# **SOMMARIO**

[SOMMARIO 1](#_Toc12980964)

[*INDICE DELLE FIGURE* 2](#_Toc12980965)

[*INDICE DELLE TABELLE* 4](#_Toc12980966)

[*CAPITOLO 2: TRADITIONAL ETL PER LA CREAZIONE DELLA DATA MART* 5](#_Toc12980967)

[***2.1 OPEN SOURCE ETL SOFTWARE: TALEND*** 5](#_Toc12980968)

[***2.2 CREAZIONE DELLA DATA MART*** 9](#_Toc12980969)

[**2.2.3 Delta Dei Dati** 11](#_Toc12980970)

[**2.2.2 Storicizzazione** 12](#_Toc12980971)

[**2.2.3 Modello Multidimensionale - DFM** 13](#_Toc12980972)

[***2.3 LEVEL L0 - DATA INGESTION*** 15](#_Toc12980973)

[**2.3.1 Data Ingestion** 15](#_Toc12980974)

[**2.3.2 I Metadati** 17](#_Toc12980975)

[***2.4 LEVEL L1 - DATA OPERATION*** 22](#_Toc12980976)

[**2.4.1 Data Quality** 23](#_Toc12980977)

[**2.4.2 Tmap Component In Talend Open Source** 25](#_Toc12980978)

[***2.5 LEVEL L2 – SNOWFLAKE MART BEST PRACTICE*** 27](#_Toc12980979)

[**2.6.1 Snowflake Schema** 33](#_Toc12980980)

[***2.6 LEVEL L2 – STARSCHEMA MART BEST PRACTICE*** 35](#_Toc12980981)

[**2.6.1 Star Schema** 37](#_Toc12980982)

[***2.7 GENERAL ETL SCHEMA & AUDIT*** 38](#_Toc12980983)

[*CAPITOLO 3: DATA MINING & MACHINE LEARNING NEL FASHION RETAIL* 40](#_Toc12980984)

[***3.1 Clustering: K-Means*** 42](#_Toc12980985)

[***3.2 Classificazione: Decison Tree*** 42](#_Toc12980986)

[***3.3 Association Rules: Apriori*** 42](#_Toc12980987)

[*CAPITOLO 4: DATA VISUALIZATION* 43](#_Toc12980988)

[***4.1 REPORT*** 43](#_Toc12980989)

[***4.2 DATA VISUALIZATION BY MICROSOFT POWER BI*** 44](#_Toc12980990)

[*~~CONCLUSIONI~~* 45](#_Toc12980991)

[***~~Results~~*** 45](#_Toc12980992)

[***~~Future Enviroments~~*** 45](#_Toc12980993)

[*REFERENCES* 46](#_Toc12980994)

# ***INDICE DELLE FIGURE***

[Figura 16: Level Tree 10](file:///C:\Users\Admin\Desktop\Mediamente%20consulting\TESI\STESURA%20TESI\TESI%20-%20Copia.docx#_Toc12980999)

[Figura 17: Hypercube OLAP 13](#_Toc12981000)

[Figura 18: Connessione al Database 16](#_Toc12981001)

[Figura 19: Caricamento da file Delimited 18](#_Toc12981002)

[Figura 20: Caricamento da file Excel 18](#_Toc12981003)

[Figura 21: Multi-Caricamento 20](#_Toc12981004)

[Figura 22: Mono-Caricamento in L0 20](#_Toc12981005)

[Figura 23: Mono-caricamento in L1 21](#_Toc12981006)

[Figura 24: Pre-Loading L1 22](file:///C:\Users\Admin\Desktop\Mediamente%20consulting\TESI\STESURA%20TESI\TESI%20-%20Copia.docx#_Toc12981007)

[Figura 25: Join & Mapping in Tmap 24](file:///C:\Users\Admin\Desktop\Mediamente%20consulting\TESI\STESURA%20TESI\TESI%20-%20Copia.docx#_Toc12981008)

[Figura 26: Trasformation in Tmap 25](file:///C:\Users\Admin\Desktop\Mediamente%20consulting\TESI\STESURA%20TESI\TESI%20-%20Copia.docx#_Toc12981009)

[Figura 27: SnowflakeDB Surrogate Key 29](#_Toc12981010)

[Figura 28: SnowflakeDB Dimension 30](#_Toc12981011)

[Figura 29: SnowflakeDB FACT 31](#_Toc12981012)

[Figura 30: Snowflake Schema 33](#_Toc12981013)

[Figura 31: Job Product Star Schema 34](#_Toc12981014)

[Figura 32: tMap Product Star Schema 35](#_Toc12981015)

[Figura 33: Star Schema 36](#_Toc12981016)

[Figura 34: Applicazione del data mining per il marketing 41](#_Toc12981017)

# ***INDICE DELLE TABELLE***

[Tabella 3: DATAWAREHOUSE vs DATA MART 8](#_Toc12980995)

[Tabella 4: Livelli ETL 9](#_Toc12980996)

[Tabella 5: Metadati 15](#_Toc12980997)

[Tabella 6: Data Quality 23](#_Toc12980998)

# ***CAPITOLO 2: TRADITIONAL ETL PER LA CREAZIONE DELLA DATA MART***

Il manifestarsi delle prime necessità di dati integrati ha portato le aziende ad affrontare il problema internamente in quanto il mercato non sapeva offrire soluzioni sufficientemente flessibili ed affidabili. Per questo il primo approccio per rispondere alle necessità di avere dati integrati fu quello di sviluppare internamente all’azienda software ad hoc soprattutto per eseguire le fasi di estrazione, trasformazione e caricamento dei dati in un ambiente unico e integrato. Nonostante i recenti progressi dei prodotti di Data Integration ancora oggi la maggior parte delle aziende utilizza soluzioni ETL personalizzate per rispondere alle necessità di integrazione.

Tuttavia, le più recenti evoluzioni del mercato hanno portato ad un aumento della domanda di prodotti completi di Data Integration, portando al 60% la percentuale delle imprese che utilizza uno dei pacchetti di prodotti di integrazione offerti sul mercato, con lo scopo di effettuare attività di Business Intelligence [8].

La recente crisi economica ha portato inoltre ad una diminuzione dei budget assegnati allo sviluppo dell’Information Technology nelle aziende, determinando un incremento dell’adozione di soluzioni di integrazione open source.

Si può quindi affermare che il mercato della Data Integration è oggi caratterizzato dalla convivenza di tre tipologie di prodotti:

* *Software personalizzati:* l’emergere delle prime necessità di integrazione dei dati molte imprese svilupparono internamente prodotti ad hoc in grado di rispondere alle esigenze specifiche del proprio ambito di business. Con la maturazione del mercato dei prodotti di Data Integration questo tipo di approccio è divenuto sempre meno conveniente. Inoltre, l’emergere di architetture SOA e applicazioni SaaS sta decretando la fine dei prodotti sviluppati in casa. Oggi le suite di Data Integration presenti sul mercato offrono sicuramente funzionalità e affidabilità migliori;
* *Software proprietari:* lo sviluppo di applicativi di Data Integration ha contribuito ad aumentare la produttività delle attività collegate all’integrazione dei dati. I prodotti di integrazione dei dati sono maturati costantemente negli anni garantendo un ventaglio di funzionalità sempre più ricco e variegato, rendendo tali applicativi idonei a supportare la grande maggioranza degli scenari di business che richiedono l’utilizzo di dati integrati. Il numero di applicativi sul mercato è oggi elevato, si va dalle suite di prodotti in grado di coprire la quasi totalità delle necessità aziendali a prodotti specializzati in particolari contesti di business o specifiche problematiche;
* *Software open source:* il limite dei maggiori prodotti proprietari presenti sul mercato sono i costi necessari per la loro implementazione. Per venire in contro alle necessità delle aziende più piccole e con risorse limitate si sono da poco affacciate sul mercato i primi prodotti open source, prodotti in grado di supportare una discreta quantità di funzioni ma con un costo decisamente minore rispetto ai prodotti proprietari (costi di licenza nulli, costi di infrastruttura ridotti, servizi pagati in base all’utilizzo).

Per la mia Tesi ho deciso di utilizzare un Software Open Source che offre tutte le funzionalità necessarie per lo svolgimento, minimizzando al massimo i costi: Talend.

## ***2.1 TALEND OPEN SOURCE***

L’approccio open source di Talend prevede la disponibilità di due prodotti:

* *Talend Open Studio*: suite gratuita scaricabile gratuitamente con licenza open source (GPL). Talend Open Studio si presenta come prodotto di Data Integration completo e contraddistinto da un’ ampia gamma di funzionalità, sufficienti per la maggior parte delle necessità;
* *Talend Integration Suite:* è una versione potenziata del prodotto gratuito che aggiunge funzionalità avanzate come lo sviluppo collaborativo, monitoraggio avanzato del progetto e il Data Masking.

Per chi non possiede l’hardware necessario per supportare il sistema c’è una terza opzione costituita da Talend On Demand, ovvero un’ offerta di tipo Software ad a Service (SaaS).

I prodotti di Talend offrono ad oggi le seguenti funzionalità:

* Ambiente di sviluppo user-friendly (basato sulla piattaforma Eclipse).
* Elevato numero di connessioni preimpostate.
* Deposito comune dei metadati.
* Supporto allo sviluppo collaborativo.
* Servizi di trasformazione di dati.
* Funzionalità di monitoraggio dell’andamento dell’integrazione.
* Data Profiling e Data Quality.

Vediamo quindi quali sono i punti di forza dell’approccio di open source di Talend [7]:

* *Nessuna barriera all’adozione*: la disponibilità gratuita del prodotto di base rende praticamente immediata l’installazione del software. Talend supporta il cliente attraverso tutorial sull’utilizzo di base, inoltre è possibile fare affidamento ad una vasta comunità di utilizzatori.
* *Curva di apprendimento veloce*: il prodotto si presenta graficamente user-friendly l’interfaccia grafica è intuitiva e l’utilizzo delle funzionalità di base non richiede particolari addestramenti.
* *Modello di prezzi stabile e prevedibile*: i prodotti proprietari prevedono spesso costi elevati man mano che si espandono le funzionalità e le capacità del prodotto, con costi di licenza che aumentano all’aumentare delle macchine installate. Questo rende spesso difficile una corretta previsione dei costi nelle fasi iniziali del progetto, soltanto a lavoro ultimato è possibile rendersi conto del costo effettivo della soluzione adottata. Talend prevede un modello di costo basato sul numero di sviluppatori e sull’utilizzo del servizio, indipendente da licenze, hardware e quantità di dati da integrare.
* *L’importanza di una comunità a supporto*: la comunità online di esperti ed utilizzatori del prodotto è già oggi molto vasta ed è un fattore di grande importanza per facilitare l’implementazione e il mantenimento delle soluzioni offerte da Talend. Forum, wiki, guide e contributi gratuiti degli utenti rappresentano un valore aggiunto che solo un prodotto di questo tipo può offrire.
* *Ampio supporto a tipologie di dati differenti*: con oltre 400 connessioni preimpostate la soluzione di Talend garantisce la compatibilità con un grande numero di sistemi, database, pacchetti di software, applicazioni gestionali, servizi web, ecc. Nessun’altra soluzione sul mercato vanta un numero di possibili connessioni così elevato.
* *Flessibilità, versatilità e riuso del prodotto*: Talend non si limita ad un supporto alle tecniche standard di ETL ma permette l’implementazione di diverse strategie di integrazione. La possibilità di riuso di progetti già perfezionati costituisce inoltre un altro punto di forza dell’approccio open source.
* *Funzionalità e performance*: il livello di funzionalità offerto è paragonabile a quello dei prodotti proprietari. Tuttavia, si registrano alcune lacune nel campo della modellazione dei dati, data quality e data mining. Un team di ricerca e sviluppo dedicato permette al prodotto di essere sempre aggiornato alle ultime esigenze del mercato e di proporre funzionalità innovative.
* *Costi e tempi ottimizzati*: le soluzioni offerte da Talend risultano da un 50% a un 80% più economiche rispetto ai prodotti tradizionali, essendo meno costose da acquisire e mantenere e permettono uno sviluppo più rapido del sistema di integrazione.

## ***2.2 CREAZIONE DELLA DATA MART***

Nello specifico, un Data Mart è un database analitico progettato per incontrarsi con le esigenze specifiche di un’impresa. Essendo sottoinsieme logico o fisico di un Data Warehouse di dimensioni maggiori, segue le stesse regole di progettazione con dati aggregati a vari livelli di dettaglio, anche se, talvolta può essere costituito anche in assenza di un sistema di dati integrato [11].

Tabella 3: DATAWAREHOUSE vs DATA MART

|  |  |  |
| --- | --- | --- |
|  | **Data Warehouses** | **Data Marts** |
| Finalità | Application-neutral  Centralizzati e condivisi  Intera impresa | Applicazioni specifiche  Dipartimenti o aree |
| Dati | Bassa denormalizzazione | Alta denormalizzazione |
| Soggetti utilizzatori | Soggetti di molte aree | Soggetti di una singola area |
| Sorgenti dei dati | Molte  Dati esterni operazionali | Poche  Dati esterni operazionali |
| Caratteristiche | Flessibile, estensibile  Lunga vita  Data-oriented | Ristretto, non estensibile  Vita breve  Project-orientation |
| Tempo d’implementazione | 9-18 mesi per il primo stadio | 4-12 mesi |

L’implementazione può essere di due tipi: *Top-Down,* costruzione del DWH, e conseguente aggregazione ed esportazione nei vari Data Mart, e *Bottom-Up,* concentrandosi su aree specifiche del business si costruiranno i vari Data Mart per poi giungere alla costruzione del DWH. In questo modo si avrà un approccio scalabile.

La fase di ETL del progetto Fashion\_Retail è basata sulla creazione di una Data Mart con implementazione *Top-Down* Data Mart con l’obiettivo di ottenere una tabella Fatto delle Immagine che contiene testo

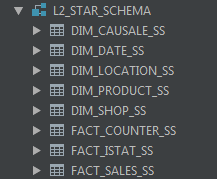
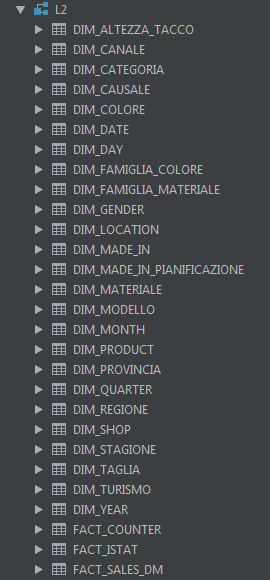
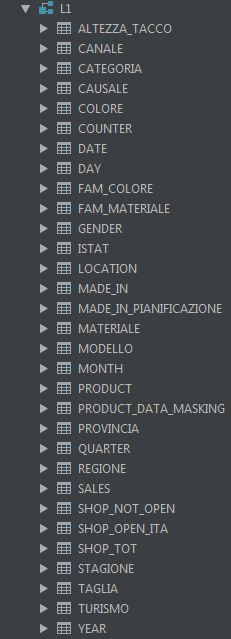
Descrizione generata con affidabilità elevata******vendite con dati specifici e elaborati per ricavarne successivamente informazioni che attualmente il cliente non conosce e che potrebbero portarlo a produrre un potenziale vantaggio competitivo ed economico. Ogni dato segue rigorosamente il sistema ACID (Atomicità, Coerenza, Isolamento e Durabilità).

Figura 18: Level Tree

Per facilitare il riconoscimento dei file, delle tabelle e per eventuali necessità di ricarico utilizzeremo una nomenclatura ferrea a livelli che renderà più semplice il riconoscimento del singolo file. Tali livelli dovranno poi risiedere in una struttura ad albero che consente la navigazione storica e concettuale all’interno delle cartelle.

Tabella 4: Livelli ETL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **L0** | **L1** | **L2** | **L2\_STARSCHEMA** |
| Definizione | Estraggo i dati da vari tipi di file senza trasforma-zioni. | Principali trasforma-zioni e le operazioni di Data Quality. | Molto veloce e tabelle snelle. uso di surrogate key per collegare i vari attributi. | Poche tabelle ma corpose, per avere tutti i dati neccessa-ri per i report. |
| Area | Staging/ Extraction Area. | Trasforma-tion Area. | ETL  Area. | Visualiza-tion Area. |
| Primary key | NO. | YES. | YES. | YES. |
| Surrogate key | NO. | YES. | YES. | NO. |
| Azione sulla Tabella | Truncate. | Nothing. | Nothing. | Nothing. |
| Azione sui dati | Insert. | Update/  Insert. | Update/  Insert. | Update/  Insert. |

***2.2.3 Delta Dei Dati***

L’alimentazione del DWH normalmente inizia con lo scatenare il processo FULL\_LOAD o Initial Load che prevede il popolamento iniziale di una situazione consistente di dati a una certa data da cui si può poi proseguire con i carichi delta.

Il carico della porzione delta dei dati può contemplare diverse situazioni:

* CDC (Change Data Capture): le tabelle vengono sottoposte a un meccanismo di change data capture che intercetta automaticamente per ogni tabella o meno il delta dei dati rispetto all’estrazione precedente e il processo nostro di replica verso L0 ottiene i dati già da caricare (consigliato per replicare le tabelle molto grandi in modo da non dover scaricare moli di dati inutili).
* MINUS: le tabelle sorgenti non vengono sottoposte a nessun meccanismo di rilevazione di cambiamenti e dobbiamo autonomamente estrarre la mole di dati e trovare il delta effettuando una minus dei dati che possediamo con quelli nuovi (consigliato per le tabelle di anagrafica che contengono una mole di dati nota e non prevedono cambiamenti eccessivi nella numerosità dei record).
* FULL: replica per intero giornaliera dei dati della tabella date le dimensioni contenute e la variabilità bassa del contenuto.

### ***2.2.2 Storicizzazione***

Le tabelle DLT possono aver una storicizzazione necessaria al ricalcolo del DWH, per evitare una perdita di dati o carico di dati parziali dovuta a problemi, di server per esempio.

Per aver una storicizzazione del dato abbiamo due scelte:

* creare tabelle ombra delle DLT partizionate per JOB\_ID di estrazione chiamate HIS. Le tabelle Delta DLT saranno non partizionate e in truncate/insert ( troncare la tabella mantenendo lo schema iniziale per poi inserire i nuovi dati) e conterranno solo i dati del JOB\_ID attuale mentre, le HIS, saranno in insert con il partizionamento per JOB\_ID, velocizzando il processo di estrazione.
* avere lo storico direttamente sulle DLT in insert con il partizionamento per JOB\_ID, sempre per velocizzare le estrazioni.

È possibile utilizzare una sola delle due modalità in modo da uniformare l’architettura delle tabelle a un modello unico e, a seconda della modalità utilizzata, sarà necessario differenziare il codice di un eventuale ricalcolo.

Nel progetto preso in considerazione, le tabelle Delta e la Storicizzazione non si utilizzeranno perché i dati derivano da file csv o Excel locali con nessuna possibilità di ricalcolo, non essendo collegati ad una sorgente con un costante aggiornamento giornaliero/mensile.

### ***2.2.3 Il Modello Multidimensionale - Dimensional Fact Model***

Il modello E-R (Modello Entità–Relazione), diffuso per progettare sistemi informativi relazionali, non è adatto per esprimere e analizzare in modo dettagliato grandi moli di dati [10].

Il modello multidimensionale o DFM (Dimensional Fact Model) è un modello concettuale dove è possibile rappresentare i dati all’interno di un ipecubo i cui spigoli rappresentano le dimensioni di analisi, che successivamente verrà suddiviso in tanti “cubetti”, ciascuno dei quali è identificato da una terna di coordinate. ogni cubetto contiene idealmente i valori assunti dalle misure per quella data terna e viene comunemente denominato “fatto” in quanto rappresenta l’accadimento di un evento di interesse per il dominio di business.

Un modello multidimensionale si basa principalmente su 4 concetti chiave:

* *Fatto:* concetto rilevante per il processo di Decision-Making. Tipicamente modella una specifica area di business (Vendite, Ordini, Produzione, etc.), ed è caratterizzato da una a più misure.
* *Misura:* rappresenta l’aspetto quantitativo del fatto che risulta di elevata importanza per l’analisi. Proprio dalle *Misure* vengono estratti dei *KPI (Key Performance Indicator)* che guideranno le imprese nelle proprie strategie di business. Alcuni esempi possono essere la Quantità prodotta, il Profitto, e il Prezzo.
* *Dimensione:* rappresenta le coordinate di analisi del *Fatto*. Tra queste possiamo trovare Data, Prodotto, Negozio.
* *Attributo Dimensionale:* è un raggruppamento logico di alcuni elementi di una stessa dimensione. Classi di elementi che consentono all'utente di selezionare i dati per specifiche caratteristiche.

Per navigare all’interno del cubo multidimensionale esistono differenti operazioni che permettono di organizzare i dati al suo interno, attraverso diverse prospettive [10].

La prima è il *Pivoting che* permette di modificare rapidamente la visualizzazione dei dati girando gli assi del cubo e ha lo scopo di cambiare il punto di vista da cui si analizza i dati del cubo. La seconda, invece, è lo *Slice* & *Dice che* seleziona e proietta i dati del cubo. Nello specifico si estrarranno sotto-cubi filtrando su una (Slice) o più (Dice) dimensioni. Infine, abbiamo il *Roll-Up* & *Drill-Down che* consentono di spostarsi all'interno di una gerarchia, scegliendo il livello di aggregazione secondo il quale l'utente desidera analizzare i dati. Nello specifico si salirà di un livello gerarchico con il roll- up, mentre si scenderà di un livello con il drill-down.

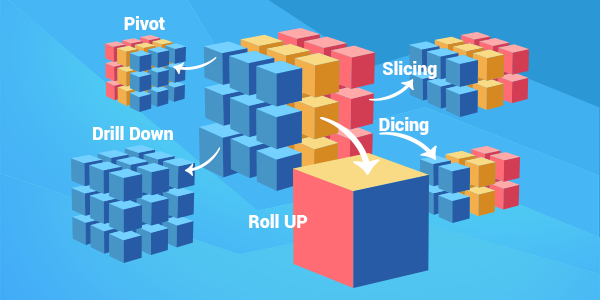


Figura 19: Hypercube OLAP

Gli scopi principali del sistema sono:

* Fornire supporto al design concettuale.
* Creare un ambiente dove gli utenti possano fare query in maniera intuitiva e formale.
* Favorire la comunicazione tra designer a utenti al fine di formalizzare i requisiti di progetto.
* Costruire una stabile piattaforma di design logico.
* Fornire una documentazione chiara e efficace.

## ***2.3 LEVEL L0 - DATA INGESTION***

Il livello L0 rappresenta la fase iniziale chiamata Staging or Extraction Area del DWH dove avviene lo scarico delle informazioni dai sistemi sorgenti.

I sistemi sorgente possono esser di diverso genere ma i più comuni sono i sistemi operazionali su database o i file prodotti dal fornitore:

* Da tabella: lettura via rete del set di dati giornaliero e replica dell’intero set di dati.
* Da file: lettura delle informazioni contenute nell’estrazione giornaliera.

La Data Ingestion è il processo di acquisizione e importazione di dati per l'uso o l'archiviazione immediata in un database.

I dati possono essere trasmessi in streaming in tempo reale o ingeriti in lotti:

Quando i dati vengono ingeriti in tempo reale, ogni elemento di dati viene importato mentre viene emesso dalla sorgente. Quando i dati vengono importati in batch, gli elementi di dati vengono importati in blocchi discreti a intervalli di tempo periodici. Un processo efficace di acquisizione dei dati inizia dando la priorità alle origini dati, convalidando i singoli file e indirizzando gli elementi di dati alla destinazione corretta.

Se esistono numerose fonti di dati di grandi dimensioni in diversi formati, può essere difficile per le aziende acquisire dati a una velocità ragionevole e elaborarli in modo efficiente al fine di mantenere un vantaggio competitivo. A tal fine, i fornitori offrono programmi software su misura per specifici ambienti di elaborazione o applicazione. Quando l'importazione dei dati è automatizzata, il software utilizzato per eseguire il processo può anche includere funzionalità di preparazione dei dati per strutturare e organizzare i dati in modo che possano essere analizzati dalla Business Intelligence (BI) e dalla Business Analytics (BA).

Le Tabelle che andremo a creare in questo livello saranno tutte precedute dal prefisso “STG”, da Staging Area, corrispondente all’importazione totale del documento di partenza con nessuna modifica allo schema sorgente, e con solo piccole trasformazioni dovute alla capienza delle variabili del Database di SQL Server usato per il progetto.

### **2.3.1 I Metadati**

Il termine “metadati” si applica ai dati usati per descrivere altri dati. 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.

La o le tabelle di metadati sono strettamente collegate al DWH vero e proprio e le applicazioni ne fanno un intenso uso sia dal lato dell’alimentazione che da quello dell’analisi.

È possibile distinguere due categorie di metadati, parzialmente sovrapposte, in base ai diversi utilizzi che ne fanno l’amministratore del sistema e gli utenti finali:

* Metadati interni: di interesse per l’amministratore, descrivono, le sorgenti, le trasformazioni, le politiche di alimentazione, gli schemi logici e fisici, i vincoli e i profili degli utenti.
* Metadati esterni: di interesse per gli utenti, riguardano, per esempio, le definizioni, la qualità, le unità di misura e le aggregazioni significative.

I metadati vengono memorizzati in un apposito contenitore al quale possono accedere tutti gli altri componenti dell’architettura.

Si possono classificare inoltre riguardo il livello in cui vengono considerati:

* Globali: contengono metadati relativi a tutti i livelli e a tutti i processi, e servono per sincronizzare le varie fasi su un livello comune temporale o di dettaglio.
* Processo: distinguiamo i metadati a seconda del sistema alimentante e del processo in cui vengono coinvolti. I metadati che descrivono il singolo processo relativo o meno a un determinato sistema (punto di sincronia interno tra le tabelle, percentuale propria del sistema di tolleranza errori) devono esser proprie per ogni sistema.

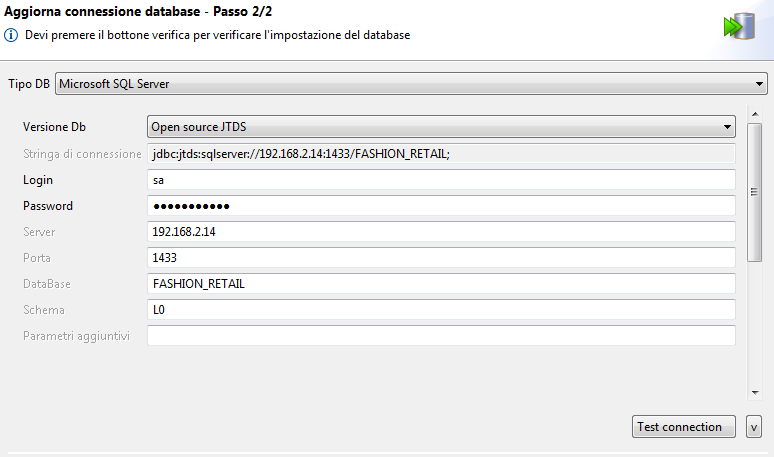
È possibile definire una reportistica sui metadati in quanto sono un punto ottimale per leggere e aver chiara la situazione in ogni istante per ogni processo. È utile avere un cruscotto dove è possibile leggere la sincronia tra i processi e i sistemi sorgente o di estrazione.

Il concetto di metadato è molto critico all’interno della gestione del DWH e viene spesso dibattuto se tenerlo interno al progetto o gestirlo con logiche esterne che svincolino la tecnologia e il prodotto utilizzato dallo scopo poi effettivo del metadato.

Normalmente registrato all’interno di una tabella, per motivi di fruibilità da parte dei più eterogenei sistemi, esso ha l’utilità di descrivere univocamente e in modo preciso un’informazione su che stato si trova un processo, al fine di evitare lanci di più istanze, lanciare un processo in un momento sbagliato, dire se il processo è terminato in modo corretto o con errori, fornire l’intervallo temporale per cui quel processo terminato o in esecuzione ha estratto i dati.

Tramite le funzionalità del software ETL Talend Open Studio è stato possibile importare vari tipi di file per costruire un nuovo Database (Data Mart) in SQL Server più semplice sulla macchina 192.168.2.14 chiamato FASHION\_RETAIL, Ma utile per valutare tutto il reparto vendite.

Per usare il database su Talend ho bisogno di creare una connessione al database per ogni livello, dove poi implementerò le Anagrafiche (insieme delle tabelle Dimensioni) e i Movimenti (insieme delle tabelle Fatto) tramite importazione dei metadati.



Server, Porta, e nome del Database per avviare la connessione

Versione del DB

Tipo di DB

Figura 20: Connessione al Database

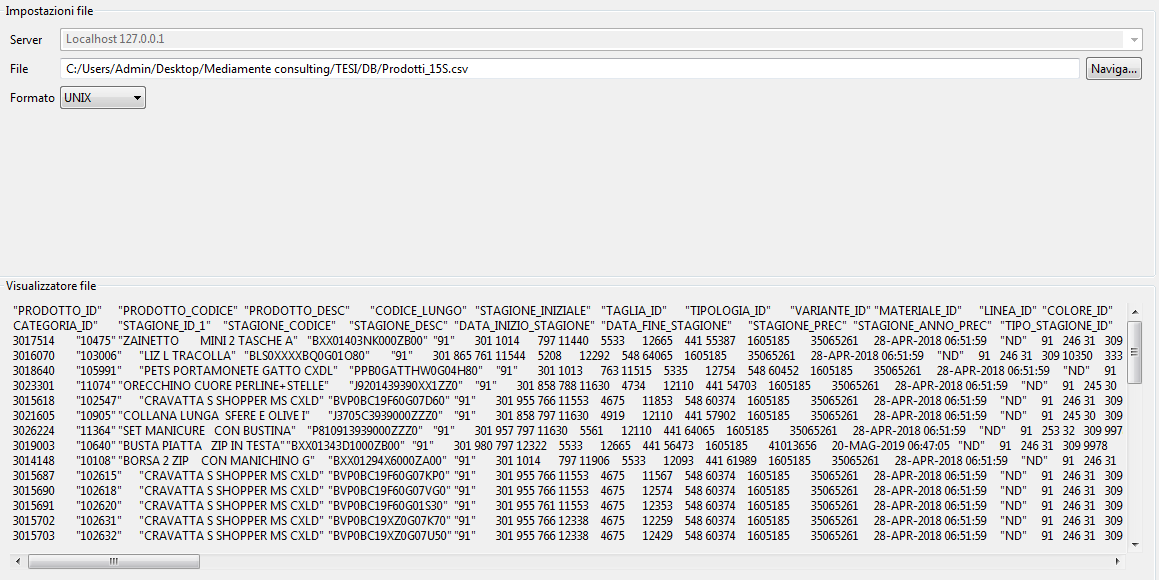
Nella tabella sottostante sono elencate le tabelle create a livello L0-Data Ingestion con i file sorgente estratti per il progetto, che sono principalmente di tipo Excel o Csv (Delimited file).

Tabella 5: Metadati

|  |  |  |
| --- | --- | --- |
| **Database Table** | **Metadata Type** | **Metadata Name** |
| STG\_Causale | File: .Csv | Causali |
| STG\_Made\_In | File: .Csv | Made\_In |
| STG\_Made\_In\_Pianificazione | File: .Csv | Made\_In\_Pianificazione |
| STG \_Product | File: .Csv | ProdottiS15, Prodotti<S15, Prodotti<S15n2 |
| STG \_Shop\_Open | File: .Excel | Negozi aperti |
| STG \_Shop\_Closed | File: .Excel | Negozi chiusi |
| STG\_Fam\_Colore | File: .Csv | Colori |
| STG\_Fam\_Materiale | File: .Csv | Materiali |
| STG \_Sales | File: .Csv | Scontrini, Scontrini2019 |
| STG \_Provincia | File: .Csv | Provincia |
| STG \_Counter | File: .Csv | Contapersone, Contapersone2019 |

Avendo una serie di file in Csv ed in Excel comporta una fase di importazione a livello di archiviazione nel software di sviluppo del processo ETL, che li ingloberà in cartelle, una per ogni tipo, sottoforma di metadati.

Nel progetto implementato in Talend, si avranno questi due tipi di visualizzazioni di caricamento:



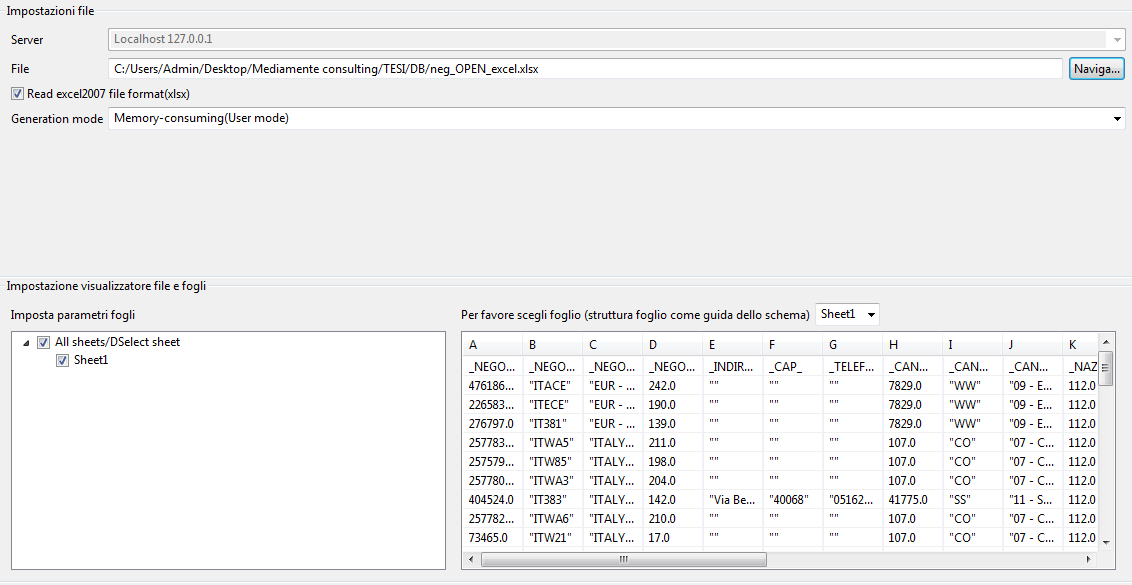
Rappresentazione dei dati nel CSV

File sorgente

Server

Figura 21: Caricamento da file Delimited

Server



File sorgente

Rappresentazione dei dati nel file EXCEL

Figura 22: Caricamento da file Excel

I Metadati, una volta importati, devono essere rielaborati per la creazione del database sul server. Concettualmente per ogni job creato in Talend, sono state effettuate le seguenti quattro operazioni:

* Importazione file csv o Excel.
* Unione dei file tramite lo strumento tUnite, se neccessario.
* Cambiamento e mappatura dei nomi, lunghezza o tipo degli attributi o unione di tabelle grazie alle chiavi primarie tramite lo strumento tMap.
* Creazione della tabella in SQL SERVER tramite lo strumento tDBOutput (tMSSQLOutput).

Per svolgere le seguenti operazioni è necessario creare un nuovo JOB, che conterrà i vari metodi di importazione.

Un esempio molto significativo, è rappresentato dalla creazione dello STG\_PRODUCT, la tabella di staging area della dimensione Prodotto.

In essa si può osservare come l’importazione di vari file vengono uniti tramite una Palette denominata tUnite, che cattura gli schemi dei file sorgenti per creane uno che si adatta a tutti; se gli schemi sorgenti sono differenti te lo segnala con un Warning, anche se il processo continua a funzionare regolarmente.

Il tool tMap vedi (2.2.4 Data Quality) permette di trasformare gli schemi di input per ottimizzare gli output (fase L1) e per creare delle relazioni di join tra le varie tabelle.

In questo specifico caso, aiuta ad identificare quali prodotti sono accettati secondo lo schema definito precedentemente e quali rigettati, con rispettivamente la creazione della staging area del prodotto e un file excel con i prodotti rifiutati.

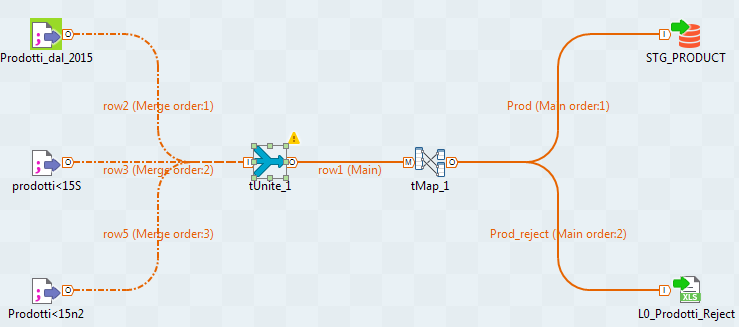


Figura 23: Multi-Caricamento

Lo stesso procedimento è stato svolto per le altre tabelle, con il caso più comune la diretta importazione del file nel database.

É importante osservare che ogni file occupa un Job (area di lavoro) diverso. Questo è necessario per prevenire errori durante il caricamento dei dati, isolando il problema.

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figura 24: Mono-Caricamento in L0

## ***2.4 LEVEL L1 - DATA OPERATION***

Il livello L1, comprende la Data Quality, la normalizzazione dei dati, e tutte le trasformazioni dei file sorgenti, e rappresenta sicuramente il cuore di tutto il processo di ETL.

A differenza del livello di estrazione L0, i dati sono estratti esclusivamente dalle tabelle create nella Staging Area elencate precedentemente, per poi essere successivamente trasformate e caricate nello stesso Database FASHION\_RETAIL, ma in uno schema differente, per valorizzarne il processo e avere un continuo controllo sulle attività che si svolgono.

Per le tabelle già presenti nella Staging Area il processo è molto semplice, in quanto si dovrà solo selezionare gli attributi nel tMap che mi interessano ed eseguire le trasformazioni adeguate per la Normalizzazione dei dati. Nel trasferimento, per un primo check di integrità, si creano le Primary Key, che andranno ad identificare univocamente l’attributo specifico di ogni tabella, spesso rappresentato da un campo ID.

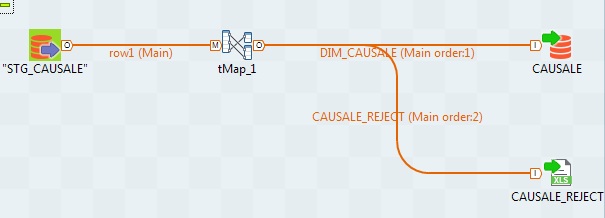


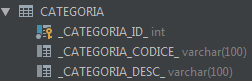
Figura 25: Mono-caricamento in L1

Importante fase è il Pre-Loading, cioè l’estrazione delle dimensioni secondarie da altre tabelle presenti nel livello L0.

Usando come input la tabella STG\_PRODUCT, per esempio, possiamo osservare la possibile formazione di nuove dimensioni e rispettivamente eseguire le queries per ciascuno di essi, estraendo univocamente, grazie alla funzione “SELECT DISTINCT” del linguaggio SQL, i campi dalla tabella sorgente relativi alla Categoria.

***~~Immagine che contiene testo, targa

Descrizione generata con affidabilità molto elevata~~***Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata***~~~~***

“SELECT DISTINCT

\_CATEGORIA\_ID\_,

\_CATEGORIA\_CODICE\_,

\_CATEGORIA\_DESC\_

FROM L0. STG\_PRODUCT “

Figura 26: Pre-Loading L1

### **2.4.1 Data Quality**

La qualità dei dati è una percezione o una valutazione dell'idoneità dei dati stessi utili per uno scopo in un determinato contesto. La qualità dei dati è determinata da fattori quali accuratezza, completezza, affidabilità, rilevanza e quanto è attuale. Poiché i dati sono diventati più strettamente collegati alle operazioni delle organizzazioni, l'enfasi sulla qualità dei dati ha guadagnato maggiore attenzione.

I Check si dividono principalmente in due tronconi:

* Di integrità referenziale: controlli tramite la verifica delle foreign key. Viene effettuato tramite join con le tabelle di L1 contenenti i padri di cui verificare le relazioni.
* Di validazione record: i dati devono subire controlli per scartare record che non soddisfano i requisiti stabiliti

1. Not Null.
2. Condizioni Simple/Complex (es.: data compresa in intervalli o campi testo di lunghezza definita ecc…)

La tabella seguente illustra (uno per tipo) tutte le operazioni di Data Quality eseguite durante la progettazione della DataWareHouse, eseguite tutte ne tool tMap o direttamente nei metadati:

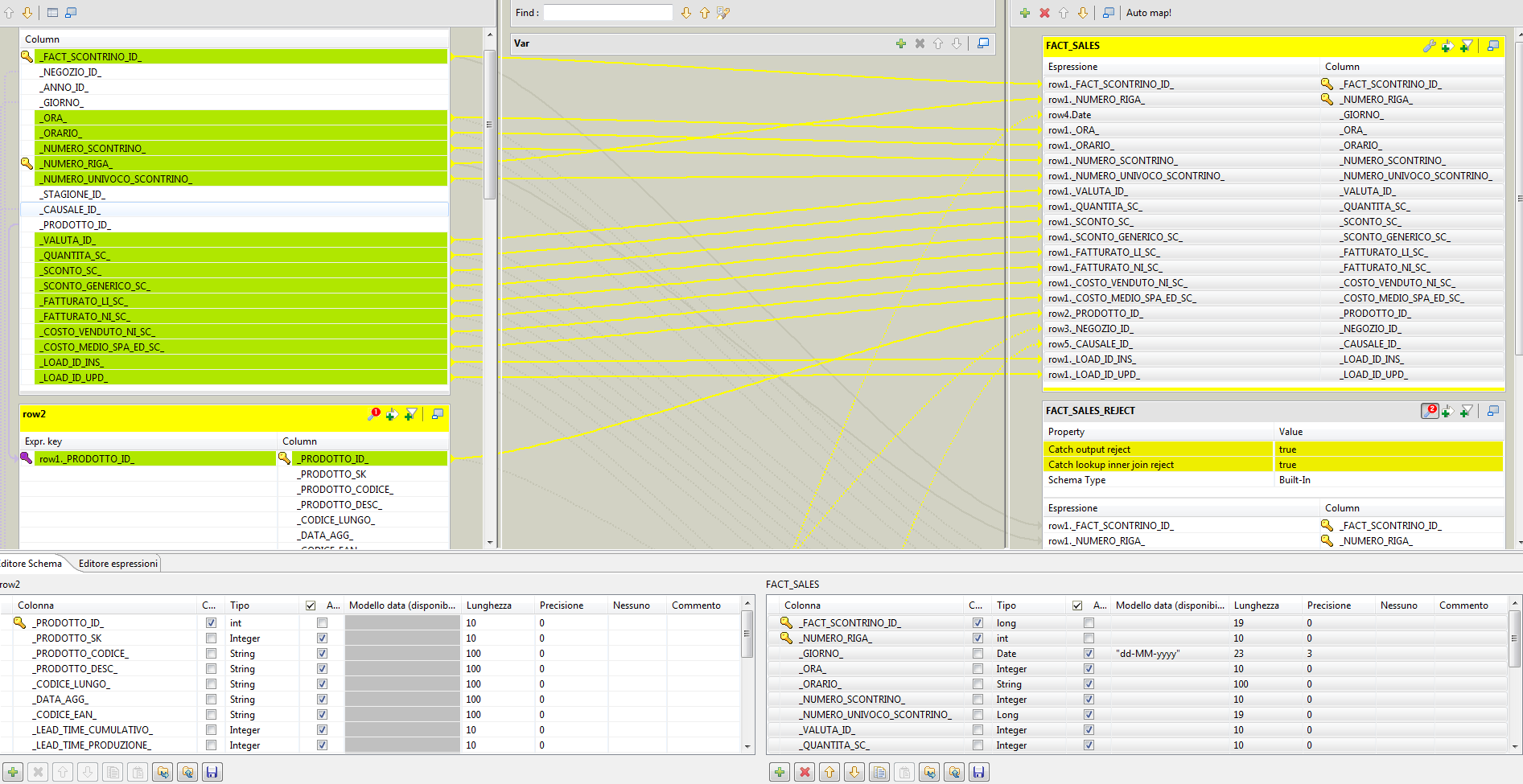
Tabella 6: Data Quality

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type**  **Sorgente** | **Sorgente** | **Trasformation for Data Normalization** | **Data Type**  **Destinazione** | **Destinazione** |
| Datetime | “dd-MMM-yyyy hh:mm:ss” | Change in the variable type directly in tMap options | Date | dd-MM-yyyy |
| String | “Piemonte” | Regione.toUpperCase() | String | “PIEMONTE” |
| String | “Trentino-Alto-Adige” | Regione.replace("-"," ") | String | “Trentino Alto Adige” |
| String | Bolzano/Bolzen | Provincia.replace("/Bolzen","") | String | “Bolzano” |
| String | “Piemonte “ | Regione.trim() | String | “Piemonte” |
| Integer | null | Totale\_ Arrivi\_2018 ==null ? 0 : row1.Totale\_esercizi\_Arrivi\_2018 | Integer | 0 |
| Integer | 23 | Integer | 23 |
| String | “23” | Double.parseDouble(SCONTO) | Double | 23.0 |

In alcuni casi, sono state volte ulteriori trasformazioni a livello di metadati, in particolare nella dimensione dei campi. Queste modifiche sono state eseguite per evitare la Data Truncation, onde evitare successive incongruenze nei dati finali.

I dati di scarsa qualità sono spesso considerati come la fonte di rapporti non accurati e strategie mal concepite in una varietà di società. Il danno economico dovuto a problemi di qualità dei dati può variare da spese varie aggiunte quando i pacchi vengono spediti a indirizzi sbagliati, fino a multe salate di conformità normativa per rapporti finanziari impropri.

### ***2.4.2 Tmap Component In Talend Open Source***

Il componente tMap [24] è uno dei componenti principali di processing di Talend Open Souce, ed è utilizzato principalmente per mappare i dati di input ai dati di output, ovvero mappare uno schema sorgente su uno di destinazione.

INNER JOIN tramite Chiave primaria

Creazione Output contenete i record rigettati

Figura 27: Join & Mapping in Tmap

Mapping

Collego gli attributi del file con gli attributi che creerò nella tabella del DB. La visualizzazione completa di essi è presente nella parte bassa della rappresentazione

Oltre a svolgere funzioni di mappatura,il tMap può anche essere utilizzato per unire più tabelle di input unendo i dati in un'unica tabella di destinazione.

Tutte le trasformazioni precedentemente elencate nella tabella riferita alla Data Quality, sono svolte in questa componente, con una ulteriore possibilità di filtrare i dati.

Un'espressione di mappatura può fare riferimento a qualsiasi numero di colonne da ciascuno degli schemi di input. Nell' editor è possibile utilizzare qualsiasi metodo di classe Java disponibile e le routine Talend, con la restrinzione di inserire un'espressione di mappatura per ciascuna colonna in ogni schema di output .

Immagine che contiene screenshot, mappa, interni, computer

Descrizione generata con affidabilità elevata

Figura 28: Trasformation in Tmap

TRASFORMAZIONI

Mapping

Come si può vedere dagli esempi sopra, ogni mappatura può avere una complessità diversa in base alle nostre esigenze.

Nella best practice bisognerebbe sempre fare in modo che i dati e i modelli creati possano essere sempre accessibili per il riutilizzo. Infatti, è molto probabile che in un recente futuro, si possa estendere la nostra capacità di uno specifico attributo di una tabella, ad esempio controllandone il formato. Perciò, per avere un codice ed un processo performante è meglio cambiare la logica in una singola routine, piuttosto che operare più volte nelle singole espressioni di mappatura.

## ***2.5 LEVEL L2 – SNOWFLAKE DATA MART BEST PRACTICE***

Nei nuovi progetti di Data Analytics una delle migliori pratiche di implementazione di un Data Warehouse o di un Data Mart è lo SnowflakeDB un Database molto simile ma, allo stesso tempo, molto diverso dagli altri database relazionali, e sviluppato principalmente per avere un processo di ETL efficace e veloce.

È definito molto diverso perché è costruito sui principi del cloud: è veramente elastico, è praticamente a zero manutenzione, è quasi real-time e ha un supporto nativo per dati strutturati e semi-strutturati (JSON). Allo stesso tempo, però, è simile perché è un database relazionale colonnare memorizzato, infatti, i fondatori di SnowflakeDB provenivano da Oracle, stessa azienda fondatrice del database relazionale.

In particolar modo, Il mio obiettivo è tato quello di portare un Database relazionale come SQL Server, ad un livello di progettazione in stile Snowflake:

* I dati devono essere classificati e contrassegnati in modo appropriato, in particolare, se altamente protetti.
* I dati devono conservare la propria storia attraverso audit e verifiche dei dati che dovranno essere convalidati alla fonte, ove possibile.
* I dati devono essere elaborati in micro-batches (lotti).
* I dati devono essere ingeriti e caricati e quindi trasformati come richiesto, ovvero in base alle regole di ELT o ETL.
* Elaborare i dati CDC end-to-end per evitare problemi prestazionali.
* Creare istanze di Data Mart separatamente in base ai requisiti specifici del Business aziendale, creando dei sistemi di governance pero ogni vista semantica.

Detto questo, ci sono alcune best practice che si applicano alla implementazione di un Database Snowflake, specifiche per le sue differenze architettoniche uniche con altri database relazionali o piattaforme di big data:

* Utilizzare i DWH indipendenti di Multi-Cluster con funzionalità di storage e con scalabilità dei dati condivise per ottimizzare le esigenze di elaborazione dei vari carichi di lavoro. Ad esempio, la Staging Area (Level L0) può trovarsi su un altro database rispetto al Core Layer (Level L1).
* Assegna un DWH virtuale separato per ogni Data Mart per avere un'esperienza ottimizzata di consumo dei dati.
* Mantenere, se possibile, i dati semi-strutturati nel formato originale per aumentare le prestazioni dell'elaborazione dei dati. Spesso i dati JSON vengono elaborati più velocemente di quelli convertiti in tabelle relazionali. Quando si memorizzano i dati semi-strutturati, SnowflakeDB ottimizza lo storage in base agli elementi ripetuti all'interno delle stringhe di semi-struttura.
* Carica i dati in piccoli blocchi invece di un file di grandi dimensioni e caricarli in parallelo utilizzando più nodi. Ad un cliente siamo stati in grado di caricare 24 mesi di dati di telemetria degli eventi in meno di 2 settimane con cluster di nodi piccoli.
* Assegnare cluster virtuali separati agli schemi di una Data Warehouse per ottimizzare le prestazioni e considerare il clustering di tabelle di grandi dimensioni per migliorare le prestazioni della query. Recluster se le prestazioni diminuiscono. A volte il cluster/reclustering può ridurre le prestazioni della query, anche se è consigliato di analizzare i dati della tabella prima di apportare tali modifiche.
* SnowflakeDB memorizza i metadati (valori min e max, valori distinti, ecc.) in modo che possa efficacemente sfoltire le micro partizioni necessarie per eseguire la scansione di una query.
* Si consiglia di eseguire l'ingestione dei dati basata su eventi per consentire l'ordine cronologico dei dati. Creare dei pipeline di dati che utilizzano la potenza di elaborazione di Snowflake. Le pipeline basate su framework flessibili di essere completamente separati dalla struttura di elaborazione effettiva. Questa operazione viene fatta tramite la creazione di chiavi surrogate (SK\_KEY) di tipo INTEGER per migliorare l’efficienza permettendo un rapido collegamento e caricamento di dati per le tabelle di grandi dimensioni, dove sarà riportata solo la chiave e non tutti gli attributi. Ciò consente modifiche minime del codice se sono presenti modifiche nello schema di destinazione o nello schema di origine, potendo comunque eseguire tutte le operazioni di caricamento, trasformazione, aggregazione e elaborazione dei dati.
* Costruire un robusto framework di controllo del bilancio di audit che traccia non solo la discendenza dei dati e la qualità dei dati, ma consente l'ottimizzazione delle prestazioni del database rispetto ai costi di elaborazione.
* Crea cloni a zero copie per creare database di test o di convalida per evitare la duplicazione dei dati.

SnowflakeDB è un potente database e ha delle caratteristiche uniche che consentono una rapida implementazione dei progetti di analisi dei dati, ma come tutti gli altri database richiede un'attenta progettazione. Seguendo la best practice appena elencata, la prima parte da svolgere nel progetto è la creazione delle Surrogate Key per tutte le tabelle Anagrafiche del livello L1, creando delle relazioni padre e figlio tra le tabelle Fatto e le tabelle Dimensioni.

**ALTER TABLE** L1.CANALE **ADD** \_CANALE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.ALTEZZA\_TACCO **ADD** \_ALTEZZA\_TACCO\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.CATEGORIA **ADD** \_CATEGORIA\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.CAUSALE **ADD** \_CAUSALE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.COLORE **ADD** \_COLORE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.COMUNI\_ISTAT **ADD** \_COMUNE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.LOCATION **ADD** \_COMUNE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.GENDER **ADD** \_GENDER\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.FAM\_COLORE **ADD** \_FAMIGLIA\_COLORE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.FAM\_MATERIALE **ADD** \_FAMIGLIA\_MATERIALE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.DATE **ADD** \_DATE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.ISTAT **ADD** \_ISTAT\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.MADE\_IN **ADD** \_MADE\_IN\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.MADE\_IN\_PIANIFICAZIONE **ADD** \_MADE\_IN\_PIANIFICAZIONE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.MATERIALE **ADD** \_MATERIALE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.MODELLO **ADD** \_MODELLO\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.PROVINCIA **ADD** \_PROVINCIA\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.REGIONE **ADD** \_REGIONE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.STAGIONE **ADD** \_STAGIONE\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.TAGLIA **ADD** \_TAGLIA\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.TURISMO **ADD** \_TURISMO\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.SHOP **ADD** \_NEGOZIO\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.SHOP\_OPEN\_ITA **ADD** \_NEGOZIO\_OPEN\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.SHOP\_NOT\_OPEN **ADD** \_NEGOZIO\_NOT\_OPEN\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.SHOP\_TOT **ADD** \_NEGOZIO\_TOT\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.PRODUCT\_DATA\_MASKING **ADD** \_PRODOTTO\_DM\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.PRODUCT **ADD** \_PRODOTTO\_DM\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.DAY **ADD** \_DAY\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.YEAR **ADD** \_YEAR\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.MONTH **ADD** \_MONTH\_SK **INT IDENTITY**(1,1) **NOT NULL**;  
**ALTER TABLE** L1.QUARTER **ADD** \_QUARTER\_SK **INT IDENTITY**(1,1) **NOT NULL**;

A questo punto, tramite la creazione di una connessione allo schema L1 del Database Fashion\_Retail in forma di metadato, procedo con l’estrazione delle tabelle aventi la Surrogate Key in esso presenti, e tramite l’elaborazione nello strumento di Talend tMap [24], diventeranno le chiave primarie delle nuove Tabelle Dimensioni.

Nello specifico, si vede come l’imput “row1” definito come la Tabella Categoria del livello L1 avente come chiave primaria l’Id, viene riportata a livello L2 con uno schema identico al precedente, ma con chiave è primaria la chiave surrogata:

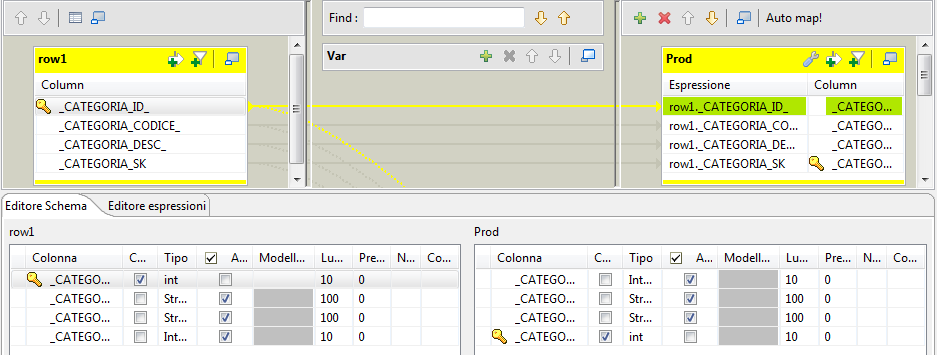


Figura 29: SnowflakeDB Surrogate Key

In generale:

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

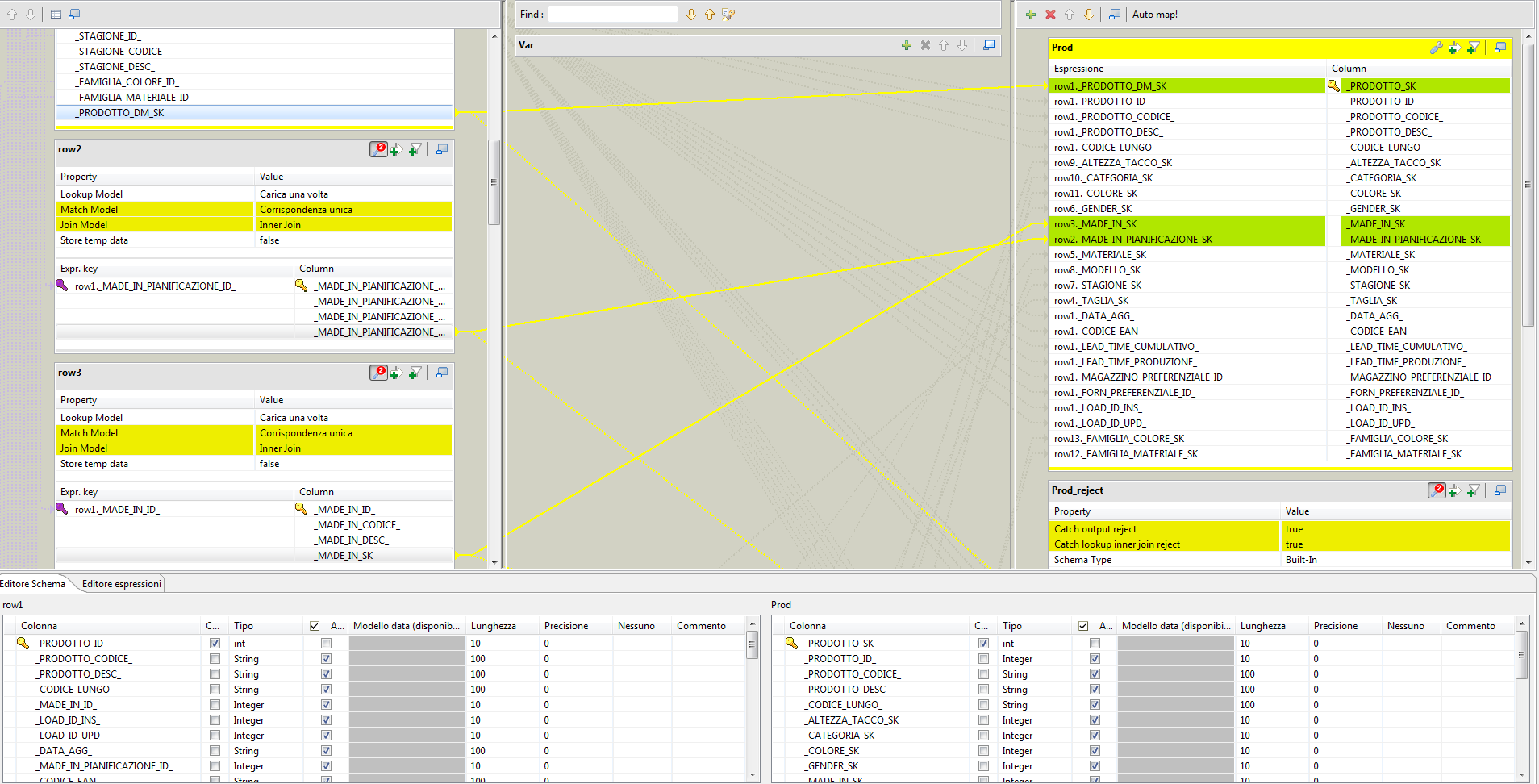
******

Figura 30: SnowflakeDB Dimension

Inoltre, è molto importante fare attenzione nella costruzione delle tabelle di fatto, in quanto, essendo tabelle padre delle tabelle dimensioni, non avranno una propria chiave surrogata. In esse, sarà riportata solo la surrogata delle tabelle dimensioni che la caratterizzano, tralasciando tutte le informazioni come, per esempio, l’Id e la descrizione, in quanto, in questo processo non ha alcun bisogno di scendere nel dettaglio in un’unica tabella ma è un sistema strutturato ad albero con un sistema di gerarchie, definito appunto tramite il comando Join tra le chiavi surrogate delle tabelle dimensioni.

***Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata***

***Immagine che contiene screenshot

Descrizione generata con affidabilità molto elevata***

Figura 31: SnowflakeDB FACT

### ***2.6.1*** ***Snowflake Schema***

Una base dati è in 3NF (*terza forma normale*) se tutti gli attributi non-chiave dipendono dalla chiave soltanto, ossia non esistono attributi non-chiave che dipendono da altri attributi non-chiave. Tale normalizzazione elimina la dipendenza transitiva degli attributi dalla chiave e prende il nome di schema SnowFlake.

Il nome Snowflake Schema deriva dal fatto che le tabelle delle dimensioni si ramificano e assomigliano, per l’appunto, ad un fiocco di neve. Osservando il modello, si evidenzia come una tabella dei fatti è circondata da delle tabelle dimensionali, con i quali si creerà la suddetta ramificazione. A differenza dello schema a stella, le tabelle di dimensioni nello schema a fiocco di neve possono avere le proprie categorie. L'idea dominante dietro lo schema fiocco di neve è che le tabelle delle dimensioni sono completamente normalizzate. Ogni tabella delle dimensioni può essere descritta da una o più tabelle di ricerca o ancora da più tabelle di ricerca aggiuntive. Questo viene ripetuto finché il modello non è completamente normalizzato.

Ovviamente, la normalizzazione crea una maggiore complessità nell’eseguire le query dello schema snowflake, in quanto, per esempio, dovremo scavare più in profondità per ottenere il nome del tipo di prodotto o il comune di un negozio. La struttura si basa su una serie di JOIN annidati, dove ad un semplice JOIN, bisognerà aggiungerne un altro JOIN per ogni nuovo livello all'interno della stessa dimensione. Naturalmente, non esiste un numero di annidazioni standard, ma dipende dal livello del dato che si vuole estrarre. Più il dato è in profondità, più il processo di scrittura delle query sarà complesso [23].

Fondamentalmente, una query eseguita su un Data Mart basato su schema snowflake verrà eseguita più lentamente rispetto ad uno su Starschema. Nella maggior parte dei casi, questo non rappresenta un problema: non importa molto se otteniamo il risultato in un secondo o in un millisecondo.

Nel progetto proposto, una vista completa dello snowflake è la seguente:

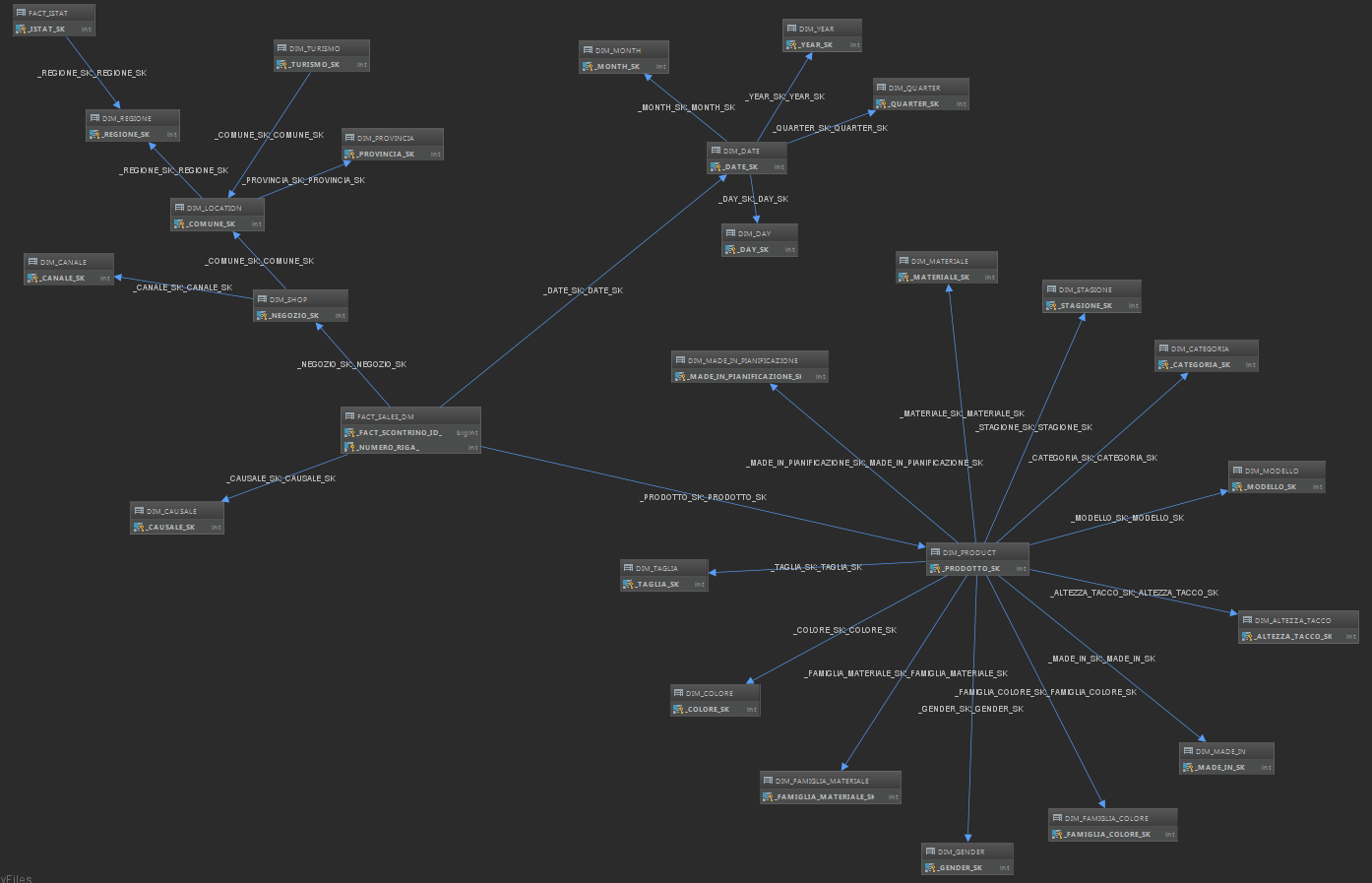


Figura 32: Snowflake Schema

## ***2.6 LEVEL L2 – STARSCHEMA DATA MART BEST PRACTICE***

I Database Relazionali, i più usati, rispetto ai SnoflakeDB presentano gli stessi livelli di Staging Area e Trasformation Area ma con una sostanziale differenziazione nel livello finale L2.

In questo caso, lo scopo non è quello di avere un processo di ETL super performante, ma di avere un minor numero di tabelle finali di grandi dimensioni aventi più informazioni possibili per facilitare, tramite software di Data Visualization, la creazione di report utili ai fini di decisioni strategiche future o per un semplice audit sull’andamento economico finanziario della azienda.

Lo schema del Job non cambia rispetto a quello dello SnowflakeDB, ma a cambiare è la mappatura delle variabili nel tMap [24].

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figura 33: Job Product Star Schema

Infatti, le tabelle del Livello L2 anche in questo caso derivano dal livello L1 ed è possibile ricavarle con una serie di JOIN tra le tabelle, non più collegate alla Surrogate Key, ma direttamente alla Primary Key, con i dovuti controlli di integrità.

Gli attributi della tabella finale possono derivare anche da tabelle differenti, in quanto ogni attributo di una tabella è collegato all’attributo della tabella finale tramite l’azione i JOIN.

Immagine che contiene screenshot, mappa

Descrizione generata con affidabilità molto elevata

Figura 34: tMap Product Star Schema

Lo stesso procedimento si è svolto per le tabelle dei fatti, ricordandosi di fare le JOIN non con le chiavi surrogate delle tabelle dimensioni, ma farli direttamente con le chiavi primarie.

### ***2.6.1 Star Schema***

Una volta costruito il Data Fact Model, viene implementato lo schema logico. Esso viene rappresentato secondo uno Star Schema, il cui centro è costituito da una tabella dei fatti; le punte della stella rappresentano invece le tabelle delle dimensioni che si diramano dal centro.

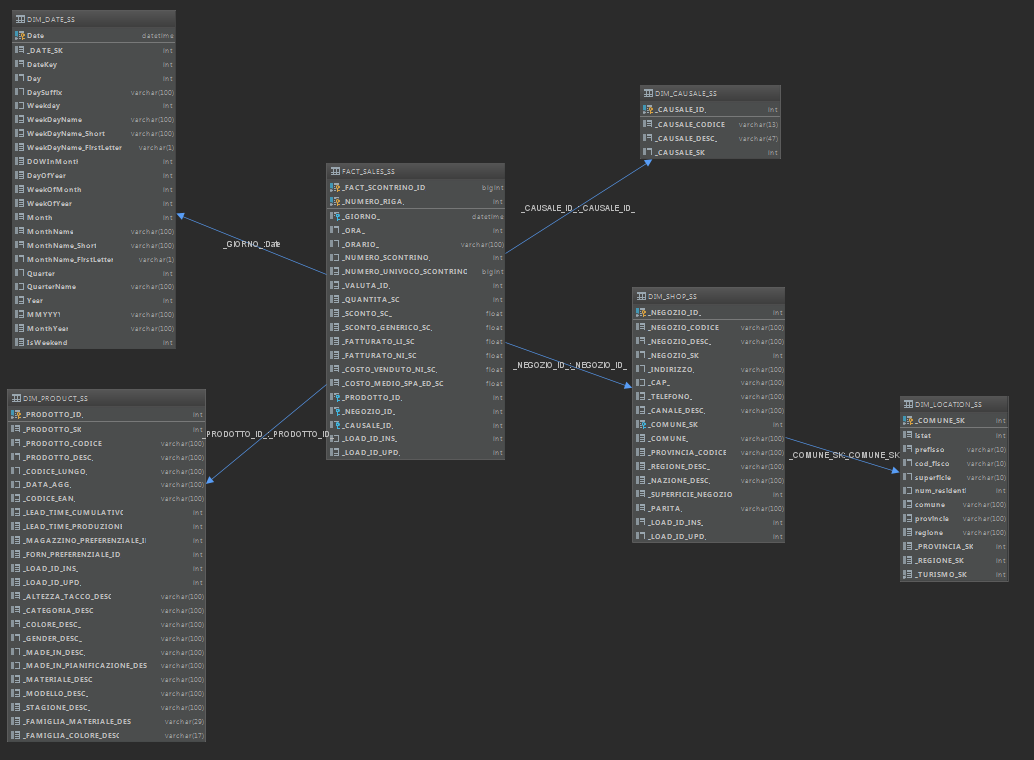


Figura 35: Star Schema

Le caratteristiche principali di uno Star Schema sono le seguenti:

* Struttura semplice di facile comprensione.
* Query molto performanti, perché riducono i join da effettuare tra tabelle.
* Tempo di caricamento dei dati relativamente lungo, perché la ridondanza dei dati dovuta alla de-normalizzazione, provoca l’aumento delle dimensioni della tabella.
* Ampiamente supportato da un gran numero di strumenti di business intelligence.

Solitamente, le tabelle dei fatti in uno Star Schema sono in terza forma normale, mentre le tabelle dimensionali sono denormalizzate [10].

## ***2.7 FULL LOAD ETL & AUDIT***

Per velocizzare tutto il processo spiegato in precedenza, il metodo migliore è quello di creare dei job che contengano altri job. In questo modo, posso racchiudere in sottogruppi per ogni livello le anagrafiche e i movimenti per poi eseguirli tutti allo stesso tempo.

Per esempio, nel livello L0 ho il Job STG\_Anagrafiche, dove raccolgo tutti i job del livello L0 riguardanti tutte le anagrafiche, come visualizzato nell’ immagine seguente.

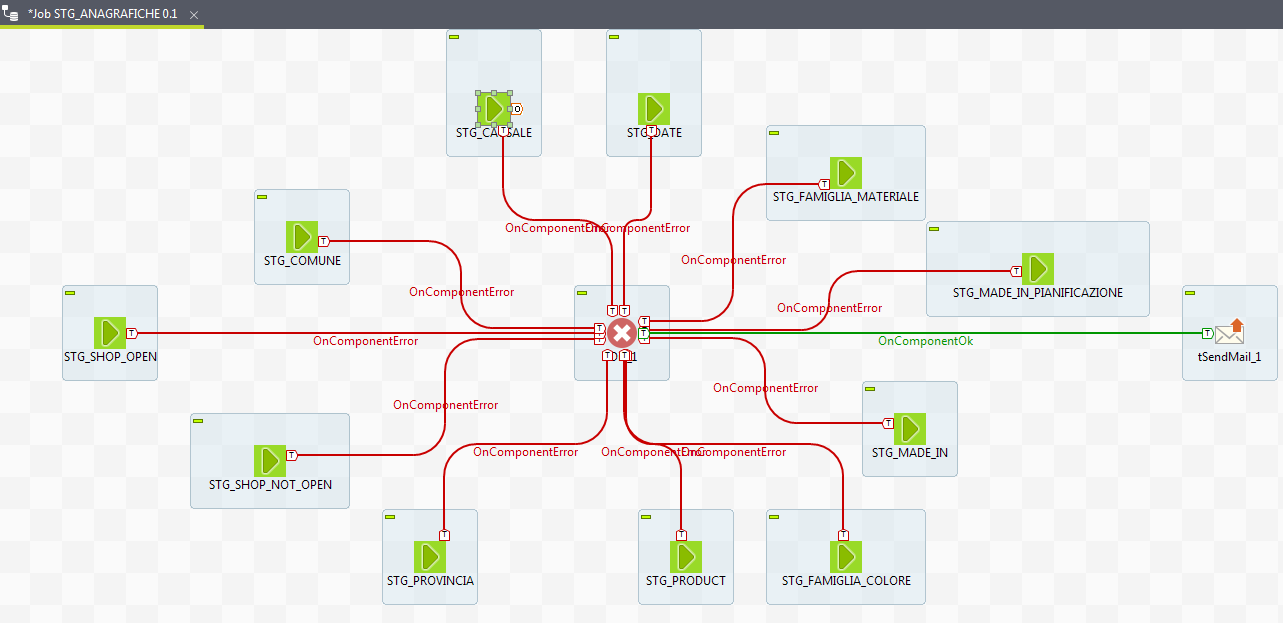
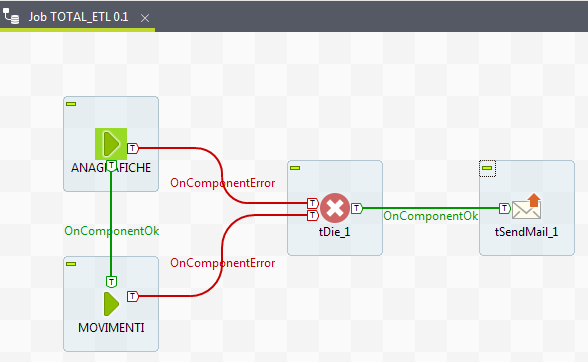


Figura 36: Job STG Anagrafiche

Lo stesso procedimento è svolto per le Anagrafiche e i movimenti di ciascun livello, arrivando al livello L2 con solo due job da unire, rispettivamente uno relativo alle anagrafiche e un ai movimenti.

Come ultimo step del processo ETL, non resta che unire i due tipi di configurazione dato con tutta la loro gerarchia in un unico job finale. Il vantaggio del Full Load ETL è la capacità tramite un univoco comando di caricare interamente tutto un database.

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Immagine che contiene testo, mappa

Descrizione generata con affidabilità molto elevata

Figura 37: Full Load ETL & Auditing

È molto importante controllare il flusso dei dati nelle varie fasi del processo. Nella figura sopra stante, oltre che all’ultimo step è importante focalizzarsi su due tools: tDie e tSendEmail. Essi funziona insieme, e quando il processo va in Errore, viene mandato un segnale al tDie, che tramite l’aiuto dello strumento tSendEmail, manda una Email, con l’allegato in forma di Script dell’errore, sia al proprietario che al gestore del Database.

### ***2.7.1 Auditing ETL***

Il controllo in un processo di estrazione, trasformazione e caricamento ha lo scopo di soddisfare i seguenti obiettivi:

* Verificare le anomalie dei dati oltre a controllare semplicemente gli errori gravi.
* Catturare e archiviare una traccia elettronica di eventuali modifiche materiali apportate ai dati durante la trasformazione

L'auditing ETL aiuta a confermare che non ci sono anomalie nei dati anche in assenza di errori. Un meccanismo di auditing ben progettato aggiunge anche l'integrità del processo ETL eliminando l'ambiguità nella logica di trasformazione, intrappolando e tracciando ogni modifica apportata ai dati lungo il percorso. Anche nelle architetture ETL più rudimentali, è possibile controllare alcune metriche di alto livello per confermare che i dati caricati sono quelli previsti.

In generale, i processi ETL di auditing dovrebbero verificare quanto segue per confermare che gli input corrispondono agli output:

* Conteggio generale delle righe.
* Totali aggregati (che potrebbero includere importi finanziari o altri dati di riepilogo).

Alcuni processi richiedono un audit più esaustivo. In altri casi, potrebbe essere necessario verificare se i dati siano entro limiti ragionevoli o se supportano tali valori. Altro aspetto che non bisogna dimenticare di verificare i casi in cui non è stato caricato alcun dato. Purtroppo, succede spesso, e le due maggiori cause sono dovute ad un file sorgente che non contiene dati, una query configurata in modo errato che non restituisce righe o una directory di origine vuota destinata a contenere uno o più file potrebbero portare al corretto completamento del processo ETL ma a caricare esattamente zero righe di dati. Tuttavia, se un determinato processo dovrebbe sempre comportare un numero di file caricato diverso da zero, assicurarsi di aggiungere una fase di controllo per verificarlo.

L'auditing ETL è raramente l'elemento più visibile nell'architettura, ma è una polizza assicurativa necessaria per proteggere l'integrità dei dati e del processo