    else '00:00:00' end as datetime) end_time
    , case when event_name = 'OnPostExecute'
    then SUBSTRING(Block_Message, CHARINDEX('time:', Block_Message) + 6, 12)
    else '00:00:00.000' end Elapsed
    from(
        SELECT e.execution_id
        , e.folder_name
        , e.project_name
        , e.package_name
        , status
        , LEN(em.package_path) - LEN(REPLACE(em.package_path, '\', ''))
            AS levelPackage
        , em.execution_path
        , CONVERT(VARCHAR(10), message_time, 120)    AS message_time
        , em.event_message_id
        , em.message_source_name
        , em.event_name
        , RIGHT(em.message, LEN(em.message)-CHARINDEX(':', em.message))
            AS Block_Message
        FROM [TIROCINI DWH].L1.ODS_EXECUTIONS e
        JOIN [TIROCINI DWH].L1.ODS_EVENT_MESSAGES em
        ON e.execution_id = em.operation_id
        WHERE event_name NOT IN ('OnInformation')
    ) a
) b
```

B.4 Query di creazione e popolamento della dimensione PACCHETTO

```
CREATE TABLE [TIROCINI DWH].L2.DIM_PACCHETTO (
    SK_PACCHETTO INT IDENTITY(1,1) NOT NULL,
    package_name VARCHAR(400) NOT NULL,
    project_name VARCHAR(400) NOT NULL,
    folder_name VARCHAR(400) NOT NULL,
    execution_path VARCHAR(4000) NOT NULL,
    level_package INT,
    message_source_name varchar(400)
    CONSTRAINT [PK_pacchetto] PRIMARY KEY CLUSTERED
    (package_name, project_name, folder_name, execution_path)
);


MERGE INTO [TIROCINI DWH].L2.DIM_PACCHETTO AS target
USING (select distinct package_name, project_name, folder_name,
        execution_path, levelPackage, message_source_name
        from [TIROCINI DWH].L2.OUT) AS source
ON target.package_name = source.package_name
AND target.project_name = source.project_name
AND target.folder_name = source.folder_name
AND target.execution_path = source.execution_path
WHEN MATCHED THEN
    UPDATE SET
    target.package_name = source.package_name,
    target.project_name = source.project_name,
    target.folder_name = source.folder_name,
    target.execution_path = source.execution_path,
    target.level_package = source.levelPackage,
    target.message_source_name = source.message_source_name
WHEN NOT MATCHED THEN
    INSERT (package_name, project_name, folder_name,
        execution_path, level_package, message_source_name)
    VALUES (source.package_name, source.project_name,
        source.folder_name, source.execution_path,
        source.levelPackage, source.message_source_name);
```

B.5 Query di MERGE per la tabella dei fatti

```
MERGE INTO [TIROCINI DWH].L2.FATTI_EVENTI_ESECUZIONI AS target
USING (
    SELECT
          O.execution_id
        , O.event_message_id
        , P.sk_pacchetto
        , S.sk_stato
        , C1.date_sk start_date
```

```
        , C2.date_sk end_date
        , ER.sk_error
        , EV.sk_evento
        , O.TotalMinutes
        , O.FLG_ONERROR
    FROM [TIROCINI DWH].L2.OUT AS O
    LEFT JOIN [TIROCINI DWH].L2.DIM_PACCHETTO AS P
        ON O.package_name = P.package_name
        AND O.project_name = P.project_name
        AND O.folder_name = P.folder_name
        AND O.execution_path = P.execution_path
    LEFT JOIN [TIROCINI DWH].L2.DIM_STATO AS S
        ON O.status = S.status
    LEFT JOIN [TIROCINI DWH].L2.DIM_CALENDARIO AS C1
        ON CAST(CAST(YEAR(O.start_time) AS VARCHAR(4)) +
        RIGHT('00' + CAST(MONTH(O.start_time) AS VARCHAR(2)), 2) +
        RIGHT('00' + CAST(DAY(O.start_time) AS VARCHAR(2)), 2) +
        RIGHT('00' + CAST(DATEPART(HOUR, O.start_time) AS VARCHAR(2)), 2)
        AS BIGINT) = C1.date_sk
    LEFT JOIN [TIROCINI DWH].L2.DIM_CALENDARIO AS C2
        ON CAST(CAST(YEAR(O.end_time) AS VARCHAR(4)) +
        RIGHT('00' + CAST(MONTH(O.end_time) AS VARCHAR(2)), 2) +
        RIGHT('00' + CAST(DAY(O.end_time) AS VARCHAR(2)), 2) +
        RIGHT('00' + CAST(DATEPART(HOUR, O.end_time) AS VARCHAR(2)), 2)
        AS BIGINT) = C2.date_sk
    LEFT JOIN [TIROCINI DWH].L2.DIM_ERRORE AS ER
        ON O.hash_error = ER.error_hash
    LEFT JOIN [TIROCINI DWH].L2.DIM_EVENTO AS EV
        ON O.EVENT_NAME = EV.EVENT_NAME
) AS source
ON target.execution_id = source.execution_id
AND target.event_message_id = source.event_message_id
WHEN MATCHED THEN
    UPDATE SET
        target.sk_pacchetto = source.sk_pacchetto,
        target.sk_stato = source.sk_stato,
        target.sk_tempo_start = source.date_sk,
        target.sk_errore = source.sk_error,
        target.sk_evento = source.sk_evento,
        target.total_minutes = source.TotalMinutes,
        target.flg_OnError = source.FLG_ONERROR
WHEN NOT MATCHED THEN
    INSERT (execution_id, event_message_id, sk_pacchetto, sk_stato,
        sk_tempo_start, sk_errore, sk_evento,
        total_minutes, flg_Onerror)
    VALUES (source.execution_id, source.event_message_id,
    source.sk_pacchetto, source.sk_stato, source.date_sk,
        source.sk_error, source.sk_evento, source.TotalMinutes,
        source.FLG_ONERROR);
```

Appendice C

Categorie e pattern di errori e warning

C.1 Categorie di errore

- **CONNESSIONE**
  - Failed to acquire connection
  - AcquireConnection method call failed
  - Communication link failure
  - Error has occurred while establishing a connection
  - Unable to complete login process
  - Verify connection object has been set
  - Connection string components
- **IMPORT FILE**
  - Cannot open the datafile
  - The file name was not valid
- **VALIDAZIONE**
  - Failed validation
  - Errors during task validation
- **PRE-EXECUTION**
  - Failed the pre-execute phase
- **OGGETTI MANCANTI/PERMESSI**
  - Does not exist
  - Permission was denied
  - User does not have required privileges

  - Variable cannot be found
  - Failed to locate
  - Not found
  - Check that the object exists
  - Cannot find the object
  - Il database non esiste
  - You do not have permissions
  - 'NT SERVICE\SQLSERVERAGENT' non ha accesso al database
  - Cannot access destination table
- **BUFFER/MEMORIA**
  - The attempt to add a row to the Data Flow task buffer failed
  - Memoria insufficiente nel pool di buffer
- **ERRORI SPECIFICATI ALTROVE**
  - There may be error messages posted before this with more information about the failure
- **CONVERSIONE/FORMATO**
  - Cannot insert duplicate key
  - Cannot insert the value NULL
  - Object cannot be cast
  - Not in a correct format
  - Invalid
  - Not valid
  - Invalid character value for cast specification
  - Arithmetic overflow
  - Conversion failed
  - The value could not be converted
  - The value violated the integrity constraints
  - Text was truncated
  - A truncation error occurred
  - Change type during execution
- **SUCCESSIONE TRANSAZIONI**
  - Transaction was deadlocked
  - Cancelled because another operation failed
- **ESECUZIONE QUERY**
  - Query failed
  - Maximum recursion

- **TIMEOUT**
  - Execution Timeout Expired
  - The session is in the kill state
- **POSSIBILE ERRORE HARDWARE**
  - Logical consistency-based I/O error
- **MODELLO TABULARE**
  - OLE DB or ODBC error: The command has been canceled
- **ERRORE GENERICO**
  - Nessuna delle precedenti

C.2 Categorie di warning

- **DUPLICATI NEI DATI**
  - Contains a duplicate value
- **DISALLINEAMENTO COLONNE ESTERNE**
  - Out of synchronization with the data source columns
  - Needs to be added to the external columns
  - Needs to be updated
  - Needs to be removed from the external columns
- **LIMITE MASSIMO DI ERRORI RAGGIUNTO**
  - SSIS Warning Code DTS_W_MAXIMUMERRORCOUNTREACHED
- **Colonne non utilizzate nel flusso di dati**
  - Unused output column
- **Espressioni o valori non validi**
  - The expression is empty
  - Provide a valid calculation expression
  - A single value was expected
- **Problemi di accesso ai dati**
  - Access is denied
  - The network path was not found