Università della Calabria

DIPARTIMENTO DI INGEGNERIA INFORMATICA MODELLISTICA ELETTRONICA E SISTEMISTICA

Corso di Laurea Magistrale in Ingegneria Informatica

Progetto Business Intelligence

Candidati:
Francesco Musmanno 216634
Ivan Scuderi 216635

Relatori:
Prof. Filippo Furfaro
Prof. Nunziato Cassavia

Anno Accademico 2021-2022

Indice

1 Introduzione

Lo sviluppo del progetto è stato modellato analizzando eventuali necessità e bisogni di un'azienda fittizia chiamata *Northwind Traders*. Non avendo a disposizione conoscenze approfondite sull'organigramma, la struttura aziendale e l'organizzazione del *business*, sono state effettuate alcune supposizioni sulle eventuali utitità e benefici che un sistema di *data-warehousing* possa apportare nello specifico scenario di interesse.

La trattazione inizia effettuando una descrizione dello scenario applicativo, all'interno della quale vengono definiti gli attori che entrano in gioco nel sistema, con annesse alcune supposizioni in merito alla struttura dell'azienda. Successivamente si procede effettuando una descrizione in merito alle fonti dati utilizzate ed eventuali operazioni di ricognizione, normalizzazione, integrazione e progettazione del livello riconciliato. Segue un'analisi degli ipotetici requisiti utente definendo in maniera informale anche le eventuali interrogazioni di interesse. Si procede poi con lo sviluppo vero e proprio del progetto, definendo in primo luogo l'architettura utilizzata, gli eventuali *data mart* e cubi implementati e in secondo luogo le tecnologie utilizzate. Successivamente vengono condotte le fasi vere e proprie dello sviluppo del sistema di *data-warehousing*, in particolare mediante la *progettazione concettuale*, la *progettazione logica*, la *progettazione dell'alimentazione* ed infine la *progettazione fisica*.

2 Descrizione dello scenario

Il *database* che ci è stato fornito dal committente contiene i dati di vendita per *Northwind Traders*, un'azienda di esportazione e importazione di specialità alimentari che opera su scala globale. Il formato dei dati in questione è basato su SQL. All'interno di questo sistema sono presenti quattro attori che svolgono le operazioni di seguito elencate:

- *customers*: sono i clienti che contattando l'azienda *Northwind Traders*, ed interagendo in particolar modo con gli *employees*, richiedono l'acquisto dei prodotti di genere alimentare, una volta generato l'ordine procedono con il pagamento;

- *employees*: sono i dipendenti presenti all'interno dell'azienda che si occupano di interagire con i *customers*, al fine di gestire gli ordini effettuati da quest'ultimi, sono in diretto rapporto con i *suppliers* che forniscono loro la merce che verrà successivamente venduta, ed infine interagiscono con gli *shippers* al fine di organizzare le spedizioni;

- *shippers*: rappresentano la compagnia che si occupa della spedizione dell'ordine richiesto dai *customers* presso l'azienda;

- *suppliers*: sono i fornitori che interagendo direttamente con gli *employees* riforniscono il magazzino dell'azienda con la merce che verrà successivamente acquistata dai *customers*.

All'interno del sistema non si ha alcuna informazione sui costi che l'azienda sostiene per acquistare i prodotti presso i *suppliers* e sulla relativa spedizione, si suppone dunque che i prodotti siano già presenti all'interno del magazzino.

Nel momento in cui un prodotto viene ricevuto dall'azienda viene posto nel magazzino e si inserisce una tupla all'interno del sistema informativo, che fornisce informazioni sul prodotto e sulle relative quantità e categorie. Quando un impiegato viene assunto dall'azienda viene inserito nel database con una serie di informazioni annesse, ed in particolare viene assegnata al suddetto impiegato una zona di competenza, un ruolo e conseguentemente un superiore a cui fare riferimento nella gerarchia aziendale. Si suppone che i vari clienti vengano registrati nel sistema una volta effettuato un acquisto e che quindi si generi un dettaglio dell'ordine. Non si hanno informazioni per quanto riguarda la scelta della compagnia di spedizione, ossia se venga selezionata dal cliente o gestita automaticamente all'interno dell'azienda. Di seguito è riportato il diagramma dei casi d'uso realativo ai servizi offerti:

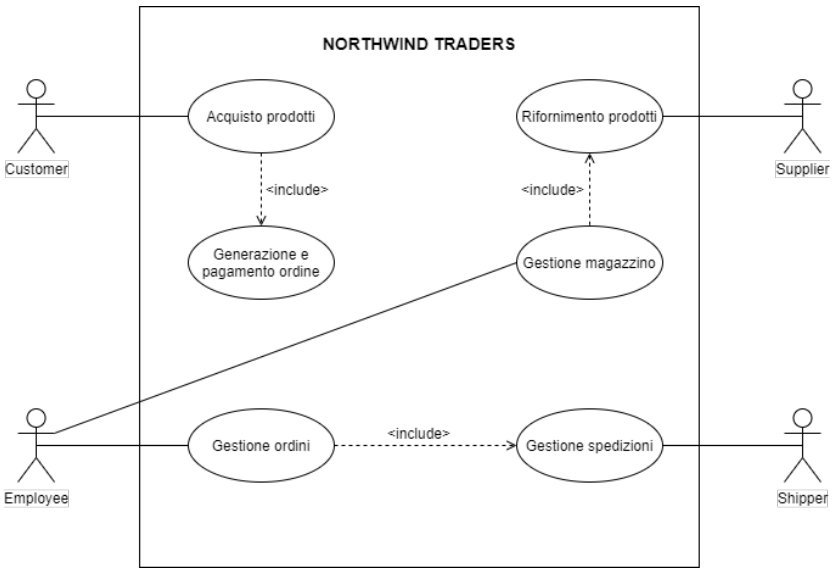

Figura 1: Use case diagram

3 Analisi e Riconciliazione delle fonti dati

All'interno di questa sezione sono descritte le fasi di ricognizione, normalizzazione, integrazione e di progettazione del livello riconciliato.

3.1 Ricognizione

Di seguito è riportato il modello entità relazione relativo al *database* in questione:

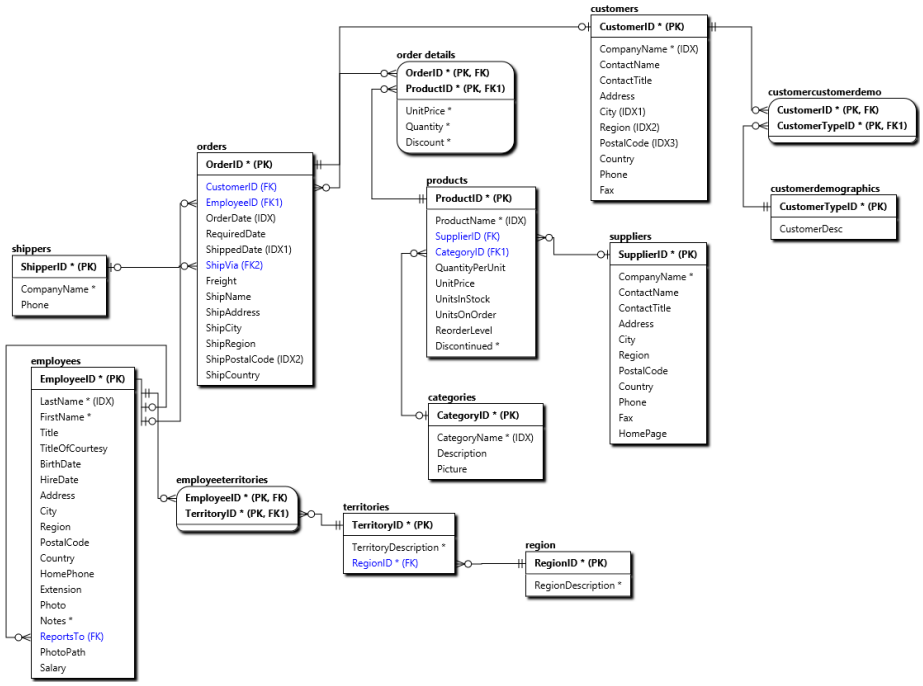

Figura 2: Modello Entità Relazione

Come si evince dallo schema di *database* sono presenti 13 tabelle, di seguito si fornisce una descrizione dettagliata di esse:

- orders : sono presenti tutte le informazioni relative all'ordine effettuato da parte di un cliente, quali la data in cui è stato effettuato, la data in cui il cliente ha necessità di ricevere l'ordine, la data di consegna ed altre informazioni come ad esempio indirizzo e spesa di spedizione (*freight*). Questa tabella presenta diverse relazioni, in particolar modo con *shippers* ed indica la compagnia incaricata del servizio di spedizione, con *employee* in cui si fa riferimento all'impiegato incaricato della gestione dell'ordine, con *customer* che indica il cliente che ha effettuato il suddetto ordine ed infine, una relazione $0 : n$ che specifica i vari prodotti inseriti all'interno dell'ordine.
- shippers : sono presenti le informazioni relative alla compagnia che si occupa di gestire le spedizione con l'eventuale recapito telefonico, è riportata una relazione verso la tabella *orders*.

- employees : fornisce informazioni relative agli impiegati, l'organigramma aziendale suddivide gli impiegati in una gerarchia (a tre livelli) e per i livelli più bassi di quest'ultima è presente il campo *reportsTo* che indica l'ID dell'impiegato a cui si deve riferire. Per ogni impiegato sono presenti informazioni relative al proprio contatto, come ad esempio numero di telefono e relativa estensione, un'immagine e il salario. Ogni impiegato è incaricato di servire una determinata zona, ciò viene espresso mediante una relazione con la tabella *employeeterritories*. Di seguito è riportato il diagramma relativo alla gerarchia, i numeri utilizzati fanno riferimento all'identificativo del dipendente:

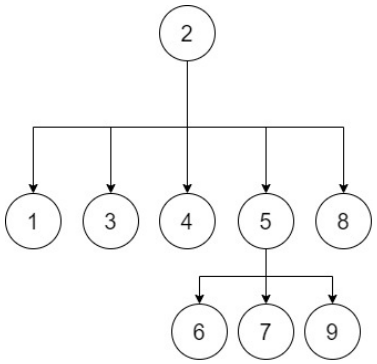

Figura 3: Diagramma gerarchia employee

- employeeterritories : al suo interno è riportato l'ID dell'impiegato e l'ID del territorio servito da quest'ultimo. É presente una relazione con la tabella *territories*.

- territories : presenta l'ID del territorio, una descrizione di quest'ultimo e l'ID della regione in cui si trova tale territorio. É presente una relazione con la tabella *region*.

- region : identificata da un ID, al suo interno sono presenti le descrizioni delle regioni.

- product : contiene al suo interno informazioni relative ai prodotti che l'azienda vende come ad esempio il nome, la categoria, il prezzo unitario, la quantità presente in magazzino. Il prezzo unitario, fa riferimento al prezzo relativo ad un'unità di prodotto e la descrizione della quantità di prodotto presente in una singola unità è specificata nell'attributo *quantityPerUnit*. Inoltre, l'attributo *discontinued* indica se tale prodotto è fuori produzione e *reorderLevel* rappresenta il numero di prodotti sotto la cui soglia debba essere inviato un nuovo ordine, al fine di rifornire il magazzino. Presenta inoltre relazioni con *categories* che specifica la categoria a cui il generico prodotto appartiene e con la tabella *supplier* per indicare il fornitore per il designato prodotto.

- orderdetails : fornisce una rappresentazione del dettaglio dell'ordine, implementa la relazione molti-a-molti tra le tabelle *orders* e *products*, in

modo tale che ad ogni ordine sia associata una lista di prodotti e analogamente consente che ogni prodotto possa trovarsi in più ordini. Per ogni tupla è specificato il prezzo unitario, la quantità ordinata e l'eventuale sconto applicato.

- categories : definisce al suo interno le varie categorie per ogni prodotto, presenta un nome ed eventuali descrizioni e fotografie.
- customers : presenta tutte le informazioni relative ai clienti.
- suppliers : contiene al suo interno le informazioni relative ai fornitori.
- customercustomerdemo e customerdemographics : per quanto riguarda entrambe queste tabelle non si hanno a disposizione sufficienti informazioni al fine di determinarne lo scopo e la semantica.

3.2 Normalizzazione

A fronte dell'analisi effettuata in fase di ricognizione, avendo analizzato in dettaglio le tuple delle varie tabelle, è stato deciso di impiegare alcune scelte progettuali in fase di normalizzazione, di seguito esposte:

- è stato deciso di non implementare la dipendenza funzionale $city \rightarrow region$ perché la seconda colonna risulta essere composta da una grande maggioranza di valori posti a *null* e da pochissimi valori validi;
- è stato scelto di implementare la dipendenza funzionale $city \rightarrow country$ poiché è presente un'eccessiva ridondanza, dato che questi attributi sono presenti all'interno delle tabelle *customers*, *suppliers*, *employees* ed *orders*;
- per quanto riguarda l'attributo *postalCode*, la scelta è stata quella di evitare di adottare dipendendenze funzionali che facessero capo a questa colonna, in quanto nel contesto specifico si hanno a disposizione molti dati che fanno riferimento a diversi stati e pertanto non si ha la certezza di avere relazioni corrette. Inoltre nelle grandi città è possibile avere *postalCode* differenti in funzione dell'indirizzo;
- è stato deciso di non realizzare la dipendenza funzionale $address \rightarrow city$ per evitare casi di omonimia dati dalla presenza di uno stesso indirizzo in città differenti.

Oltre alle considerazioni fatte non sono state riscontrate altre dipendenze funzionali sulle quali porre attenzione.

3.3 Integrazione

Nel contesto di riferimento avendo a disposizione un'unica fonte dati (database SQL), non è risultato necessario effettuare la fase di integrazione.

3.4 Progettazione del livello riconciliato

All'interno di questa fase sono state apportate varie modifiche ed è stata implementata la normalizzazione sul *database* operazionale ed operazioni di pulizia ed aggiunta di campi, in particolare:

- sono state eliminate due righe dalla tabella *customer* che avevano valori per la maggior parte posti a *null* in ogni attributo.
- É stata eliminata una tupla dalle tabelle *employee_territory* e *region* in quanto era presente un valore duplicato relativo alla *entry* che faceva riferimento alla città di New York.
- É stata eseguita un'operazione di ripulitura all'interno delle tabelle *orders*, *employees*, *customers* e *suppliers*, in particolar modo per quanto riguarda l'attributo città in cui erano presenti molti errori che sono stati corretti manualmente.
- Al fine di ridurre la ridondanza è stata creata una nuova tabella denominata *city_country* che al suo interno riporta le città, impostate a chiave primaria e lo stato in cui quella determinata città si trova. Una volta creata la tabella sono state eliminate le colonne *country* per le tabelle *orders*, *employees*, *customers* e *suppliers* e create le opportune chiavi esterne.
- Per quanto riguarda il prezzo unitario sono state trovate incosistenze nei dati delle tabelle *orderDetails* e *products* (prezzi unitari differenti), di conseguenza è stato deciso di eliminare l'attributo *unitPrice* nella tabella *orderDetails* al fine di rendere i dati consistenti e il tutto meno ridondante.
- É stato deciso di eliminare alcuni attributi ridondanti presenti nella tabella *orders*, in particolare *ShipName*, *ShipAddress*, *ShipCity* e *ShipPostalCode* che fanno riferimento alla tabella *customers* rispettivamente agli attributi *CompanyName*, *Address*, *City* e *PostalCode*.
- É stato inserito manualmente l'attributo *sex* all'interno della tabella *employees*, il quale rappresenta il sesso del relativo impiegato, poiché l'informazione potrebbe essere utile al fine di query analitiche.
- Sono presenti diverse colonne che non rappresentano informazioni utili ai fini dello sviluppo del *Data Warehouse*, che non sono state eliminate, ma non verranno considerate nel proseguo dello sviluppo del progetto, come ad esempio foto di categorie ed impiegati, i link alle homepage dei vari *supplier* ecc.

In figura di seguito, è riportato il modello entità relazione dello schema di database riconciliato.

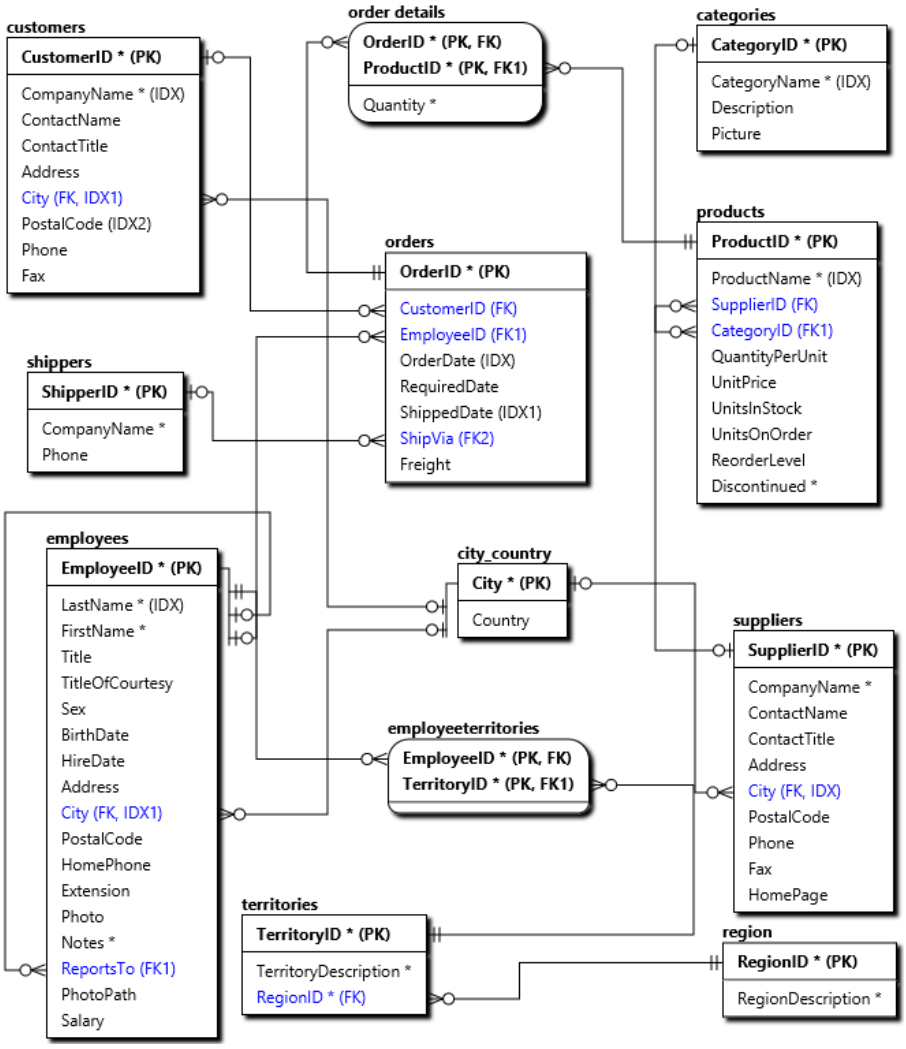

Figura 4: Modello Entità Relazione dello schema riconciliato

4 Analisi dei requisiti utente

Le informazioni relative all'utilizzo e allo scopo del *Data-Warehouse* sono state ottenute mediante interviste con i diretti interessati. Gli utenti che faranno uso del sistema sono in particolar modo i *manager* dell' azienda, i *resposabili del marketing*, gli eventuali *addetti alle vendite* o alla logistica ed organizzazione delle spedizioni. Per l'analisi del carico di lavoro ed eventuali query che verranno sottoposte al sistema viene fornita una descrizione nel seguente diagramma:

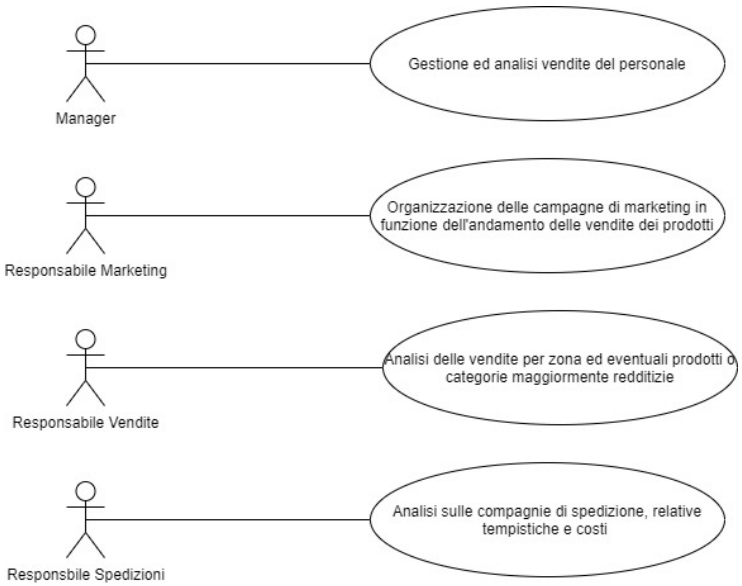

Figura 5: Diagramma finalità utenti

Le interrogazioni di interesse per uno specifico utente che utilizza il sistema sviluppato possono essere ad esempio:

- *Manager*: può avere interesse ad analizzare la qualità del lavoro svolto dal personale, analizzando le eventuali aree di competenza maggiormente congestionate, al fine di effettuare un ipotetico avvicendamento in modo tale da gestire in maniera efficiente i territori di vendita, con l'interesse di incrementare i profitti ed ottimizzare i costi di gestione.

- *Responsabile Marketing*: può condurre analisi valutando l'andamento delle vendite anche in base alle tipologie di clienti che effettuano acquisti, in modo tale da adottare politiche di marketing al fine di massimizzare le vendite per una determinata tipologia di prodotto in funzione dell'area geografica a cui il cliente appartiene, fondamentale inoltre anche per migliorare la visibilità dell'azienda sul territorio.

- *Responsabile Vendite*: può effettuare analisi relative all'andamento vero e proprio delle vendite, considerando tipologie di prodotti e spese sostenute per eventuali spedizioni, analizzando le categorie più popolari in funzione dell'area geografica e dei clienti, è inoltre in grado di ottenere informazioni

relative anche a prodotti non più sul mercato, ma che avevano riscontrato successo.

- *Responsabile Spedizioni*: può essere interessato ad analizzare l'andamento dei prezzi di spedizione nel tempo, relativamente anche alla tipologia di prodotto spedito e alla fascia temporale in cui quest'ultimo è stato ordinato, valutando anche i tempi di consegna per prodotto o categoria in funzione della compagnia di spedizione.

5 Pianificazione del DW

La scelta dal punto di vista del *deployment* è virata su un architettura a due livelli, in quanto, avendo effettuato ricognizione e normalizzazione di un'unica fonte dati, non è risultato necessario sviluppare un livello di *operational-data-store* in cui riconciliare più fonti dati provenienti dall'azienda. Questo ha consentito di snelirre il carico di lavoro e di ridurre i tempi di sviluppo dal punto di vista pratico dell'intero sistema. Il *Data-Warehouse* presenta un unico *Data-Mart*, che a sua volta è composto da un unico cubo che consente di aggregare le informazioni in modo da poter rispondere efficacemente alle eventuali interrogazioni da parte degli attori, che operano all'interno dello scenario aziendale.

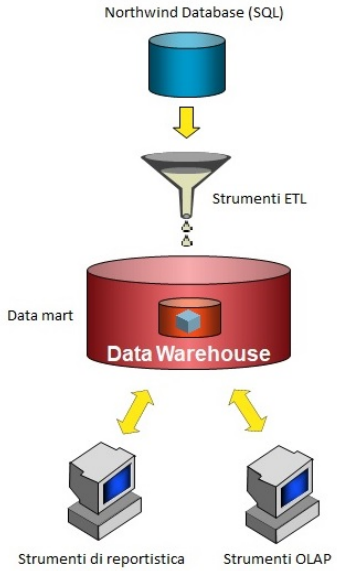

Figura 6: Diagramma di deployment

Per quanto riguarda le tecnologie utilizzate per il progetto si fa menzione dei seguenti strumenti:

- *Pentaho Data Integration*: per lo sviluppo ed implementazione dell'ETL;
- *Pentaho Schema Workbench*: per la modellazione del cubo;
- *Pentaho Aggregation Designer*: per la progettazione delle viste;
- *Pentaho Server*: come piattaforma di supporto per il *Data-Warehousing*;
- *MySQL*: come RDBMS impiegato sia per i dati operazionali che per l'implementazione del cubo (ROLAP);
- *ApricotDB*: come *software* per la modellazione degli schemi di *database*.

6 Progettazione concettuale

Per risponedere in maniera completa a tutte le query descritte per i vari utenti finali, la scelta del fatto di interesse per la costruzione del cubo, è ricaduta sulla tabella *order details*. La fase di progettazione concettuale ha avuto inizio a partire dalla suddetta tabella, che ci ha consentito di sviluppare il seguente albero degli attributi:

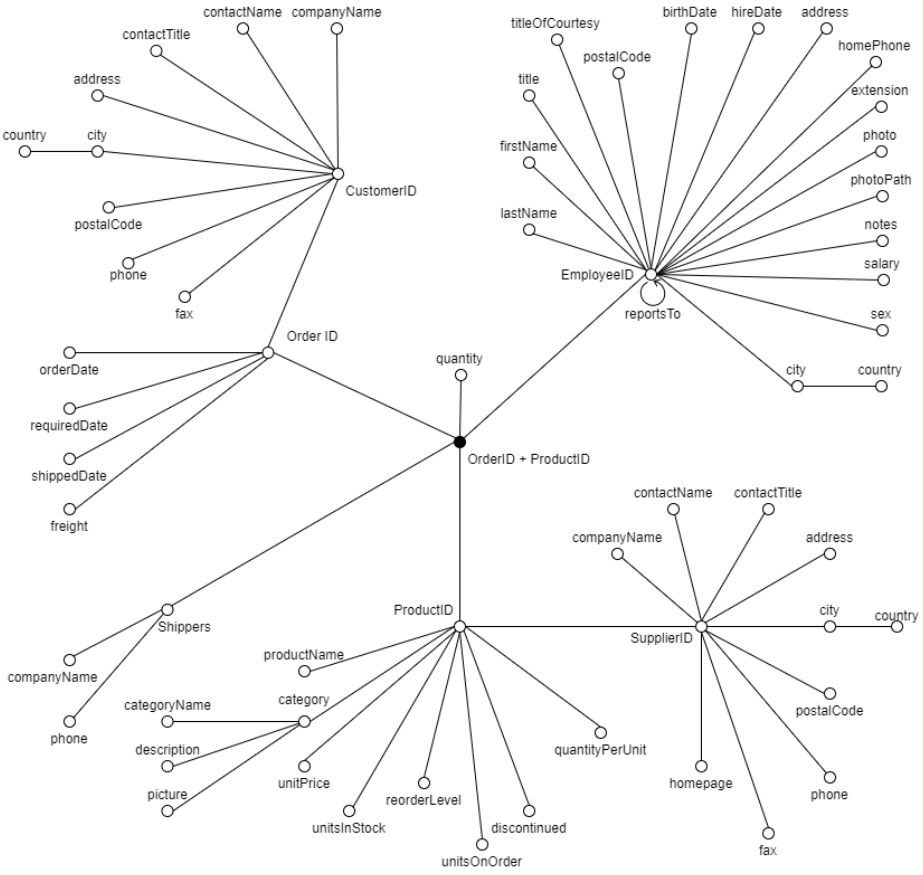

Figura 7: Albero degli attributi iniziale

Al fine di modellare il tutto in conformità alle richieste degli utenti finali, le operazioni di *editing* apportate all'albero, sono state le seguenti:

- rimozione degli attributi *phone* e *fax* in ogni occorrenza poiché poco influenti dal punto di vista informativo.
- In merito ai *suppliers* è stato rimosso l'attributo *homePage*, mentre sono stati posti ad attributi descrittivi: *companyName*, *contactName*, *contactTitle*, *address*, *postalCode*.
- Per quanto riguarda gli *employees* è stato generato l'attributo *sex* e sono stati rimossi gli attributi *photo*, *homePhone*, *extension*, *photoPath*, *hireDa-*

te, *birthDate* e *postalCode*; la rimozione delle date è avvenuta in quanto non apportano benefici dal punto di vista informativo in fase di analisi delle vendite. Sono stati unificati gli attributi *firstName* e *lastName* generando l'attributo completo *fullName*; *salary*, *address*, *title* e *fullName* diventano attributi descrittivi. É stata inoltre appiattita la gerarchia ricorsiva, poiché analizzando i dati non esistono *employees* che abbiano più di due *supervisor*, ovviamente sono stati opportunamente aggiunti gli archi opzionali.

- Per quanto riguarda *product*, gli attributi *productName* e *quantityPerUnit* sono stati resi descrittivi; sono stati potati *unitsInStock*, *unitsOnOrder* e *reorderLevel*; l'attributo *unitPrice* è stato reso descrittivo, inoltre è stato generato l'attributo *rangeUnitPrice* che tiene conto dell'intervallo in cui il prezzo unitario relativo al prodotto oscilla, potrebbe essere interessante effettuare analisi sulle vendite dei prodotti in base al loro prezzo di listino. Gli intervalli definiti sono stati fissati da 0 a 15, da 15 a 30 ed oltre 30, scelti empiricamente in modo tale da avere le cardinalità degli insiemi dei prodotti bilanciate. Relativamente a *category* è stato eliminato l'attributo *picture* e sono stati posti a descrittivi *categoryName* e *description*.

- In *shippers* è stato posto *companyName* ad attributo descrittivo;

- Relativamente agli ordini è stato potato l'attributo *OrderID* e sono stati innestati tutti i suoi figli al fatto di interesse, inoltre sono stati sostituiti gli attributi *shippedDate* e *requiredDate* con *daysToShip* che rappresenta i giorni che intercorrono tra la ricezione dell'ordine e la sua spedizione. É stata generata la gerarchia temporale (mese, trimestre e anno) a partire da *orderDate*.

- Sono state generate le gerarchie condivise a partire dagli attributi *city* e *sex*.

- Sono stati inseriti gli archi multipli per quanto riguarda i *territories* serviti dai diversi *employees*, ovviamente è stato anche inserito l'attributo *region*, figlio di *territories*.

A fronte delle modifiche sopra elencate, è stato ottenuto il seguente albero degli attributi e conseguentemente lo schema di fatto:

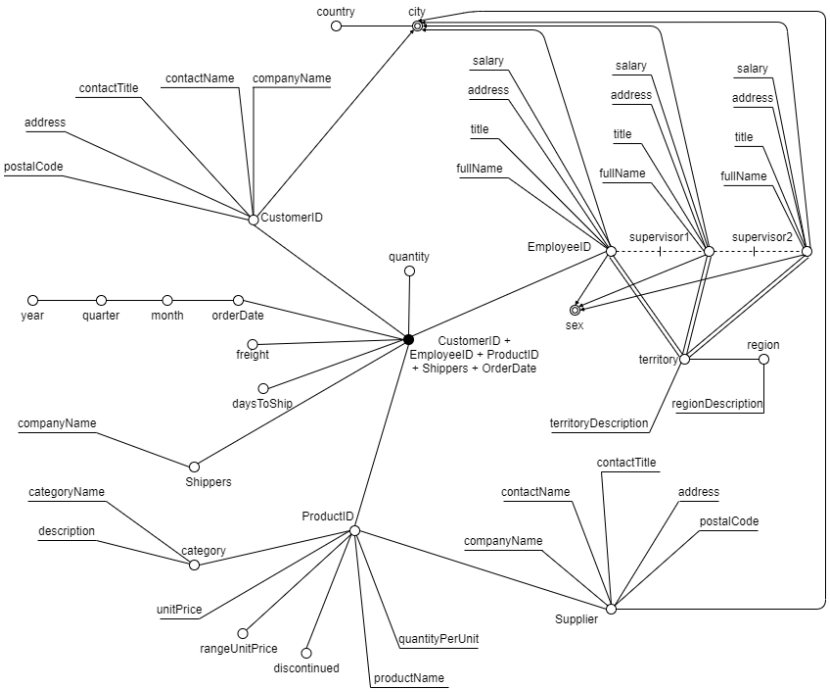

Figura 8: Albero degli attributi editato

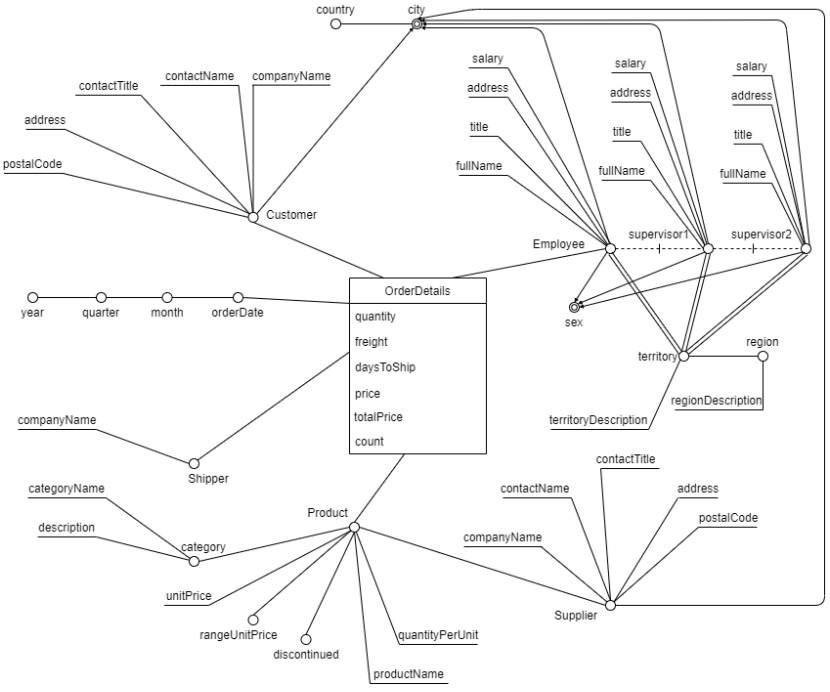

Figura 9: Dimensional Fact Model

In seguito alla potatura di *orders* si ha un aumento della granularità del fatto, e ciò implica una prospettiva di analisi rivolta ad un insieme di ordini che fanno capo ad un determinato *customer*, gestiti dal relativo *employee*, eseguiti in una specifica data, inerenti ad un prodotto, con la spedizione a carico di una designata compagnia. La scelta delle misure è ricaduta su:

- *quantity*: indica la quantità di prodotti presenti in un determinato ordine;
- *freight*: rappresenta la spesa di spedizione sostenuta;
- *daysToShip*: indica il tempo di consegna medio per un determinato ordine;
- *price*: misura derivata che fornisce informazioni sul prezzo di acquisto per il relativo ordine, ottenuta moltiplicando la quantità per il prezzo unitario del prodotto;
- *totalPrice*: misura derivata e rappresenta la somma del prezzo e delle spese di spedizione;
- *count*: misura di supporto che tiene conto del numero di tuple contenute all'interno dell'aggregazione.

Di seguito è riportata la tabella che fornisce indicazioni sull'additività delle misure rispetto alle varie gerarchie del fatto di interesse:

	quantity	freight	daysToShip	price	totalPrice	count
customer	SUM, AVG	SUM, AVG	AVG	SUM	SUM	SUM
employee	SUM, AVG	SUM, AVG	AVG	SUM	SUM	SUM
product	SUM, AVG	SUM, AVG	AVG	SUM	SUM	SUM
shipper	SUM, AVG	SUM, AVG	AVG	SUM	SUM	SUM
orderDate	SUM, AVG	SUM, AVG	AVG	SUM	SUM	SUM

Figura 10: Tabella analisi delle misure

7 Progettazione logica

Per rappresentare la struttura multidimensionale dei dati è stato adottato il modello ROLAP, in particolare è stato scelto di utilizzare uno *star schema* in quanto, le cardinalità dei valori degli attributi presenti all'interno delle *dimension tables* appartengono allo stesso ordine di grandezza, di conseguenza è stato deciso di non effettuare *snowflaking*. Avere utilizzato costrutti avanzati all'interno del DFM, ha consentito di impiegare soluzioni che garantissero in ogni caso un minimo livello di normalizzazione. Ad esempio, per quanto riguarda la gerarchia condivisa *city* è stata impiegata un'unica *dimension table* secondaria, cui fanno riferimento all'occorenza le varie *dimension table* primarie.

Maggiore attenzione è stata posta nell'implementazione della gerarchia ricorsiva (*reportsTo*), utilizzando una *navigate table* alimentata con tutti i possibili valori di *supervisor* ed *employee* e memorizzando inoltre il livello della gerarchia, per consentire le interrogazioni al fine di determinare i sottoposti, diretti ed indiretti di un *supervisor* e la foglia che consente di determinare se un dato impiegato abbia o meno subordinati. All'atto pratico è stato necessario aggiungere anche l'autoanello relativo a *supervisor* all'interno della *dimension table* primaria *employee*, perchè l'*OLAP Cube Engine Mondrian (Pentaho Schema Workbench)* non consente di generare la gerarchia utilizzando la sola *navigate table*, che nello specifico contesto prende il nome di *closure table*.

É stato scelto di implementare la gerarchia condivisa *sex* inserendo tale attributo all'interno della *dimension table* relativa agli *employee*, in quanto nel sistema informativo aziendale avendo a disposizione solo 9 impiegati, non è stato ritenuto opportuno appesantire il *Data-Warehouse* inserendo un'ulteriore tabella con le relative chiavi surrogate ed esterne.

Infine, sono stati implementati gli archi multipli per la gerarchia *employee-territory* mediante l'utilizzo di una *bridge table*, non è stato ritenuto conveniente aggiungere il peso nella relazione in quanto non si hanno informazioni sufficienti per stabilire l'importanza dei vari territori di vendita, ciò rende dunque impossibile effettuare interrogazioni pesate. Anche in questo caso, dal punto di vista pratico *Mondrian* non consente l'implementazione delle relazioni molti-a-molti, per questo motivo la scelta progettuale è stata quella di sviluppare una vista, in modo da raggruppare preventivamente tali informazioni in un'unica tabella, per poterla utilizzare al fine di effettuare interrogazioni su tale gerarchia.

Oltre alla vista necessaria per l'implementazione della *bridge table* all'interno del cubo OLAP, a valle di un'attenta analisi, sono state sviluppate le viste materializzate sulla base delle seguenti aggregazioni:

- $\{CustomerCity - ProductCategory\}$
- $\{Month - Shipper\}$
- $\{CustomerCity - RangeUnitPrice\}$
- $\{Customer - ProductDiscontinued\}$
- $\{OrderDate - CustomerCity - Product - Employee\}$
- $\{Quarter - Shipper - Employee\}$
- $\{Year - RangeUnitPrice - EmployeeTerritory\}$

La scelta è stata dettata dai requisiti specificati dagli utenti finali, oltre ad eventuali *group-by-set* selezionati al fine di rendere maggiormente efficienti le interrogazioni con i diversi operatori OLAP applicabili.

Di seguito è riportato lo *star schema* del *data warehouse*, implementato mediante l'utilizzo di *ApricotDB*:

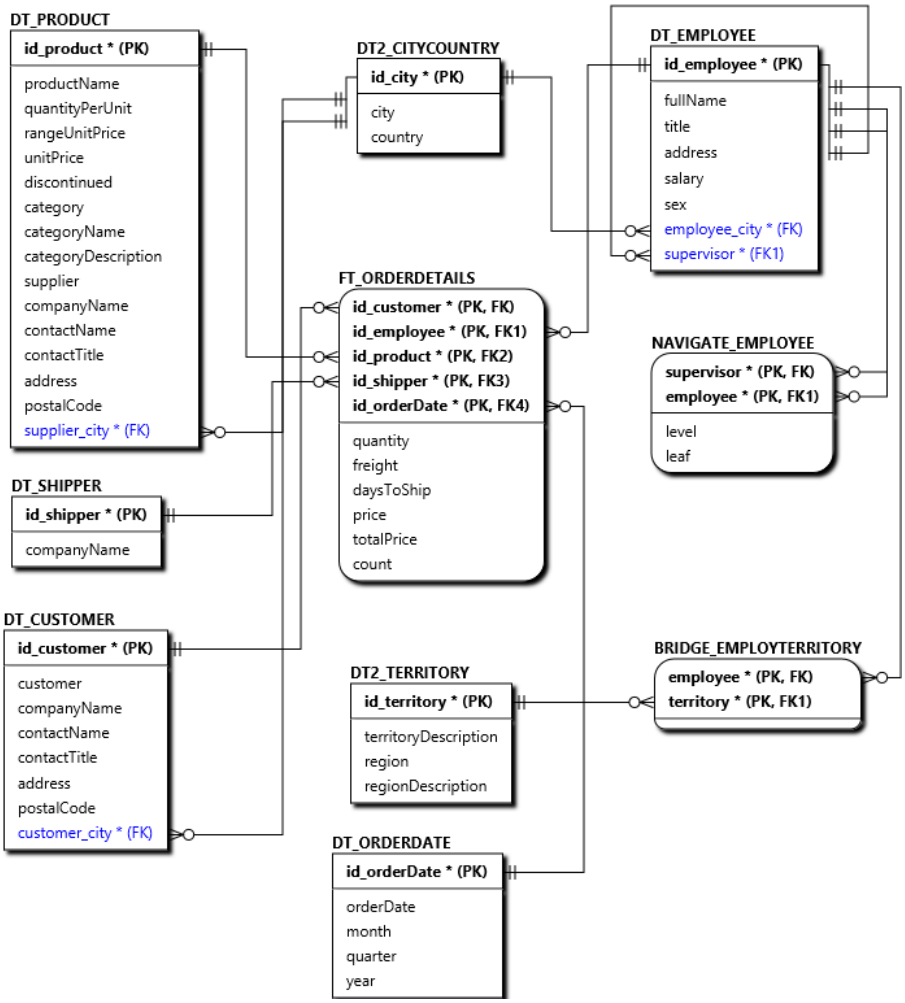

Figura 11: Schema a stella

8 Progettazione dell'alimentazione

La progettazione dell'ETL ha avuto luogo in diverse fasi distinte, cronologiacamente di seguito elencate. Al fine rispettare i vincoli di chiave esterna, sono state generate in primo luogo le *dimension tables* secondarie, in particolare la tabella *dt2_citycountry* e la tabella *dt2_territory*, non è stato ritenuto opportuno inserire delle chiavi surrogate per quanto riguarda la seconda, sono state dunque impiegate le chiavi primarie già presenti all'interno del database operazionale. Di seguito è riportata la trasformazione *kettle* necessaria ad ottenere le suddette *dimension tables*:

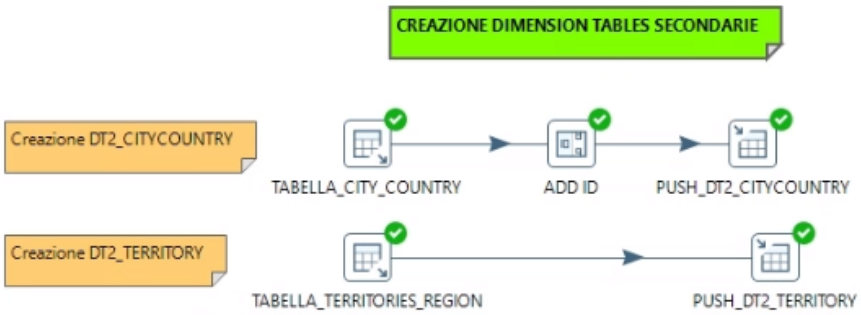

Figura 12: Creazione Dimension Tables secondarie

Successivamente sono state create ed opportunamente alimentate le *dimension tables* primarie, effettuando delle operazioni di *lookup* al fine di inserire i valori di chiavi surrogate relativi alla *dimension table* secondaria *dt2_citycountry*, ove necessario. Scelta obbligata, è stata la generazione di chiavi surrogate per quanto riguarda la gerarchia *dt_orderdate* e *dt_customer*. Per quanto riguarda le altre tabelle sono stati riutilizzati tutti gli identificatori già presenti all'interno del *database* operazionale. Al fine della costruzione della gerarchia relativa alla *dimension table dt_orderdate* sono state opportunamente generate le marche temporali. Di seguito è riportata la trasformazione *kettle* necessaria alla generazione delle *dimension tables* primarie:

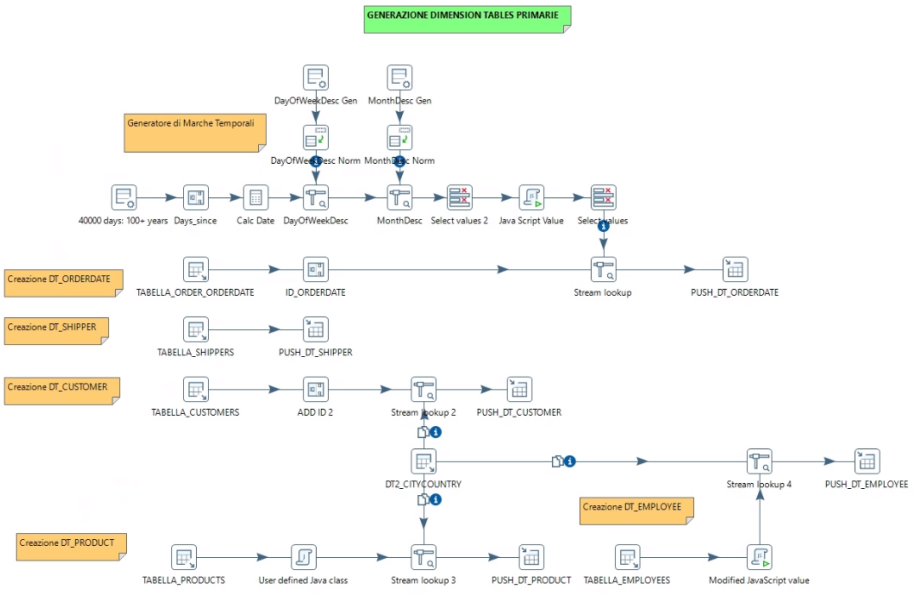

Figura 13: Creazione Dimension Tables primarie

Avendo ora a disposizione la *dimension table* primaria *dt_employee* e la *dimension table* secondaria *dt2_territory* è stato possibile sviluppare la trasfomazione per la costruizione della *bridge table* e della *navigate table*, al fine di implementare la gerachia ricorsiva e gli archi multipli. Per quanto riguarda la prima, era già disponibile all'interno del *database* operazionale, mentre per la generazione della seconda è stata eseguita una *query* SQL. Non è stato necessario effettuare alcuna operazione di *lookup* per reperire le chiavi esterne in quanto per entrambe le *dimension tables* sono state mantenute le chiavi primarie presenti nel *database* operazionale, per questo motivo i vincoli di integrità referenziale sono automaticamente rispettati. Di seguito è riportata la trasformazione *kettle* eseguita:

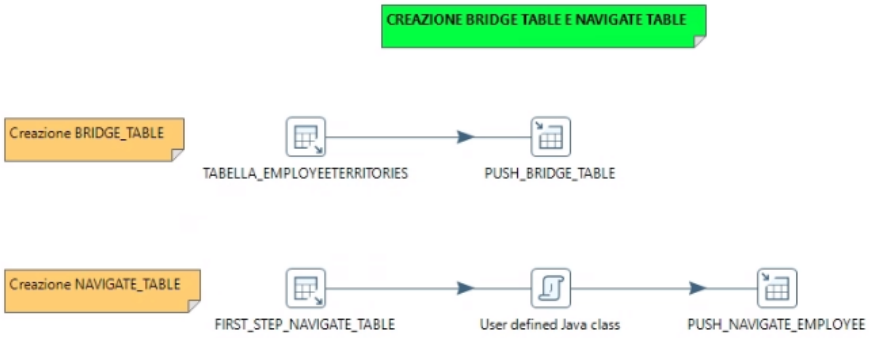

Figura 14: Creazione Navigate e Bridge Tables

Per quanto riguarda lo sviluppo della *fact table*, i dati sono stati pre-aggregati per reperire i valori corretti di misura in funzione delle gerarchie costruite, ed è stato necessario effettuare *lookup* per ottenere le chiavi surrogate unicamente delle *dimension tables*: *dt_customer* e *dt_orderDate*. Di seguito è riportata la trasformazione *kettle* eseguita:

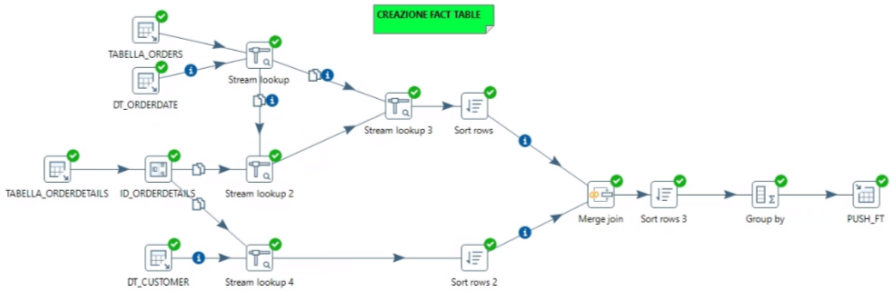

Figura 15: Creazione Fact Table

In ultima fase, come è possibile evincere dal *job* di seguito riportato, sono state eseguite in serie le vari operazioni per l'alimentazione del *Data-Warehouse*, precedute dallo *script* SQL addetto alla creazione delle tabelle sul DBMS. Ovviamente sono stati inseriti all'interno della sequenza anche i processi che hanno permesso la materializzazione ed il popolamento delle viste selezionate.

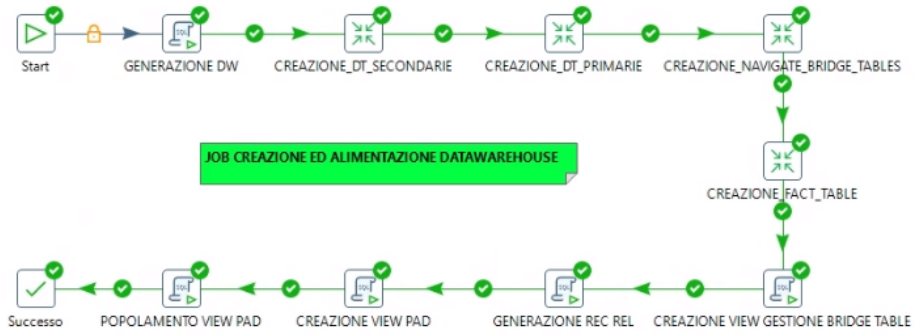

Figura 16: Definizione job per creazione e alimentazione del DW

Come è possibile evincere dal grafico di seguito riportato, mediante l'utilizzo dei sette aggregati, precedentemente definiti, vengono coperte un numero di righe pari a 12'257 con un consumo approssimativo di spazio su disco di $1.08MB$:

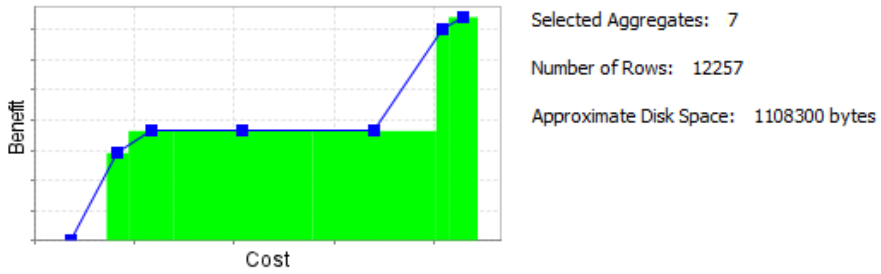

Figura 17: Diagramma viste costi-benefici

Il ridotto consumo di spazio dipende fortemente dalla ridotta dimensione del database operazionale ($0.96MB$). Sulla base del grafico è possibile notare che si ha un elevato costo dal punto di vista dello spazio occupato, ma un netto beneficio per quanto riguarda le aggregazioni sia in termini di righe sviluppate, che conseguentemente sui tempi di esecuzione delle diverse interrogazioni. Tuttavia, essendo il *trade-off* accettabile si è deciso di proseguire con le decisioni e considerazioni fatte precedentemente.

9 Progettazione fisica

Dal punto di vista implementativo, non è stato possibile svolgere in maniera opportuna questa fase, in quanto *mysql* presenta una forte limitazione sulla selezione delle tipologie di indici da adottare, in particolare è possibile impiegare soltanto *B-Tree* e *Hash-Index*. In ogni caso, al fine di rendere maggiormente efficienti le query da sottoporre al sistema, supponendo di utilizzare un tool differente, è possibile effettuare le seguenti scelte implementative:

- è consigliabile la costruzione di *bitmap-index* sulle colonne:

  dt_product.category;

  dt_orderDate.month;

  dt_orderDate.quarter;

  dt_orderDate.year;

  dt_employee.sex;

  dt2_citycountry.country;

  essendo la cardinalità di questi attributi molto ridotta, la dimensione occupata da questa struttura dati non dovrebbe essere eccessivamente elevata.

- É possibile implementare uno *star-index* tra la *fact table* e tutte le *dimension table* primarie, in modo tale da rendere maggiormente efficienti le query che pongono complessivamente in *join* tali tabelle.

- All'occorrenza è possibile implementare dei *bitmapped-join-index* per quanto riguarda tutte le *dimension table* primarie che hanno relazioni con la *dimension table* secondaria *dt2_citycountry*.

- Eventualmente è possibile implementare due *bitmapped-join-index* tra la *fact table* e le *dimension table* primarie *dt_product* e *dt_customer*, in quanto presentano la maggiore cardinalità.

Oltre alle considerazioni sopra elencate, non è stata ritenuta rilevante la selezione di ulteriori indici al fine di rendere maggiormente efficiente il sistema di *data-warehousing* sviluppato.