Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Gestionale

ICT e Data Analytics per il Management

A.a. 2024/2025

Sessione di Laurea Marzo/Aprile 2025

Automatizzazione e personalizzazione di modelli tabulari con Tabular Editor in Power BI

Relatrice

Tania Cerquitelli

Candidata

Valentina Nalesso

ABSTRACT

Il processo di modellazione dei dati è il principale strumento per la creazione di dashboard efficaci e intuitive. La produzione di un modello coerente e quanto più completo facilita la realizzazione di rappresentazioni grafiche leggibili e chiare, in modo da fornire informazioni determinanti per il decision-making di un'azienda. Si rende necessario un efficace processo di creazione del modello, in grado di minimizzare tempi e costi. Il seguente progetto, sviluppato per Mediamente Consulting Srl, si pone l'obiettivo di ottimizzare il processo di creazione e mantenimento di modelli di dati tramite l'utilizzo del tool Tabular Editor. Il metodo presentato si propone come il più generale possibile e non limitato ad uno specifico contesto aziendale: dall'individuazione delle linee comuni applicabili a molteplici settori, sono stati elaborati degli script in C# capaci di automatizzare operazioni fondamentali per la gestione di modelli tabulari (e.g., creazione di relazioni tra tabelle, formattazione di colonne, misure con DAX). In tal modo, l'analista si occupa quasi esclusivamente della personalizzazione del modello in accordo con le esigenze dei clienti. Il progetto di tesi applica il metodo proposto a un caso studio reale: da un database preesistente, si adoperano gli script di automazione per la creazione di uno star schema e si personalizza in base alle necessità del cliente. Infine, con l'utilizzo di Power BI, viene realizzata una dashboard interattiva e navigabile per una chiara esposizione delle informazioni.

INDICE

INDICE DELLE TABELLE

INDICE DELLE FIGURE

INTRODUZIONE

Contesto aziendale

Mediamente Consulting è una società di consulenza specializzata nella gestione e valorizzazione dei dati. L'azienda nasce con la missione di guidare l'innovazione delle imprese in ambito Business Analytics, migliorandone le performance e abilitando le organizzazioni alle nuove sfide del mercato e della concorrenza.

Fondata nel 2013 come startup presso l'incubatore I3P del Politecnico di Torino, Mediamente Consulting ha completato il processo di integrazione in Var Group nel 2022, divenendo parte integrante della Line of Business (LOB) Data Science e venendo automaticamente controllata dal gruppo SeSa, leader italiano nella distribuzione di soluzioni IT a valore per le imprese, attivo sul mercato telematico azionario di Borsa Italiana dal 2013. L'azienda offre consulenza a medie e grandi imprese in settori diversificati, tra cui Retail, Manufacturing e Fashion. Al fine di garantire gli obiettivi di cui sopra, Mediamente Consulting si compone di cinque Business Units specializzate:

- Data Management: i servizi offerti spaziano dalla consulenza su varie tecnologie Oracle, compresi i sistemi ingegnerizzati, all'analisi delle prestazioni e alla valutazione dell'architettura, fino alla gestione e all'aggiornamento sia dell'infrastruttura tecnologica "on premise" che in cloud;
- Data Integration & Data Governance: l'obiettivo è centralizzare e armonizzare tutte le fonti di dati aziendali, interne ed esterne, per rendere i sistemi più efficienti e trasformare le aziende in organizzazioni basate sui dati;
- Low Code: sviluppa applicazioni aziendali e moduli di data entry, sfruttando strumenti visivi e tecnologie come APEX per automatizzare processi e semplificare la gestione dei dati.
- Corporate Performance Management: questo settore mira ad affiancare le imprese nella definizione delle strategie aziendali, nella loro implementazione attraverso piani

d'azione e budget, nel monitoraggio dei risultati e nella simulazione di scenari futuri, facilitando la comunicazione con gli stakeholder;

- Business Intelligence & Data Visualization: trasforma i dati grezzi in informazioni strategiche, fornendo strumenti essenziali per prendere decisioni più informate e orientate al successo aziendale.

Obiettivo della tesi

Nel panorama attuale, l'analisi dei dati e la loro visualizzazione rappresentano elementi chiave per il processo decisionale aziendale. L'adozione di strumenti avanzati per la gestione dei dati consente di ottimizzare le operazioni e migliorare la qualità delle informazioni a disposizione del management. Power BI si distingue come una delle soluzioni più utilizzate per la creazione di dashboard interattive e per la gestione di modelli tabulari. In questo contesto, Tabular Editor emerge come uno strumento fondamentale per automatizzare e personalizzare la modellazione dei dati in Power BI, riducendo i tempi e i costi di sviluppo.

Questa tesi si propone di analizzare l'automazione e la personalizzazione di modelli tabulari utilizzando Tabular Editor in Power BI. L'obiettivo è sviluppare un metodo che ottimizzi il processo di creazione e mantenimento dei modelli tabulari, riducendo le attività manuali e migliorando la flessibilità delle analisi. A tal fine, verranno sviluppati script in C# per automatizzare operazioni chiave, applicando il metodo a un caso studio reale per valutarne l'efficacia.

Il primo capitolo introduce i concetti fondamentali dell'analisi dei dati, illustrando il ruolo del Data Warehousing, il confronto tra OLTP e OLAP, le architetture dei Data Warehouse e il processo ETL. Vengono inoltre trattate le principali tecniche di Data Visualization e il loro impatto sul decision-making aziendale.

Il secondo capitolo approfondisce il concetto di modellazione dei dati, distinguendo tra le diverse tipologie di modelli: relazionale, multidimensionale e non relazionale. Vengono descritti gli schemi di modellazione più comuni, come lo Star Schema e lo Snowflake Schema, evidenziando vantaggi e limitazioni.

Il terzo capitolo è dedicato a Power BI, illustrandone le funzionalità principali: importazione e trasformazione dei dati, modellazione, creazione di dashboard e pubblicazione dei report. Viene inoltre discusso l'utilizzo di Power Query per la preparazione dei dati e le diverse modalità di connessione ai dati disponibili.

Nel quarto capitolo viene introdotto Tabular Editor, evidenziandone le caratteristiche principali e le funzionalità di automazione. Si analizzano le possibilità offerte dagli script in C# per la gestione delle relazioni tra tabelle, la formattazione delle colonne e la creazione di misure con DAX, mostrando come questi strumenti possano migliorare l'efficienza nella modellazione tabulare.

L'ultimo capitolo applica il metodo proposto a un caso studio reale. A partire da un database preesistente, si utilizzano gli script sviluppati per automatizzare la creazione dello schema tabulare e personalizzarlo in base alle esigenze del cliente. Infine, viene creata una dashboard interattiva in Power BI per una visualizzazione efficace delle informazioni.

La tesi si conclude con una riflessione sui benefici dell'adozione di un approccio automatizzato alla modellazione dei dati in Power BI, discutendo le potenziali applicazioni future e le possibili evoluzioni della metodologia proposta.

Capitolo 1

Data Analysis

La Data Analysis, o Analisi Dati, è una disciplina che mira a trasformare grandi quantità di dati non elaborati in informazioni significative per orientare i processi decisionali e la strategia delle imprese. Questo processo è reso possibile grazie all'utilizzo di metodologie avanzate, strumenti software e algoritmi analitici, che attraverso operazioni di estrazione, trasformazione e centralizzazione ricavano informazioni dai dati.

L'importanza dell'analisi dei dati giace nella sua capacità di identificare pattern nascosti, predire future tendenze e soluzioni a situazioni complesse. A livello aziendale, questo implica un miglioramento delle risorse utilizzate, un'ottimizzazione dell'efficienza operativa e una personalizzazione dei servizi per i clienti.

Il ciclo di vita dell'analisi dei dati include diverse fasi chiave:

1. Definizione del problema: lo scopo è delineare un quesito aziendale correlato agli obiettivi dell'organizzazione;
2. Raccolta dei dati: i dati vengono estratti da fonti eterogenee, come database aziendali, sensori IoT, social media o transazioni finanziarie;
3. Pulizia e preparazione: vengono eliminati errori, dati duplicati e informazioni irrilevanti per garantire l'accuratezza e l'affidabilità dell'analisi;
4. Analisi ed elaborazione: i dati vengono analizzati attraverso tecniche statistiche, modelli predittivi o machine learning;
5. Visualizzazione e interpretazione: i risultati vengono rappresentati attraverso grafici e report per facilitare la comprensione e il processo decisionale.

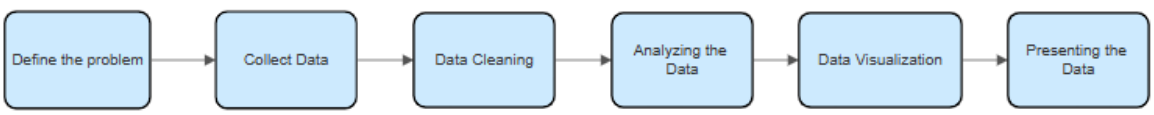

Figura 1. Processo di analisi dati

1.1 Definizione di Data Analysis

L'analisi dati ha come obiettivo principale l'individuazione di schemi e pattern all'interno dei dati, che si traducono nell'identificazione di relazioni tra le variabili e nella rilevazione di somiglianze tra le diverse unità [1].

L'analisi dei dati si classifica in quattro categorie principali:

- Analisi descrittiva: si occupa di sintetizzare ciò che è accaduto, fornendo statistiche riassuntive, come medie, mediane e frequenze;
- Analisi diagnostica: esplora le cause degli eventi osservati, utilizzando tecniche come la correlazione e la regressione per comprendere le relazioni tra variabili;
- Analisi predittiva: prevede risultati futuri basandosi su modelli matematici e machine learning. Esempi includono previsioni delle vendite e analisi dei rischi;
- Analisi prescrittiva: suggerisce azioni ottimali per raggiungere obiettivi specifici, combinando tecnologie avanzate come l'intelligenza artificiale e l'ottimizzazione operativa;

Questa classificazione permette di affrontare problemi complessi da prospettive differenti, rendendo l'analisi dei dati uno strumento essenziale in numerosi settori, tra cui sanità, finanza, produzione e marketing.

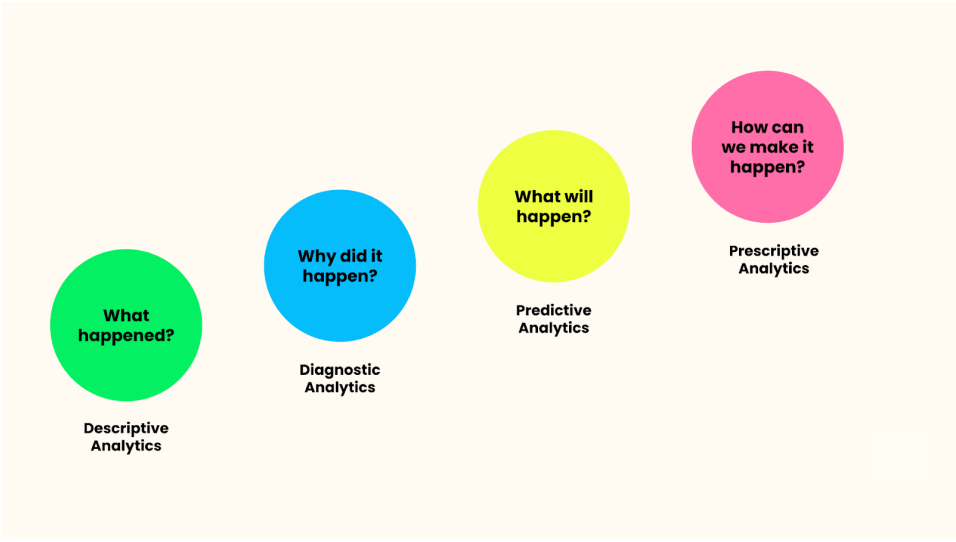

Figura 2. Tipologie di Data Analysis
(Fonte: DataCamp, "The impact of machine learning across verticals and teams." Available: https://www.datacamp.com/blog/the-impact-of-machine-learning-across-verticals-and-teams)

1.2 Introduzione al Data Warehousing

Un Data Warehouse (DWH) è una piattaforma centrale progettata per raccogliere e conservare dati provenienti da diverse fonti operative in un formato omogeneo e accessibile. La sua architettura è orientata a supportare attività di Business Intelligence (BI), eseguendo query e analisi che spesso esaminano grandi quantità di dati storici. Queste capacità analitiche consentono alle organizzazioni di ricavare importanti insight sul business dai loro dati per migliorare il processo decisionale [2]. La presenza di un Data Warehouse distinto dai database operazionali è essenziale per raggiungere tali obiettivi.

I database transazionali, utilizzati a livello operativo, posseggono caratteristiche che non rendono la loro prestazione adatta a scopi di business intelligence e decision-making. Le informazioni di tali database, infatti, derivano da una varietà di fonti dei dati diverse, impostate su formati e strutture differenti, impedendo un'analisi complessiva coerente e affidabile. Per garantire queste proprietà, occorre effettuare un processo di estrazione e integrazione dei dati, riconciliandoli in una struttura complessa e centralizzata. Inoltre, i database operazionali contengono informazioni poco significative per la BI, considerabili superflue e non adatte per fornire indicazioni su possibili azioni tattico-strategiche aziendali. Per ultimo, i dati contenuti nelle sorgenti dati hanno un livello di granularità molto elevato, con un livello di dettaglio non utile alle analisi di BI, sovraccaricando considerevolmente il volume dei dati e i tempi di elaborazione. Le attività di BI, infatti, sono ottimizzate per dati aggregati e sintetizzati, al fine di ottenere una visione più complessiva dell'andamento dell'azienda.

Le caratteristiche principali di un Data Warehouse includono [3]:

- Orientamento ai soggetti: distingue il Data Warehouse dai sistemi transazionali tradizionali, poiché organizza i dati in base ai principali processi aziendali, come vendite, marketing e produzione, invece di concentrarsi sulle operazioni quotidiane. Questo approccio tematico consente di ottenere informazioni mirate su argomenti specifici, facilitando l'analisi strategica;
- Integrazione: i dati provenienti da diverse fonti o database vengono armonizzati per garantire coerenza e uniformità. Ciò significa che il Data Warehouse segue uno schema globale dei dati, una naming convention standardizzata e un sistema unificato di unità

di misura e codifiche. Questo processo assicura che le informazioni abbiano un valore aziendale complessivo, piuttosto che essere frammentate a livello dipartimentale;

- Non volatilità: i dati, una volta caricati, non vengano modificati o cancellati. A differenza dei database operazionali, in cui le informazioni vengono aggiornate dinamicamente, nel Data Warehouse i dati vengono storicizzati e aggiornati solo a intervalli prestabiliti. Di conseguenza, le uniche operazioni eseguite sui dati sono l'accesso per l'analisi e il caricamento di nuovi dati, senza alterare quelli già esistenti;
- Temporalità: ogni record conserva un riferimento temporale, permettendo di analizzare l'evoluzione delle informazioni nel tempo. Questo aspetto è essenziale per condurre analisi storiche e identificare tendenze e pattern basati su dati consolidati.

Tra i principali vantaggi del data warehousing vi è la possibilità di eseguire query veloci e dettagliate, supportare la creazione di report complessi e favorire l'adozione di strumenti di Business Intelligence. Adottare un Data Warehouse, quindi, permette di raggiungere quel vantaggio competitivo ottenibile attraverso sistemi per il supporto decisionale (DSS) in grado di estrarre informazioni utili sul piano strategico a partire da un'elevata quantità di dati, in linea con l'attività modellata dai decision maker [2].

1.2.1 OLTP vs OLAP

Per interrogare i dati il Data Warehouse utilizza analisi OLAP (Online Analytical Processing), a differenza dei database operazionali che si affidano ad analisi OLTP (Online Transaction Processing). La necessità di utilizzo dei database OLAP risiede nella ricerca della qualità dei dati da analizzare, nell'integrazione dei dati da diverse fonti, nell'efficienza e rapidità delle analisi e nell'estensione temporale dei dati, che devono avere sufficiente profondità storica. Le differenze tra i due sistemi possono essere riassunte nella seguente tabella:

Tabella 1. Confronto OLTP - OLAP

	OLTP	**OLAP**
Funzione	Operativa, esecuzione di transazioni	Analitica, supporto alle decisioni
Progettazione	Orientata alle applicazioni	Orientata ai soggetti
Frequenza di aggiornamento	Molto frequente, aggiornamenti in tempo reale o quasi	Meno frequente, aggiornamenti periodici (ETL o ELT)
Dati	Recenti, dettagliati	Storici, aggregati, multidimensionali
Sorgente	Singolo DB	DB multipli
Uso	Guidato, processi ripetitivi	Interrogazioni ad hoc
Accesso	Read/Write	Read
Flessibilità accesso	Uso di programmi precompilati	Generatori di query
Tipo utenti	Operatori, applicazioni operative	Manager, analisti, data scientist
Tempo di risposta	Breve, ottimizzato per la velocità	Maggiore, accettabile per query analitiche più lunghe
Volume dei dati	Relativamente basso, focalizzato sul tempo presente	Elevato, include dati storici per analisi temporali

1.2.2 Architettura del Data Warehouse

L'architettura di un Data Warehouse è un framework strategico concepito per centralizzare, organizzare e rendere disponibili grandi volumi di dati provenienti da fonti eterogenee. Essa rappresenta la base tecnologica per l'analisi dei dati e il supporto alle decisioni aziendali, consentendo l'integrazione e la trasformazione dei dati operativi in informazioni analitiche e strategiche. La progettazione di un Data Warehouse è guidata da obiettivi chiave come la scalabilità, l'affidabilità e la capacità di fornire prestazioni elevate anche in presenza di query complesse e dataset estesi. Un Data Warehouse deve garantire la capacità di integrare dati da molteplici sistemi, come database transazionali (OLTP), file esterni, applicazioni cloud e sensori IoT, e trasformarli in un formato coerente e strutturato per analisi dettagliate o riassuntive. Questa struttura consente agli utenti finali, come analisti e manager, di accedere ai dati in modo intuitivo attraverso strumenti di Business Intelligence (BI) e piattaforme analitiche. In base al grado di complessità e alle esigenze specifiche, esistono tre principali tipologie di architettura: a un livello, a due livelli e a tre livelli.

Architettura a un livello

L'architettura a un livello, la più semplice tra tutte, consiste in un unico repository centrale in cui i dati vengono direttamente caricati dalle sorgenti operative senza un processo intermedio di staging o trasformazione. Tale DWH è virtuale, nel senso che viene implementato come una vista multidimensionale dei dati operazionali generata da un apposito “middleware”, ossia da uno strato d’elaborazione intermedio [4].

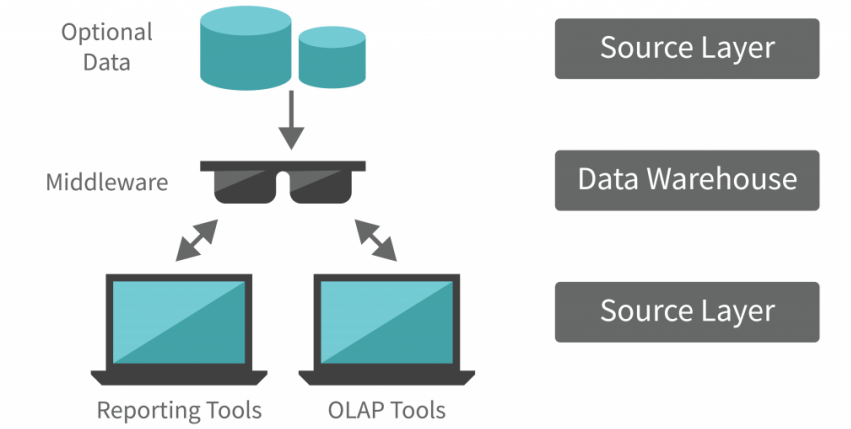

Figura 3. Architettura ad un livello
(Fonte: InterviewBit. “Data warehouse architecture.”
Available: https://www.interviewbit.com/blog/data-warehouse-architecture/)

Tale approccio è tipico di piccole organizzazioni con esigenze analitiche limitate, dove le fonti di dati sono relativamente omogenee e i requisiti di integrazione e pulizia sono minimi. I vantaggi di questa architettura includono semplicità di implementazione e costi operativi ridotti, data la minimizzazione dei dati memorizzati grazie all'eliminazione delle ridondanze. Tuttavia, la mancanza di una fase di trasformazione intermedia limita la qualità e la consistenza dei dati, rendendola inadatta per analisi complesse o contesti con molteplici fonti di dati eterogenee. Inoltre, questa architettura non rispetta il requisito definito da Kimball di separazione tra l'elaborazione analitica OLAP e quella transazionale OLTP [5]. Le interrogazioni di analisi, venendo ridirette sui dati operazionali dopo essere state reinterpretate nel middleware, possono così interferire con il normale carico di lavoro transazionale.

Architettura a due livelli

L'architettura a due livelli si compone di due insiemi di dati: il livello sorgente, contenente l'insieme dei dati operazionali dell'azienda o esterni all'azienda, e il livello del Data Warehouse. In questo caso, a differenza dell'architettura ad un livello, il Data Warehouse è fisicizzato. La ridondanza di dati derivante dall'esistenza dei due livelli permette di rispettare il requisito di separabilità tra operazioni d'analisi e transazionali, fondamentale del data warehousing. Il Data Warehouse è alimentato tramite strumenti di ETL (Extraction-Trasformation-Loading), che hanno il compito di estrarre e pulire i dati dal livello sorgente. I dati possono venire memorizzati all'interno di un unico Data Warehouse, denominato Data Warehouse primario o aziendale, oppure possono venire raggruppati nei cosiddetti Data Mart. Un Data Mart è un sottoinsieme dipartimentale dei dati contenenti l'insieme delle informazioni rilevanti per una particolare area di business, una particolare divisione dell'azienda, una particolare categoria di soggetti [6]. Esso può venire alimentato dal Data Warehouse primario o direttamente dalle sorgenti in modo indipendente. Il livello d'analisi, infine, include l'insieme degli strumenti che permettono di effettuare operazioni di reportistica, OLAP e data mining.

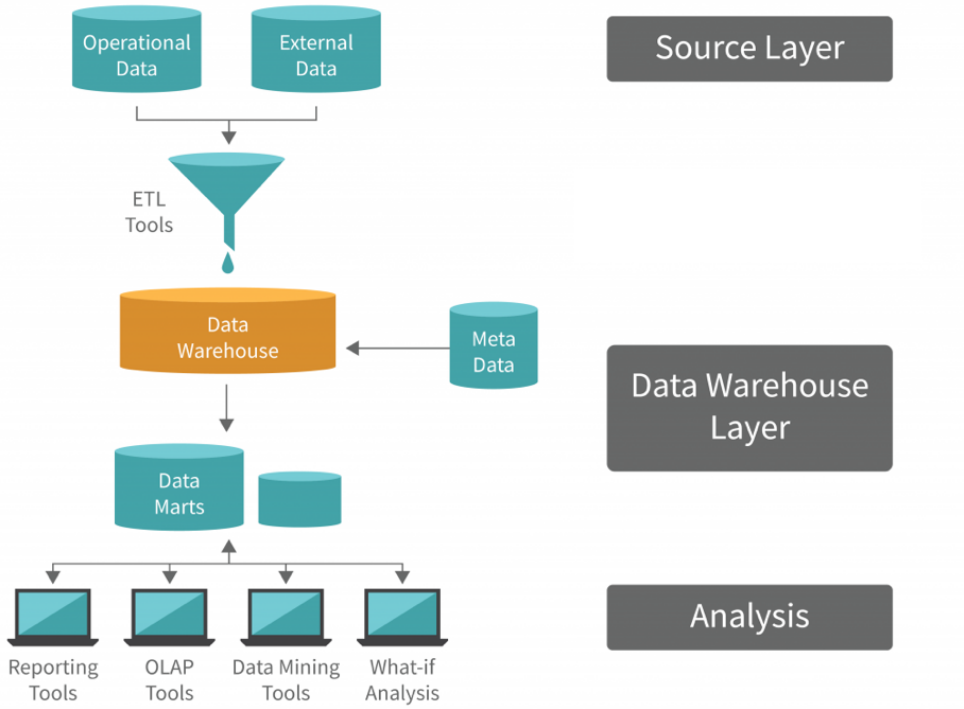

Figura 4. Architettura a due livelli

(Fonte: InterviewBit. "Data warehouse architecture." Available: https://www.interviewbit.com/blog/data-warehouse-architecture/)

Questo approccio è spesso utilizzato in contesti moderni che fanno uso di cloud Data Warehouse, dove gli strumenti di caricamento e trasformazione (ETL o ELT) sono integrati nella piattaforma. Tuttavia, l'assenza di un livello intermedio può limitare la capacità di gestire trasformazioni complesse o di eseguire controlli di qualità approfonditi sui dati.

Architettura a tre livelli

L'architettura a tre livelli è la struttura più completa e articolata, adottata in ambienti complessi che richiedono requisiti analitici avanzati. A differenza dell'architettura a due livelli, in questa situazione i dati provenienti dalle sorgenti vengono riconciliati a livello della staging area. In questo modo i dati, prima di essere caricati all'interno del Data Warehouse, subiscono trasformazioni avanzate (pulizia, integrazione, arricchimento). Questa fase migliora la qualità e l'affidabilità dei dati e consente una gestione efficiente delle trasformazioni complesse.

Una volta trasformati, i dati sono memorizzati in un Data Warehouse ottimizzato per l'analisi, dove vengono organizzati secondo schemi analitici (es. schema a stella o a fiocco di neve) e resi accessibili agli utenti finali tramite strumenti BI e piattaforme analitiche avanzate.

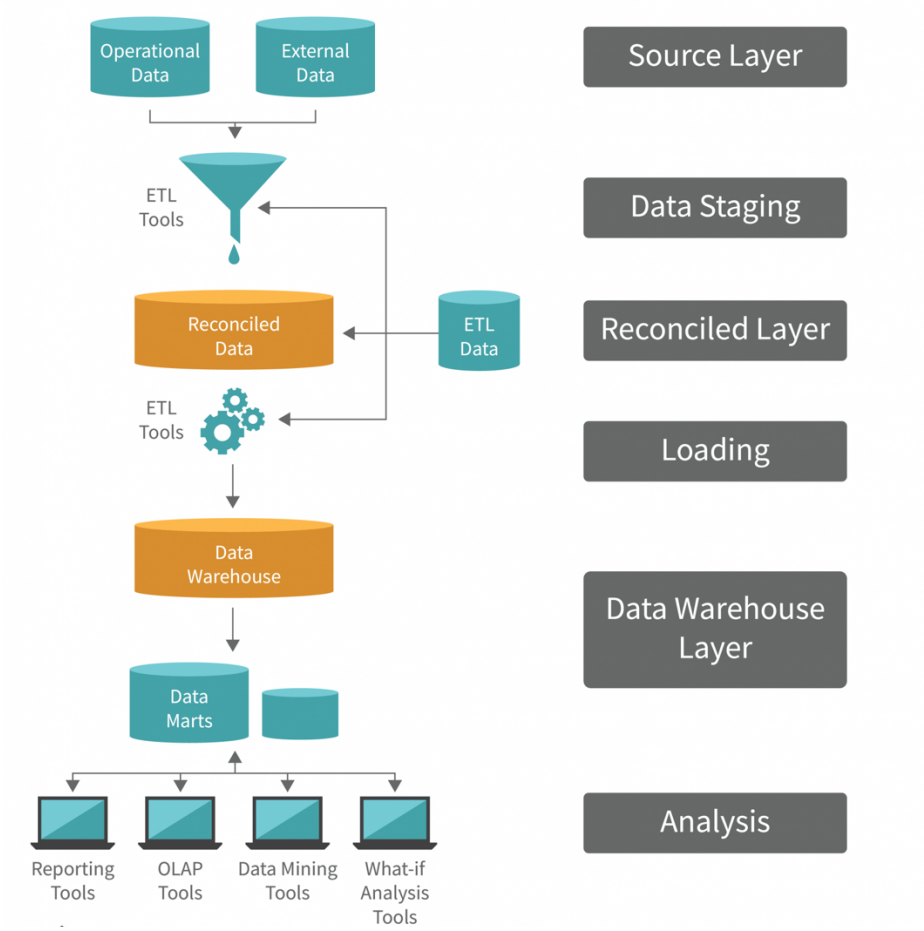

Figura 5. Architettura a tre livelli

(Fonte: InterviewBit. "Data warehouse architecture." Available: https://www.interviewbit.com/blog/data-warehouse-architecture/)

Questa architettura garantisce la massima flessibilità e scalabilità, rendendola ideale per analisi multidimensionali e scenari con grandi volumi di dati e fonti eterogenee. Tuttavia, è più complessa da implementare e richiede risorse maggiori in termini di tempo e costi.

1.2.3 Metadati

Per ottenere un accesso universale ai dati di Data Warehouse è necessaria l'esistenza di un repository di informazioni relative ai dati stessi: i metadati. I metadati sono informazioni che descrivono i dati stessi, relativamente alla struttura, alla creazione e al loro significato. Nel contesto del Data Warehousing, in cui giocano un ruolo sostanziale, essi indicano le sorgenti, il valore, l'uso e le funzioni dei dati memorizzati nel DWH e descrivono come i dati vengono alterati e trasformati durante il passaggio attraverso i diversi livelli dell'architettura [7]. Il repository dei metadati è strettamente collegato al DWH primario, risultando essenziale sia in fase di alimentazione che di analisi.

In un DWH si possono classificare due tipologie di metadati: metadati tecnici e metadati di business [2]. I metadati tecnici vengono impiegati da analisti e sviluppatori per la gestione e realizzazione del Data Warehouse e si suddividono in:

- Metadati infrastrutturali: sono creati e utilizzati nelle fasi di progettazione e sviluppo del Data Warehouse, in stretta relazione con il sistema di supporto alle decisioni. Comprendono informazioni come i nomi e le dimensioni di tabelle e colonne, la descrizione delle sorgenti dati, la documentazione delle trasformazioni e i dettagli sui formati dei dati.
- Metadati di controllo: controllano le operazioni in corso sui dati all'interno del Data Warehouse, includendo elementi quali i log delle attività, le regole per garantire la qualità del dato, le metriche di performance e la gestione delle versioni dei dati.

I metadati di business, invece, sono organizzati per aiutare gli utenti finali a interpretare correttamente i dati aziendali. Queste informazioni facilitano l'accesso delle risorse del Data Warehouse, fornendo descrizioni dettagliate su diversi aspetti dei dati e delle funzionalità del sistema decisionali. Tra questi rientrano le definizioni di termini e attributi, le regole aziendali che disciplinano i dati e il linguaggio utilizzato per la loro rappresentazione.

1.3 Processo ETL

Il ruolo degli strumenti di ETL è quello di alimentare una sorgente dati singola o multipla, dettagliata, esauriente e di alta qualità che possa a sua volta alimentare il Data Warehouse [3]. Le operazioni da essi svolte vengono spesso indicate con il termine *riconciliazione* e,

durante il processo di alimentazione del Data Warehouse, avvengono in due occasioni: quando il DWH viene popolato per la prima volta (Initial Load), e periodicamente quando il DWH viene aggiornato (Change Data Capture) [7]. La riconciliazione consiste di tre distinti processi detti rispettivamente:

1) Extraction o Capture;
2) Transformation;
3) Loading.

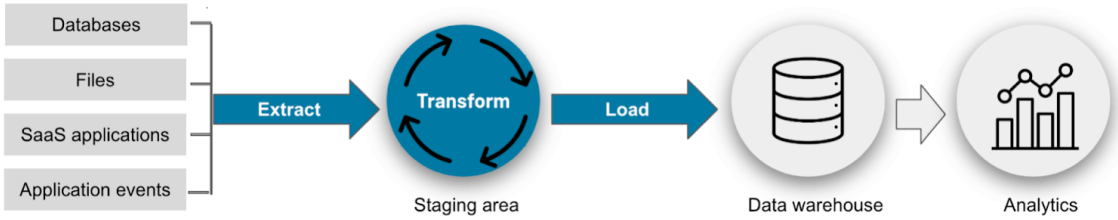

Figura 6. Processo ETL

(Fonte: Coupler.io. "What is ETL?"
Available: https://blog.coupler.io/what-is-etl/)

1.3.1 Extraction

L'integrazione dei dati a partire dalla sorgente inizia con la fase di estrazione, durante la quale vengono prelevate le informazioni rilevanti in base alle specifiche esigenze. Questa operazione può essere di tipo statico o incrementale.

L'estrazione statica (*Initial Load - IL*) viene effettuata quando il Data Warehouse deve essere popolato per la prima volta e consiste, concettualmente, in una fotografia completa dei dati operazionali in un determinato momento.

L'estrazione incrementale (*Change Data Capture - CDC*), invece, viene utilizzata per aggiornare periodicamente il Data Warehouse, catturando solo i dati modificati rispetto all'ultima estrazione. Tale approccio riduce il volume di dati elaborati, migliorando le prestazioni e l'efficienza del processo. Attraverso le tecniche di CDC, è possibile

identificare i cambiamenti direttamente a livello sorgente, ottimizzando la manutenzione del DWH e garantendo un aggiornamento più rapido e mirato.

L'estrazione dei dati dalle sorgenti può avvenire in modalità sincrona o asincrona. La modalità sincrona permette di prelevare i dati in tempo reale, registrando immediatamente eventuali aggiornamenti. La modalità asincrona, invece, prevede l'estrazione dei dati a intervalli regolari o pianificati, indipendentemente dal momento in cui sono stati modificati, consentendo un'elaborazione più strutturata e meno impattante sulle prestazioni del sistema sorgente.

1.3.2 Transformation

La fase di trasformazione, centrale nel processo di riconciliazione, implementa una sottofase di pulitura del dato. Durante questo processo viene migliorata la qualità dei dati delle sorgenti, andando ad eliminare dati "sporchi" dovuti a duplicazioni, inconsistenze, dati mancanti, valori errati etc.

La fase di trasformazione vera e propria, invece, ha l'obiettivo di convertire i dati dal formato operazionale sorgente a quello del DWH. Per l'alimentazione dei dati riconciliati vengono effettuate operazioni di conversione e normalizzazione al fine di uniformare i dati, operazioni di matching per stabilire corrispondenze tra campi equivalenti in sorgenti diverse, e operazioni di selezione per ridurre, se necessario, il numero di campi e record rispetto alle sorgenti.

Nella successiva alimentazione del DWH, la normalizzazione viene sostituita dalla denormalizzazione e si introduce l'aggregazione che realizza le opportune sintesi dei dati.

1.3.3 Loading

L'ultima fase del processo di ETL consiste nella strutturazione fisica e nel caricamento dei dati sul Data Warehouse. Ciò può avvenire secondo due modalità [8]:

- refresh: questa tecnica effettua la scrittura dei dati nel Data Warehouse, con la sostituzione integrale dei dati già presenti. In generale questo processo viene utilizzato

solo durante la fase iniziale di popolamento perché, pur garantendo un insieme di dati recenti e totale, risulta intensivo in termini di tempo e risorse;

- update: questa tecnica aggiorna in modo incrementale i dati del Data Warehouse solo con i cambiamenti avvenuti, senza sovrascrivere ad ogni iterazione tutti i dati. Tale tecnica viene solitamente utilizzata in abbinamento all'estrazione incrementale per l'aggiornamento periodico.

1.3.4 ELT

L'ELT (Extraction-Loading-Transformation) è un alternativo processo di integrazione dei dati. A differenze dell'ETL, nell'ELT, in seguito alla fase di estrazione da uno o più sistemi sorgenti, i dati estratti vengono prima caricati nel database di destinazione, per poi essere trasformati, ovvero convertiti dal formato di origine al formato richiesto per l'analisi [8]. La pulizia, la trasformazione e l'arricchimento dei dati avvengono, in questo caso, all'interno del Data Warehouse.

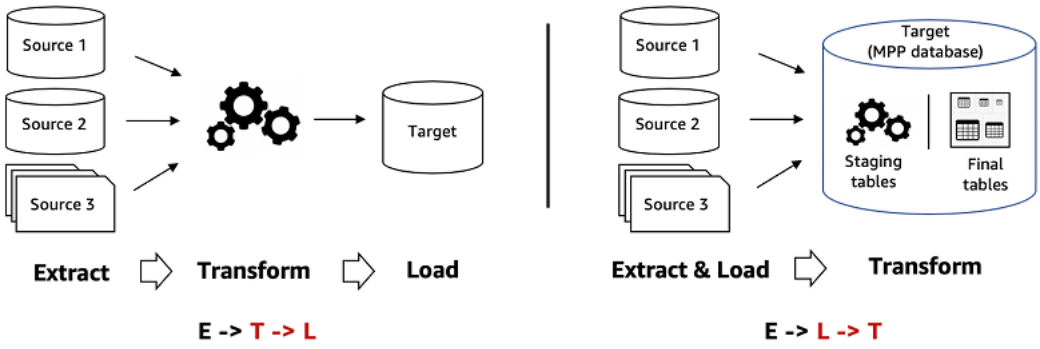

Figura 7. Confronto tra processi ETL e ELT

(Fonte: TecizEverything. "ETL vs. ELT Solutions."
Available: https://tecizeverything.com/technology/etl-elt-solutions/)

Generalmente, si considera ELT più utile per l'elaborazione di grandi insiemi di dati necessari per la Business Intelligence e l'analisi dei big data [9]. Infatti, l'evoluzione delle tecnologie cloud permette alle aziende di memorizzare un numero illimitato di dati non elaborati, per analizzarli in un secondo momento. Inoltre, tipi di dati come immagini, audio,

video, non sono archiviabili in formato tabellare. ETL viene considerato un approccio preferibile all'ELT quando è necessaria un'ampia pulizia dei dati prima di caricarli sul sistema di destinazione, quando sono necessari numerosi calcoli complessi su dati numerici e quando tutti i dati di origine provengono da sistemi relazionali.

1.4 Tecniche di Data Visualization

La Data Visualization è una disciplina che traduce dati e informazioni complesse in chiare rappresentazioni grafiche visive. Lo scopo principale della rappresentazione grafica dei dati è quello di fornire uno strumento per agevolare il lavoro di analisti, permettendo di identificare tendenze, modelli e anomalie nei dati, appoggiando in questo modo il processo decisionale e strategico [10].

Un aspetto cruciale della Data Visualization è la sua capacità di semplificare la comunicazione dei dati, rendono l'analisi molto più semplice ed intuitiva e permettendo una rapida individuazione delle informazioni utili. La maggior parte delle persone risponde molto meglio alle visualizzazioni piuttosto che al testo: il 90% delle informazioni inviate al cervello è visivo e il cervello elabora gli elementi visivi 60.000 volte più velocemente di quelli di testo [11]. Ad esempio, un grafico a linee può mostrare chiaramente l'andamento di un valore nel tempo, mentre una tabella di numeri potrebbe risultare opprimente e difficile da interpretare. In questo contesto, l'uso di colori, forme e layout strategici diventa fondamentale per migliorare la chiarezza e l'impatto del messaggio visivo.

La Data Visualization è un elemento fondamentale della Business Intelligence, dal momento che deve essere usata anche da coloro che non sono esperti in materia di analisi dati. Questo obiettivo è favorito dall'utilizzo di colori, dimensioni e immagini utili a comprendere l'andamento complessivo delle dimensioni in analisi. I punti cardine su cui si basa la Data Visualization sono:

- L'utente è il target: utilizzare un linguaggio comprensibile che tenga conto delle capacità e nozioni dell'utente;
- Evidenziare KPI: utilizzare pochi numeri che permettono di comprendere l'andamento complessivo dell'azienda;

- Gestire i confronti: è importante confrontare il valore di un KPI con il numero corrispondente del passato o con un valore target;
- Condividere il layout: rendere utile e comprensibile l'ambiente di analisi per l'utente finale;
- Certificare i dati: l'informazione deve essere sicura, consistente;

Gli strumenti di visualizzazione sul mercato, come Oracle BI, Power BI, IBM Cognos e Tableau, permettono di creare grafici interattivi, mappe e dashboard che sintetizzano informazioni complesse.

1.5 Business Intelligence

La Data Analysis, con i suoi processi di raccolta, pulitura, trasformazione e visualizzazione dei dati, rappresenta le fondamenta per un efficace sistema di Business Intelligence. La Business Intelligence, infatti, utilizza le informazioni derivanti dall'analisi dei dati contestualizzandole, per permettere alle organizzazioni di ottenere una visione d'insieme dei vari processi e ambienti aziendali, al fine di prendere decisioni data-driven in linea con la strategia della società. Attraverso dashboard interattive e report dettagliati, la BI fornisce una visione panoramica delle performance aziendali, consentendo ai decisori di monitorare i KPI e individuare tendenze emergenti. Tale approccio, permettendo di agire in modo informato rispetto a dati concreti e riducendo quindi il rischio di errori, consente di ottimizzare non solo i processi interni ma anche la gestione delle risorse. Inoltre, i tempestivi e aggiornati insight derivanti dalle analisi consentono di reagire rapidamente alle mutevoli esigenze del mercato, migliorando la competitività dell'azienda.

Capitolo 2

Modellazione dei dati

2.1 Introduzione alla modellazione dei dati

La modellazione dei dati rappresenta uno dei pilastri fondamentali per lo sviluppo di database, sistemi di gestione delle informazioni e soluzioni di Data Analysis e Business Intelligence (BI). Essa consiste nel processo di definizione e strutturazione dei dati di un'organizzazione in un formato strutturato, facilmente consultabile e analizzabile. Ciò permette una rappresentazione logica e coerente delle informazioni, garantendo al contempo un'alta qualità, consistenza e comprensione dei dati stessi, necessarie per analisi, reporting e supporto decisionale.

Nel contesto della BI, la modellazione dei dati si colloca al centro delle attività di analisi, poiché consente di tradurre le esigenze di business in una struttura tecnica efficace. Una modellazione ben progettata garantisce non solo l'efficienza del sistema, ma anche la sua scalabilità, mantenendo bassi i costi di gestione e agevolando l'adozione da parte degli utenti.

2.2 Tipologia di modellazione dei dati

La modellazione dei dati si articola in tre livelli principali: concettuale, logico e fisico, ognuno dei quali contribuisce a rappresentare i dati con un livello di dettaglio crescente.

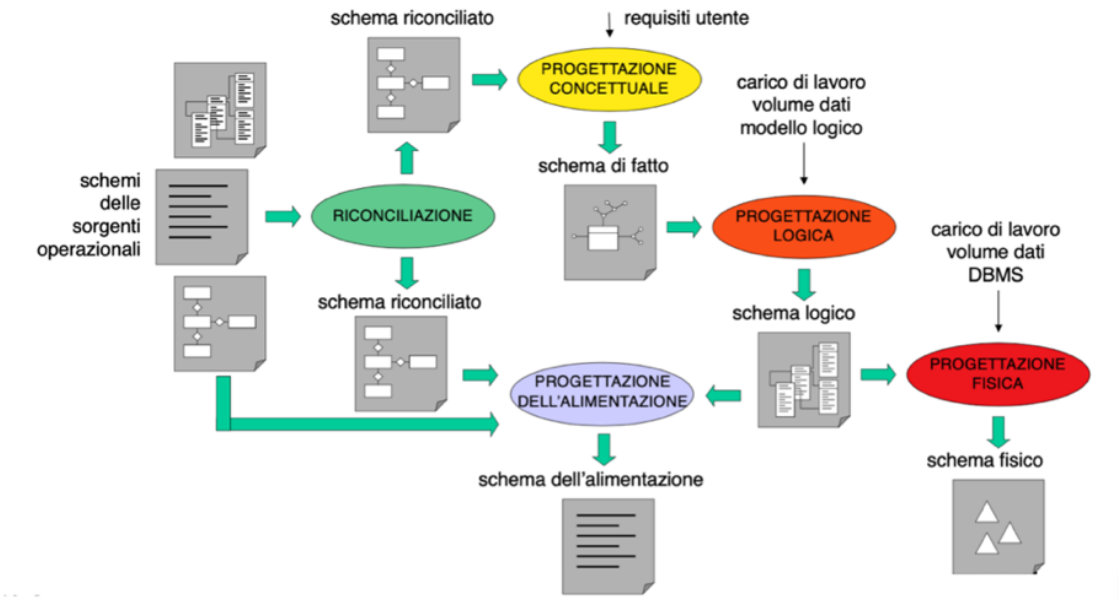

Figura 8. Progettazione di un Data Mart

La progettazione di un Data Mart comincia con la creazione dello schema dei dati riconciliato, provenienti dalle sorgenti informative operazionali. Essendo le fonti diverse, la riconciliazione è un passaggio necessario per ottenere una corretta visione d'insieme globale. Con la ricezione dei requisiti da parte dell'utente futuro utilizzatore si procede alla progettazione del **modello concettuale**, rappresentante la realtà di interesse in un modo formale e completo ma indipendente dal DBMS (Database Management System) utilizzato. Il modello di dati concettuale più popolare è il modello Entità-Associazione (ER: Entity-Relationship), usato prevalentemente per i database operazionali in quanto orientato a interrogazioni di navigazione più che di sintesi dei dati [5]. Per la modellazione concettuale di un Data Warehouse, invece, gli elementi di tale formalismo non riescono a rappresentare tutti i concetti d'interesse.

Il modello concettuale per la progettazione di un Data Mart è il Dimensional Fact Model (DFM), che prevede un insieme di schemi di fatto i cui elementi costituenti sono i fatti, le misure, le dimensioni e le gerarchie [12].

Tabella 2. Elementi del Dimensional Fact Model (DFM)

Fatto	Concetto di interesse per il processo decisionale; tipicamente modella un insieme di eventi che accadono nell'impresa
Misura	Proprietà numerica di un fatto che ne descrive un aspetto quantitativo di interesse per l'analisi
Dimensione	Proprietà con dominio finito di un fatto che ne descrive una coordinata di analisi.
Gerarchia	Albero direzionato i cui nodi sono attributi dimensionali e i cui archi modellano associazioni molti-a-uno tra coppie di attributi dimensionali. Essa racchiude una dimensione, posta alla radice dell'albero, e tutti gli attributi dimensionali che la descrivono.

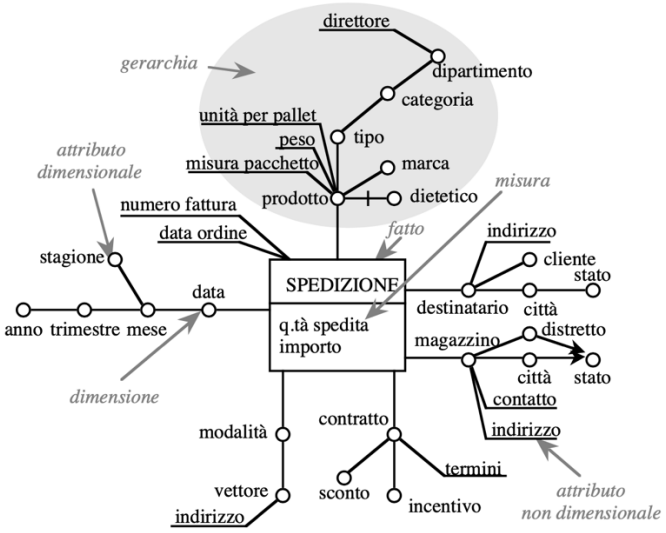

Figura 9. Esempio di schema DFM

(Fonte: M. Golfarelli e S. Rizzi, "Data Warehouse: teoria e pratica della progettazione", McGraw-Hill, 2006)

La progettazione logica rappresenta la fase intermedia che permette di definire, a partire dal modello concettuale, uno schema della base di dati nel modello adatto all'implementazione in un DBMS. A differenza del modello concettuale, nel **modello logico** i concetti d'interesse sono definiti con l'aggiunta delle informazioni sull'organizzazione dei dati. Nello specifico, si definiscono tabelle, colonne, tipi di dati e vincoli come chiavi primarie e chiavi esterne. Il prodotto ottenuto da questa fase è il cosiddetto schema logico, utile per semplificare ed ottimizzare le operazioni di interrogazione e manipolazione dei dati. Tuttavia, non viene ancora specificata l'implementazione fisica su uno specifico DBMS.

Il livello di dettaglio aggiuntivo raggiunto nel modello logico è il contesto di cui hanno bisogno gli architetti e progettisti per garantire la compatibilità delle nuove applicazioni con i dati coinvolti nel progetto. In poche parole, un modello dati logici fornisce le fondamenta necessarie per una progettazione produttiva dei database, garantendo la soddisfazione di tutti i requisiti, velocizzando il time-to-market e diminuendo i costi totali associati al processo di sviluppo [13].

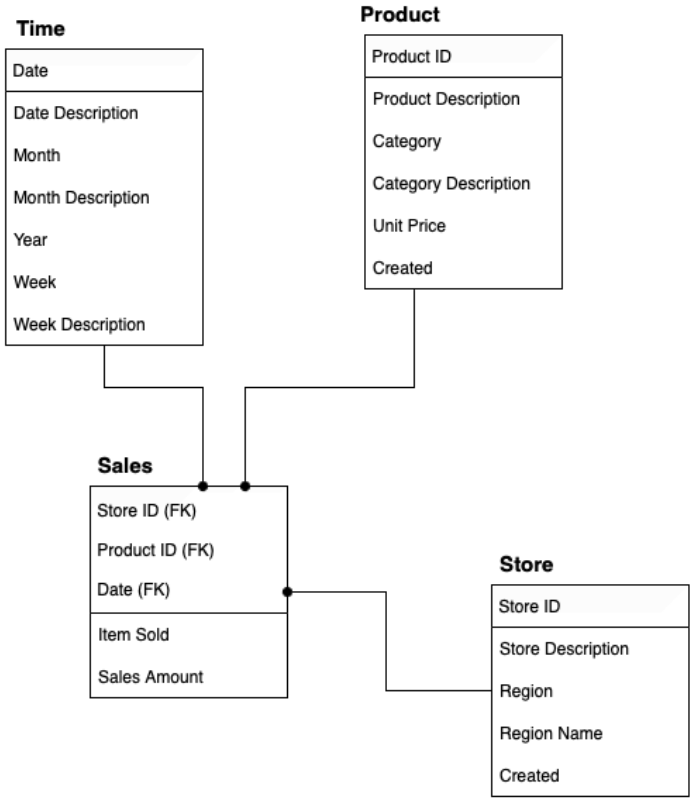

Figura 10. Esempio di schema logico

La modellazione di database fisica si occupa della progettazione del database effettivo in base ai requisiti raccolti durante la modellazione del database logico. Lo **schema fisico** deve contenere dettagli a sufficienza per consentire ai tecnici di creare l'effettiva struttura del database in hardware e software a supporto delle applicazioni che lo utilizzeranno: le tabelle e le colonne sono realizzate in base alle informazioni fornite durante la modellazione logica; chiavi primarie, chiavi univoche e chiavi esterne vengono definite per fornire vincoli. Questo modello include dettagli tecnici come l'organizzazione dello storage, gli indici, le partizioni e i meccanismi di ottimizzazione, mirando a massimizzare le prestazioni del database e garantendo che il sistema sia in grado di gestire carichi di lavoro complessi e query intensive. Ovviamente, il modello di dati fisico è specifico di un sistema software di database designato. Se si utilizzano diversi sistemi di database, possono esserci vari modelli fisici derivati da un singolo modello logico [14, 15] .

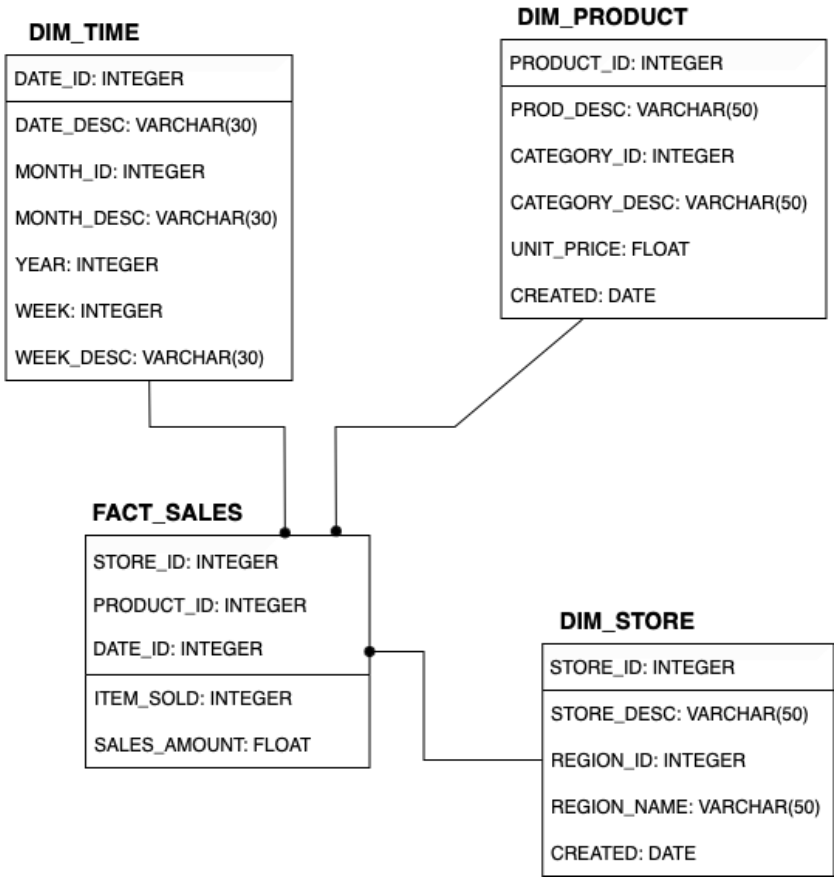

Figura 11. Esempio di schema fisico

2.3 Classificazione dei modelli dati

Il tipo di modello definisce la struttura logica, ossia il modo in cui i dati sono archiviati logicamente e, pertanto, la modalità di archiviazione, organizzazione e recupero. I sistemi di Data Warehousing possono presentare modelli dei dati differenti, ciascuno con specifici vantaggi e limiti:

- Relazionale
- Multidimensionale
- Non relazionale

2.3.1 Modello Relazionale

Il modello di dati relazionale, introdotto da Edgar F. Codd nel 1970 allo scopo di favorire l'indipendenza dei dati ed eliminarne la ridondanza, basa la gestione e l'organizzazione dei dati sulla teoria delle relazioni matematiche. In questo modello, i dati vengono archiviati in record di formato fisso, organizzati in tabelle (relazioni) composte da righe (tuple) e colonne (attributi) [16]. In tale contesto, si evidenziano due principali elementi tra i dati: le misure, identificate da valori numerici e utilizzate nei calcoli (somme o medie), e le dimensioni, rappresentanti informazioni qualitative. Sebbene un database relazionale sia definito da vari termini e requisiti strutturali, il fattore importante è rappresentato dalle relazioni definite all'interno di tale struttura. Ogni tabella rappresenta un'entità, o una relazione tra entità, e viene identificata da una **chiave primaria**, che garantisce l'univocità dei record. Le relazioni tra le tabelle sono gestite attraverso **chiavi esterne**, che collegano i dati senza bisogno di duplicazione, riducendo in questo modo la ridondanza dei dati. Le tabelle possono essere correlate anche in modo esplicito, per esempio con relazioni tra elementi sovraordinati e subordinati, tra cui relazione uno a uno, uno a molti o molti a molti [15].

Uno dei vantaggi principali apportati da un modello dati relazionale è la presenza di una struttura chiara e organizzata, che permette una facile comprensione del database. Inoltre, viene garantita integrità referenziale e consistenza dei dati, tramite chiavi primarie, che assicurano l'unicità dei record, e chiavi esterne, per mantenere la coerenza delle relazioni

tra le tabelle. Questo obiettivo è raggiunto anche tramite la conformità del modello alle transazioni ACID (Atomicità, Coerenza, Isolamento, Durabilità):

- Atomicità: ogni transazione è eseguita completamente o annullata;
- Coerenza: il database rimane in uno stato valido prima e dopo la transazione;
- Isolamento: le transazioni non interferiscono tra loro;
- Durabilità: i dati sono persistenti anche dopo guasti;

Ulteriormente, questo modello supporta potenti linguaggi di interrogazione come SQL (Structured Query Language), che permette di eseguire interrogazioni avanzate per recuperare, modificare e analizzare i dati in modo efficiente. Tuttavia, le operazioni su dati molto normalizzati, risultanti in molte tabelle interconnesse, possono risultare inefficienti in contesti di analisi complesse e Big Data. Ogni join tra tabelle, infatti, aggiunge un costo computazionale, rallentando la velocità delle query. I database relazionali non sono ottimizzati per elaborare rapidamente grandi aggregazioni di dati rispetto ai modelli denormalizzati [17].

I principali strumenti per gestire database relazionali sono MySQL, PostgreSQL e Oracle, che sono stati a lungo il punto di riferimento per la gestione dei dati aziendali grazie alla loro robustezza, flessibilità e compatibilità con il linguaggio SQL.

2.3.2 Modello Multidimensionale

Il modello multidimensionale è una tipologia di organizzazione dei dati utilizzato principalmente nei sistemi di Business Intelligence e nei Data Warehouse per facilitare l'analisi e il reporting. Il vantaggio che rende il modello multidimensionale molto adottato per rappresentare i dati nei DWH è legato tanto alla sua capacità di fornire informazioni aggregate ed essenziali in ambito strategico-tattico aziendale quanto alla sua semplicità ed intuitività nell'utilizzo. Per poter selezionare e raggruppare agevolmente gli eventi che accadono nell'azienda, si immagina di collocarli in uno spazio n-dimensionale i cui assi, chiamati dimensioni di analisi, definiscono varie prospettive per la loro identificazione. I dati vengono quindi rappresentati in una struttura a cubo (ipercubo se le dimensioni sono più di tre). All'interno di ogni "cubetto" sono memorizzati i valori delle misure, che rappresentano i dati numerici aggregati, e per identificarli basta fissare il valore di ogni

dimensione e tracciare, per ognuna di esse, le perpendicolari agli assi: il punto d'intersezione è l'evento da considerare [18].

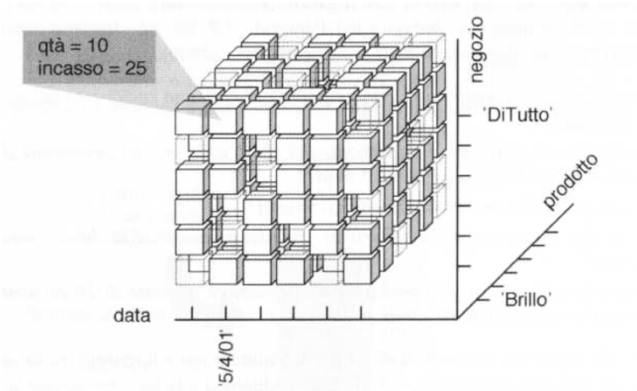

Figura 12. Cubo multidimensionale

(Fonte: G. Quattrone e D. Ursino, Appunti per il Corso di Data Warehousing, Università degli Studi Mediterranea di Reggio Calabria, 2007-2008)

Un aspetto fondamentale del modello multidimensionale è la possibilità di eseguire operazioni OLAP (Online Analytical Processing), quali drill-down, roll-up, slice e dice e pivoting, tecniche vantaggiose per ridurre la quantità di dati e che, andando ad interrogare direttamente il repository, rendono più semplici e rapide le query [19].

L'analisi dei dati tramite un cubo OLAP permette la visione di diversi livelli di sintesi delle informazioni. L'operazione di **drill-down** consente un passaggio da informazioni di riepilogo a dati con un focus più ristretto, andando ad aumentare la granularità dell'analisi [20].

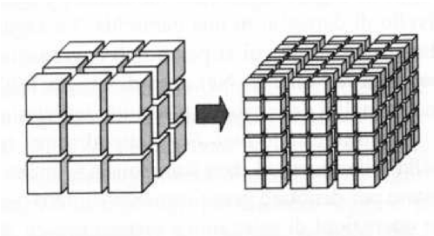

Figura 13. Operazione di drill-down

(Fonte: G. Quattrone e D. Ursino, Appunti per il Corso di Data Warehousing, Università degli Studi Mediterranea di Reggio Calabria, 2007-2008)

L'operazione di **roll-up** è duale al drill-down: esso aumenta l'aggregazione dei dati, eliminando un livello di dettaglio in una gerarchia. Il roll-up può anche portare alla diminuzione della dimensionalità del risultato, qualora tutti i dettagli di una gerarchia vengano eliminati [21].

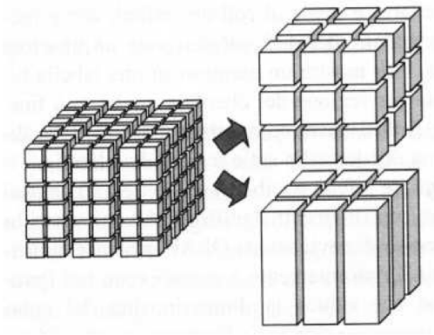

Figura 14. Operazione di roll-up

(Fonte: G. Quattrone e D. Ursino, Appunti per il Corso di Data Warehousing, Università degli Studi Mediterranea di Reggio Calabria, 2007-2008)

L'operazione di **slicing** riduce la dimensionalità del cubo, fissando un valore per una delle dimensioni, mentre la **selezione**, o filtraggio, è l'operazione che riduce l'insieme dei dati oggetto di analisi attraverso la formulazione di un criterio di selezione.

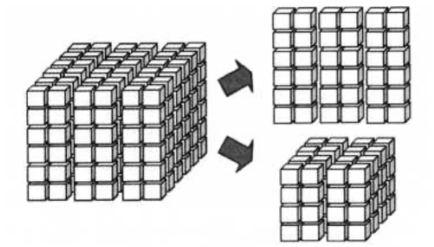

Figura 15. Operazioni di slice (sopra) e dice (sotto)

(Fonte: G. Quattrone e D. Ursino, Appunti per il Corso di Data Warehousing, Università degli Studi Mediterranea di Reggio Calabria, 2007-2008)

L'operazione di **pivoting** comporta un cambiamento nella modalità di presentazione con l'obiettivo di analizzare le stesse informazioni sotto diversi punti di vista. In altre parole,

effettuare il pivoting significa ruotare il cubo in modo da riorganizzare le celle secondo una nuova prospettiva, ovvero portando in primo piano una diversa combinazione di dimensioni.

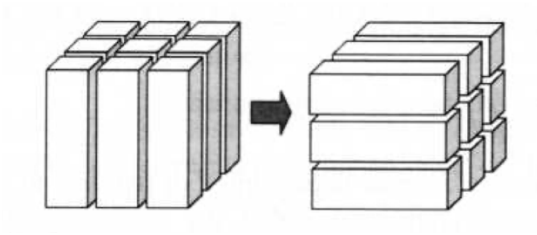

Figura 16. Operazione di pivoting

(Fonte: G. Quattrone e D. Ursino, Appunti per il Corso di Data Warehousing, Università degli Studi Mediterranea di Reggio Calabria, 2007-2008)

Architetture OLAP: ROLAP

I modelli logici che rappresentano la struttura multidimensionale dei dati sono: ROLAP, MOLAP e HOLAP.

Il ROLAP (Relational On-Line Analytical Processing) è un approccio all'analisi multidimensionale basato su database relazionali. In questo modello, i dati sono memorizzati in un DBMS relazionale, dove ogni riga di una tabella rappresenta un'istanza e include una colonna per ogni dimensione e una per ogni misura.

L'architettura ROLAP si basa su tre livelli principali:

1. Database relazionale: gestisce l'archiviazione e l'organizzazione dei dati;
2. Motore analitico OLAP: fornisce funzionalità analitiche e genera query SQL per interrogare il database;
3. Strumento di front-end: consente agli utenti di esplorare i dati e visualizzare i risultati delle analisi.

In seguito alla progettazione del Data Warehouse, i dati vengono estratti e caricati dai sistemi transazionali. Il motore OLAP, integrato nel livello di presentazione, traduce le richieste degli utenti in query SQL ottimizzate da eseguire sul database relazionale, abilitando analisi

multidimensionali. I risultati ottenuti sono successivamente elaborati e messi a disposizione per un'esplorazione interattiva e approfondita.

L'architettura ROLAP offre una gestione efficiente dei dati, potendo sfruttare sia informazioni precalcolate, se disponibili, sia il calcolo dinamico dei risultati da dati elementari. Inoltre, supporta diverse tecniche di ottimizzazione per migliorare le prestazioni delle query, tra cui: partizionamento delle tabelle, denormalizzazione per ridurre il numero di join, supporto multithreading per elaborazioni parallele, precomputazione dei dati aggregati, indici avanzati per velocizzare le ricerche.

Grazie a queste caratteristiche, il ROLAP è una soluzione scalabile e flessibile, adatta a gestire grandi volumi di dati e analisi complesse.

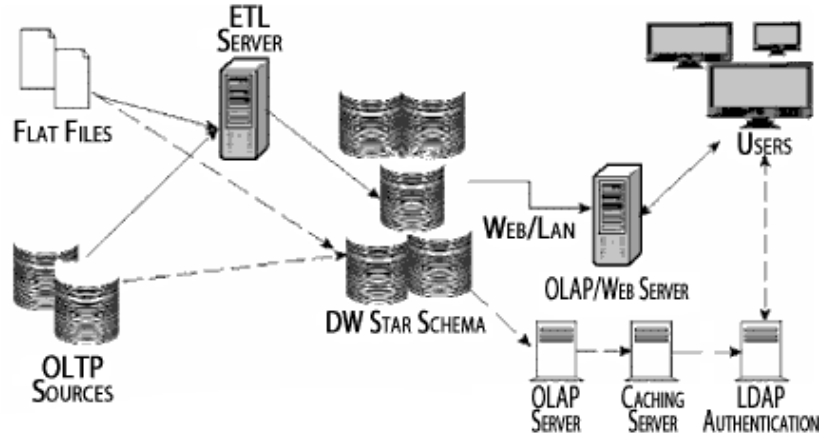

Figura 17. Architettura ROLAP

(Fonte: Drexel University. "ROLAP (Relational OLAP)"
Available: https://cci.drexel.edu/faculty/song/courses/info%20607/tutorial_OLAP/ROLAP.htm)

Architetture OLAP: MOLAP

Il MOLAP (Multidimensional On-Line Analytical Processing) è un approccio all'analisi multidimensionale basato su DBMS multidimensionali. Questo modello utilizza un'architettura a due livelli:

1. Database multidimensionale (MDB): gestisce la memorizzazione, l'accesso e il recupero dei dati;
2. Motore analitico OLAP: esegue le query OLAP e fornisce funzionalità analitiche;

3. Strumento di front-end: offre agli utenti un'interfaccia per visualizzare e analizzare i dati;

I dati provenienti dai sistemi transazionali vengono caricati nel sistema MOLAP. Una volta memorizzati nel BDMD, vengono eseguiti calcoli per ottenere aggregazioni multidimensionali, che popolano la struttura dati del database.

Dopo questa fase, il sistema ottimizza l'accesso alle query, generando indici e utilizzando algoritmi di hash table per ridurre i tempi di risposta. Una volta completato il popolamento del database multidimensionale, il sistema è pronto per l'uso: gli utenti possono eseguire analisi e report attraverso l'interfaccia, mentre la logica applicativa del MDB recupera i dati rapidamente. Un server MOLAP memorizza i dati su disco in strutture ottimizzate per l'accesso multidimensionale, spesso utilizzando array densi, che tipicamente richiedono quattro o otto byte per cella.

Una delle principali caratteristiche distintive del MOLAP è il preconsolidamento dei dati. In un database relazionale, rispondere a una query richiede la ricerca e l'aggregazione di tutti i record pertinenti. In un MDB, invece, le aggregazioni sono precalcolate e archiviate, consentendo risposte più rapide tramite operazioni su array.

Uno dei limiti principali del MOLAP è la gestione della sparsità. Questo fenomeno si verifica quando non tutte le combinazioni di elementi del modello multidimensionale contengono effettivamente dati [22]. Ad esempio, in una struttura a cubo, ogni evento è rappresentato da una cella, ma solo alcune celle contengono informazioni, ovvero quelle che corrispondono a eventi realmente accaduti. Per ottimizzare lo spazio di archiviazione, ogni strumento MOLAP utilizza meccanismi di compressione per evitare di salvare esplicitamente le celle vuote. Tuttavia, questa compressione introduce un costo computazionale aggiuntivo per la decompressione durante l'accesso ai dati. Per ottimizzare lo spazio di archiviazione, ogni strumento MOLAP utilizza meccanismi di compressione per evitare di salvare esplicitamente le celle vuote. Tuttavia, questa compressione introduce un costo computazionale aggiuntivo per la decompressione durante l'accesso ai dati.

Grazie alla sua elevata velocità di elaborazione delle query, il MOLAP è particolarmente adatto a scenari che richiedono analisi rapide e frequenti di dati aggregati. Tuttavia, la gestione della sparsità e la necessità di pre-elaborare i dati rappresentano delle sfide che possono influire sulla scalabilità del sistema.

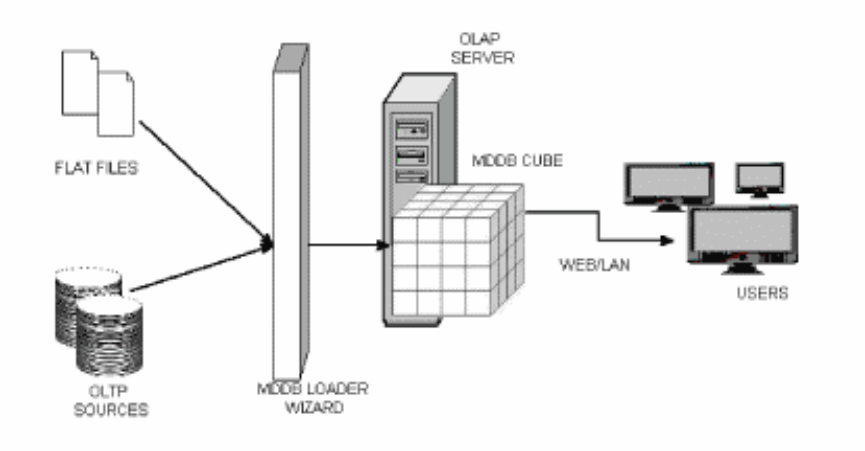

Figura 18. Architettura MOLAP

(Fonte: Drexel University. "ROLAP (Relational OLAP)"
Available: https://cci.drexel.edu/faculty/song/courses/info%20607/tutorial_OLAP/MOLAP.htm)

Architetture OLAP: HOLAP

Un sistema HOLAP (Hybrid On-Line Analytical Processing) combina i vantaggi di ROLAP e MOLAP, ottimizzando l'archiviazione e l'accesso ai dati. Per risolvere il problema della sparsità, i dati granulari (meno aggregati) vengono mantenuti nel database relazionale (DB relazionale), mentre gli aggregati sono memorizzati in un formato multidimensionale, riducendo così la presenza di celle vuote e migliorando le prestazioni delle query. Il precalcolo degli aggregati diventa essenziale quando il set di dati è molto esteso, poiché senza di esso alcune query potrebbero richiedere la scansione completa della tabella dei fatti, con un impatto significativo sulle prestazioni.

Un elemento fondamentale nella gestione delle aggregazioni è la cache, che memorizza gli aggregati precalcolati, permettendo di velocizzare l'accesso ai dati senza dover recuperare le informazioni direttamente dal disco. Se la cache contiene dati a un livello di aggregazione inferiore, può calcolare dinamicamente aggregati di livello superiore quando necessario. La cache è una parte cruciale della strategia di ottimizzazione in HOLAP, grazie alla sua natura adattiva. La scelta di quali aggregati precompilare è complessa: il sistema deve bilanciare velocità di accesso e spazio di archiviazione, soprattutto in presenza di molte dimensioni o query imprevedibili. Inoltre, nei sistemi in cui i dati cambiano frequentemente in tempo reale, mantenere aggregati precompilati potrebbe non essere pratico. Una cache ben dimensionata consente al sistema di rispondere in modo efficiente a

query imprevedibili, riducendo la dipendenza dagli aggregati precalcolati e migliorando la scalabilità complessiva del sistema.

Star Schema

I principali modelli multidimensionali utilizzati in contesti aziendali sono lo Star Schema e lo Snowflake Schema.

Lo Star Schema, o Schema a Stella, è un modello di organizzazione dei dati composto da due tipi di tabelle: tabelle dei fatti e tabelle delle dimensioni. Al centro del modello si trova la fact table (tabella dei fatti), che memorizza le informazioni sugli eventi di interesse. Attorno ad essa si dispongono le dimension tables (tabelle delle dimensioni), che forniscono dettagli descrittivi sui vari aspetti del contesto [23]. La fact table è collegata alle dimension tables tramite chiavi surrogate che fanno riferimento alle chiavi primarie delle tabelle dimensionali. Mentre le dimension tables archiviano informazioni qualitative, le fact tables conservano le misure numeriche che possono essere aggregate o analizzate per ottenere insight significativi [24]. Un elemento fondamentale nell'implementazione dello Star Schema è la cardinalità dei rapporti tra le tabelle. In questo modello, la relazione tra la fact table e ciascuna dimension tables è tipicamente di tipo molti-a-uno: una singola riga della fact table, che rappresenta un evento o una misurazione, è associata a una e una sola riga corrispondente nelle dimension tables. Questa configurazione garantisce che ogni misura numerica sia contestualizzata in maniera univoca da attributi descrittivi, facilitando il processo di aggregazione e analisi dei dati. L'assenza di relazioni dirette tra le tabelle dimensionali e il loro collegamento univoco alla fact table determina una denormalizzazione dei dati, semplificando le query e migliorando le performance di lettura. Tuttavia, questa struttura introduce ridondanza, che può aumentare lo spazio di archiviazione richiesto e, in alcuni casi, rendere meno efficienti le operazioni di aggiornamento dei dati.

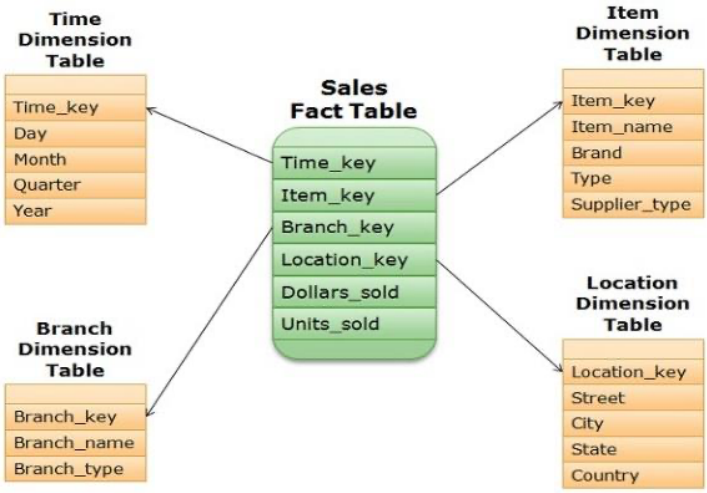

Figura 19. Star Schema

(Fonte: TechDifferences. "Difference between star and snowflake schema" Available: https://techdifferences.com/difference-between-star-and-snowflake-schema.html)

Snowflake Schema

Lo Snowflake Schema, o Schema a Fiocco di Neve, è un'estensione dello Star Schema, in cui le tabelle dimensionali vengono normalizzate per ridurre la ridondanza e ottimizzare lo spazio di archiviazione. In questo modello, le dimension tables possono essere suddivise in più sotto-tabelle collegate tra loro, formando una struttura gerarchica che ricorda un fiocco di neve [5]. Questa normalizzazione comporta la suddivisione delle informazioni in più livelli, eliminando dati ridondanti e migliorando la gestione degli aggiornamenti. La fact table rimane il nucleo centrale del modello, contenente le misure numeriche e le chiavi surrogate che la collegano alle varie dimensioni. Tuttavia, a differenza dello Star Schema, le dimension tables non si collegano direttamente alla fact table con un solo livello di relazione, ma possono essere ulteriormente suddivise in sotto-tabelle con chiavi esterne. La fact table rimane il nucleo centrale del modello, contenente le misure numeriche e le chiavi surrogate che la collegano alle varie dimensioni. Tuttavia, a differenza dello Star Schema, le dimension table non si collegano direttamente alla fact table con un solo livello di relazione, ma possono essere ulteriormente suddivise in sotto-tabelle con chiavi esterne.

Uno dei principali vantaggi dello Snowflake Schema è la maggiore efficienza nello spazio di archiviazione, grazie alla riduzione della duplicazione dei dati tramite la normalizzazione. Inoltre, agevola la manutenzione e l'aggiornamento delle dimensioni, evitando che le

modifiche si propaghino su più record ridondanti. Tuttavia, questa struttura introduce un numero maggiore di join nelle query, che possono ridurre le prestazioni di lettura rispetto allo Star Schema, soprattutto per analisi complesse su grandi volumi di dati. Per questo motivo, lo Snowflake Schema viene spesso adottato quando l'ottimizzazione dello spazio è una priorità o quando le dimensioni sono particolarmente ampie e soggette a frequenti aggiornamenti.

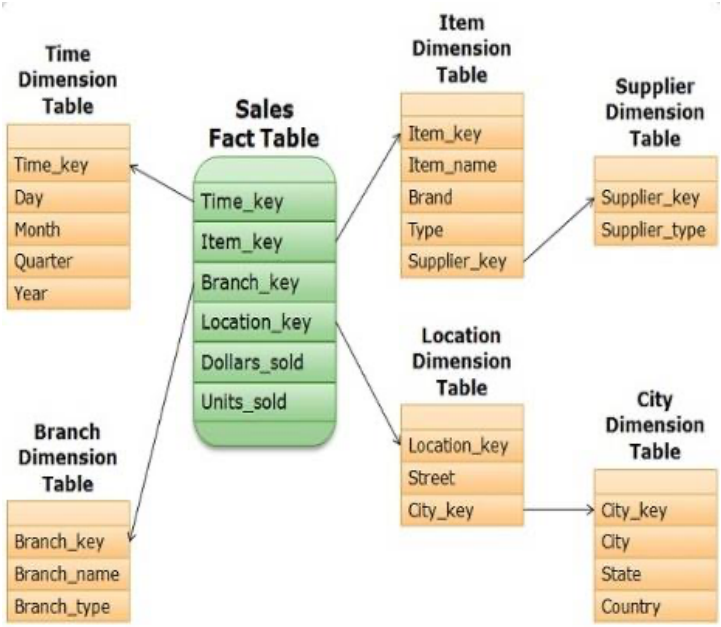

Figura 20. Snowflake Schema

(Fonte: TechDifferences. "Difference between star and snowflake schema" Available: https://techdifferences.com/difference-between-star-and-snowflake-schema.html)

La scelta tra Star Schema e Snowflake Schema dipende dal bilanciamento tra semplicità e performance. Lo Star Schema è spesso preferito per applicazioni BI con esigenze analitiche elevate, mentre lo Snowflake Schema è utile in contesti dove la ridondanza deve essere minimizzata.

2.3.3 Modello Non Relazionale

I sistemi di gestione di basi di dati relazionali sono stati fino ad oggi la tecnologia predominante per la memorizzazione di dati strutturati per varie applicazioni aziendali. Con l'avvento della rivoluzione tecnologica degli ultimi anni, tuttavia, la quantità di dati è

in aumento e risulta necessario scegliere un giusto modello di database in grado di gestire in modo efficiente questo enorme volume di dati. Il modello relazionale ha uno schema piuttosto rigido, dal momento che lo schema deve essere progettato in anticipo prima che i dati siano stati caricati e data l'uniformità degli attributi per tutti gli elementi [17]. Di conseguenza, se i requisiti di estrazione dei dati e della loro gestione è in continua evoluzione, uno schema così rigido può risultare limitante e diventare un ostacolo [25]. Questi problemi hanno portato allo sviluppo dei cosiddetti database non relazioni, conosciuti anche come database NoSQL (Not Only SQL), ottimizzati per i nuovi network e per accelerare il caricamento dei dati. I database NoSQL hanno una struttura di base diversa da quelli relazionali: non memorizzano i dati in tabelle ma in documenti, non richiedono uno schema definito prima dell'inserimento dei dati e non devono cambiare struttura in caso di modifiche di regole di gestione dei dati. Usano, invece, delle chiavi identificative, e i dati possono essere rintracciati tramite tali chiavi assegnate.

Oltre alla flessibilità, un rilevante vantaggio dei modelli non relazionali è l'alta scalabilità orizzontale, ossia l'abilità di distribuire i dati e il carico di semplici operazioni su molti server, senza RAM o dischi di archiviazione condivisi. Tale proprietà permette elevate prestazioni nelle operazioni di scrittura/lettura, con un elevata velocità di elaborazione delle interrogazioni, mantenendo l'efficacia di esecuzione di un gran numero di transazioni al secondo.

D'altra parte, però, i database non relazionali presentano anche degli svantaggi. Generalmente, i database non relazioni sacrificano le proprietà ACID a favore di quelle BASE (Basically available, Soft state, Eventually consistent); ciò potrebbe portare ad una compromissione della consistenza dei dati e della loro attendibilità. Inoltre, molti dei database NoSQL risultano essere open source, compromettendone l'affidabilità dal momento che nessuno è responsabili di eventuali errori nel tempo [26].

Capitolo 3

Power BI

Power BI, strumento sviluppato da Microsoft nel 2010, è una delle piattaforme leader di business intelligence disponibili sul mercato. Si presenta come soluzione alle organizzazioni per migliorare le proprie capacità di visualizzazione dei dati, trasformando i dati grezzi e non strutturati in preziosi insights. Grazie all'interfaccia intuitiva, alle dashboard interattive e alla perfetta integrazione con varie fonti di dati, Power BI consente di dare un senso a un insieme di dati complessi in modo rapido ed efficiente. Questa capacità di visualizzare i dati attraverso grafici, diagrammi e report dinamici offre alle aziende di tutti i settori una visione completa delle loro attività, portando a processi decisionali più informati. Inoltre, le funzionalità di collaborazione di Power BI consentono ai team dei vari reparti di condividere le intuizioni, assicurando che le strategie basate sui dati siano applicate in modo coerente in tutta l'organizzazione [27].

3.1 Funzionalità di Power BI

Power BI offre molteplici funzionalità per la creazione di soluzioni di Business Intelligence, la gestione dei dati e la costruzione di report, permettendo agli utenti di condividere e sfruttare al meglio le informazioni aziendali a livello professionale. Queste funzionalità sono raggruppate nei seguenti strumenti che, integrati, consentono l'esplorazione e l'analisi dei dati in modo interattivo:

- Power BI Desktop: applicazione installabile sul computer locale. È lo strumento principale per creazione dei modelli di dati e dei report, consentendo l'importazione dei dati da molteplici fonti, la trasformazione e la visualizzazione attraverso dashboard interattive [28].
- Power BI Service: servizio SaaS (Software as a Service), piattaforma cloud che consente agli utenti di caricare, condividere e collaborare su report e dashboard.

Permette di pubblicare i report creati tramite Power BI Desktop e di monitorare i dati in tempo reale grazie alla funzionalità di aggiornamento automatico dei dataset [29].

- Power BI Mobile: applicazione installabile su dispositivo mobile iOS e Android. Permette di visualizzare e interagire con dashboard e report creati attraverso Power BI Desktop, in locale o nel cloud [30].
- Power Query: strumenti per la trasformazione e preparazione dei dati. Power Query permette di connettere Power BI a fonti dati diverse e, tramite un editor, consente la pulizia e trasformazione dei dati prima di importarli nel modello [31].
- Power BI Report Server: server di report locale con un portale Web in cui visualizzare e gestire report e KPI, soluzione ideale per le organizzazioni che non vogliono o non possono archiviare i propri dati nel cloud [32].

3.1.1 Importazione e trasformazione dei dati

Tramite il servizio Power Query, Power BI offre la possibilità di connessione a una vasta gamma di origini dei dati, tra cui database privati, servizi cloud, file locali e online [33].

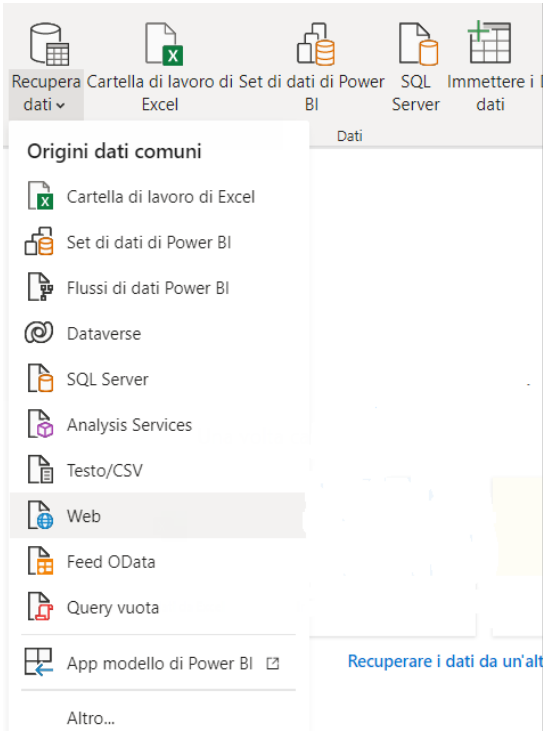

Figura 21. Importazione dati Power BI

Le origini dei dati sono circa 80, tra cui file CSV, fogli Excel, tabelle web HTML e varie tipologie di database. Inoltre, i report di Power BI possono attingere ad una cartella come origine dati per importare contemporaneamente più file dello stesso tipo. Questo può essere particolarmente utile quando si lavora con dati provenienti da sensori o sistemi di telecamere che possono generare un gran numero di file. Per gli utenti del cloud storage OneDrive di Microsoft, i file possono anche essere referenziati tramite link web per accedere ai dati archiviati nel cloud. Nella sezione di data preparation è possibile scegliere la tipologia di origine dei dati, al fine di ottimizzare i tempi di aggiornamento ed evitando, quando possibile, l'importazione dei dati statici.

In fase di importazione, l'utente è tenuto a scegliere tra le seguenti modalità di importazione: Import Mode, DirectQuery o Live Connection.

L'*Import Mode* è l'opzione più comune, prevede l'estrazione dei dati dalla sorgente e la conseguente archiviazione direttamente nella memoria di Power BI. Questa modalità offre velocità di esecuzione elevata grazie all'esecuzione di query in memoria, ma comporta anche spazio occupato sul disco locale, se si lavora da Power BI Desktop, o sul disco nel cloud di Power BI, se si lavora da portale web. Tuttavia, quando vengono aggiornati, i dati subiscono una compressione e ottimizzazione, con archiviazione su disco tramite il motore di archiviazione VertiPaq. La compressione può determinare una riduzione del 20% delle dimensioni di archiviazione su disco [34].

La modalità di importazione *DirectQuery* è una connessione diretta con l'origine dati. Con questo sistema i modelli sviluppati importano solamente i metadati delle tabelle e colonne, e non i dati stessi. Non essendoci *caching*, non viene mantenuto in memoria alcun dato: per questo motivo, ogni query eseguita dall'utente finale sul modello scatena un'interrogazione diretta all'origine dati. Con questa modalità di connessione, le dimensioni dei file sono molto più piccole. Tuttavia, questa tipologia, non supportata da tutte le origini dei dati, potrebbe comportare prestazioni delle query più lente rispetto all'Import Mode, dal momento che i tempi di attesa dipenderanno dalle prestazioni della sorgente. Questa modalità è quindi consigliata quando il tasso di produzione dei dati è alto e l'aggiornamento deve avvenire in tempo reale. Un ulteriore punto a favore riguarda la garantita sicurezza dei dati: solo chi può accedere al data base originale potrà accedere alle dashboard [34].

Il metodo *Live Connection*, come DirectQuery, prevede una connessione diretta con la sorgente con importazione solo di metadati. Le origini a cui ci si può collegare, però, devono

essere a loro volta motori di modellazione, consentendo in questo modo l'aggiornamento dei dati senza necessità di accesso al report. Usando la connessione dinamica è possibile connettere il report a una delle origini dati seguenti [35]:

- Modello semantico già esistente nel servizio Power BI
- Un database di Azure Analysis Services (AAS)
- Istanza locale di SQL Server Analysis Services (SSAS)

Tabella 3. Confronto modalità di connessione ai dati in Power BI

	Vantaggi	**Svantaggi**
IMPORT MODE	La connessione è generalmente più veloce. Le query eseguite dai visual in Power BI vengono valutate sui dati caricati in memoria; di conseguenza, non si hanno ritardi o lentezza nel report.	C'è una limitazione alle dimensioni del modello dati. Il file di Power BI, per funzionare correttamente, non dovrebbe superare 1 GB di memoria. Con Power BI Premium per user è possibile raggiungere i 10 GB, e i 400 GB con la Premium per capacity.
	Power BI resta completamente funzionale. È possibile utilizzare tutti i suoi componenti, compreso Power Query, per la trasformazione e modellazione del dato. Inoltre, non ci sono restrizioni al linguaggio DAX.	
	È permessa la combinazione di dati provenienti da origini diverse.	

DIRECT QUERY	Dati sempre aggiornati senza necessità di refresh.	Prestazioni dipendenti dalla fonte dati: se il database è lento o sovraccarico, anche Power BI avrà tempi di risposta elevati.
	Gestione di dataset di grandi dimensioni, evitando limiti imposti dalla memoria di Power BI.	Limiti sulle trasformazioni dei dati: rispetto alla modalità di importazione, DirectQuery ha limitazioni nelle trasformazioni avanzate disponibili in Power Query.
	Riduzione del tempo di aggiornamento, specialmente per dati che cambiano frequentemente.	Limiti di query e di riga: alcune origini dati impongono restrizioni sul numero massimo di righe restituite per query.
LIVE CONNECTION	Non vi è alcuna limitazione alle dimensioni del modello Analysis Services.	Impossibile combinare dati provenienti da più origini.
	Si può effettuare la connessione direttamente con Power BI senza dover necessariamente replicare il modello importando i dati.	
	È possibile creare ulteriori misure a livello di report.	Non è consentito l'utilizzo di Power Query. Non è possibile cambiare le relazioni tra le tabelle e nemmeno effettuare modifiche sui dati.

Una volta completata la fase di importazione si passa alla trasformazione dei dati, con operazioni di preprocessing, filtraggio e modifiche. Le funzionalità di trasformazione di Power BI sono molto intuitive ed avvengono nell' Editor di Power Query, la cui schermata può essere suddivisa in quattro sezioni:

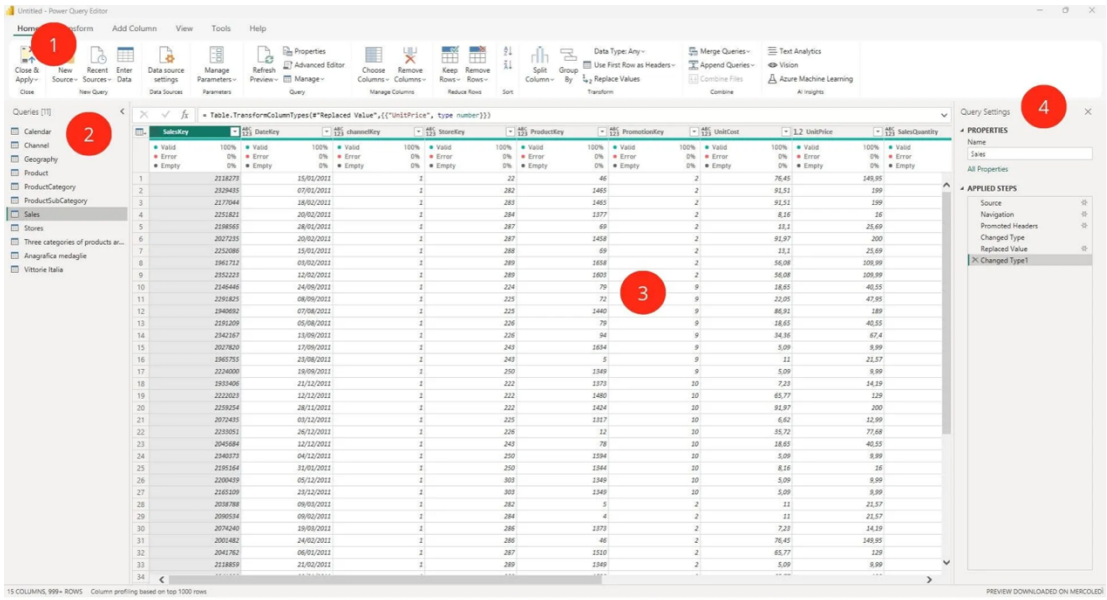

Figura 22. Editor Power Query

1. barra multifunzionale, dove è possibile interagire con i dati;
2. pannello Queries, dove le query caricate vengono elencate e rese disponibili per la selezione, la visualizzazione e il data shaping;
3. riquadro centrale, dove vengono visualizzate le query selezionate, disponibili per essere interrogate e modellate;
4. pannello Query Settings, dove è possibile modificare il nome della query selezionata e visualizzare una pipeline degli step di trasformazione eseguiti. Attraverso il pannello di Applied Steps è quindi possibile selezionare il passaggio, spostarlo prima o dopo un altro passaggio ed eventualmente eliminarlo.

Ogni fonte di dati genera una tabella la cui struttura può essere modificata: filtraggio delle righe, eliminazione delle colonne e cambio del tipo di dati contenuti. È anche possibile analizzarne la distribuzione attraverso processi automatici di verifica della qualità, utili per individuare eventuali anomalie prima del caricamento. Inoltre, è consentito combinare tabelle provenienti da diverse origini dati utilizzando comandi analoghi alle istruzioni JOIN di SQL. Le modifiche apportate non alterano la fonte dati originale, ma solo questa vista specifica, permettendo così di aggiornare i file di origine e applicare automaticamente le modifiche salvate.

Le trasformazioni, selezionabili intuitivamente attraverso le opzioni disponibile nella barra multifunzionale, per essere applicate vengono in realtà convertite dal software stesso in script in linguaggio “M”. Tra le tecniche comuni di trasformazioni si trovano:

- ordinamento e filtraggio dei dati
- aggiunta o rimozione di righe
- aggiunta o rimozione di colonne
- modifica del formato tipo di dato
- gestione dei duplicati
- merge o duplicazioni tabelle
- trasformazioni pivot

Una volta completate le operazioni di cleaning sulle tabelle, attraverso l’icona “Close and Apply” è chiudere l’Editor Power Query e applicare le modifiche di query apportate in Power BI Desktop.

3.1.2 Modellazione dei dati

Terminata la fase di data preparation inizia quella di data modeling, fase di definizione del modello relazionale presente nei dati, dove vengono definite gerarchie e relazioni tra le tabelle che presentano campi comuni, così da poterle elaborare congiuntamente attraverso grafici complessi. La creazione di modelli è, infatti, il processo che consente di ottenere dati connessi pronti per l'uso.

Per costruire un modello efficace, è necessario creare relazioni tra le origini dati, in modo che le tabelle possano interagire tra loro attraverso campi condivisi. Questo collegamento è fondamentale per aggregare le informazioni in modo coerente, senza dover duplicare i dati. Una volta caricati i dati, Power BI Desktop tenta di trovare e creare automaticamente le relazioni, impostando autonomamente le opzioni di “cardinalità”, “direzione filtro incrociato” e “imposta come relazione attiva”. Tuttavia, ciò avviene solo se la corrispondenza viene rilevata con un elevato livello di attendibilità, e spesso risulta necessario creare manualmente le relazioni oppure apportarvi alcune modifiche [36].

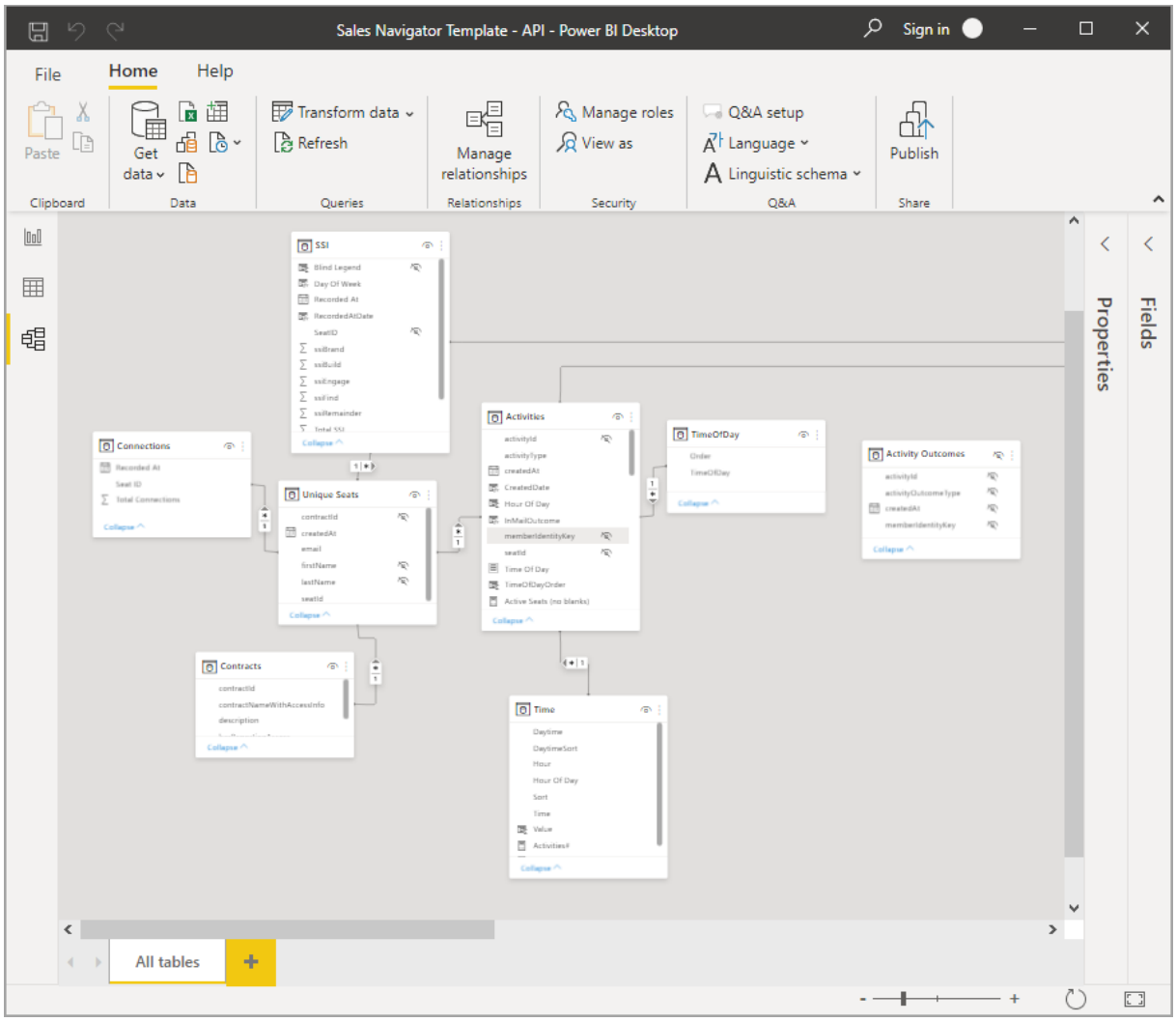

Figura 23. Esempio modello su Power BI

Un altro passaggio importante è la creazione di colonne calcolate, che permettono di generare nuovi campi derivati da quelli esistenti, arricchendo così l'analisi con informazioni aggiuntive senza modificare la fonte originale. Le colonne calcolate usano formule DAX per definire i valori di una colonna. DAX (Data Analysis Expressions) è un potente linguaggio per le formule che consente di creare calcoli avanzati.

L'ottimizzazione del modello passa anche attraverso la riorganizzazione della visualizzazione, nascondendo campi non necessari e ordinando i dati in modo più leggibile. Questo rende il modello più chiaro e fruibile, migliorando l'esperienza dell'utente.

Inoltre, l'utilizzo di misure consente di eseguire calcoli dinamici, come somme, medie o percentuali, adattandoli al contesto della visualizzazione. Le misure, scritte tramite il linguaggio DAX, sono calcoli definiti sui dati che vengono eseguiti al momento della query durante l'interazione con i report, non venendo in questo modo archiviate nel database.

Quando le relazioni tra le tabelle non sono immediatamente disponibili, è possibile utilizzare tabelle calcolate per generare connessioni aggiuntive, aggregare informazioni o filtrare i dati in modo più efficace.

Un altro aspetto fondamentale del data modeling è la creazione di gerarchie, che consentono di strutturare i dati in livelli logici, come ad esempio una gerarchia geografica (continente → paese → città) o una gerarchia temporale (anno → trimestre → mese → giorno). Le gerarchie migliorano l'esperienza di navigazione nei report, permettendo agli utenti di esplorare i dati in modo intuitivo e dettagliato, utilizzando il drill-down per esplorare i dati a diversi livelli di granularità, come anno, trimestre, mese e giorno.

Un concetto chiave nella modellazione dati con Power BI è l'utilizzo dello Star Schema, un approccio che prevede una tabella centrale di fatti collegata a più tabelle di dimensioni. Questo schema semplifica l'organizzazione dei dati, migliora le prestazioni delle query e facilita la creazione di relazioni, evitando le complessità di un modello eccessivamente normalizzato. Lo Star Schema è particolarmente consigliato in Power BI perché ottimizza i tempi di risposta nei report e rende il modello più efficiente e scalabile.

3.1.3 Creazione delle dashboard

Dopo aver elaborato i dati, è possibile creare report interattivi utilizzando la finestra del foglio di lavoro, che offre strumenti per la visualizzazione grafica, la creazione di colonne calcolate e l'aggiunta di misure utili all'analisi.

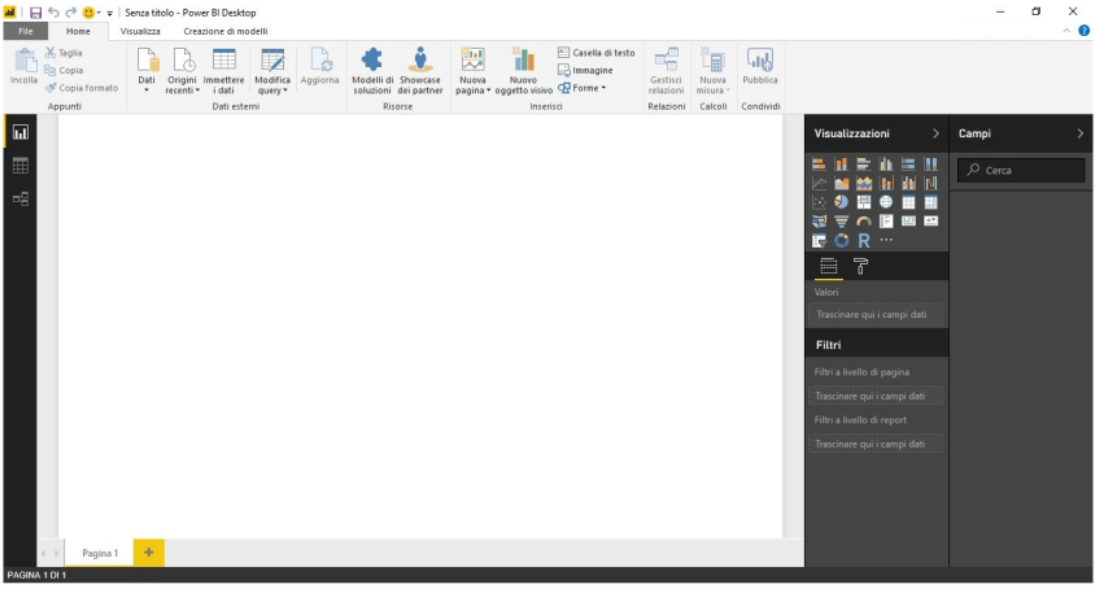

Figura 24. Schermata del foglio di lavoro di Power BI

Sulla sinistra dell'interfaccia di Power BI sono presenti tre icone che rappresentano le principali modalità di visualizzazione disponibili:

- *Report*: consente di usare le query create dall'utente per generare rappresentazioni grafiche dei dati, organizzate in più pagine e personalizzabili secondo le proprie esigenze. Questi report possono essere condivisi con altri utenti per una collaborazione efficace.
- *Dati*: di esaminare le informazioni caricate all'interno del modello, mostrando i dati in formato tabellare. In questa sezione è possibile creare nuove colonne, definire misure personalizzate e gestire le relazioni tra le tabelle per ottimizzare l'analisi.
- *Modello*: fornisce una rappresentazione grafica delle connessioni tra le tabelle nel modello di dati. Da qui è possibile modificare o creare nuove relazioni, assicurandosi che la struttura del database sia ottimizzata per l'analisi e la creazione dei report.

Nella parte superiore dell'interfaccia si trovano tutte le voci del menu di navigazione, insieme alle varie opzioni ad esse collegate. Al centro della schermata è presente l'area di lavoro, in cui vengono elaborati e visualizzati i grafici dei report. Sul lato destro, invece, sono disponibili tre pannelli principali:

- Menù Campi: consente di selezionare le tabelle e gli attributi da includere nel report, permettendo di organizzare i dati in modo strutturato e coerente con l'analisi da svolgere.
- Menù Visualizzazioni: consente di scegliere tra diversi elementi visivi disponibili, come tabelle, matrici, grafici a torta o ad anello, mappe, grafici a barre, ad area o a linee. Oltre ai diagrammi predefiniti presenti nel programma, è possibile aggiungerne di nuovi da un portale dedicato, in cui gli sviluppatori possono pubblicare e condividere le proprie visualizzazioni personalizzate.
- Menù Filtri: permette di impostare filtri per affinare la visualizzazione dei dati. È possibile applicare filtri a livello di pagina, di drill-through o di singolo report, migliorando così la precisione e la rilevanza delle informazioni mostrate nei grafici.

Menù Visualizzazioni

Nella seguente figura viene riportata la schermata delle visualizzazioni presenti su Power BI.

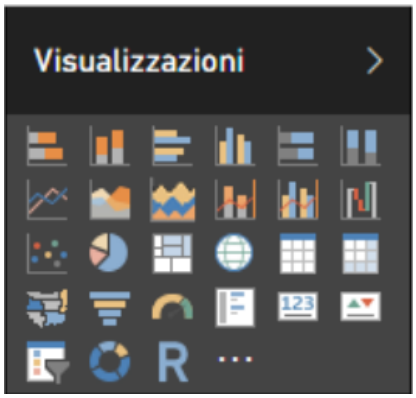

Figura 25. Schermata delle visualizzazioni presenti su Power BI

I cruscotti utilizzano diverse tecniche di visualizzazione per rappresentare i dati in modo chiaro e intuitivo:

- Le **tabelle** sono costituite da righe e colonne e vengono utilizzate per confrontare diverse variabili. Offrono una modalità strutturata per visualizzare una grande quantità

di informazioni, ma possono risultare poco efficaci quando l'obiettivo è individuare rapidamente tendenze di alto livello.

- I **grafici a torta e a barre sovrapposte** suddividono i dati in sezioni che rappresentano parti di un insieme. Questi grafici offrono un metodo intuitivo per organizzare e confrontare le dimensioni di ciascun componente, aiutando a identificare rapidamente la distribuzione delle categorie.
- I **grafici a linee e ad aree** vengono utilizzati per rappresentare la variazione di una o più variabili nel tempo, tracciando una serie di punti dati. I grafici a linee collegano questi punti con linee continue per mostrare l'andamento delle informazioni, mentre i grafici ad area enfatizzano le differenze tra le variabili sovrapponendo le aree colorate. Queste visualizzazioni sono spesso utilizzate nell'analisi predittiva.
- Gli **istogrammi** rappresentano la distribuzione di un insieme di dati attraverso un diagramma a barre senza spazi tra le colonne. Questo tipo di grafico consente di identificare facilmente la frequenza con cui i dati rientrano in determinati intervalli, aiutando a individuare eventuali valori anomali.
- I **grafici a dispersione** sono utili per mostrare la relazione tra due variabili, rendendoli strumenti fondamentali per l'analisi della regressione. Tuttavia, possono essere confusi con i grafici a bolle, che invece rappresentano tre variabili utilizzando l'asse x, l'asse y e la dimensione della bolla per indicare un'ulteriore misura.
- Le **mappe di calore** sono utilizzate per rappresentare i dati comportamentali in base alla posizione. Possono essere applicate a una mappa geografica per evidenziare concentrazioni di valori o a una pagina web per visualizzare il comportamento degli utenti in base alle aree più cliccate.
- Le **mappe ad albero** offrono una rappresentazione gerarchica dei dati utilizzando forme annidate, generalmente rettangoli. Sono particolarmente utili per confrontare le proporzioni tra diverse categorie, dove la dimensione di ciascun rettangolo indica il peso relativo di una determinata variabile rispetto alle altre.

Menù Filtri

Una funzionalità molto utile di Power BI Desktop è il filtraggio dei dati, che consente di affinare l'analisi e migliorare la comprensione delle informazioni. Nella visualizzazione Report, il riquadro "Filtri" permette di applicare diverse tipologie di filtri agli oggetti visivi e alle pagine, consentendo di esplorare i dati in modo più dettagliato.

Power BI offre tre tipologie principali di filtri:

- Il filtro della pagina, che si applica a tutti gli oggetti visivi presenti in una singola pagina del report.
- Il filtro visivo, che agisce su un singolo oggetto visivo all'interno di una pagina del report, senza influenzare gli altri elementi.
- Il filtro di report, che viene applicato globalmente a tutte le pagine del report, garantendo coerenza nell'analisi.

Un aspetto particolarmente utile del riquadro "Filtri" è la possibilità di filtrare i dati anche utilizzando campi che non sono direttamente presenti negli oggetti visivi del report. Ogni filtro può inoltre essere raffinato grazie all'uso dei sottofiltri, che permettono di restringere ulteriormente il campo di analisi.

I principali tipi di sottofiltri includono:

- Il filtro avanzato, che consente di applicare condizioni specifiche sui dati, come ad esempio valori maggiori o minori di una determinata soglia, uguali a un certo valore o che soddisfano determinati criteri.
- Il filtro di base, ideale per selezionare valori precisi come anno, mese o giorno, utile per suddividere i dati in categorie ben definite.
- Il filtro "Primi N", che permette di concentrarsi su un numero limitato di valori, selezionando ad esempio i primi 10 risultati più rilevanti per un'analisi specifica.

Grazie a queste funzionalità avanzate di filtraggio, Power BI consente di personalizzare la visualizzazione dei dati in modo efficace, offrendo maggiore flessibilità nell'interpretazione e nella creazione di report dettagliati.

Formule DAX

A volte i dati disponibili non contengono tutte le informazioni necessarie per rispondere a determinate domande analitiche. Per questo motivo, in Power BI è possibile creare nuove misure, ovvero calcoli avanzati personalizzati utilizzando formule scritte in DAX.

DAX (Data Analysis Expressions) è il linguaggio di formule utilizzato in Power BI. Si tratta di un linguaggio funzionale, nel quale tutto il codice eseguito è contenuto all'interno di una funzione. DAX mette a disposizione una libreria con oltre 200 funzioni, operatori e costrutti, offrendo un'elevata flessibilità nella creazione di misure personalizzate per l'analisi dei dati. Le funzioni DAX possono essere annidate, contenere istruzioni condizionali e fare riferimento a valori dinamici, permettendo così di costruire calcoli complessi.

L'esecuzione delle espressioni in DAX segue una logica dall'interno verso l'esterno, ovvero inizia dalla funzione o dal parametro più interno e procede progressivamente verso l'esterno. Un aspetto fondamentale è che i risultati delle misure DAX vengono aggiornati dinamicamente in base alle interazioni dell'utente nei report, rendendo l'esplorazione dei dati veloce e interattiva.

Oltre alle misure, il linguaggio DAX consente anche di creare colonne calcolate e tabelle calcolate, elementi utili per arricchire ulteriormente i dataset con informazioni derivate da calcoli specifici.

Grazie alla potenza di DAX, è possibile trasformare i dati grezzi in informazioni strutturate e dettagliate, migliorando la capacità di analisi e il valore dei report creati in Power BI.

3.1.4 Pubblicazione e condivisione delle dashboard

Una volta creata una dashboard in Power BI Desktop, è possibile pubblicarla e condividerla attraverso il servizio Power BI Service, per consentire ad altri utenti di visualizzarla e interagire con i dati. Gli utenti possono accedere alle dashboard pubblicate tramite un browser web o l'app mobile di Power BI, garantendo flessibilità e accessibilità. Il servizio online di Power BI, all'interno dell'azienda, è un luogo di condivisione e collaborazione, dove gli elaborati sviluppati vengono utilizzati per monitorare l'andamento di determinati processi e comunicare alle varie funzioni aziendali le informazioni chiave emerse dall'analisi dei dati. Attraverso questa piattaforma, gli utenti possono condividere

report sia all'interno che all'esterno dell'azienda in tempo reale, garantendo in questo modo l'accuratezza e la rilevanza dei dati. Questa interazione dinamica favorisce una maggiore efficacia decisionale, permettendo alle aziende di agire in modo tempestivo e strategico.

Un aspetto fondamentale della condivisione è la gestione della sicurezza e dei permessi, che consente di controllare chi può visualizzare o modificare i report. Con l'uso di Row-Level Security (RLS), è possibile filtrare i dati in base ai ruoli degli utenti, garantendo che ciascuno veda solo le informazioni a lui pertinenti.

Capitolo 4

Analysis Services e Tabular Editor

4.1 Introduzione a SQL Server Analysis Services

SQL Server Analysis Services (SSAS) è un componente fondamentale di Microsoft SQL Server, progettato per l'analisi e l'elaborazione **multidimensionale** dei dati. SSAS permette di creare modelli di dati analitici ottimizzati per il reporting e il data mining, supportando sia modelli multidimensionali (OLAP) sia modelli tabulari più moderni e performanti. Tra le sue caratteristiche distintive si evidenziano:

- Modelli di dati centralizzati: creare un modello in SSAS consente di evitare che ogni processo di aggiornamento e caricamento dati debba interrogare direttamente il software di origine, garantendo così maggiore efficienza e riducendo il carico sul sistema sorgente.
- Capacità di gestione dei dati: SSAS può contenere volumi di dati molto più grandi rispetto a strumenti come Power BI. Questo permette di aggregare e analizzare grandi quantità di informazioni senza compromettere le prestazioni.
- Flessibilità di reporting: il modello creato in SSAS può essere collegato a numerosi report realizzati con Power BI, Excel o altri strumenti di visualizzazione, offrendo così una piattaforma centralizzata per la distribuzione e l'analisi dei dati.
- Prestazioni elevate: poiché i dati vengono caricati in memoria (RAM) anziché su disco, l'accesso e l'elaborazione dei dati avvengono in maniera estremamente rapida, migliorando notevolmente le performance delle query.
- Linguaggi di espressione avanzati: SSAS si avvale dei linguaggi MDX e DAX, consentendo una manipolazione sofisticata e flessibile dei dati.

- Gestione avanzata della sicurezza: SSAS offre strumenti per definire ruoli di sicurezza personalizzati, assicurando che l'accesso ai dati sia controllato e riservato agli utenti autorizzati.
- Integrazione con ecosistemi BI: l'integrazione nativa con strumenti come Power BI o Excel permette di sfruttare al meglio le capacità di visualizzazione e analisi, rendendo SSAS uno strumento essenziale per la Business Intelligence.

4.2 Introduzione a Tabular Editor

Un software strettamente collegato a SSAS in modalità tabulare è Tabular Editor, uno strumento avanzato per la gestione e la modifica di modelli di dati in Analysis Services Tabular, Power BI e Azure Analysis Services. Sviluppato per migliorare l'efficienza dei modelli tabulari, consente di lavorare in un ambiente più flessibile rispetto a quello offerto dall'interfaccia standard di Power BI Desktop o SQL Server Data Tools (SSDT). Grazie alla sua interfaccia intuitiva e alle funzionalità avanzate, Tabular Editor è particolarmente apprezzato per la creazione e gestione di misure DAX, colonne calcolate e oggetti di metadati, offrendo un controllo più preciso sui modelli tabulari. Esistono due versioni di Tabular Editor: Tabular Editor 2 (gratuito e open-source) e Tabular Editor 3 (a pagamento, con funzionalità avanzate come la creazione guidata di modelli e debugging DAX).

4.3 Caratteristiche principali di Tabular Editor

Tabular Editor si contraddistingue per molteplici caratteristiche chiave che lo rendono uno strumento vantaggioso per la gestione dei modelli tabulari:

- Modifica offline dei modelli tabulari: consente di modificare file BIM senza necessità di accesso e caricamento dei dati in memoria ad ogni process, migliorando così le prestazioni, riducendo i costi computazionali ed i tempi di elaborazione. Inoltre, vi è la possibilità di accedere a più file BIM contemporaneamente [37].
- Automatizzazione tramite script avanzati in C#: tramite uno scripting engine basato sul linguaggio di programmazione C#, è possibile automatizzare operazioni ripetitive e

ottimizzare la gestione dei modelli attraverso funzioni non realizzabili semplicemente tramite la User Interface [38].

- Supporto completo per il linguaggio DAX: permette la creazione e la gestione di misure, colonne calcolate e gerarchie direttamente nell'editor, che risulta a tal scopo completo ed intuitivo.
- Best Practice Analyzer (BPA): strumento per definire regole sui metadati del modello, verificandone la conformità a best practice e riducendo in questo modo il rischio di errori, con conseguente miglioramento prestazionale del modello [39].
- Gestione dei ruoli: permette la configurazione della Data Security, definendo la Row-Level Security (RSL) e la Object-Level Security (OLS). Lo scopo della Data Security è quello di garantire che gli utenti vedano e utilizzino solo i dati a cui sono autorizzati, sia nei report pubblicati che quando creano le proprie soluzioni di dati self-service. Agli utenti utilizzatori sono quindi assegnati dei ruoli con regole di filtro RLS e di restrizione OLS predefinite.

Una volta caricato il file BIM o dopo aver effettuato il collegamento con un database da Analysis Services, l'interfaccia di Tabular Editor si presenta organizzata in diverse sezioni che facilitano la navigazione e la gestione dei modelli tabulari.

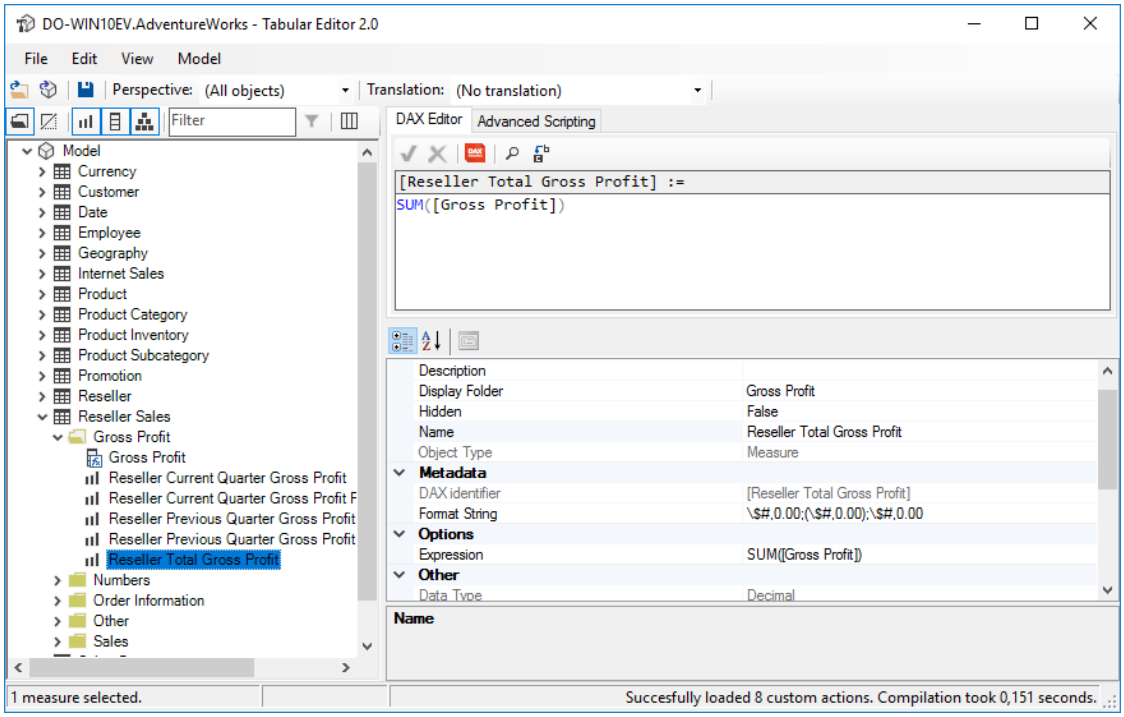

Figura 26. Interfaccia di Tabular Editor

A sinistra, una visualizzazione gerarchica mostra tutti gli oggetti del modello tabulare, consentendo di vedere e modificare tabelle, colonne, misure o gerarchie, raggruppate in base alle cartelle di visualizzazione (Display Folder).

Nella parte superiore dell'interfaccia si trovano tutte le voci del menu di navigazione, insieme alle varie opzioni ad esse collegate. Sul lato destro, invece, sono visibili tre pannelli:

- Griglia delle proprietà degli oggetti: consente di esaminare e configurare rapidamente gli attributi di ogni elemento del modello, come formati, descrizione, tipi di aggregazione e gerarchie;
- Editor DAX: un'area dedicata alla scrittura e modifica di espressioni DAX;
- Script Editor: un'area dedicata alla scrittura e modifica di script in C#.

4.4 Automatizzazione

Il progetto di tesi si propone di sviluppare script in C# per automatizzare operazioni ripetitive nel processo di creazione del modello dati. Di seguito vengono riportati i codici elaborati.

Relazioni tra tabelle

Per una corretta modellazione è necessario relazionare tra loro le tabelle. Nel caso di uno Star Schema, ottimizzato per l'integrazione su Power BI, ogni dimension table è collegata alla fact table tramite una chiave surrogata.

```
var keySuffix = "_SK";  // Suffisso delle chiavi surrogate
var dimensionPrefix = "DIM_";  // Prefisso dimension tables
var factPrefix = "FACT_"; // Prefisso fact tables

// Loop su tutte le FACT tables:
foreach (var fact in Model.Tables.Where(f => f.Name.StartsWith(factPrefix,
StringComparison.OrdinalIgnoreCase))) // No sensitive case
{
    // Loop su tutte le colonne SK della tabella:
    foreach (var factColumn in fact.Columns.Where(c =>
    c.Name.EndsWith(keySuffix, StringComparison.OrdinalIgnoreCase)))
    {
        // Cerca una dimension table con lo stesso nome della SK:
        var matchingDimensionTable = Model.Tables
            .Where(t => t.Name.StartsWith(dimensionPrefix,
                StringComparison.OrdinalIgnoreCase))  // Solo dimension tables
            .FirstOrDefault(t => t.Name.StartsWith(dimensionPrefix +
```

```
                factColumn.Name.Replace(keySuffix, ""),
                StringComparison.OrdinalIgnoreCase)); // Cerca corrispondeza
                                                        nel nome della tabella

            // L'operatore ?? ritorna il valore di sinistra se esiste,
               altrimento quello di destra: se trova una dimension table con
               nome corrispondente, la usa, altrimenti cerca nelle altre
               dimension tables:
        var dim = matchingDimensionTable ?? Model.Tables
            .Where(t => t.Name.StartsWith(dimensionPrefix,
               StringComparison.OrdinalIgnoreCase))  // Solo dimension tables
            .FirstOrDefault(t => t.Columns.Any(c => c.Name ==
             factColumn.Name)); // Cerca corrispondeza nel nome della colonna

         if (dim != null)
         {
              // Trova la colonna chiave nella dimension table:
              var dimColumn = dim.Columns.FirstOrDefault(c => c.Name ==
                   factColumn.Name);

              if (dimColumn != null)
              {
            // Controlla che non esista già una relazione tra le due colonne:
                   if (!Model.Relationships.Any(r => r.FromColumn ==
                      factColumn && r.ToColumn == dimColumn))
                   {
                       // Se esiste già una relazione tra le due tabelle,
                      Model.Relationships.Any ritorna TRUE e la nuova
                      relazione verrà salvata come inattiva:
                       var makeInactive = Model.Relationships.Any(r =>
                            r.FromTable == fact && r.ToTable == dim);

                       // Aggiunge una nuova relazione:
                       var rel = Model.AddRelationship();
                       rel.FromColumn = factColumn;
                       rel.ToColumn = dimColumn;
                       factColumn.IsHidden = true;  // Nasconde la colonna
                                                      della fact table

                    // Setta la relazione inattiva se già ne esisteva un'altra
                       tra le due tabelle:
                       if (makeInactive) rel.IsActive = false;
                   }
            }
        }
    }
}
```

Lo script utilizza cicli for per analizzare le colonne di ogni fact table nel modello, identificando le chiavi surrogate grazie a un suffisso nel nome. Per ciascuna chiave surrogata, il processo cerca prima una dimension table il cui nome, rimosso l'eventuale prefisso, coincida con quello della chiave. Se non esiste una corrispondenza diretta, vengono esaminate tutte le dimension tables per trovare una colonna con lo stesso nome della chiave surrogata nella fact table. Una volta individuata la corretta dimension table, la chiave

surrogata viene identificata e viene creata una relazione molti-a-uno tra la fact table e la dimension table. L'analista, in base al tipo di progetto in lavorazione, può adattare lo script semplicemente modificando l'inizializzazione delle variabili iniziali.

Formattazione colonne

Tutte le colonne numeriche delle fact tables, rappresentanti le misure degli eventi presi in considerazione, vanno formattate con l'impostazione di aggregazione di sommatoria. Ciò permette il corretto funzionamento della dashboard realizzata a partire dal modello, consentendo un'esatta analisi dei dati.

```
var factPrefix = "FACT_"; // Prefisso fact tables
var keySuffix = "_SK";  // Suffisso delle chiavi surrogate

// per ogni fact table:
foreach(var table in Model.Tables.Where(t => t.Name.StartsWith(factPrefix,
StringComparison.OrdinalIgnoreCase)))
{
    // per ogni colonna numerica (no sk):
    foreach(var c in table.Columns.Where(a => a.DataType == DataType.Decimal
                                          || a.DataType == DataType.Double
                                          || (a.DataType == DataType.Int64
                                              && !a.Name.EndsWith(keySuffix,
                                        StringComparison.OrdinalIgnoreCase))
                                        )
            )
    {
        c.SummarizeBy = AggregateFunction.Sum;

    }
}
```

Lo script scansiona le fact tables nel modello utilizzando cicli for, individuando tutte le colonne. Per ciascuna colonna numerica (intera o decimale) che non rappresenta una chiave, imposta automaticamente la funzione di aggregazione su *"Sum"*.

Le colonne numeriche delle dimension tables, invece, vanno impostate con la funzione di aggregazione nulla. I campi in tali tabelle, infatti, rispecchiano proprietà qualitative delle singole istanze.

```
var dimensionPrefix = "DIM_"; // Prefisso dimension table

// per ogni dimension table:
foreach(var t in Model.Tables.Where(table =>
table.Name.StartsWith(dimensionPrefix, StringComparison.OrdinalIgnoreCase)))
{
    foreach(var c in t.Columns.Where(a => a.DataType == DataType.Decimal ||
                                          a.DataType == DataType.Double ||
                                          a.DataType == DataType.Int64))
    {
        c.SummarizeBy = AggregateFunction.None;

    }
}
```

Lo script scansiona le dimension tables nel modello utilizzando cicli for, individuando tutte le colonne. Per ciascuna colonna numerica (intera o decimale), imposta automaticamente la funzione di aggregazione su *"None"*.

Creazione misure con DAX

Le metriche delle fact tables vengono spesso utilizzate in un contesto di minor livello di granularità. Per tale motivo viene automatizzata la creazione delle misure di sommatoria.

```
var factPrefix = "FACT_"; // Prefisso fact tables

// per ogni fact table:
foreach(var table in Model.Tables.Where(t => t.Name.StartsWith(factPrefix,
StringComparison.OrdinalIgnoreCase)))
{
    // Crea una misura SUM per ogni colonna selezionata, ma solo se la misura
non esiste già
    foreach(var c in table.Columns.Where(a => a.SummarizeBy ==
AggregateFunction.Sum)) // Per ogni colonna con SummarizeBy SUM
    {
        // Verifica se la misura con lo stesso nome esiste già
        var existingMeasure = c.Table.Measures.FirstOrDefault(m => m.Name ==
"Sum of " + c.Name);

        // Se la misura non esiste:
        if(existingMeasure == null)
        {
            var newMeasure = c.Table.AddMeasure(
                "Sum of " + c.Name,                    // Nome della misura
                "SUM(" + c.DaxObjectFullName + ")",    // Espressione DAX
                "Misure"                               // Cartella di
```

```
visualizzazione
            );

            // Imposta la stringa di formato per la nuova misura:
            newMeasure.FormatString = "#,##0.00"; // numero decimale con
separatore delle migliaia

            // Aggiunge una descrizione per la misura:
            newMeasure.Description = newMeasure.Expression;

            // Nasconde la colonna originale:
            c.IsHidden = true;
        }
    }
}
```

Il codice analizza le tabelle nel modello di dati, identificando le fact tables, il cui nome inizia con il prefisso assegnato. Per ogni tabella trovata, esamina le colonne che hanno l'aggregazione predefinita impostata su “Sum” e verifica se esiste già una misura calcolata di sommatoria. Se la misura non è presente, ne crea una nuova utilizzando l'espressione DAX (`SUM('Nome tabella'[Nome colonna])`), la organizza in una cartella a scelta, imposta il formato numerico con separatori delle migliaia e decimali, aggiunge una descrizione esplicativa e infine nasconde la colonna originale per evitare ridondanze nei report.

In molti progetti applicabili a svariati settori aziendali vengono calcolate misure per permettere il confronto temporale delle misure stesse. Il seguente codice automatizza la creazione di analisi temporali, migliorando la coerenza e la rapidità di sviluppo del modello.

```
var dateColumn = "'DIM_CALENDAR'[Date]";
var displayFolder = "Time Intelligence";

// Crea misure di time intelligence per ogni misura selezionata:
foreach(var m in Selected.Measures) {
    // Year-to-date:
    m.Table.AddMeasure(
        m.Name + " YTD",                                          // Nome
        "TOTALYTD(" + m.DaxObjectName + ", " + dateColumn + ")",// Espressione
                                                                    DAX
        displayFolder                                           // Cartella
    ).FormatString = "#,##0.00";                                  // Formato

    // Previous year:
    m.Table.AddMeasure(
        m.Name + " PY",
        "CALCULATE(" + m.DaxObjectName + ", SAMEPERIODLASTYEAR(" + dateColumn
         + "))",
        displayFolder
    ).FormatString = "#,##0.00";

    // Year-over-year
```

```
    m.Table.AddMeasure(
        m.Name + " YoY",
        m.DaxObjectName + " - [" + m.Name + " PY]",
        displayFolder
    ).FormatString = "#,##0.00";

    // Year-over-year %:
    m.Table.AddMeasure(
        m.Name + " YoY%",
        "DIVIDE([" + m.Name + " YoY], [" + m.Name + " PY])",
        displayFolder
    ).FormatString = "0.0 %";

    // Quarter-to-date:
    m.Table.AddMeasure(
        m.Name + " QTD",
        "TOTALQTD(" + m.DaxObjectName + ", " + dateColumn + ")",
        displayFolder
    ).FormatString = "#,##0.00";

    // Month-to-date:
    m.Table.AddMeasure(
        m.Name + " MTD",
        "TOTALMTD(" + m.DaxObjectName + ", " + dateColumn + ")",
        displayFolder
    ).FormatString = "#,##0.00";
}
```

Questo codice genera automaticamente misure di time intelligence per ogni misura selezionata all'interno del modello dati, utilizzando una colonna specificata come riferimento per la data nel modello temporale.

Per ogni misura selezionata, vengono create diverse versioni calcolate:

- Year-to-Date (YTD): tramite la funzione TOTALYTD, calcola il valore della misura considerando il periodo dall'inizio dell'anno fino alla data corrente;
- Previous Year (PY): tramite la funzione SAMEPERIODLASTYEAR, calcola il valore della misura dell'anno precedente;
- Year-over-Year (YoY): calcola la differenza tra la misura corrente e il valore dell'anno precedente;
- Year-over-Year % (YoY%): calcola la variazione percentuale tra i due periodi;
- Quarter-to-Date (QTD): tramite la funzione TOTALQTD, calcola il valore della misura considerando il periodo dall'inizio del trimestre corrente fino alla data attuale;
- Month-to-Date (MTD) tramite la funzione TOTALMTD, calcola il valore della misura considerando il periodo dall'inizio del mese corrente fino alla data attuale.

Tutte queste misure possono essere organizzate all'interno di cartelle scelte e formattate in base alle esigenze specifiche.

Formattazione misure

Una formattazione corretta e coerente dei numeri all'interno delle tabelle dei fatti garantisce una migliore leggibilità, ottimizzando la presentazione dei dati nel modello.

```
var factPrefix = "FACT_"; // Prefisso fact tables
var keySuffix = "_SK";  // Suffisso delle chiavi surrogate
var jobid = "JOBID"; // eventuali altri suffissi

// per ogni fact table:
foreach(var table in Model.Tables.Where(t => t.Name.StartsWith(factPrefix,
StringComparison.OrdinalIgnoreCase)))
{
    // per ogni colonna numerica (no sk):
    foreach(var c in table.Columns.Where(a =>   a.DataType == DataType.Decimal
                                            || a.DataType == DataType.Double
                                            || (a.DataType == DataType.Int64 &&
!a.Name.EndsWith(keySuffix, StringComparison.OrdinalIgnoreCase)

&& !a.Name.EndsWith(jobid, StringComparison.OrdinalIgnoreCase))))
    {
        c.FormatString = "#,##0";

    }
}
```

Questo codice scansiona tutte le fact tables nel modello e applica un formato numerico alle colonne che soddisfano determinati criteri. Per ogni fact table trovata, esamina le colonne e seleziona quelle di tipo intero, decimale o double, escludendo quelle di carattere funzionale, quali le chiavi surrogate o altre eventuali. Alle colonne che rispettano queste condizioni viene assegnato il formato numerico “#,##0”, che applica il separatore delle migliaia e rimuove eventuali decimali.

Lo stesso codice può essere riadattato per assegnare alle misure il formato “#,##0.00”, che applica il separatore delle migliaia mantenendo però due cifre decimali.

```
var factPrefix = "FACT_"; // Prefisso fact tables
var keySuffix = "_SK";  // Suffisso delle chiavi surrogate
var jobid = "JOBID";

// per ogni fact table:
```

```
foreach(var table in Model.Tables.Where(t => t.Name.StartsWith(factPrefix,
StringComparison.OrdinalIgnoreCase)))
{
    // per ogni colonna numerica (no sk):
    foreach(var c in table.Columns.Where(a =>    a.DataType == DataType.Decimal
                                              || a.DataType == DataType.Double
                                              || (a.DataType == DataType.Int64
&& !a.Name.EndsWith(keySuffix, StringComparison.OrdinalIgnoreCase)

&& !a.Name.EndsWith(jobid, StringComparison.OrdinalIgnoreCase))))
    {
        c.FormatString = "#,##0.00";

    }
}
```

Modifiche funzionali

Il seguente approccio consente di mantenere il modello più conciso e focalizzato sui dati rilevanti, nascondendo automaticamente le colonne non necessarie per l'analisi.

```
// Array di prefissi e suffissi
var suffixes = new string[ ] { "_SK" };
var prefixes = new string[ ] { "JOBID_", "INS_", "UPD_" };

// Loop attraverso tutte le tabelle del modello
foreach(var table in Model.Tables)
{
    // Loop attraverso tutte le colonne della tabella corrente
    foreach(var c in table.Columns.Where(k => suffixes.Any(p =>
k.Name.EndsWith(p))
                                          || prefixes.Any(p =>
k.Name.StartsWith(p))
                                    )
    )
    {
        // Nascondi la colonna
        c.IsHidden = true;
    }
}
```

Il codice scansiona tutte le tabelle presenti nel modello e nasconde automaticamente le colonne che soddisfano determinati criteri di denominazione. In particolare, una colonna viene nascosta se il suo nome termina con uno dei suffissi definiti nell'array *"suffixes"* (ad esempio "_SK", che solitamente indica chiavi surrogate) oppure se inizia con uno dei prefissi specificati nell'array *"prefixes"* (come "JOBID_", "INS_" e "UPD_", che potrebbero rappresentare metadati di gestione o tracciamento).

Per migliorare la leggibilità dei nomi delle tabelle, può essere utile il processo di ridenominazione. Per sostituire un carattere ricorrente (ad esempio “ _ ”), è possibile applicare il seguente script.

```
foreach (var table in Model.Tables)
{
    // Trova il primo underscore
    int underscoreIndex = table.Name.IndexOf("_");

    // Se c'è un underscore, sostituisci solo il primo
    if (underscoreIndex >= 0)
    {
        table.Name = table.Name.Substring(0, underscoreIndex) + " " +
table.Name.Substring(underscoreIndex + 1);
    }
}
```

Il codice scorre tutte le tabelle nel modello e modifica il loro nome nel seguente modo: cerca il primo underscore (_) nel nome della tabella; se l'underscore è presente, sostituisce il primo con uno spazio. Ad esempio, se una tabella si chiama "FACT_SALES", il nome verrà modificato in "FACT SALES". L'operazione viene effettuata solo sul primo underscore trovato nel nome della tabella, lasciando intatti gli eventuali altri underscore presenti.

Le potenzialità di questi script risiedono nella loro flessibilità e adattabilità a qualsiasi tipo di modello, rendendoli strumenti estremamente versatili per diversi contesti. Le operazioni svolte sono, infatti, necessarie nella maggior parte dei modelli di dati e possono essere facilmente personalizzate per rispondere a esigenze specifiche. Le variabili e i criteri, come formati numerici, nomi di tabelle e colonne, prefissi e suffissi, sono totalmente configurabili, consentendo di adattare lo script a qualsiasi struttura e convenzione di naming. Questa personalizzazione rende gli script particolarmente utili per progetti complessi o modelli in continua evoluzione, dove la capacità di modificare rapidamente le impostazioni è fondamentale per garantire un flusso di lavoro efficiente e scalabile.

Capitolo 5

Case study

Il caso studio elaborato utilizza il database di un cliente di Mediamente Consulting appartenente al settore della grande distribuzione organizzata. La GDO in esame è una società distributiva multibrand e multicanale di prodotti alimentari e altri generi. L'azienda presenta un'area di competenza concentrata principalmente nel Centro-Sud d'Italia, con una maggiore diffusione nelle regioni della Basilicata e della Campania.

5.1 Introduzione ai dati

L'evento preso in considerazione rappresenta le giacenze dell'intera società, entrando nel dettaglio dei diversi magazzini, punti vendita e prodotti offerti dell'azienda, al fine di osservarne l'andamento ed il mantenimento. Le tabelle coinvolte sono le seguenti:

Tabella 4. Tabelle case study

TABELLA	**Categoria**	**Descrizione**
DIM_AZIENDE	DIMENSIONE	Contiene informazioni riguardo le aziende facenti parte della GDO
DIM_CLASSEAGGREGATA	DIMENSIONE	Differenzia le due tipologie di aggregazione dei dati (*giacenza*, *giacenza cumulata*)
DIM_MAGAZZINI	DIMENSIONE	Identifica i differenti centri di distribuzione
DIM_PDV	DIMENSIONE	Fornisce informazioni dettagliate sui punti di vendita, tra cui nome del brand,

		codice azienda, indirizzo, località, data di apertura ed eventuale data di chiusura, stato del punto di vendita (attivo o meno)
DIM_PRODOTTO	DIMENSIONE	Fornisce informazioni dettagliate sui prodotti offerti, tra cui nome, categoria di appartenenza, marca, produttore e varie altre informazioni sul confezionamento
DIM_SCENARIO	DIMENSIONE	definisce i diversi contesti o stadi in cui vengono analizzate le giacenze, distinguendo tra dati effettivi (*consuntivo*), preliminari (*bozza*), rivisti (*revised*) e previsionali (*forecast*)
DIM_TIPO_GIACENZA	DIMENSIONE	Identifica la tipologia di giacenza (*mensile*, *settimanale*, *giornaliera*)
DIM_PROMOTION	DIMENSIONE	raccoglie le informazioni relative alle promozioni e agli eventi, offrendo un contesto per analizzare l'impatto delle attività promozionali su vendite e giacenze
FACT_GIACENZE	FATTO	Oltre alle varie chiavi surrogate delle tabelle dimensionali, è composta dall'insieme delle metriche di interesse per l'evento preso in considerazione, quali il valore delle giacenze e delle giacenze cumulate per diverse tipologie di analisi finanziarie, la quantità dei prodotti fornita secondo varie unità di misura (*pezzi*, *kg*, *colli*, *pallet*)

5.2 Creazione del modello

La prima operazione necessaria per la costruzione del modello consiste nell'instaurare una connessione tra il progetto di Tabular Editor e il database di interesse, presente su SQL Server Management Studio (SSMS). Questa connessione è essenziale per consentire l'accesso ai dati e garantire un'integrazione fluida tra il database relazionale e l'ambiente di sviluppo del modello tabulare.

Dopo aver configurato correttamente la connessione, sono stati importati i metadati delle tabelle di interesse all'interno di Tabular Editor. In questa fase iniziale, si definiscono le tabelle sorgenti, che rappresentano il nucleo del modello dati.

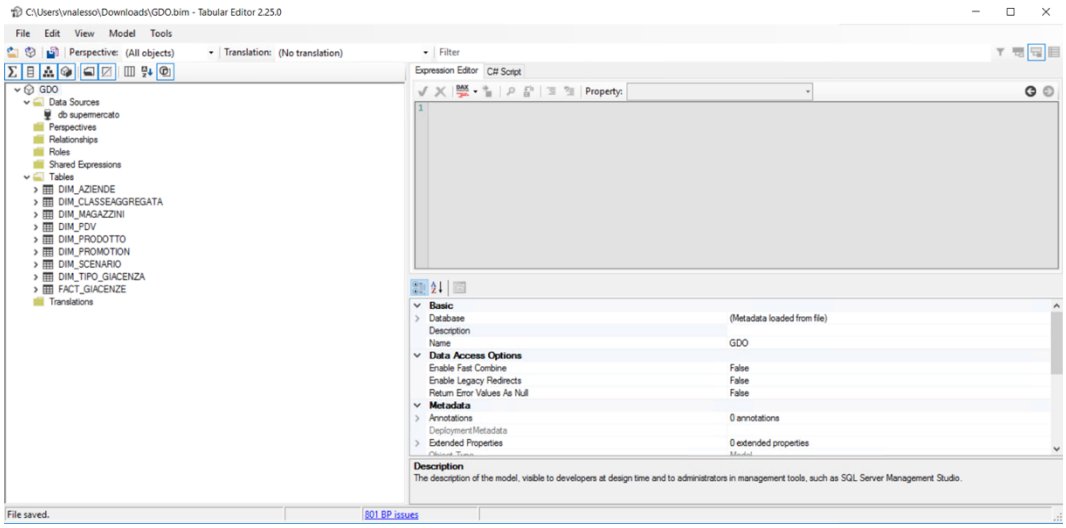

Figura 27. Collegamento al database e import delle tabelle

Per garantire un'adeguata gestione delle analisi temporali, è stata creata una tabella calcolata (DIM_TEMPO) contenente informazioni temporali, che funge da calendario per il modello. Questa tabella include un elenco di date e ulteriori attributi temporali utili per le analisi, come l'anno, il mese e il quadrimestre.

Una volta definite le basi del modello, si è proceduto con una serie di operazioni di ottimizzazione e strutturazione attraverso l'uso degli script personalizzati, i quali hanno permesso di automatizzare rapidamente i seguenti passaggi:

- Creazione delle relazioni tra la tabella dei fatti (FACT_GIACENZE) e le relative tabelle dimensionali, stabilendo i legami necessari per navigare correttamente i dati e consentire aggregazioni e filtri coerenti all'interno della dashboard.

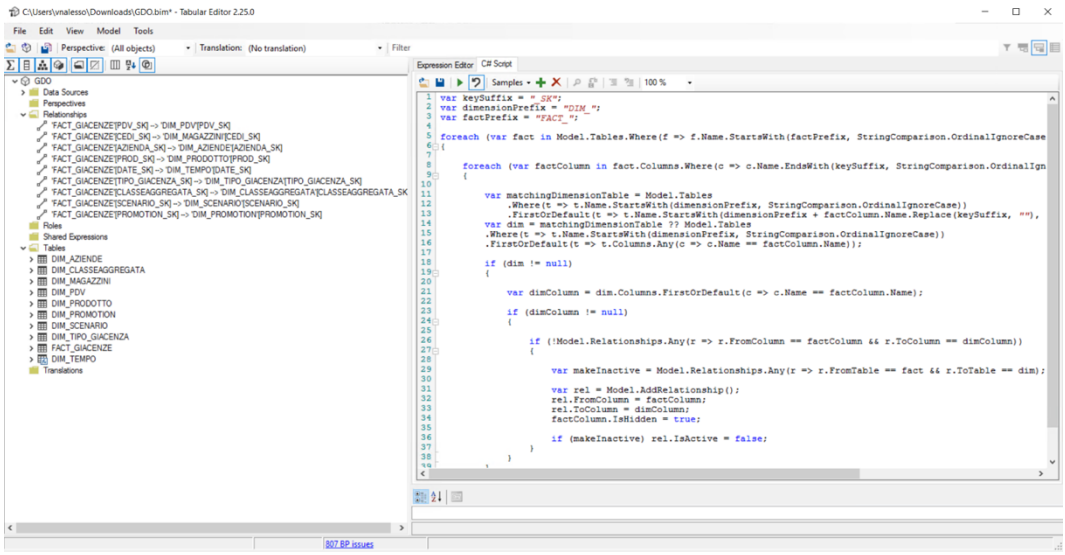

Figura 28. Creazione automatizzata delle relazioni tra tabelle

- Formattazione delle colonne numeriche per garantire una corretta aggregazione. In particolare, nelle fact tables, è stato definito il tipo di aggregazione di sommatoria ed è stato applicato un formato numerico coerente per migliorare la leggibilità dei dati nei report. Per le dimensioni, invece, è stata disattivata la funzione di aggregazione.

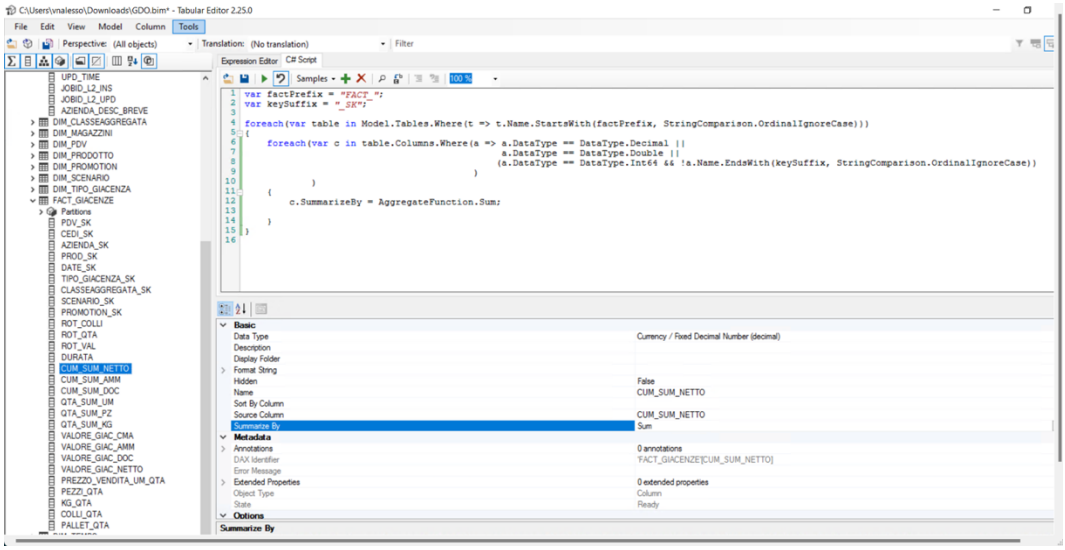

Figura 29. Formattazione automatizzata della funzione di aggregazione nella fact table

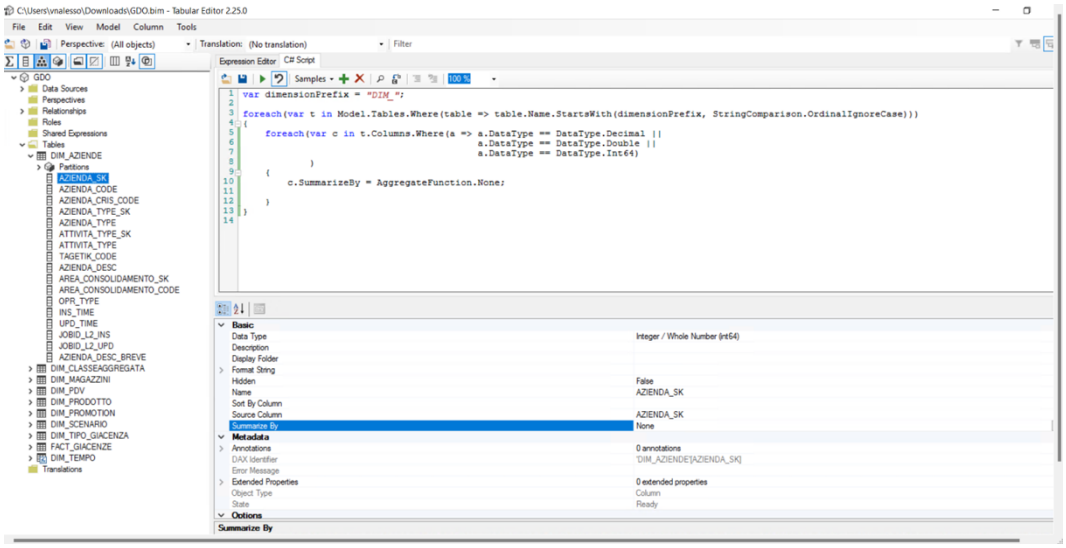

Figura 30. Formattazione automatizzata della funzione di aggregazione nelle dimension tables

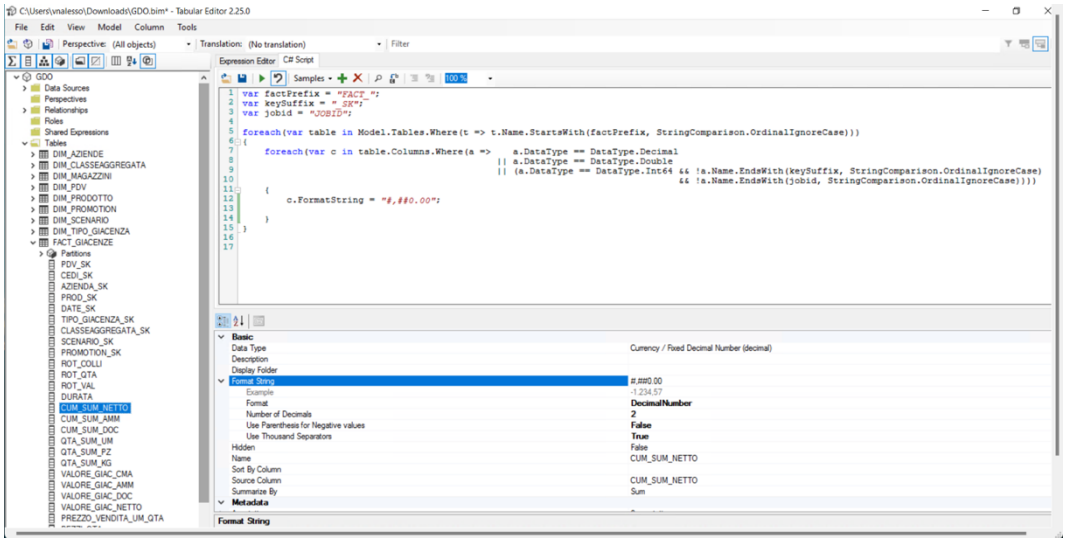

Figura 31. Impostazione automatizzata del formato delle colonne nella fact table

- Creazione delle misure basate sulle metriche presenti in FACT_GIACENZE e relativa formattazione. Questo ha permesso di ottenere indicatori sul valore della GIACENZA CUMULATA immediatamente utilizzabili all'interno della dashboard, senza necessità di ulteriori calcoli lato reportistica.

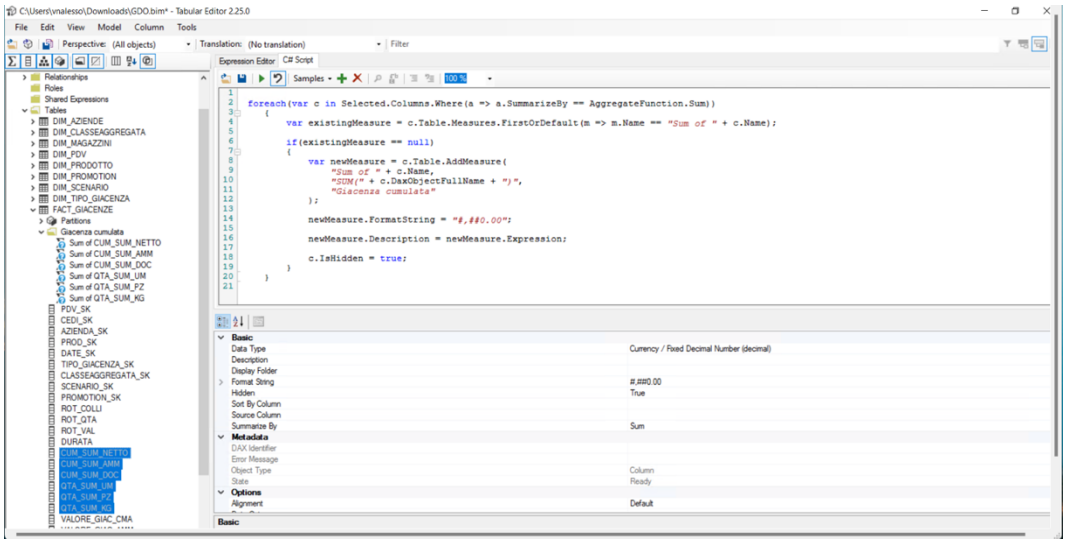

Figura 32. Creazione automatizzata delle sum measures delle colonne selezionate

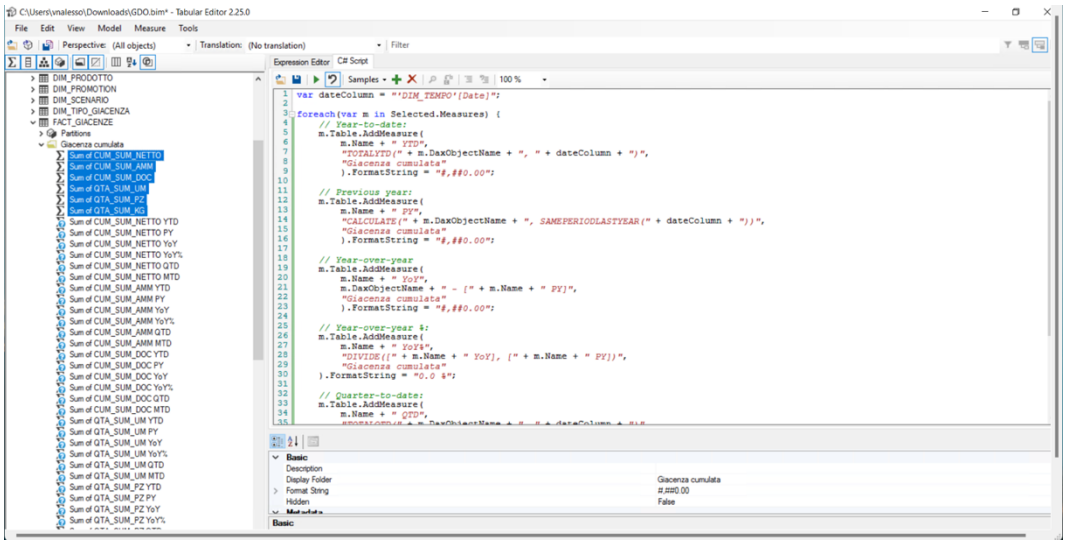

Figura 33. Creazione automatizzata delle time intelligence measures delle colonne selezionate

Completate queste operazioni, il modello è pronto per essere utilizzato e integrato in una dashboard. Tuttavia, prima di procedere al deploy su un'istanza di Analysis Services, sono state apportate ulteriori modifiche e integrazioni per adattarlo alle specifiche esigenze del caso in esame.

Personalizzazioni e Ottimizzazioni Finali

- Calcolo della variazione temporale (VARIAZIONE_CUM_SUM) della giacenza tramite colonne calcolate (una per ogni tipologia di analisi finanziaria). Questa colonna consente di monitorare entrate e uscite dai magazzini nel tempo.
- Definizione di misure specifiche per valutare i KPI, quali:

$$\text{Giacenza media Mensile} = \frac{\text{Giacenza cumulata}}{\#\text{mesi considerati}}$$

$$\text{Uscite} = \sum \text{Variazione giacenza negativa}$$

$$\text{Entrate} = \sum \text{Variazione giacenza positiva}$$

$$\text{Indice di Rotazione} = \frac{\sum \text{Valore Uscite}}{\text{Valore Giacenza media}}$$

- Utilizzo dello script per creare le misure di time intelligence riferite ai KPI calcolati per analisi temporali avanzate;

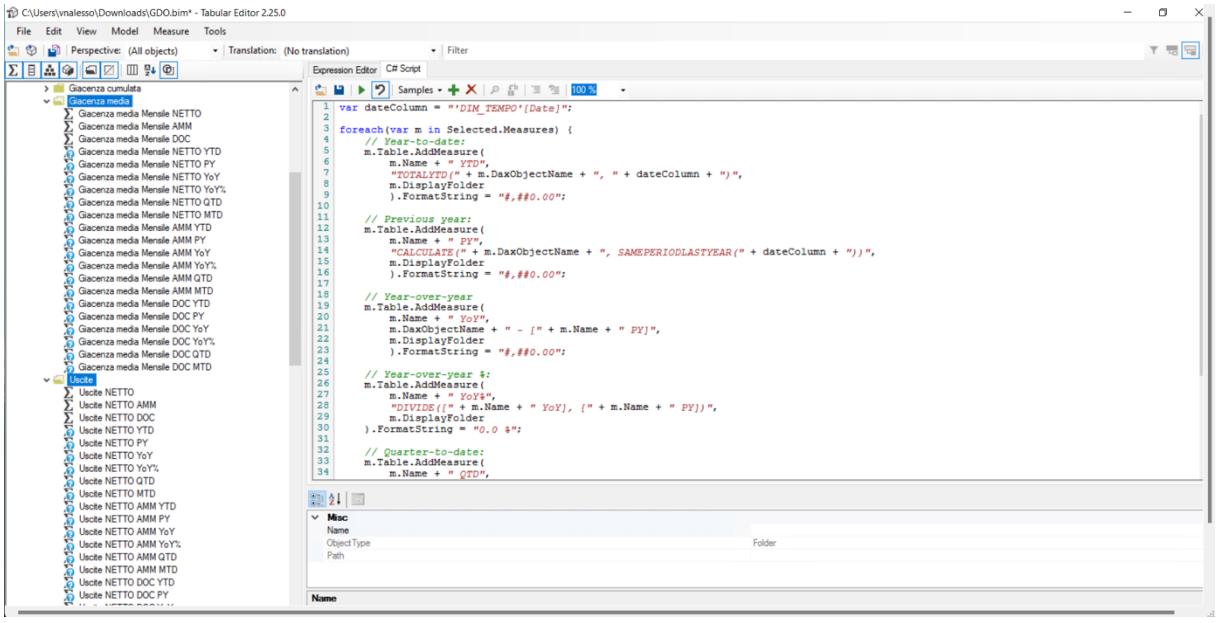

Figura 34. Creazione automatizzate delle time intelligence measures dei KPI

- Creazione della tabella *Analisi finanziaria* per supportare un'analisi più dinamica. Questa tabella contiene le tre diverse tipologie di rendicontazione (amministrativa, da documento aziendale, netta), permettendo all'utente di selezionare rapidamente la modalità di visualizzazione desiderata nella dashboard. Poiché questa tabella non ha una dipendenza diretta con FACT_GIACENZE, non è stata creata alcuna relazione diretta, ma viene utilizzata come filtro indipendente.
- Implementazione di misure aggiuntive derivate dai KPI esistenti per abilitare la trasformazione dinamica dei valori visualizzati, in modo da adattare l'analisi ai diversi contesti finanziari selezionati.

(per maggior dettagli sulle formule DAX utilizzate, si veda Appendice – Allegato A)

Il modello è stato completato mediante operazioni volte a migliorare la leggibilità dei dati, come l'inattivazione delle colonne funzionali non necessarie per l'analisi e la ridenominazione delle tabelle per rendere la navigazione più intuitiva. Tali modifiche sono state realizzate attraverso l'uso di script dedicati alle operazioni funzionali.

Alcune delle operazioni effettuate possono anche venire implementate tramite il pacchetto ufficiale di Best Practice Rules fornito da Microsoft, il quale include regole quali il controllo dell'esistenza delle relazioni, la possibilità di modificare il formato delle colonne secondo criteri specifici, la rimozione delle colonne superflue e l'adozione di una naming convention strutturata.

Deploy su Analysis Services e Integrazione con Power BI

Dopo aver completato la configurazione e l'ottimizzazione del modello, si procede con il deploy su Analysis Services. Il deploy consiste nel pubblicare il modello tabulare su un server, rendendolo disponibile per strumenti di reporting come Power BI.

Una volta eseguito il deploy, viene avviata la *process* del modello. La process è un'operazione che carica i dati dal database sorgente, popolando il modello tabulare con i valori aggiornati. Esistono diversi tipi di process, tra cui:

- Process Full: ricarica interamente il modello, utile quando vengono aggiunte nuove tabelle o modificati i metadati.

- Process Data: aggiorna solo i dati senza modificare la struttura del modello.
- Process Recalc: ricalcola le misure e le colonne calcolate senza ricaricare i dati.

Una volta completata la process, il modello è pronto per essere utilizzato in Power BI. L'utente può connettersi ad Analysis Services direttamente da Power BI tramite la funzione di Live Connection, sfruttando il modello tabulare per creare dashboard interattive, report avanzati e analisi dettagliate, garantendo prestazioni elevate e dati sempre aggiornati.

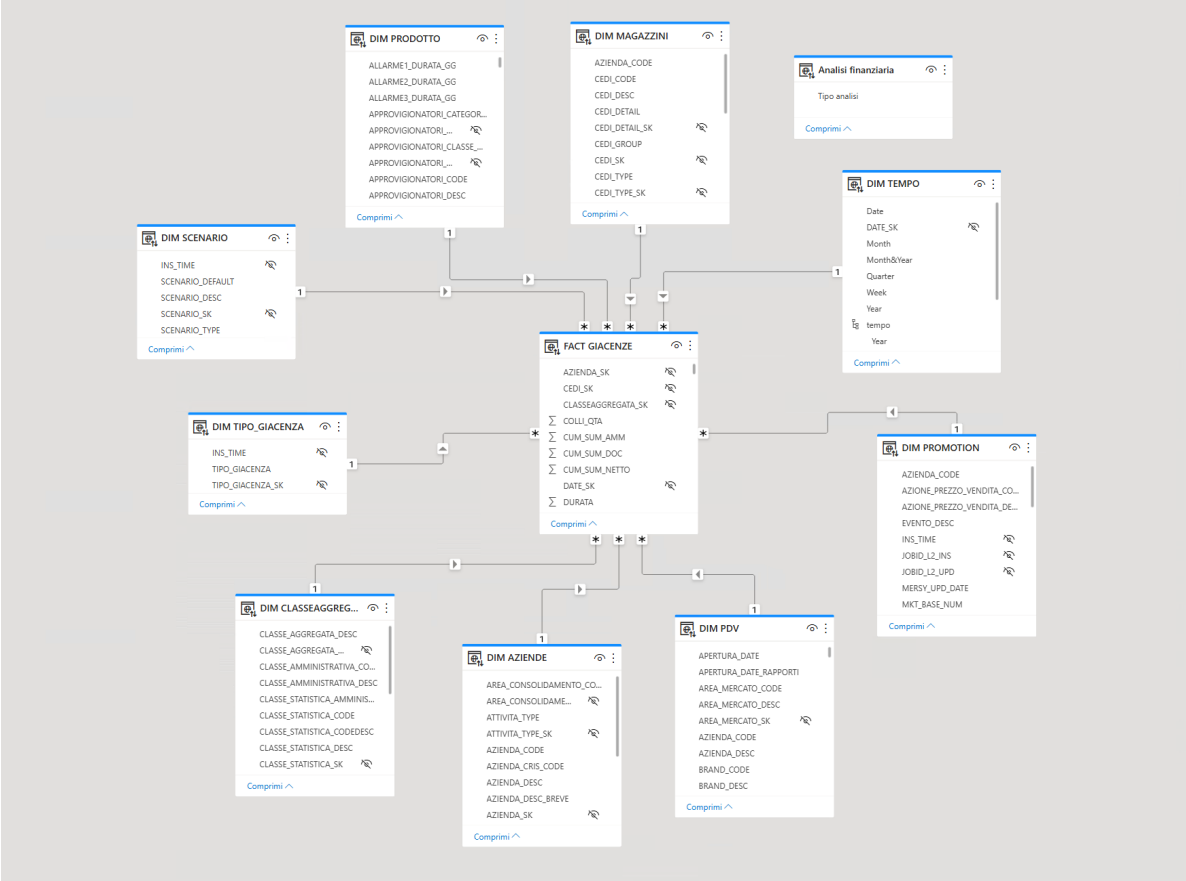

Figura 35. Modello Star Schema – case study

5.3 Dashboard

Il modello attualmente implementato in SSAS può essere integrato con una moltitudine di report realizzati con Power BI e altri strumenti di visualizzazione, garantendo così elevata flessibilità e centralità. Nel caso specifico, sono state create delle dashboard basate su questo modello, pur essendo solo un esempio di ciò che i software utilizzati possono realizzare.

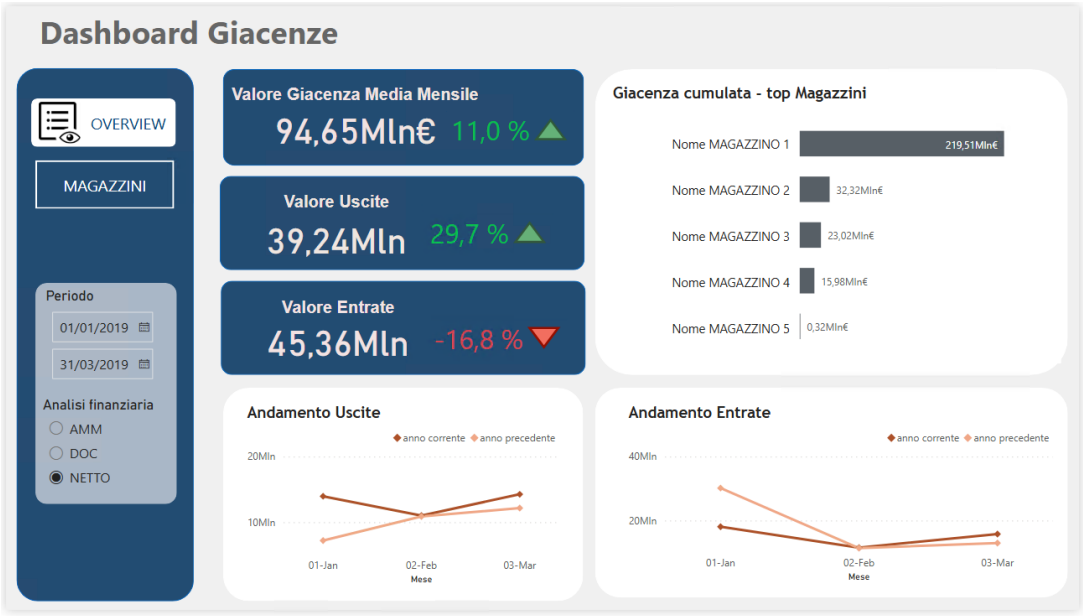

Figura 36. Dashboard – Overview

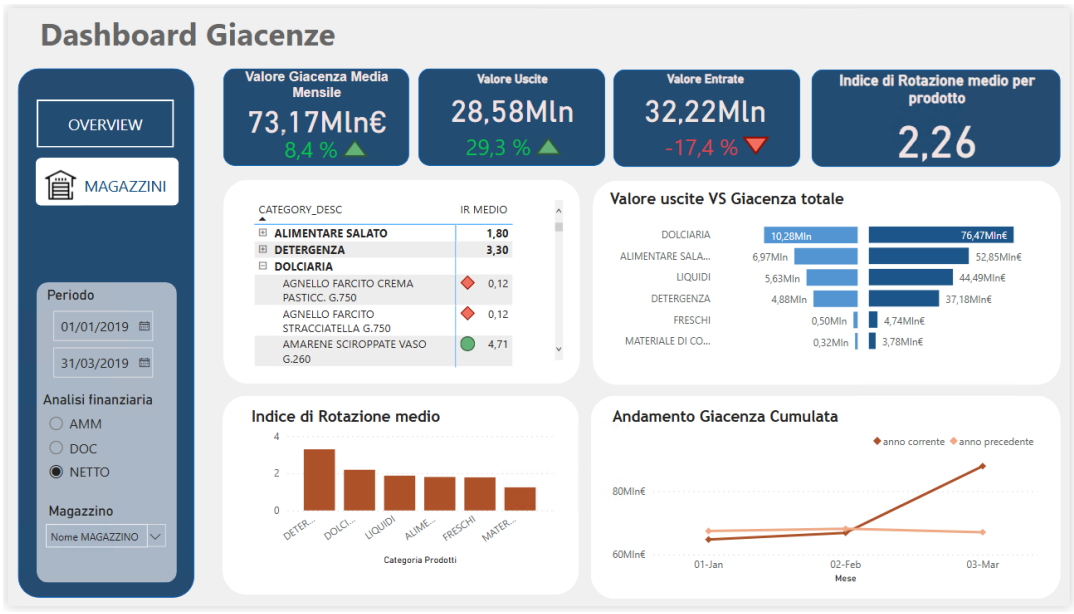

Figura 37. Dashboard – Dettaglio Magazzini

Una dashboard di gestione delle giacenze fornisce un quadro chiaro sull'andamento delle scorte di magazzino, consentendo di monitorare diversi indicatori chiave per ottimizzare la logistica e la distribuzione. Sono stati considerati i seguenti KPI:

- Valore della giacenza media mensile: consente di comprendere se l'azienda sta mantenendo livelli di stock adeguati o se vi sono eccessi o carenze che potrebbero influire sulle operazioni;
- Andamento entrate/uscite del magazzino: permette di valutare il flusso dei prodotti, evidenziando eventuali squilibri tra approvvigionamento e vendita;
- Indice di Rotazione dei prodotti: aiuta a distinguere tra articoli a elevata domanda e prodotti che rimangono troppo tempo in giacenza, indicando possibili azioni correttive come promozioni, riduzione degli ordini o sostituzione con articoli più performanti.

La pagina *OVERVIEW* del report consente, inoltre, di individuare quali magazzini detengono la maggior parte dello stock, permettendo di ottimizzare la distribuzione delle scorte per evitare concentrazioni inutili o rischi di esaurimento nei punti vendita. La pagina *DETTAGLIO MAGAZZINI*, invece, effettua un focus sul singolo centro di distribuzione, evidenziando i prodotti critici caratterizzati da un indice di rotazione minore di 1. L'indice di Rotazione (IR), infatti, è un KPI essenziale per valutare l'efficienza della gestione delle scorte. Questa misura consente di analizzare con precisione la velocità di rotazione delle scorte, supportando decisioni strategiche di gestione del magazzino. Infine, il confronto con i dati storici permette di individuare trend di miglioramento o peggioramento, fornendo un supporto per decisioni strategiche basate su dati concreti, al fine di ridurre i costi di gestione e migliorare la disponibilità dei prodotti sul mercato.

Le dashboard di Power BI consentono un'interazione avanzata e dinamica con i dati, migliorando l'esplorazione e l'analisi delle informazioni. Grazie alle funzionalità di interazione tra i vari elementi visivi, selezionando un valore in un grafico, tutti gli altri si aggiornano automaticamente per riflettere la scelta effettuata. Questo meccanismo permette di analizzare i dati da diverse prospettive, approfondire un determinato campo e identificare correlazioni in modo rapido e intuitivo, senza la necessità di applicare filtri manualmente o navigare tra diverse pagine di report.

Da queste dashboard l'azienda cliente può estrapolare informazioni strategiche per il processo decisionale, che supportano le politiche aziendali volte all'ottimizzazione degli stock e al miglioramento dell'efficienza operativa. Le informazioni consentono di monitorare la riduzione delle giacenze per prevenire rotture di stock, mantenendo un equilibrio tra disponibilità e domanda: lo stock, infatti, non rappresenta solo il capitale investito in merci, ma è anche un indicatore della capacità dell'azienda di rispondere alle variazioni del mercato. Allo stesso modo, particolare attenzione va posta ai valori dell'indice di rotazione (IR), che misura la frequenza con cui le scorte vengono vendute e rinnovate; un IR elevato indica una gestione efficiente delle giacenze, mentre un valore basso può segnalare problematiche di overstocking e immobilizzazione del capitale. Pertanto, diviene essenziale ridurre gli acquisti di prodotti a bassa rotazione, valutandone eventualmente promozioni o eliminazioni, e confrontare l'andamento delle uscite dal magazzino con quello delle vendite per identificare eventuali disallineamenti tra la logistica interna e la domanda di mercato, consentendo così di intraprendere azioni correttive mirate per migliorare la sincronizzazione tra rifornimenti e vendite. Anche l'andamento delle entrate riveste un ruolo fondamentale nel garantire un flusso costante e coordinato di rifornimenti, essenziale per mantenere un equilibrio ottimale tra disponibilità e domanda. Un trend positivo nelle entrate suggerisce una gestione efficiente degli ordini e una collaborazione stabile con i fornitori, permettendo all'azienda di anticipare le necessità del mercato senza sovraccaricare le giacenze. Al contrario, diminuzioni o fluttuazioni improvvise possono indicare problematiche nella catena di fornitura o strategie di rifornimento poco allineate con la domanda effettiva, rischiando così di causare stock-out o un eccessivo accumulo di prodotti. Analizzando l'andamento delle entrate in sinergia con altri indicatori come uscite e giacenze, l'azienda può individuare criticità, ottimizzare la pianificazione degli acquisti e migliorare la resilienza della propria supply chain, garantendo così continuità operativa e competitività sul mercato.

Conclusioni

Il presente lavoro di tesi intende approfondire il processo di modellazione dei dati nel contesto della Business Intelligence e della Data Analysis, evidenziando come l'utilizzo di metodologie automatizzate possa migliorare l'efficienza e la qualità delle analisi aziendali. L'obiettivo principale è quello di ottimizzare la creazione e il mantenimento dei modelli tabulari in Power BI, sfruttando le potenzialità di Tabular Editor per automatizzare operazioni fondamentali quali la creazione di relazioni tra tabelle, la formattazione delle colonne e la definizione di misure DAX attraverso script in C#. Questa automazione accelera il processo di sviluppo, riducendo il tempo necessario per attività ripetitive e meccaniche. Di conseguenza, si ottiene una maggiore flessibilità delle risorse, migliorando la profittabilità complessiva dei progetti.

I risultati ottenuti dimostrano che l'automazione, implementata con un approccio generalizzato e adattabile a molteplici contesti aziendali, comporti benefici significativi, quali la riduzione dei tempi e dei costi di sviluppo. Tali successi permettono agli analisti di focalizzarsi sulla personalizzazione del modello in base alle specifiche esigenze dei clienti, garantendo così una maggiore precisione nell'interpretazione dei dati. Il case study presentato in questo lavoro ha dimostrato l'efficienza del metodo proposto nella generazione di uno star schema (da un database preesistente) e nella conseguente creazione di dashboard interattive e intuitive tramite Power BI. La qualità e l'organizzazione del modello tabulare rappresentano infatti un elemento essenziale per la costruzione di una dashboard efficace, in quanto garantiscono una gestione strutturata delle informazioni e una navigabilità chiara e immediata. A testimonianza dell'efficacia del processo standardizzato, una indagine interna è stata effettuata nel team di Data Visualization riguardo i tempi medi di sviluppo di una dashboard per piccole-medie aziende. Il confronto tra il processo standardizzato e quello manuale è sintetizzato nel seguente grafico:

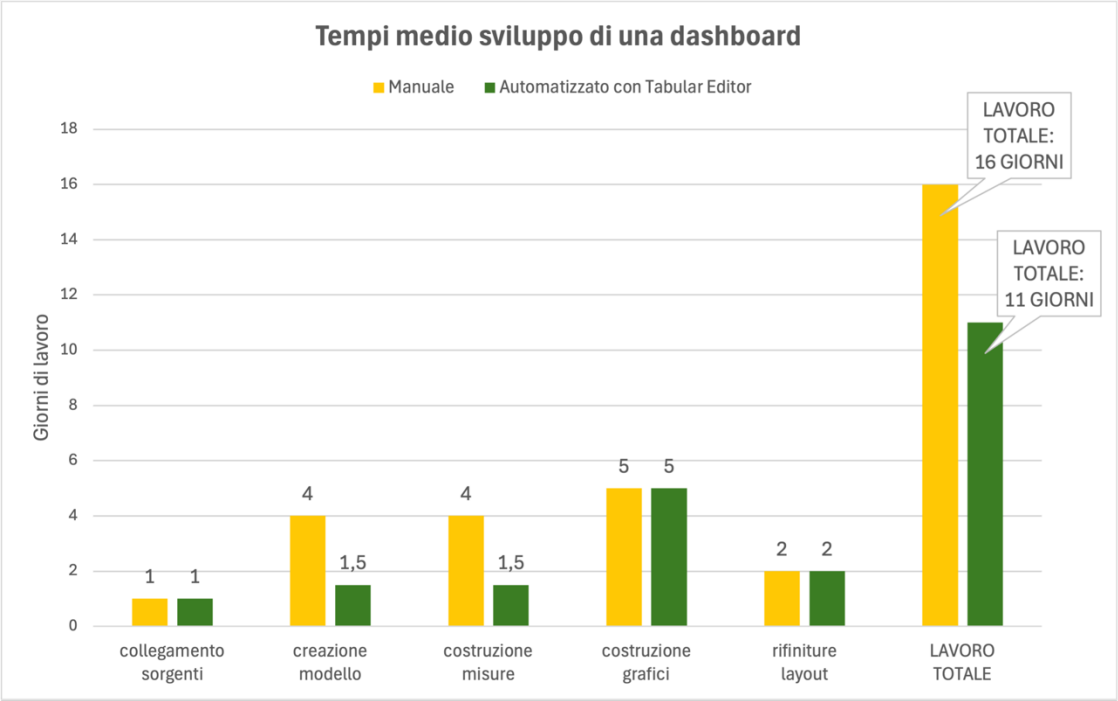

Figura 38. Grafico di confronto dei tempi medi di sviluppo di una dashboard

Dal grafico presentato emerge come l'adozione del metodo semi-automatizzato per un case study semplice riduca il tempo medio di sviluppo di circa il 30%. Nel caso di progetti con modelli più complessi e richieste più articolate da parte del cliente, l'approccio automatizzato potrebbe comportare una riduzione maggiore dei tempi di sviluppo (>30%) e un incremento della redditività complessiva progettuale.

L'approccio proposto permette quindi la standardizzazione dell'intero processo, riducendo il margine di errore e facilitando la manutenzione e l'aggiornamento dei modelli. Tale sistema crea le basi per future evoluzioni, aprendo prospettive di integrazione con tecniche di Machine Learning e analisi predittiva, oltre ad estendere l'applicazione della metodologia a diversi settori aziendali.

In sintesi, l'adozione di un processo automatizzato e personalizzato per la modellazione dei dati, basato su Tabular Editor e Power BI, si configura come una soluzione innovativa e competitiva per supportare le aziende nella trasformazione digitale. I risultati ottenuti confermano che investire in tecnologie capaci di ottimizzare il processo decisionale non solo migliora l'efficienza operativa, ma contribuisce anche a definire strategie aziendali più mirate ed efficaci in un mercato sempre più dinamico e complesso.

APPENDICE

A. Case study - formule DAX

Tabella **DIM_TEMPO**:

```
DIM_TEMPO =
ADDCOLUMNS(
    CALENDAR(
        DATE(2018,01,01),
        DATE(2019,04,30)
    ),
    "Year", YEAR([Date]),
    "Month", FORMAT([Date],"mm"),
    "Month&Year", FORMAT([Date],"mm") & "/" & YEAR([Date]),
    "Quarter", "Q" & FORMAT([Date],"Q"),
    "DATE_SK", YEAR([Date]) & FORMAT([Date], "mm")
                            & FORMAT([Date], "dd")
)
```

Tabella **Analisi finanziaria**:

```
DATATABLE(
    "Riferimento", STRING,
    {{"AMM"}, {"DOC"}, {"NETTO"}}
    )
```

Per facilità, vengono ora riportate la colonna calcolata e le misure riferite solo alla tipologia di analisi finanziaria “NETTO”. Le altre rendicontazioni risultano equivalenti.

Colonna calcolata **VARIAZIONE**:

```
var pdv = FACT_GIACENZE[pdv_sk]
var azienda = FACT_GIACENZE[azienda_sk]
var prodotto = FACT_GIACENZE[prod_sk]
var tipo = FACT_GIACENZE[tipo_giacenza_sk]
var classe = FACT_GIACENZE[classeaggregata_sk]
var current = FACT_GIACENZE[CUM_SUM_NETTO]

var previousDate = MAXX(FILTER(FACT_GIACENZE,
             FACT_GIACENZE[date_sk]<EARLIER(FACT_GIACENZE[date_sk]
                 && FACT_GIACENZE[classeaggregata_sk]=classe
                 && FACT_GIACENZE[pdv_sk]=pdv
                 && FACT_GIACENZE[azienda_sk]= azienda
                 && FACT_GIACENZE[prod_sk]= prodotto
                 && FACT_GIACENZE[tipo_giacenza_sk]=tipo),
           FACT_GIACENZE[date_sk])

var previous = MAXX(FILTER(FACT_GIACENZE,
                 FACT_GIACENZE[date_sk]=previousDate
                 && FACT_GIACENZE[classeaggregata_sk]=classe
                 && FACT_GIACENZE[pdv_sk]=pdv
                 && FACT_GIACENZE[azienda_sk]= azienda
                 && FACT_GIACENZE[prod_sk]= prodotto
                 && FACT_GIACENZE[tipo_giacenza_sk]=tipo),
           FACT_GIACENZE[CUM_SUM_NETTO])

return current-previous
```

Misura **Giacenza media mensile**:

```
CALCULATE (
      DIVIDE (SUM (FACT_GIACENZE[CUM_SUM_NETTO],
                 DISTINCTCOUNT (DIM_TEMPO[Month])
      ),
      FILTER (FACT_GIACENZE, FACT_GIACENZE[TIPO_GIACENZA_SK] = 2)
)
```

Misura **Sum_uscite**:

```
ABS(
   CALCULATE(
       SUM(FACT_GIACENZE[VARIAZIONI_NETTO]),
          FILTER(FACT_GIACENZE, FACT_GIACENZE[VARIAZIONI_NETTO]<0
                          && FACT_GIACENZE[TIPO_GIACENZA_SK]=2 )
      )
)
```

Misura **Sum_entrate**:

```
ABS(
   CALCULATE(
       SUM(FACT_GIACENZE[VARIAZIONI_NETTO]),
          FILTER(FACT_GIACENZE,'FACT_GIACENZE'[VARIAZIONI_NETTO]>0
                              && FACT_GIACENZE[TIPO_GIACENZA_SK]=2 )
      )
)
```

Misura **IR SETTIMANALE**:

```
DIVIDE([Sum of Uscite],
        CALCULATE(
            AVERAGE(FACT_GIACENZE[CUM_SUM_NETTO]),
            FILTER(FACT_GIACENZE, FACT_GIACENZE[TIPO_GIACENZA_SK]=2
                         && FACT_GIACENZE[CLASSEAGGREGATA_SK]=1494
            )
        )
)
```

Misura **IR MEDIO**:

```
AVERAGEX(
      SUMMARIZE('DIM_PRODOTTO', 'DIM_PRODOTTO'[PROD_DESC]),
                  [IR SETTIMANALE])
```

Bibliografia

[1] S. Zani e A. Cerioli, Analisi dei dati e data mining per le decisioni aziendali.

[2] F. La Noce e L. D'Ercole, Data warehousing. Dal dato all'informazione, Consiel, 2000.

[3] W. Inmon, Building the data warehouse, Wiley, 2002.

[4] B. Devlin, Data Warehouse: From Architecture to Implementation, Wesley, 1997.

[5] M. Ross e R. Kimball, The data warehouse toolkit, John Wiley & Sons, 2002.

[6] M. Ross e R. Kimball, Data Warehouse, teoria e pratica della progettazione, McGraw-Hill, 2005.

[7] M. Humphries, M. W. Hawkins e M. C. Dy, Data Warehousing: Architecture and implementation., Prentice Hall, 1999.

[8] C. Barbi, «ETL: significato, funzionamento e vantaggi,» Deda Tech, 10 July 2024. [Online]. Available: https://www.dedatech.com/blog/etl-significato-funzionamento-e-vantaggi.

[9] E. Núria, «From ETL to ELT or How Big Data Has Transformed ETL Processes,» *Bismart,* 2023.

[10] A. Unwin, «Why is Data Visualization Important? What is Important in Data Visualization?,» *Harvard Data Science Review,* 2020.

[11] S. Ilemann, «Visual versus Text: What does the brain prefer?,» 14 December 2020. [Online]. Available: https://simpleshow.com/blog/visual-versus-text/.

[12] M. Golfarelli e S. Rizzi, Data Warehouse: teoria e pratica della progettazione, McGraw-Hill, 2006.

[13] Erwin, «Modellazione dati logica,» [Online]. Available: https://www.erwin.com/it-it/solutions/data-modeling/logical.aspx.

[14] I. Amazon Web Services, «Modelli di dati logici e fisici - Differenza nella modellazione dei dati,» [Online]. Available: https://aws.amazon.com/it/compare/the-difference-between-logical-and-physical-data-model/.

[15] SAP, «Cos'è la modellazione dati?,» [Online]. Available: https://www.sap.com/italy/products/technology-platform/datasphere/what-is-data-modeling.html.

[16] E. F. Codd, «A relational model of data for large shared data banks,» *Communications of the ACM,* 1970.

[17] C. Gyorödi, R. Gyorödi e R. Sotoc, «A Comparative Study of Relational and Non-Relational Database Models in a Web-Enabled Application,» *International Journal of Advanced Computer Science and Applications,* 2015.

[18] M. R. Guarracino e S. Cuciniello, «Progettazione concettuale e logica di un data warehouse per dati genomici,» Consiglio Nazionale delle Ricerche Istituto di Calcolo e Reti ad Alte Prestazioni, Napoli, 2006.

[19] E. Thomsen, OLAP Solutions - Building multidimensional Information Systems, John Wiley & Sons, 2002.

[20] «Overview of OLAP cubes for advanced analytics,» Microsoft Learn, 1 November 2024. [Online]. Available: https://learn.microsoft.com/en-us/system-center/scsm/olap-cubes-overview?view=sc-sm-2025.

[21] G. Quattrone e D. Ursino, *Appunti per il Corso di Data Warehousing,* Università degli Studi Mediterranea di Reggio Calabria, 2007-2008.

[22] W. Lehner, H. Albrecht e H. Wedekind, «Normal forms for multidimensional databases,» in *Proceedings. Tenth International Conference on Scientific and Statistical Database Management*, 1998.

[23] M. Karmani, G. Mustafa, N. Sarwar, S. Wajid, J. Nasir e S. Siddque, «A Review of Star schema and Snowflakes Schema,» *Communications in Computer and Information Science,* 2020.

[24] E. Sidi, M. El Merouani e E. Abdelouarit, «Star Schema Advantages on Data Warehouse: Using Bitmap Index and Partitioned Fact Tables,» *International Journal of Computer Applications,* vol. 134, n. 13, 2016.

[25] C. Bazar e C. S. Iosif, «The Transition from RDBMS to NoSQL. A Comparative Analysis of Three Popular Non-Relational Solutions: Cassandra, MongoDB and Couchbase,» *Database Systems Journal,* vol. 5, n. 2, 2014.

[26] N. Leavitt, «Will NoSQL databases live up to their promise?,» *IEEE Computer Society,* vol. 43, n. 2, 2010.

[27] K. K. Tirupati, A. Joshi, D. S. Singh, A. Chhapola, S. Jain e D. A. Gupta, «Leveraging Power BI for Enhanced Data Visualization and Business Intelligence,» *Universal Research Reports,* June 2023.

[28] Microsoft Learn, «What is Power BI,» 22 March 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview.

[29] Microsoft Learn, «What is the Power BI service?,» 7 August 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-service-overview.

[30] Microsoft Learn, «What are the Power BI mobile apps?,» 26 July 2024. [Online]. Available: https://learn.microsoft.com/it-it/power-bi/consumer/mobile/mobile-apps-for-mobile-devices.

[31] Microsoft Learn, «What is Power Query?,» 27 January 2025. [Online]. Available: https://learn.microsoft.com/en-us/power-query/power-query-what-is-power-query.

[32] Microsoft Learn, «What is Power BI Report Server?,» 24 June 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/report-server/get-started.

[33] Microsoft Learn, «Connectors in Power Query,» 23 January 2025. [Online]. Available: https://learn.microsoft.com/en-us/power-query/connectors/.

[34] Microsoft Learn, «Semantic model modes in the Power BI service,» 10 November 2023. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/connect-data/service-dataset-modes-understand#import-mode.

[35] Microsoft Learn, «Live connection and DirectQuery comparison,» 13 May 2024. [Online]. Available: https://learn.microsoft.com/en-us/power-bi/connect-data/service-live-connect-dq-datasets.

[36] Microsoft Learn, «Creare e gestire le relazioni in Power BI Desktop,» 1 October 2024. [Online]. Available: https://learn.microsoft.com/it-it/power-bi/transform-model/desktop-create-and-manage-relationships.

[37] Superior consulting services, «Benefits of using Tabular Editor 2 for working with tabular data models,» 8 July 2024. [Online]. Available: https://www.teamscs.com/superior-spotlight-blogs/benefits-of-using-tabular-editor-2-for-working-with-tabular-data-models.

[38] Tabular Editor Documentation, «Advanced Scripting,» [Online]. Available: https://docs.tabulareditor.com/te2/Advanced-Scripting.html.

[39] Tabular Editor Documentation, «Best Practice Analyzer,» [Online]. Available: https://docs.tabulareditor.com/te2/Best-Practice-Analyzer.html.