UNIVERSITA' DEGLI STUDI DI PADOVA

FACOLTA' DI SCIENZE STATISTICHE

Corso di laurea in
Statistica e gestione delle imprese

relazione finale

Realizzazione di un data mart orientato alle vendite con l'utilizzo di software open source

relatore: Prof. ssa Susi Dulli

laureando: Frison Simone

anno accademico 2007/2008

.... alla mia famiglia

SOMMARIO

INTRODUZIONE

In un mercato ogni giorno più competitivo, il tradizionale approccio al marketing di massa sta diventando inefficace, poiché i clienti sono diventati più esigenti ed informati, e di conseguenza, meno fedeli. Le Aziende, quindi, vivono in modo sempre più pressante l'esigenza di analizzare i dati della propria gestione, attuale e passata, in modo da individuare le migliori strategie da adottare per essere presenti sul mercato in modo propositivo. Una risposta corretta e tempestiva a tale esigenza consentirebbe di suggerire in tempo utile gli eventuali interventi correttivi necessari in un processo produttivo, o di individuare nuove opportunità di mercato, o ancora di migliorare il controllo di qualità del prodotto assicurandosi la soddisfazione del cliente. In effetti ogni giorno le Aziende trattano e accumulano grandi quantità di dati: questi dati però spesso rimangono un patrimonio che non riesce ad esprimere proprio le informazioni di cui i Manager vorrebbero disporre per migliorare le decisioni. Poter disporre di informazioni sufficienti in modo tempestivo e fruibile è reso possibile oggigiorno dagli strumenti della così detta Business Intelligence. Tali tecniche garantiscono anche la possibilità di analizzare tali informazioni così che il Manager assicuri per l'Azienda un impatto positivo sulle strategie, sulle tattiche e sulle operazioni di business. Nell'ambito della Business Intelligence un'attività fondamentale è la raccolta dei dati aziendali. Questa raccolta non si deve limitare ai soli dati transazionali, generati e usati nei processi produttivi o operativi di un'impresa, ma deve essere orientata anche ai dati decisionali (o business data), caratterizzati da una natura aggregata, una struttura flessibile, un uso non ripetitivo, un orizzonte temporale più ampio. A questo scopo si rende necessaria la progettazione e la costruzione di un "magazzino dei dati" (Data Warehouse) che, attingendo periodicamente sia dal sistema transazionale aziendale sia da altre sorgenti informative, raccolga e sintetizzi le informazioni secondo regole ben definite dettate dal proprio business, e poi le organizzi in una forma comprensibile per chi in azienda deve prendere decisioni tattiche e/o strategiche. In questo modo si utilizza in pieno l'informazione, garantendo sempre tempi di risposta accettabili in un'analisi interattiva di tipo manageriale, e al tempo stesso salvaguardando l'integrità del sistema transazionale di partenza che non viene caricato di responsabilità e funzionalità diverse da quelle per cui è stato progettato. Le attuali soluzioni di Data Warehousing, però, sono molto costose e quindi spesso non sostenibili dalle Piccole e Medie Imprese (PMI), spesso comunque costrette a competere su mercati globali, pur senza disporre della struttura e delle risorse delle grandi Aziende. Questo

progetto intende offrire alle PMI un nuovo approccio al Data Warehousing che consenta ai manager di accedere ai dati di business in modo veloce, consistente ed interattivo. Il Data Warehousing (o meglio il data mart) proposto è caratterizzato da costi ragionevoli, tempi di realizzazione ridotti, e minor mole di dati immagazzinati. Sulla base delle strutture informative rese disponibili dal Data Warehouse, questo Progetto offrirà all'utente finale anche alcuni strumenti di analisi dei dati a supporto del processo decisionale aziendale.

OBBIETTIVI

Questo progetto si è svolto nell'ambito di collaborazione con l'azienda Gruppo Valente S.p.A. di Campodarsego (PD) che mi ha incaricato di realizzare uno strumento per facilitare l'analisi delle vendite, in particolare per osservare i dati di fatturato nell'arco degli ultimi 4 anni e nello stesso tempo visionare le vendite stesse per ogni zona fino al dettaglio della provincia.

L'azienda desiderava avere uno strumento flessibile in grado di rispondere velocemente alle richieste e che permetta di esplodere i dati fino ad arrivare al dettaglio.

Lo strumento avrebbe dovuto essere in grado di soddisfare richieste del tipo:

- Quanto ho fatturato in Italia nel 2003? E nella regione X nello stesso anno? E nelle relative province?
- Quali prodotti ho venduto nella provincia Y? In che quantità e che fatturato mi hanno realizzato?
- I clienti/rivenditori della Regione Z, quali sono? Quanto hanno fatturato? Quali prodotti hanno acquistato nel corso degli anni? In che quantità?

L'azienda in passato per estrarre i dati richiesti, si serviva di semplici report statici presenti nel gestionale da loro utilizzato, Microsoft Navision (rinominato recentemente in Microsoft Dynamics).

L'esigenza però è quella di avere una sorta di strumento dinamico in grado di soddisfare una vasta gamma di richieste e che non richieda la costruzione di una query ad hoc ogni qual volta si presenti una nuova richiesta.

La mia proposta è stata quella di fornire un navigatore OLAP, a mio parere adeguato per la risoluzione di questo tipo di problematiche, a costi contenuti o nulli e che offrisse buone prestazioni, flessibilità e scalabilità.

Nel corso della relazione verranno descritte le fasi di sviluppo.

CAPITOLO 1: CICLO DI VITA DEI SISTEMI DI DATAWAREHOUSING

I sistemi di data warehousing stanno acquisendo popolarità via via che aziende e imprese nei più disparati settori ne intuiscono i benefici. D'altronde, larga parte delle organizzazioni mancano della necessaria esperienza e capacità per affrontare con successo le sfide implicite nei progetti di data warehousing. In particolare, uno dei fattori che tutt'oggi maggiormente minaccia la riuscita di questi progetti è la mancata adozione di una approccio metodologico, che riproponga l'esperienza fatta da altri e minimizzi i rischi di insuccesso, essendo basato su un'analisi costruttiva degli errori commessi.

Verranno illustrate alcune metodologie per la gestione dell'intero ciclo di vita dei sistemi di data warehousing e alla definizione di un quadro metodologico per lo sviluppo di ciascun data mart [2].

1.1 FATTORI DI RISCHIO

E possibile evidenziare quattro categorie di rischio:

- Rischi legati alla gestione del progetto. Un progetto di data warehousing coinvolge trasversalmente tutti i livelli aziendali e porta con sé importanti implicazioni politiche e organizzative. Da questo punto di vista, un fattore che spesso ostacola l'attuazione dei progetti di data warehousing è la necessità di condivisione dell'informazione tra i reparti, che viene vissuta con riluttanza dai responsabili aziendali poiché può comportare "perdita di potere" e far emergere le carenze del proprio operato. Altre frequenti cause di insuccesso sono legate alla definizione dell'ambito e delle finalità del sistema e alla sua sponsorizzazione; in particolare, parecchi progetti in aziende medio-piccole sono stati stroncati sul nascere dall'incapacità del progettista di presentare una convincente valutazione dei costi e dei benefici.
- Rischi legati alle tecnologie. Le tecnologie per la progettazione, l'implementazione, l'uso e il mantenimento dei sistemi di data warehousing si stanno evolvendo rapidamente, e l'architettura proposta deve essere in grado di tenere il passo. Problemi classici a questo livello sono la scarsa scalabilità dell'architettura in termini di volume dati e numero di utenti; l'assenza di estendibilità per accogliere nuove tecnologie, componenti o applicazioni; la gestione inefficace dell'interscambio di meta-dati[1] tra i diversi componenti.

1 letteralmente "dato su un (altro) dato", è l'informazione che descrive un insieme di dati.

- Rischi legati ai dati e alla progettazione. Questi fattori di rischio dipendono dalla qualità dei dati e dei progetto realizzato. Citiamo in particolare il rischio di ottenere risultati di scarso valore a causa dell'instabilità e inaffidabilità delle sorgenti, oppure a causa dell'inaccurata specificazione dei requisiti utente. Un altro frequente motivo di insuccesso dipende dall'incapacità di fornire un elevato valore aggiunto agli utenti alla consegna dei primi prototipi, che mina la fiducia nell'intero progetto.
- Rischi legati all'organizzazione. Assolutamente non ultime in ordine di importanza tra le cause di insuccesso, l'incapacità di impegnare l'utente finale in un reale interesse e sostegno per il progetto, la difficoltà di trasformare radicalmente la cultura aziendale per renderla conscia del valore dell'informazione, l'impossibilità degli utenti di trarre vantaggio dai risultati ottenuti causa inerzia organizzativa.

Complessivamente, il rischio di ottenere un risultato insoddisfacente nei progetti di data warehousing è particolarmente alto proprio a causa delle elevatissime aspettative degli utenti. Nella cultura aziendale contemporanea, è infatti diffusissima la credenza che attribuisce al data warehousing il ruolo di panacea in grado di sanare tutte le incongruità dell'organizzazione e le manchevolezze del sistema informativo aziendale. In realtà, come si è appena visto, una larga parte della responsabilità della riuscita del progetto ricade sulla qualità dei dati sorgente e sulla lungimiranza, disponibilità e dinamismo del personale dell'azienda.

1.1.1 APPROCCIO METODOLOGICO AL DATA WAREHOUSE

Prima di affrontare la realizzazione di un progetto di un datawarehouse occorre definire una “strategia”. Gli approcci generalmente utilizzati per l'implementazione sono:

- approccio top down
- approccio bottom up
- approccio incrementale

Approccio top down

L’approccio top-down è quello che prevede una implementazione estensiva del sistema, il cui disegno originale tiene in considerazione tutte le principali aree di interesse aziendale.
Si parla in questo caso di Enterprise data warehouse, che può essere successivamente suddiviso in un insieme di data mart, per motivi tecnici ed organizzativi, pur mantenendo rigorosamente accorpata la visione totalitaria d’insieme della soluzione.

I data mart dipendenti costituiscono un subset di dati aziendali altamente specializzati per aree di interesse o dipartimenti aziendali.

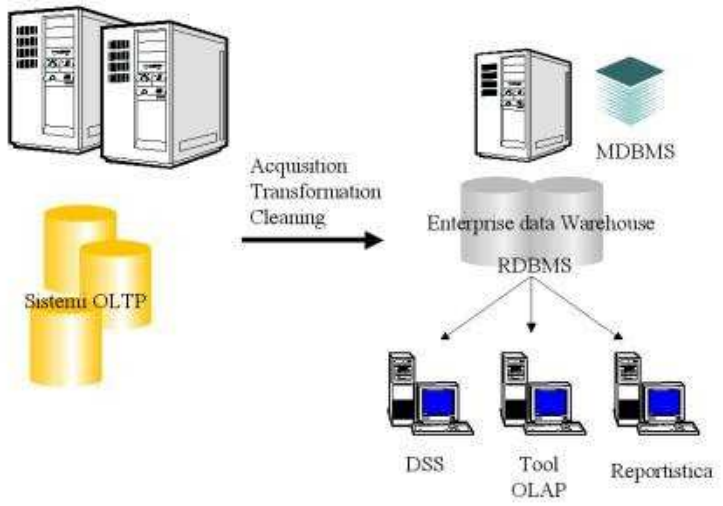


figura 1.1 – approccio top down [4]

Il punto debole di questo approccio teoricamente rigoroso è nella difficoltà di gestione del progetto onnicomprensivo, che rischia di paralizzare l'attività e di fornire risultati troppo in avanti nel tempo. Questo tipo di metodo (caldamente suggerito da Bill Inmon [1]) rappresenta l'approccio migliore per la costituzione del data warehouse e va valutata la sua applicabilità relativamente allo sforzo che il cliente prevede di sostenere in termini di tempi e costi.

Approccio bottom-up

L'approccio bottom-up prevede un'implementazione non coordinata nella quale ogni data mart viene realizzato per rispondere ad uno specifico fabbisogno informativo di una utenza dipartimentale.

In questo caso l'Enterprise data warehouse è il risultato dell'insieme dei singoli data mart indipendenti, che si alimentano direttamente dai sistemi operazionali.

Il vantaggio di tale approccio pragmatico è unicamente nel conseguire risultati utili per l'utente in un arco temporale limitato con costi diretti relativamente contenuti.

Tale approccio può essere valutato solo per progetti con budget di sviluppo molto bassi e, soprattutto, il cliente deve essere ben informato sui rischi del costo di manutenzione di un data warehouse di tale tipo.

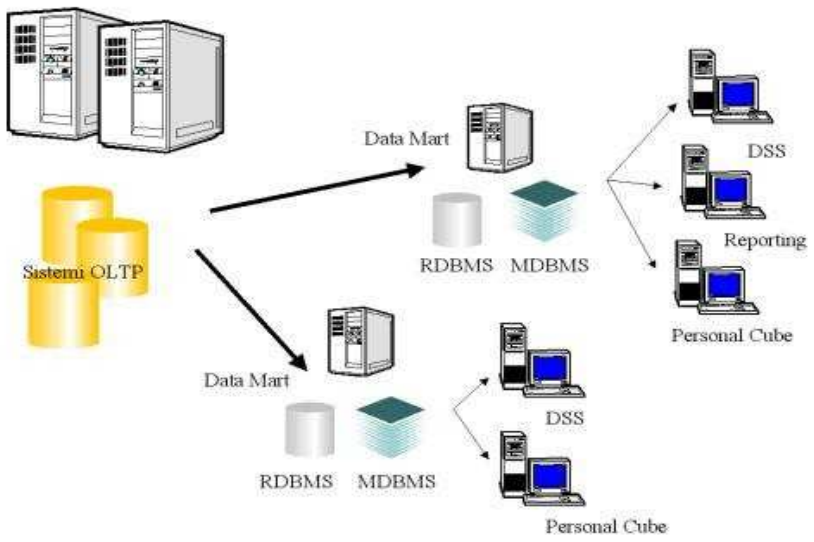


figura 1.2 – approccio bottom-up [4]

Approccio “incrementale” : Enterprise data mart Architecture (EDMA)

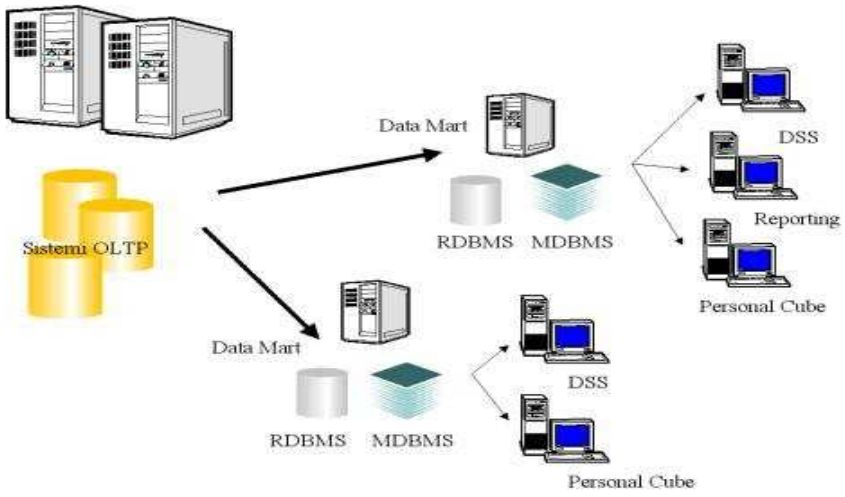


figura 1.3 – approccio incrementale [4]

L’approccio “incrementale” combina i vantaggi dei due approcci sopra descritti (e vede in Ralph Kimball [4] il primo sostenitore). Alla base di questo approccio, definito in letteratura anche approccio “federato”, vi è infatti la creazione di un modello informativo comune.
Dal modello informativo comune vengono sviluppati in maniera coerente modelli dati dell’Enterprise data warehouse e/o dei data mart; questi ultimi possono essere sia dipendenti che indipendenti.

L’implementazione prevede di mettere a fattore comune tra diversi progetti di data mart i processi di acquisizione di dati dai sistemi transazionali.
E' tuttavia fondamentale segnalare che è comunque opportuno consigliare l'eliminazione data mart indipendenti, non appena sviluppati quelli derivanti direttamente dall'ambiente integrato del data warehouse; è importante verificare tempi e costi dello sviluppo temporaneo di data mart indipendenti.
Il risultato dei processi di acquisizione viene centralizzato su aree di appoggio comuni (cosiddette aree di staging) su cui vengono svolti i successivi processi di trasformazione.
Il modello informativo comune e la fruizione delle aree di staging minimizzano i problemi di integrazione tra data mart mentre l’utilizzo di un’architettura data warehouse Bus consente l’individuazione e la condivisione di dimensioni e fatti razionalizzate rispetto ai processi aziendali.
Il concetto di Warehouse Bus (sviluppato da Ralph Kimball [4]) è “l'uovo di Colombo” che in molti casi consente un approccio incrementale paragonabile nei risultati a quello Top Down di Inmon [1].
Alla definizione di dettaglio nel data warehouse di un particolare processo basato sul concetto di dimensioni e fatti conformi è possibile rilasciare velocemente il data mart aggregato, certi che le successive attività a livello di data warehouse non potranno creare isole informative. In questo tipo di approccio si rivela strategico la scelta del primo data mart che farà da guida a tutti i successivi (generalmente il primo è quello che ha come oggetto di interesse le vendite).

APPROCIO ADOTTATO: Si è scelto di adottare una metodologia di tipo bottom-up per riuscire ad ottenere velocemente il risultato desiderato anche in considerazione del fatto che l'interesse non è quello di avere uno data warehouse completo, ma limitato ad analizzare le sole vendite come richiesto dall'azienda.

1.2 IL “BUSINESS DIMENSIONAL LIFECYCLE”

Il business dimensionai lifecycle è il ciclo di vita per la progettazione, lo sviluppo e l'attuazione di sistemi di data warehousing riportato da Kimball [6]. Evolutosi nel corso di quasi vent'anni di esperienza professionale, ha attualmente la struttura evidenziata in figura 1.4.
Il punto fondamentale di questo approccio è che i tre percorsi sono paralleli così da non influenzarsi a vicenda evitando quindi di aspettare la fine fine di un percorso per iniziarne

un altro. Durante tutte le fasi del ciclo di vita, un'accurata gestione del progetto permette di mantenere le diverse attività sincronizzate, di monitorare lo stato corrente del progetto e di assicurare che il team dei progettisti resti costantemente in stretto contatto con l'organizzazione aziendale. Le fasi presenti nel modello sono:

- La *pianificazione* del progetto include la definizione degli scopi e dei confini del sistema, la valutazione degli impatti organizzativi, la stima dei costi e dei benefici, l'allocazione delle risorse necessarie e infine l'identificazione di un piano di massima per l'attuazione del progetto.
- La *definizione dei requisiti* ha un ruolo primario nell'assicurare che le richieste degli utenti siano correttamente e interamente recepite dai progettisti, così da massimizzare l'utilità e la redditività del sistema in via di definizione. Durante questa fase, i progettisti devono catturare i fattori chiave del processo decisionale e trasformarli in specifiche che guidino la progettazione. La definizione dei requisiti dà il via a tre percorsi paralleli, ciascuno articolato in differenti fasi: quello dei dati, quello della tecnologia e quello delle applicazioni.

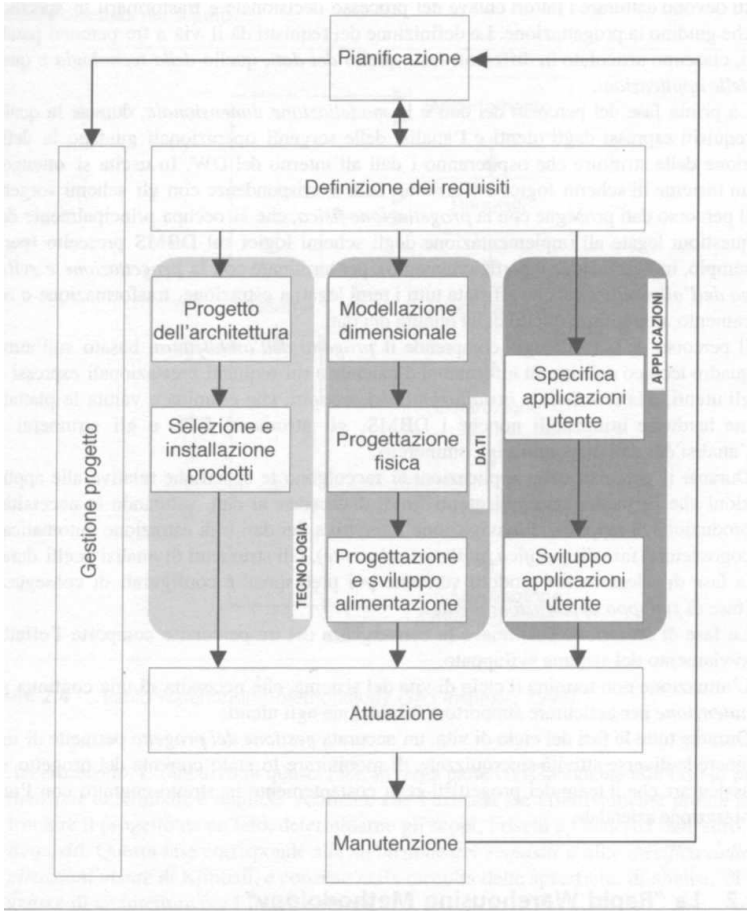


figura 1.4 – Il "Business Dimensional Lifecycle" [2]

- La prima fase del percorso dei dati è la *modellazione dimensionale*, durante la quale i requisiti espressi dagli utenti e l'analisi delle sorgenti operazionali guidano la definizione delle strutture che ospiteranno i dati all'interno del DW. In uscita si ottengono un insieme di schemi logici e l'insieme delle corrispondenze con gli schemi sorgente. Il percorso dati prosegue con la *progettazione fisica*, che si occupa principalmente delle questioni legate all'implementazione degli schemi logici sul DBMS prescelto (per esempio, indicizzazione e partizionamento), per terminare con la *progettazione e sviluppo dell'alimentazione* che affronta tutti i temi legati a estrazione, trasformazione e caricamento e non ultimo quello della qualità dei dati.
- Il percorso della tecnologia comprende il *progetto dell'architettura*, basato sull'attuale quadro tecnico dei sistemi informatici d'azienda e sui requisiti prestazionali espressi dagli utenti, e la *selezione e installazione dei prodotti*, che esamina e valuta le piattaforme hardware utilizzabili nonché i DBMS, gli strumenti ETL e gli strumenti per l'analisi dei dati disponibili in commercio.
- Durante il percorso delle applicazioni si raccolgono le specifiche relative alle applicazioni che permetteranno agli utenti finali di accedere ai dati, valutando le necessità di produzione di rapporti, di navigazione interattiva dei dati e di estrazione automatica di conoscenza (*fase di specifica applicazioni utente*). Gli strumenti di analisi scelti durante la fase di selezione dei prodotti verranno poi predisposti e configurati di conseguenza (*fase di sviluppo applicazioni utente*).
- La fase di *attuazione* costituisce la convergenza dei tre percorsi e comporta l'effettivo avviamento del sistema sviluppato.
- L'attuazione non termina il ciclo di vita del sistema, che necessita di una continua *manutenzione* per assicurare supporto e formazione agli utenti.

1.3 LA “RAPID WAREHOUSING METHODOLOGY”

La *rapid warehousing methodology* è una metodologia iterativa ed evolutiva per la gestione di progetti di data warehousing, pensata da una delle aziende leader del settore (SAS Institute). Essa si basa sulla suddivisione di progetti vasti e potenzialmente rischiosi in sotto progetti, più piccoli e molto meno rischiosi, chiamati *build*. Ciascun sotto progetto riprende l'ambiente sviluppato dal build precedente, estendendolo per conseguire nuove funzionalità e facendolo evolvere per far sì che continui a soddisfare i mutevoli bisogni degli utenti. Questo approccio mantiene vivo nel tempo il coinvolgimento e l'interesse degli utenti, gettando le basi per il successo a lungo termine del progetto.

Le fasi individuate sono:

- *Accertamento*. L'obiettivo di questa fase, in larga parte corrispondente alla fase di pianificazione di Kimball, è duplice: verificare che l'azienda sia effettivamente pronta ad affrontare il progetto da un lato, determinarne gli scopi, i rischi e i benefici dall'altro.
- *Requisiti*. Questa fase corrisponde alla definizione dei requisiti e alla specifica delle applicazioni utente di Kimball, e consiste nella raccolta delle descrizioni analitiche di progetto e di architettura per l'intero sistema.
- *Progettazione*. L'attenzione è ora concentrata su un solo sotto progetto per volta. Le specifiche di analisi vengono raffinate per generare il progetto logico e fisico dei dati e il progetto dell'alimentazione; vengono inoltre selezionati gli strumenti di implementazione.
- *Costruzione*. Il DW viene implementato e popolato con i dati estratti dalle sorgenti, le applicazioni di front-end vengono sviluppate e collaudate.

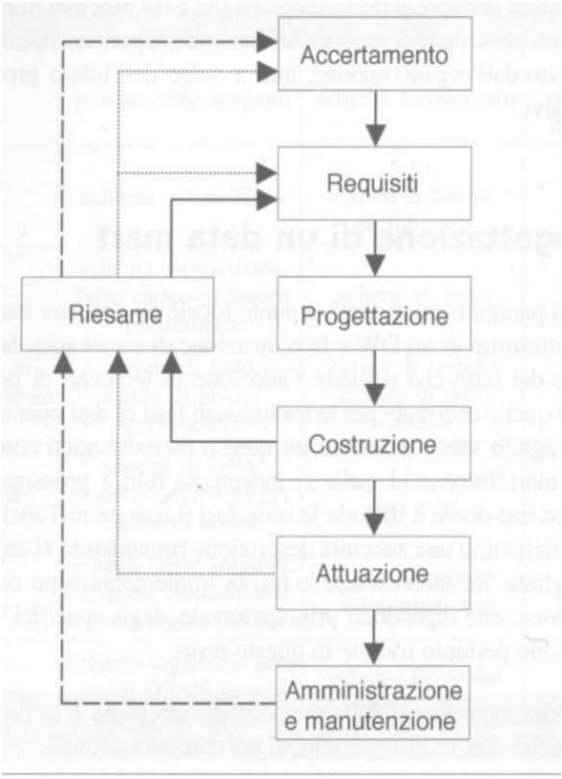


figura 1.5 – il modello “Rapid Warehouse Methodology” si SAS institute [2]

- *Attuazione*. Finalmente il sistema viene consegnato e avviato, dopo che gli utenti sono stati adeguatamente addestrati.
- *Amministrazione e manutenzione*. Questa fase continua durante tutta la vita del

sistema, e prevede l'estensione delle sue funzionalità, il ridimensionamento dell'architettura per venire incontro ai nuovi fabbisogni, il controllo della qualità dei dati.

- *Riesame*. Ciascun sotto progetto si accompagna a tre processi di riesame: uno a verifica dell'implementazione, uno a seguito dell'attuazione per accertarsi che il sistema sia stato ben accettato dall'organizzazione, uno a valle dell'intero processo per misurarne i benefici effettivi.

1.4 LA PROGETTAZIONE DI UN DATA MART

Il punto focale di ciascuna iterazione prevista durante lo sviluppo bottom-up di un DW è la costruzione di un singolo data mart, processo reso complesso anche dal fatto che richiede l'adozione di tecniche di progettazione completamente diverse da quelle utilizzate per le tradizionali basi di dati operazionali.

Nella tabella sottostante vengono riassunte le sette fasi di progettazione di un data mart:

Fase	*Ingresso*	*Uscita*	*Figure coinvolte*
Analisi e riconciliazione delle fonti dati	schemi delle sorgenti	schema riconciliato	progettista; amministratori db operazionale
Analisi dei requisiti	schema riconciliato	fatti; carico di lavoro preliminare	progettista; utenti finali
Progettazione concettuale	schema riconciliato; fatti; carico di lavoro preliminare	schemi di fatto	progettista; utenti finali
Raffinamento carico di lavoro, validazione schema concettuale	schemi di fatto; carico di lavoro preliminare	carico di lavoro; schemi di fatto validati	progettista; utenti finali
Progettazione logica	schemi di fatto; modello logico target; carico di lavoro	schema logico del data mart	progettista
Progettazione dell'alimentazione	schemi delle sorgenti; schema riconciliato; schema logico del d.m.	procedure di alimentazione	progettista; amministratori db operazionale
Progettazione fisica	schema logico del data mart; DBMS target; carico di lavoro	schema fisico del data mart	progettista

Tabella 1.1 – le fasi di progettazione di un data mart [3]

1.4.1 Analisi e riconciliazione delle fonti dati

La prima fase della progettazione richiede di definire e documentare lo schema del livello dei dati operazionali a partire dal quale verrà alimentato il data mart: quello che dal punto di vista architetturale viene chiamato livello dei dati riconciliati. Occorre quindi analizzare e comprendere gli schemi delle sorgenti disponibili (*ricognizione*), eventualmente trasformarli per portare alla luce correlazioni utili precedentemente inespresse (*normalizzazione*), determinare quali porzioni siano utili ai fini del processo decisionale nel

settore aziendale cui il data mart è dedicato e infine valutare la qualità dei dati. Qualora le sorgenti da utilizzare siano più d'una, i loro schemi dovranno inoltre essere sottoposti a un'attività di omogeneizzazione e integrazione tesa a individuare i tratti comuni e a sanare le eventuali inconsistenze. Oltre al progettista, in questa fase sono coinvolti gli amministratori dei database operazionali, che tipicamente sono gli unici in grado di attribuire un significato a schemi e tracciati record spesso incomprensibili; inoltre, la loro conoscenza del dominio applicativo è indispensabile per normalizzare gli schemi. La tipica carenza o addirittura assenza di documentazione delle basi di dati operazionali fa sì che questa fase risulti essere la più lunga (può infatti richiedere anche il 50% dei tempi dell'intero progetto) e, in genere, la più ardua.

1.4.2 Analisi dei requisiti

Nella fase di analisi dei requisiti il progettista raccoglie e filtra i requisiti degli utenti finali, con l'obiettivo di delineare quali sono le informazioni di interesse. In uscita da questa fase vengono prodotte specifiche sui fatti da modellare e indicazioni preliminari sul carico di lavoro.

I fatti sono concetti di interesse primario per il processo decisionale; le loro occorrenze sono eventi che accadono dinamicamente nell'azienda. La scelta dei fatti è essenzialmente responsabilità dell'utente finale, guidato in questo compito cruciale dal progettista sulla base della documentazione del livello riconciliato prodotta al passo precedente. Per ogni fatto occorre definire l'intervallo di storicizzazione, ovvero quale arco temporale dovranno abbracciare gli eventi memorizzati.

Il carico di lavoro preliminare può essere espresso in linguaggio pseudo-naturale e ha l'obiettivo di permettere al progettista di identificare dimensioni e misure per la progettazione concettuale; per ciascun fatto, esso dovrebbe specificare le misure quantitative e le aggregazioni più interessanti. Naturalmente abbinata alla determinazione del carico di lavoro è la scelta del livello minimo di sintesi delle informazioni, ovvero la definizione della granularità nella rappresentazione dei fatti, la cui adeguatezza costituisce un fattore critico per la riuscita dell'intero progetto poiché determina la flessibilità di interrogazione del data mart. La granularità è determinata da un compromesso tra velocità di risposta richiesta al sistema e livello di dettaglio massimo delle interrogazioni. Quando ciò non appesantisce troppo il sistema può essere conveniente utilizzare una granularità più fine di quella necessaria agli utenti, poiché ciò conferisce una maggiore flessibilità di utilizzo.

1.4.3 Progettazione concettuale

Secondo l'approccio seguito indicato dalla teoria e adottato in buona parte nello sviluppo di questa applicazione, la progettazione concettuale comporta l'utilizzo dei requisiti utente catturati durante la fase precedente per disegnare, a partire dallo schema riconciliato, uno schema concettuale per il data mart. Il modello concettuale adottato è il *Dimensional Fact Model* che prevede la creazione di uno schema di fatto per ciascun scenario di interesse evidenziato dall'utente. Uno schema di fatto è in grado di descrivere graficamente tutti i concetti del modello multidimensionale: fatti, misure, dimensioni e gerarchie; comprende inoltre un insieme di costrutti avanzati in grado di rappresentare accuratamente le diverse sfumature concettuali che caratterizzano gli scenari reali più complessi.

1.4.4 Raffinamento del carico di lavoro e validazione dello schema

Al termine della fase di progettazione concettuale occorre raffinare il carico di lavoro, già espresso in forma preliminare, formulando ciascuna interrogazione direttamente sullo schema concettuale. Ciò permette di verificare che tutte le interrogazioni previste siano effettivamente esprimibili, e quindi in ultima analisi di validare lo schema concettuale prodotto al passo precedente.

Tale fase discute inoltre il problema della determinazione del volume dati, che riveste fondamentale importanza per le successive fasi di progettazione logica e fisica.

1.4.5 Progettazione logica e fisica

Sebbene la successione delle tre fasi di progettazione concettuale, logica e fisica sia apparentemente simile a quella che regola il processo di costruzione di una base di dati operazionale, è importante evidenziare che la progettazione di un data mart segue principi molto diversi, poiché diverse sono le caratteristiche del carico di lavoro.
Evidentemente, requisito essenziale per affrontare la progettazione logica è la scelta del modello logico di riferimento: si tratta cioè di scegliere tra un'implementazione ROLAP[2] e una MOLAP[3]. Il resto della trattazione sarà incentrato sulla prima, di gran lunga più diffusa sul mercato, e sarà di conseguenza riferito al modello relazionale. Una volta definito lo schema logico di base secondo il cosiddetto schema a stella, la tecnica che maggiormente impatta sulle prestazioni è la materializzazione delle viste; può inoltre risultare vantaggioso applicare tecniche di frammentazione verticale e orizzontale.
Entrambe le tecniche sono guidate dal carico di lavoro presunto per il data mart e dal

2 sistema OLAP che lavora direttamente con database relazionali
3 sistema OLAP che utilizza un database di riepilogo con uno specifico motore per l'analisi multidimensionale

volume dati stimato.

Il problema di maggior rilievo durante la progettazione fisica è la scelta degli indici da costruire per ottimizzare le prestazioni. In questa fase non è più sufficiente avere scelto il modello relazionale come piattaforma a livello logico ma occorre anche riferirsi a uno specifico DBMS. Anche per la progettazione fisica, il carico di lavoro e il volume dati svolgono un ruolo sostanziale.

1.4.6 Progettazione dell'alimentazione

Durante questa fase vengono prese, di concerto con gli utenti e gli amministratori del sistema informatico, tutte le decisioni che riguardano il processo di alimentazione del livello riconciliato, se presente, e del data mart. Tra queste, particolarmente importante è la scelta dell'intervallo di aggiornamento periodico del data mart a partire dalle sue sorgenti.

CAPITOLO 2: DALLA TEORIA ALLO SVILUPPO

2.1 PRIMA FASE DI ANALISI

La prima fase prevede l'esplorazione dei sistemi informativi presenti nell'azienda, in particolare modo il nostro oggetto di interesse sarà il software gestionale ERP[4].

2.1.1 Il software gestionale e il DBMS

L'azienda si avvale di un software gestionale prodotto da Microsoft denominato “Navision” che si serve del database management system (DBMS) “SQL Server 2000” prodotto sempre da Microsoft.

Ho provveduto a visionare i dati di vendita, ovvero le fatture emesse, attraverso il gestionale stesso inizialmente per capire quali dati avevo a disposizione.

Essenzialmente in ogni fattura è indicata l'anagrafica del cliente, la data di fatturazione e dell'ordine, gli articoli acquistati e infine l'agente che ha effettuato la vendita; inoltre ad ogni fattura naturalmente corrisponde un codice identificativo.

Vista dal gestionale una fattura si presenta come in figura 2.1

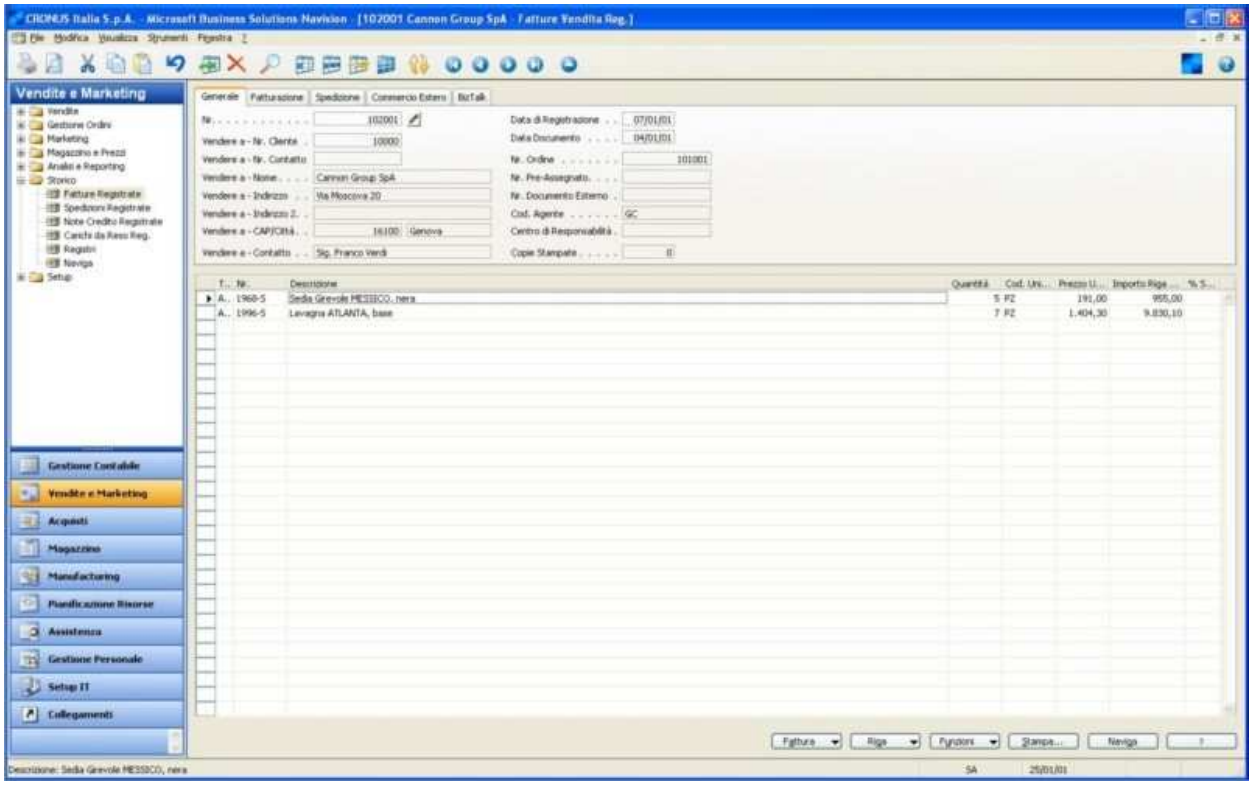


figura 2.1 – screenshot dell'ERP Microsoft Navision

4 Enterprise Resource Planning. Sistema che integra (o cerca di integrare) tutti i dati e i processi di una azienda all'interno di un sistema unificato

Dopo tale fase preliminare, ho proseguito ad analizzare le tabelle presenti nel database servendomi del tool grafico “Enterprise Manager” presente nel pacchetto del DBMS.
Sotto il punto di vista implementativo il database presenta una struttura relazionale piuttosto semplificata con poche tabelle e con tanti campi contenuti in esse, tale da farlo sembrare semi-denormalizzato.
Inoltre non esistono vincoli di integrità referenziale e controlli sui campi con tecniche di trigger in quanto i controlli sui dati vengono delegati al gestionale stesso.
Per esplorare le tabelle e osservarne il contenuto, mi sono servito di Microsoft Access con il quale mi sono collegato al DBMS attraverso il driver MS-SQL ODBC. Per ragioni di sicurezza ho creato un account con privilegi in sola lettura sui dati, onde evitare qualsiasi sorta di danno al database.
Non è stato difficile individuare le tabelle interessate contenenti i dati riguardanti le fatture in quanto presentano il nome in inglese e senza alcun tipo di abbreviazione; analogamente anche i campi fanno uso dello stesso principio.

2.1.2 La scelta di realizzare un data mart non virtuale

Come precisato pocanzi, il database presenta una struttura semi-denormalizzata e volendo avrei potuto sfruttare questa caratteristica per realizzare facilmente delle query “on-fly” al fine di creare un data mart di tipo virtuale (Virtual Data mart).
Ho però deciso di abbandonare questa soluzione e di servirmi di un nuovo DBMS installato una macchina diversa da quella del DBMS del gestionale e che avesse come obbiettivo quello di contenere solo i dati realmente necessari per l'analisi delle vendite in modo da ottenere migliori prestazioni; inoltre tale soluzione ha il vantaggio di essere completamente autonoma e indipendente dalle operazioni svolte dal software gestionale che già spesso sovraccarica di richieste il DBMS a esso collegato visti i numerosi utenti collegati.

2.1.3 Struttura logico-fisica delle vendite

Le vendite, oggetto del data mart, sono l'insieme delle fatture di vendita e delle note di credito. Ciascuna di essa è individuata da due tabelle: la prima contenente l'intestazione della fattura identificando il cliente, l'agente di vendita, la data di registrazione contabile; la seconda contiene le righe relative ai prodotti acquistati e il relativo prezzo di vendita.
Nelle tabelle relative alle vendite, oltre ad essere i presenti il codice del cliente, dell'agente e dei prodotti, vengono scritti anche la maggior parte dei campi delle tabelle d'origine dove

sono presenti per l'appunto i dati completi dei clienti, degli agenti e dei prodotti.
In pratica Navision al momento della compilazione di una fattura preleva la maggior parte dei campi e li ricopia anche nelle tabelle della fattura/nota di credito. Tale tecnica è utilizzata per ottenere una maggiore velocità e semplicità utile ad esempio nel momento in cui si volessero scorrere le fatture attraverso le form o realizzare report basati su query che potrebbero fare uso della singola tabella senza ricorrere a join fra più tabelle.

2.2 PROGETTAZIONE CONCETTUALE

In questa fase si individua il modello DFM più adatto tra *star schema* e *snowflake* analizzando vantaggi e svantaggi di ciascuno; si traccia poi un primo disegno dello schema concettuale.

2.2.1 Snowflake o no snowflake?

Una delle prime scelte in fase di progettazione di un data mart è la scelta del tipo di modello da adottare. La scelta ricade su due particolari modelli:

- star schema
- snowflake

Lo star schema prevede una tabella centrale, detta Fact-Table (FT), che determina l'oggetto dello studio e più tabelle di appoggio, denominate Dimensional-Table (DT), che rappresentano le dimensioni utilizzate per l'analisi.

- La chiave della FT è composta dalle chiavi delle varie DT.
- Le sottoparti della chiave della FT sono chiavi importate delle DT.
- Esiste una relazione di tipo 1-a-n tra le Dimension-Table e la Fact-Table
- L'accesso ai dati avviene tramite join tra le Dimension-Table e la Fact-Table

Il modello snowflake (figura 2.2) è invece ottenuto normalizzando una o più dimensioni dello star schema di partenza.
Essendo la mia prima esperienza di data warehousing, ho preferito affidarmi a documentazione esistente che trattasse l'argomento e ho preso spunto da un libro [1] che tratta la progettazione di data warehouse, nel quale vengono spiegate i pregi e i vantaggi dei due tipi di modelli.
Tralasciando i generosi ragionamenti dell'autore, in linea di massima, in un data warehouse la struttura a fiocco di neve non è vantaggiosa.

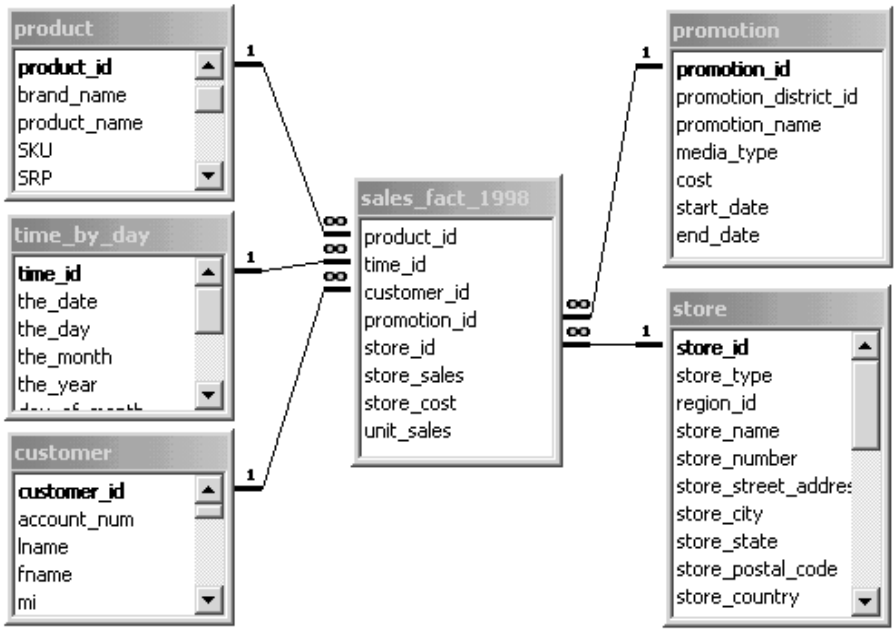


figura 2.1 – esempio di modello star schema

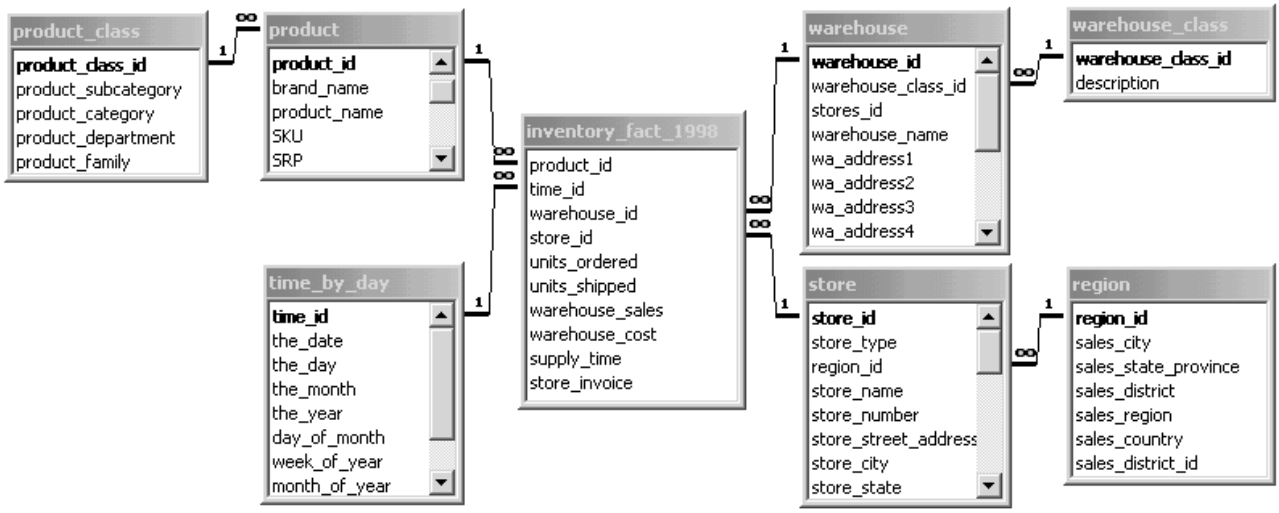


figura 2.2 – esempio di modello snowflake

Pro:

- vantaggioso, se l'utente interroga la tabella dei fatti limitatamente alla massima granularità del dato.

Contro:

- denormalizzando, si aumenta il numero di tabelle coinvolte nelle query, aumentando i

tempi di esecuzione.

- peggiora le prestazioni dei database che sfruttano i bitmap index.
- in fase di stesura di una query, l'utente finale deve avere una buona conoscenza della base dati e della sua normalizzazione.

La mia scelta è stata quindi quella di utilizzare un modello di tipo star schema.

2.2.2 DFM del data mart orientato alle vendite

In figura 2.1 viene mostrata una prima struttura del data mart.

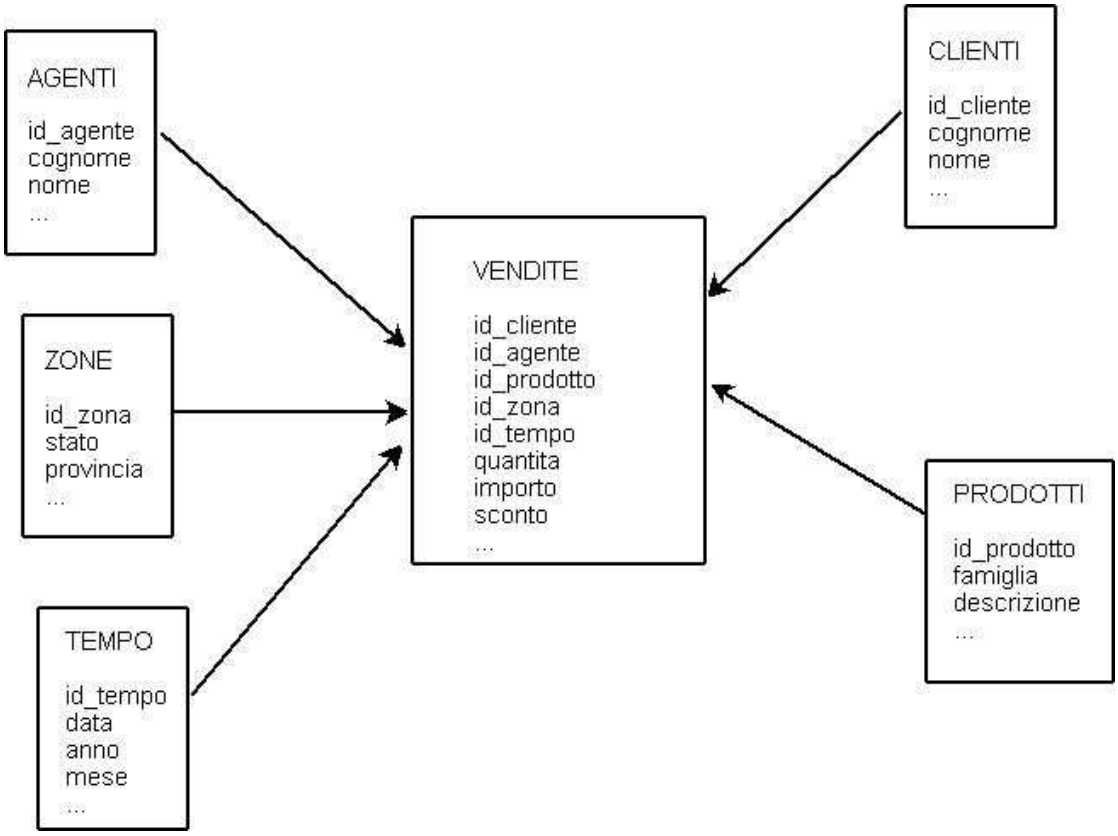


figura 2.1 – DFM del datamart orientato alle vendite

2.3 PROGETTAZIONE FISICA E PROGETTAZIONE DELL'ALIMENTAZIONE

Si tratterà di seguito le soluzioni adottate, nonché la scelta dei software, per quanto riguarda la fase di progettazione fisica (DBMS) e dell'alimentazione (ETL).

2.3.1 Scelta del DBMS

La scelta di un DBMS adeguato per l'implementazione del data mart è fondamentale per il buon esito del progetto.

I requisiti da me individuati sono:

- installazione su sistemi Linux e Windows.
- possibilità di implementare le relazioni.
- possibilità di utilizzare i triggers.
- pieno supporto della creazione di indici.
- ottimizzazione del database attraverso l'analisi storica delle query.
- disponibilità di drivers ODBC, JDBC.
- buoni risultati generali derivati dall'analisi dei benchmark[5].

Ho passato quindi al vaglio tre diverse soluzioni open source disponibili nel mercato:

- Postgresql.
- MySQL.
- Firebird.

Riassumo nella tabella 2.1 le motivazioni della mia scelta attraverso un'analisi delle funzionalità offerte dai singoli prodotti

La scelta è ricaduta su Postgresql vista la maturità del prodotto e che rispetta pienamente i requisiti da me richiesti.

Dato il budget limitato, l'installazione è stata effettuata su un PC di fascia desktop con le seguenti caratteristiche:

- CPU Amd Athlon XP 2500+
- 2 Gigabyte di memoria RAM del tipo DDR 400
- 2 x Hard Disk SCSI da 40GB in modalità mirroring
- Sistema operativo Linux Ubuntu Server versione 6.06LTS

Avrei preferito una soluzione basata su doppio processore, e almeno 4GB di ram ECC ma ciò avrebbe aumentato notevolmente i costi del sistema.

5 i benchmark visionati sono disponibili sul sito: http://benchw.sourceforge.net

	Postgresql	MySQL	Firebird
Installazione su sistemi Linux e Windows	SI	SI	SI
Possibilità di implementare le relazioni	SI	SI (supporto nativo solo a partire dalla versione 5)	SI
Possibilità di utilizzare i triggers	SI	SI (supporto nativo solo a partire dalla versione 5)	SI
Pieno supporto di tecniche di indicizzazione	SI	SI	SI
Ottimizzazione del database attraverso l'analisi delle query eseguite sul database	SI	SI	SI
Disponibilità di drivers ODBC, JDBC	SI	SI	SI (drivers JDBC in versione in beta)
Risultati generali dei benchmark	MIGLIORI	BUONI	PEGGIORI

Tabella 2.1 – DBMS open source a confronto

Tralasciando i dettagli relativi all'installazione e messa a punto di Postgresql, mi limito brevemente alla descrizione generica dei passaggi effettuati:

1. installazione di Linux
2. configurazione base di Linux e attivazione di SSH (Secure Shell) per il controllo remoto del server
3. installazione di Postgresql
4. modifica del file di configurazione per abilitare l'accesso al database ad indirizzi IP della rete aziendale (di default Postgresql rimane in ascolto solo su “localhost”)
5. creazione di 3 account per l'accesso a Postgresql:
   - amministratore per l'accesso completo al DBMS (già attivato in fase di installazione)
   - amministratore del solo database “BI” (Business Intelligence) da me creato per l'accesso completo alle tabelle del data mart
   - account in sola lettura sul database “BI”

2.3.2 Implementazione del motore ETL

La fase più lunga e delicata è stata l'implementazione di un motore ETL[6] ovvero un applicativo in grado di estrarre i dati dal database a cui si appoggia l'ERP, trasformarli e ripulirli se necessario e infine caricarli al fine di ottenere il data mart.

In un primo momento è stata tentata l'implementazione del motore ETL attraverso una serie di script scritti in linguaggio PHP che attraverso delle query SQL[7] estraevano i dati, li trasformavano ed infine, sempre attraverso delle query SQL, venivano scritti nel mio database.

Nonostante la semplicità del linguaggio PHP, mi sono reso conto della poca flessibilità degli script da me scritti e inoltre necessitavo di un buon supporto multi-threading per l'esecuzione di più processi in contemporanea e di una libreria per mappare in qualche modo le tabelle facilmente senza ricorrere necessariamente a lunghe query.

E' stata provata successivamente l'implementazione di una soluzione basata sulla tecnologia JAVA abbinata alla libreria Hibernate (http://www.hibernate.org) utilizzata per mappare le tabelle in classi e rendere più semplice quindi l'accesso.

Nonostante l'abbinata JAVA e Hibernate fosse ottima a mio avviso per la realizzazione dello strato ETL, ho abbandonato tale soluzione in quanto il progetto avrebbe richiesto troppo tempo per essere completato.

Ho quindi analizzato due software ETL open source KETTLE (http://kettle.pentaho.org/) e TALEND (http://www.talend.com) che offrono diverse funzionalità, tra queste le principali:

- collegamento alle sorgenti database attraverso drivers JDBC.
- interfaccia visuale semplice simile a quella offerta da prodotti commerciali.
- possibilità di utilizzare script per la pulizia dei dati (data cleaning).
- modo intuitivo e rapido da apprendere.
- esecuzione in multi-threading dei singoli processi.
- possibilità di schedulare le attività grazie allo scheduler integrato.

L'idea è quella di realizzare una trasformazione composta dai sequenti step:

1. collegamento ad entrambi i DBMS
2. esecuzione della query di estrapolazione dei dati dal database a cui si appoggia

6 Extract, Transform, Load. Espressione inglese che si riferisce al processo estrazione, trasformazione e caricamento dei dati in un sistema di data warehouse

7 Structured Query Language. E' un linguaggio creato per l'accesso a informazioni memorizzate nei database.

Navision

3. filtro dei dati attraverso script e/o procedure già presenti nel software
4. inserimento o aggiornamento dei dati all'interno della tabella della relativa dimensione

2.3.3 Kettle VS Talend: due software ETL a confronto

Kettle

Scritto in linguaggio java, e giunto alla versione 2.5 con un percorso di maturazione concreto, è oramai diventato uno strumento utilizzabile in progetti reali. Dispone di una interessante batteria di componenti, nella quale spicca un blocco "javascript" che permette di scrivere veri e propri pezzi di codice (diciamo delle funzioni o routines).
Altro punto di forza sono la disponibilità di strumenti di lookup web (attraverso web services) e di XML input/output. Permette ovviamente di scrivere propri plugin java implementando un numero ragionevole di interfacce. Interessante anche la parte di schedulazione dei processi.

PRO:

- uno dei primi software ETL open source. Ha una vasta comunità in supporto che recentemente è divenuta ancora più numerosa grazie all'integrazione nel progetto Pentaho[8].
- può essere utilizzato anche al di fuori di Pentaho in quanto è un software indipendente dalla suite.
- strumenti ad-hoc per le dimensioni dinamiche e variabili nel tempo che facilitano notevolmente lo sviluppo di questi scenari.
- buona disponibilità di documentazione.

CONTRO:

- supporta l'esecuzione delle trasformazioni all'interno di cluster, ma il processo di partizionamento e riunificazione dei dati all'interno di thread paralleli non è automatizzato.
- mancano strumenti per verificare la qualità dei dati e/o individuare dati anomali.
- in caso di alto volume di traffico dati, le performance non sono migliori rispetto alle altre soluzioni.

8 Una suite di Business Intelligence open source

Talend Open Studio

Strumento Eclipse[9] based (con il notevole vantaggio di avere un interfaccia molto conosciuta ai programmatori java che utilizzano eclipse come ambiente sviluppo). Genera uno script in linguaggio perl che implementa la trasformazione. Permette una maggiore visibilità ed interazione con il codice del motore, forse anche per questo si presenta un po' più povero di componenti sebbene, potendo caricare una qualunque libreria perl per usarne poi le funzioni, ha un bacino di oggetti pronti all'uso molto molto vasto.

PRO:

- riesce a bilanciare bene il traffico dati tra il Talend processing server, eventuali cluster, il database sorgente e il database target.
- ben integrato con suite di business intelligence opensource, quali SpaboBI e JasperIntelligence.
- disponibilità di documentazione e presenza di una comunità di supporto.
- interfaccia visuale ricca di funzionalità in grado di generare il codice velocemente per lo sviluppatore.

CONTRO:

- l'accesso alle sorgenti dati richiede l'installazione di driver JDBC, non presenti nel software distribuito.
- la crescita del progetto è pilotato da altre società (come JasperSoft) e soffre di una piena autonomia; nonostante ciò il programma è in costante crescita.
- mancano strumenti di validazione della qualità dei dati e inoltre la gestione dei meta-dati è ancora poco funzionale, tale da richiedere l'utilizzo di software di terze parti.

LA SCELTA: Entrambi i software offrono funzionalità quasi identiche. Ho preferito Kettle per il semplice fatto che è sembrato essere un prodotto più maturo e con un più rapido apprendimento nell'utilizzo.

L'interfaccia si è rilevata a mio parere più intuitiva e la possibilità di utilizzare semplici script in javascript per il trattamento dei dati è un vero punto di forza del software. Inoltre, da non sottovalutare, la documentazione disponibile più ampia e di qualità maggiore essendo presente da almeno quattro anni sul mercato.

9 Ambiente di sviluppo molto conosciuto dai programmatori java, ma utilizzabile anche con altri linguaggi

2.3.4 Note sulla scelta di semplificare la teoria dei sistemi data warehouse

La mia soluzione di implementazione semplifica la teoria dei data warehouse, in particolare per ogni dimensione non mantengo uno storico delle modifiche ma mi limito ad aggiornare i dati; non sono quindi previsti campi che indicano la cancellazione e/o modifica per ogni riga della dimensione al fine di storicizzare i cambiamenti.

Nel mio caso, ho potuto fare ciò in quanto per ogni dimensione analizzata, dal lato ERP vengono di norma solo inseriti dati (nuovi clienti, nuovi prodotti, ...) e non vengono cancellate mai le fatture di vendita visto che l'ERP stesso mantiene uno storico in sola lettura delle fatture registrate, nonché degli ordini di vendita stessi.

I dati che vengono più aggiornati sono quelli riferiti alle anagrafiche cliente, ma le modifiche effettuate servono principalmente a correggere errori di data-entry (es. l'indirizzo errato) e non vi è alcun interesse a mantenere uno storico delle modifiche dei clienti. Stesso discorso vale per la dimensione degli agenti.

Per quanto riguarda i prodotti ciò che viene aggiornato più spesso sono i prezzi di vendita, ma essendo riportati nelle righe ordine non ho nemmeno in questo caso la necessità di mantenere uno storico delle modifiche dei prezzi di listino; inoltre va tenuto conto che la dimensione relativa ai prodotti contiene per scelta solamente campi descrittivi in quanto la decisione è stata di inserire tutte le misure additive (come il prezzo del prodotto) solamente nella fact-table.

Altra cosa importante da notare è stata la fortuna di avere come sorgente OLTP[10] un unico database, il che ha reso più semplice il progetto evitando di ricorrere all'utilizzo di una nuova chiave primaria per le dimensioni allo scopo di integrare ad esempio i dati proveniente da più sorgenti. Ho potuto quindi "riciclare" la chiave primaria (es. il codice cliente) per ogni dimensione, tranne per la dimensione temporale e geografica dove la chiave è stata creata ex-novo.

2.3.5 Esempio di popolazione di una dimensione attraverso Kettle

Per meglio comprendere il funzionamento di KETTLE vediamo ora i passaggi per creare la "trasformation[11]" che si occuperà del popolamento della dimensione dei clienti.

Per popolare la dimensione dei clienti occorre:

1) collegare KETTLE ai due database (quello di Navision e quello contenente il mio

10 Acronimo di On Line Transaction Processing, è un insieme di tecniche software utilizzate per l'analisi dei dati.

11 In kettle una "trasformation" è definita come l'insieme dei processi di una trasformazione dati. L'estrazione e l'inserimento delle anagrafiche clienti nella relativa dimensione è un esempio di trasformation

data mart) attraverso drivers JDBC.

2) attraverso query SQL estrarre i campi interessati dalle tabelle relative ai clienti (in questo caso è stata sufficiente una sola tabella).
3) Applicare se richiesto filtri e/o trasformazioni
4) Inserimento/aggiornamento dei dati all'interno della mia tabella relativa alla dimensione dei clienti.

Preciso che l'implementazione dei filtri e delle trasformazioni in KETTLE sono ottenibili attraverso dei semplici script in linguaggio javascript.
Il test della transformation può essere fatto direttamente da KETTLE e permette di individuare eventuale errori attraverso i log files visibili direttamente nella finestra del programma.

2.3.6 Creazione della dimensione temporale e geografica

La creazione della tabella relativa alla dimensione temporale è stata creata ex-novo
La tabella contiene i seguenti campi:

- chiave primaria (ID tempo)
- data in formato giorno/mese/anno
- stagione
- mese
- settimana
- anno

Ho popolato la dimensione attraverso uno programma scritto in JAVA collegato tramite JDBC al database. Utilizzando le classi JAVA per la gestione delle date presenti nel pacchetto “java.util.Date” , il programma genera tutte le date a partire dal 01/01/2000 al 31/12/2020 e per ogni data viene estratta la stagione, il mese, la settimana e l'anno per poi essere inseriti direttamente nella tabella.
Per quanto riguarda la dimensione geografica, anche in questo caso ho provveduto a creare una tabella con i seguenti campi:

- chiave primaria (ID zona)
- sigla stato (in formato europeo)
- nome stato

- sigla provincia
- nome provincia
- nome regione
- area Nielsen di appartenenza

Considerando che l'azienda è interessata principalmente alle vendite effettuate in Italia e appurato che la compilazione delle anagrafiche cliente è generalmente corretta fino al dettaglio della provincia per i clienti residenti in Italia (per i clienti esteri purtroppo spesso e volentieri indicano solo lo stato di appartenenza), ho inserito le righe per tutte le province italiane (complete naturalmente di regione, stato e area nielsen) e mi sono limitato ad indicare solo lo stato per tutti gli stati esteri omettendo in questo caso le province, quindi in definitiva una riga singola per ogni stato estero.

2.3.7 Caricamento della fact-table relativa alle vendite con Kettle

La fact-table relativa alle vendite si popola (figura 2.2) in modo analogo alle dimensioni con l'unica differenza che le chiavi esterne relative alla dimensione temporale e geografica sono aggiunte in una trasformazione successiva a quella del caricamento della fact-table.
La trasformazione per l'inserimento nella fact-table della chiave esterna della dimensione temporale non fa altro che confrontare la data del documento (fattura o nota di credito) con la data presente nella dimensione temporale, estrae il relativo codice (chiave primaria) e lo inserisce nella riga della fact-table.
In modo analogo per l'inserimento nella fact-table della chiave esterna della dimensione geografica, si confronta la sigla della provincia e dello stato. Se lo stato è Italia, estraggo il codice della chiave cercando la sigla della provincia tra nella dimensione geografica, mentre se lo stato è estero mi limito a ricercare il codice relativo confrontando lo stato. Entrambe le procedure di individuazione delle chiavi sono effettuate con delle semplici strumenti di lookup presenti nel programma.
A differenza dell'individuazione della chiave esterna della dimensione temporale, la ricerca della chiave esterna della dimensione geografica in alcuni rari casi non è possibile in quanto ad esempio capita che in fase di inserimento dell'anagrafica cliente non siano compilati correttamente i campi (es. la sigla della provincia è errata, o non viene indicato lo stato). Grazie al modulo per l'invio delle email integrato in KETTLE è stata implementata una semplice procedura che notifica tramite email ai commerciali quali anagrafiche non sono state compilate correttamente, ovvero quei clienti dei quali non è possibile

individuare la chiave relativa nella dimensione geografica. Una volta corrette le anagrafiche clienti, alla successiva importazione verrà individuata correttamente la dimensione geografica.

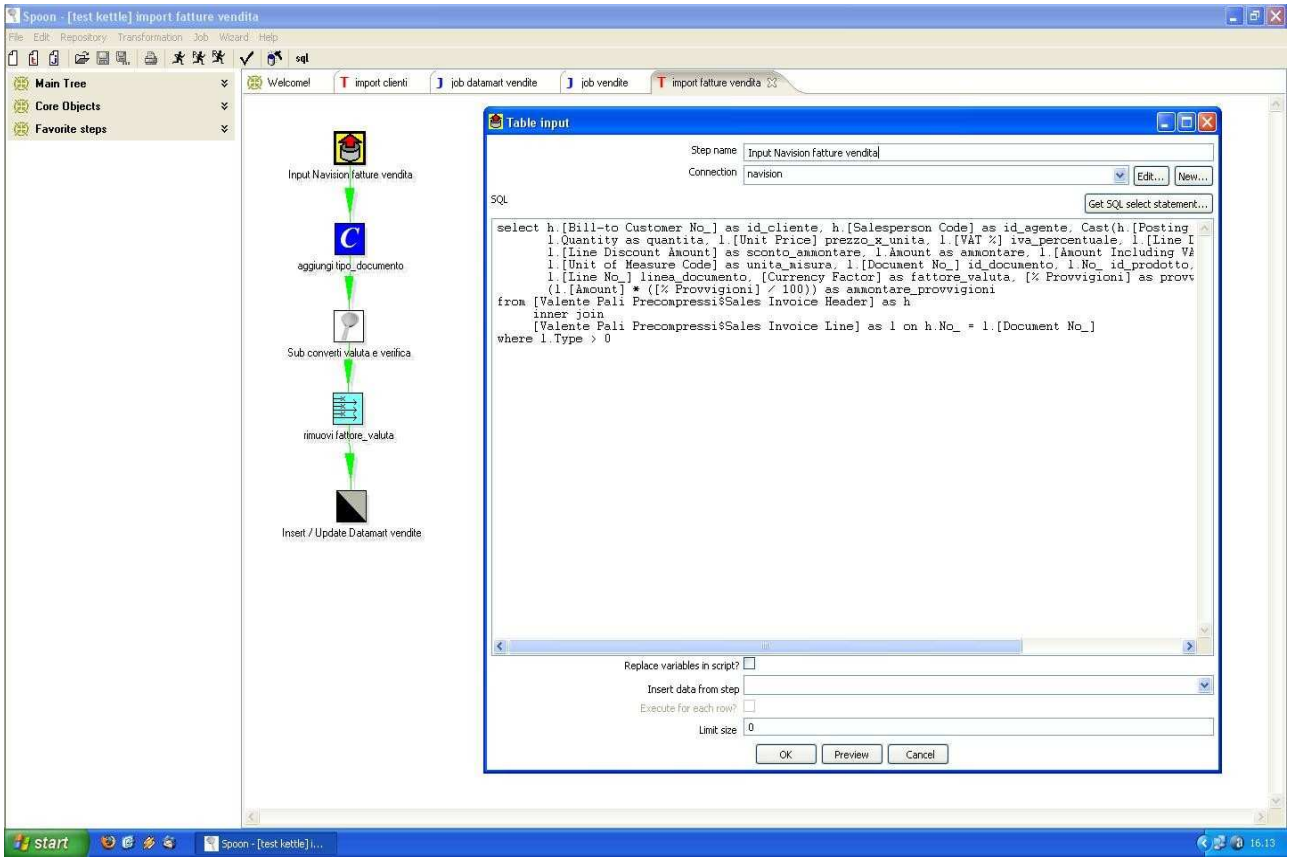


figura 2.2 - il caricamento della fact-table con KETTLE

2.3.8 La gestione job in Kettle

E' stato creato un job[12] per ogni dimensione e per la fact-table per meglio raggruppare le singole trasformation. Prendiamo ad esempio il job relativo al caricamento della fact-table:

In un primo momento vengono caricate nella tabella le fatture di vendita e le note di credito.

Nelle fasi successive importo nella dimensione relativa ai clienti, tutti i clienti non più presenti in anagrafica ma a cui in passato è stata effettuata una vendita; il gestionale stesso comunque non permette la cancellazione di un cliente dall'anagrafica, ma durante la mia analisi ho scoperto che alcune anagrafiche erano eliminate non riuscendo quindi ad individuare il cliente nella relativa tabella. E' stato quindi necessario aggiungere alla dimensione questi clienti ricavando le informazioni direttamente dalla righe vendita visto che, come già detto, Navision al fine di storicizzare alcune informazioni riporta alcuni

12 Insieme di una o più trasformation

campi della tabella clienti.

Analogamente anche per i prodotti ho effettuato una simile procedura.

Infine vengono eliminate le righe vendita non verificate e viene azzerato il relativo campo di verifica il quale è stato da me aggiunto che serve per contrassegnare quelle righe caricate in una prima importazione, ma che non sono più disponibili alla succesiva importazione. Ciò è necessario in quanto è possibile ad esempio che vengano eliminati degli ordini dai commerciali nel corso del tempo perché errati.

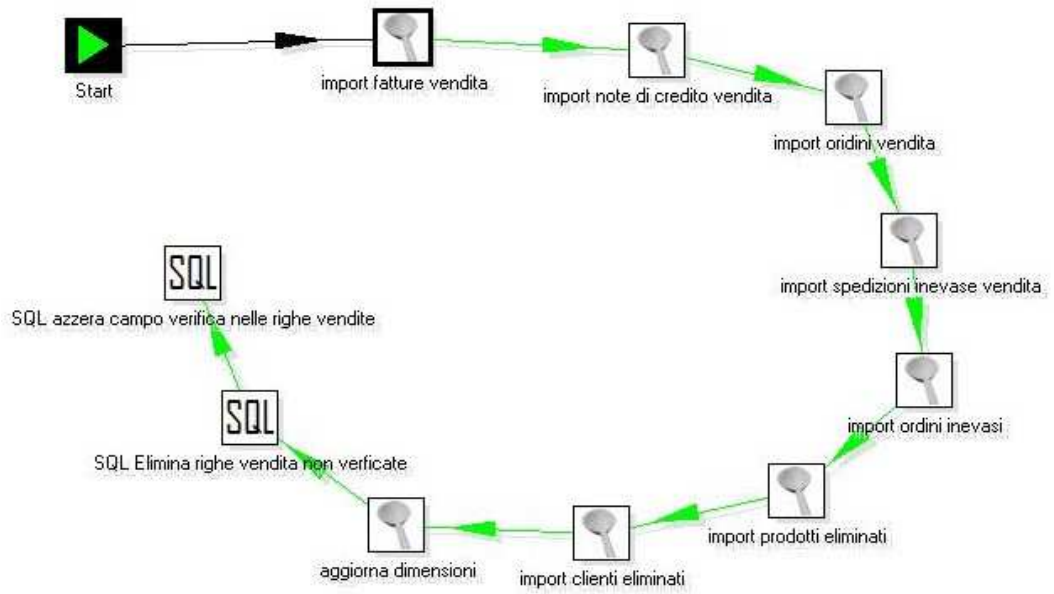


figura 2.3 – il job per il caricamento della fact-table composto dalle trasformazioni

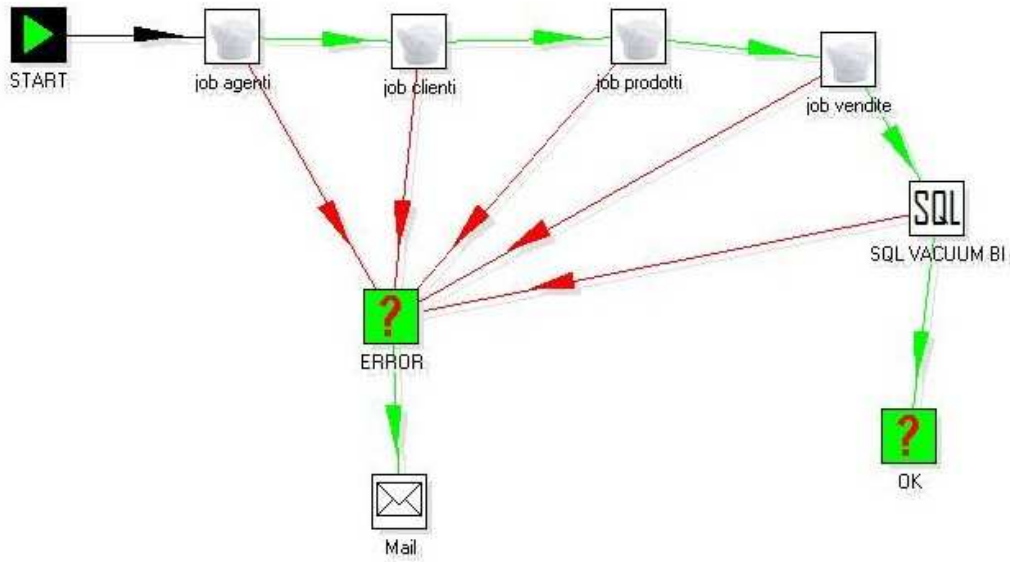


figura 2.4 – il job “madre” responsabile dell'esecuzione di tutti i processi

Infine i singoli job sono stati uniti in uno unico che potremmo definire come il job “madre” visibile in figura 2.4, responsabile dell'esecuzione di tutti i processi che concorrono alla creazione del data mart. Tale job è stato poi schedulato per essere eseguito ogni notte.
Da notare che in caso di errori, essi vengono segnalati all'amministratore attraverso una semplice email contenente in allegato il log. Ho inoltre aggiunto una funzione di “vacuum” disponibile in Postgresql per ottimizzare e compattare il database al fine di ottenere migliori prestazioni.

CAPITOLO 3: DAL DATA MART ALLA NAVIGAZIONE MULTIDIMENSIONALE

Una volta realizzato il data mart, occorre fornire agli utilizzatori finali gli strumenti per effettuare le analisi dati come richiesto dagli obbiettivi.

3.1 INTRODUZIONE ALL'OLAP

OLAP, acronimo che sta per l'espressione On-Line Analytical Processing, designa un insieme di tecniche software per l'analisi interattiva e veloce di grandi quantità di dati, che è possibile esaminare in modalità piuttosto complesse.

Questa è la componente tecnologica base di accesso ai dati del data warehouse, serve ad esempio alle aziende per analizzare i risultati delle vendite, l'andamento dei costi di acquisto merci, al marketing per misurare il successo di una campagna pubblicitaria, ad una università i dati di un sondaggio ed altri casi simili.

La creazione di un database OLAP consiste nell'effettuare una fotografia di informazioni (ad esempio quelle di un database relazionale) in un determinato momento e trasformare queste singole informazioni in dati multidimensionali.

Eseguendo successivamente delle interrogazioni sui dati così strutturati è possibile ottenere risposte in tempi decisamente ridotti rispetto alle stesse operazioni effettuate su altre tipologie di database, anche perché il DB di un sistema OLTP non è stato studiato per consentire analisi articolate.

Una struttura OLAP creata per questo scopo è chiamata "cubo" multidimensionale.

Un sistema OLAP permette di:

- studiare una grande quantità di dati
- vedere i dati da prospettive diverse
- supportare i processi decisionali

Partendo dai concetti di base appena descritti, possiamo aggiungere che esistono tre tipologie di sistemi OLAP: multidimensionale (MOLAP: Multidimensional OLAP), relazionale (ROLAP: Relational OLAP) e ibrido (HOLAP: Hybrid OLAP).

- MOLAP è la tipologia più utilizzata e ci si riferisce ad essa comunemente con il termine OLAP. Utilizza un database di riepilogo che ha uno specifico motore per l'analisi multidimensionale e crea le "dimensioni" con un misto di dettaglio ed aggregazioni.

- ROLAP lavora direttamente con database relazionali; i dati e le tabelle delle dimensioni sono memorizzati come tabelle relazionali e nuove tabelle sono create per memorizzare le informazioni di aggregazione.

- HOLAP utilizza tabelle relazionali per memorizzare i dati e le tabelle multidimensionali per le aggregazioni "speculative".

Ogni tipologia presenta vantaggi, ma non c'è un accordo completo relativamente a questi vantaggi:

- MOLAP risulta la scelta migliore per quantità di dati ridotte, perché è più veloce nel calcolare le aggregazioni e restituire risultati, ma crea enormi quantità di dati intermedi (per via delle viste materializzate).

- ROLAP è considerato più scalabile e necessita di minor spazio disco e uso di RAM, ma è il più lento nella fase di creazione tabelle e nel produrre il risultato delle interrogazioni.

- HOLAP si pone nel mezzo, è in grado di essere creato più velocemente di ROLAP ed è più scalabile di MOLAP.

La difficoltà nell'implementazione di un database OLAP parte dalle ipotesi delle possibili interrogazioni utente; scegliere la tipologia di OLAP, lo schema e creare una base dati completa e consistente è un'operazione piuttosto complicata. Decisamente complicata per una base di utenza ampia ed eterogenea.

3.2 OPERAZIONI ED ANALISI EFFETTUABILI

Di seguito verranno verranno mostrate le operazioni comunemente effettuabili attraverso un qualsiasi tool OLAP, prendendo spunto dal libro di data warehousing [3].

3.2.1 Roll-Up, Drill-Down

Le operazioni di Roll-Up e Drill-Down permettono di muoverci lungo la gerarchia di una dimensione passando a visioni dei dati più o meno dettagliate. Se ad esempio stiamo visualizzando dei dati mensili delle vendite, potremmo volere una visione d'insieme dei dati a livello trimestrale per apprezzare globalmente l'andamento delle vendite di tutto l'anno; in questo caso mediante un'operazione di Roll-Up come mostrato in figura 3.1, ci porteremo ad un livello superiore nella gerarchia temporale. Un'altra operazione di Roll-Up e arriveremo al dettaglio annuale, da qui potremo scendere nel dettaglio per quanto riguarda la dimensione geografica: decidiamo di evidenziare i dati delle regioni del centro attraverso un Drill-Down (figura 3.2). A questo punto possiamo passare ai dati provinciali e, se la gerarchia lo prevede, a quelli a livello comunale.

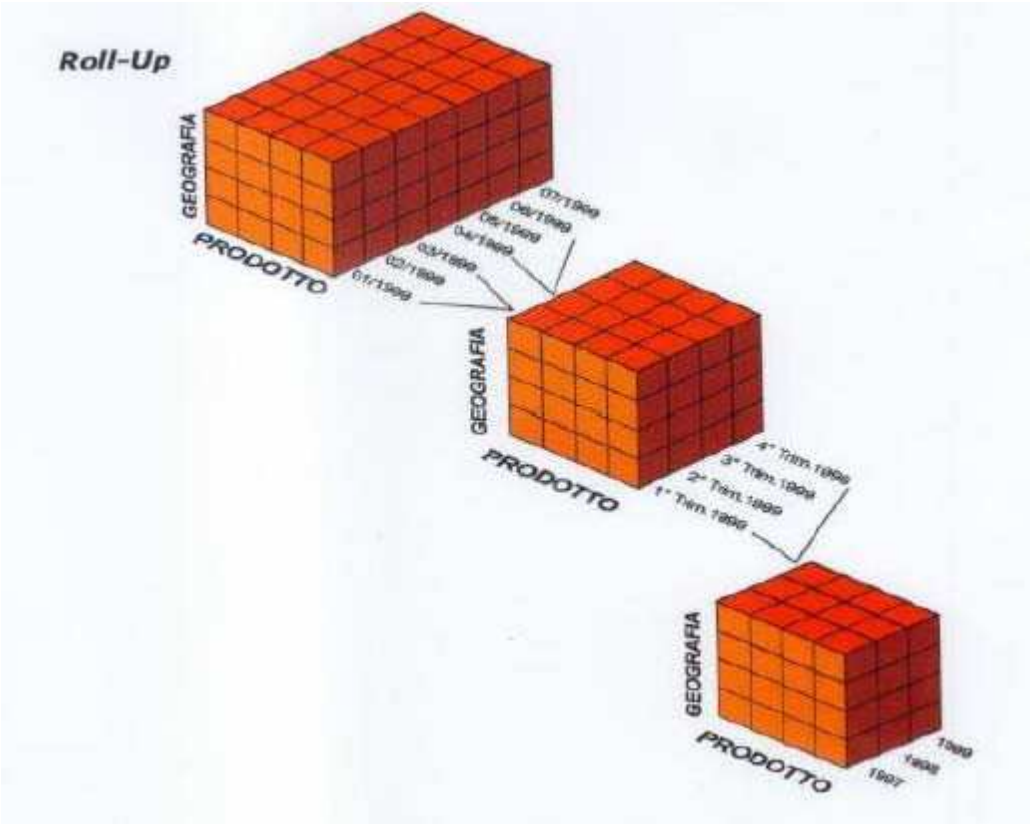


Figura 3.1 – Roll-Up [3]

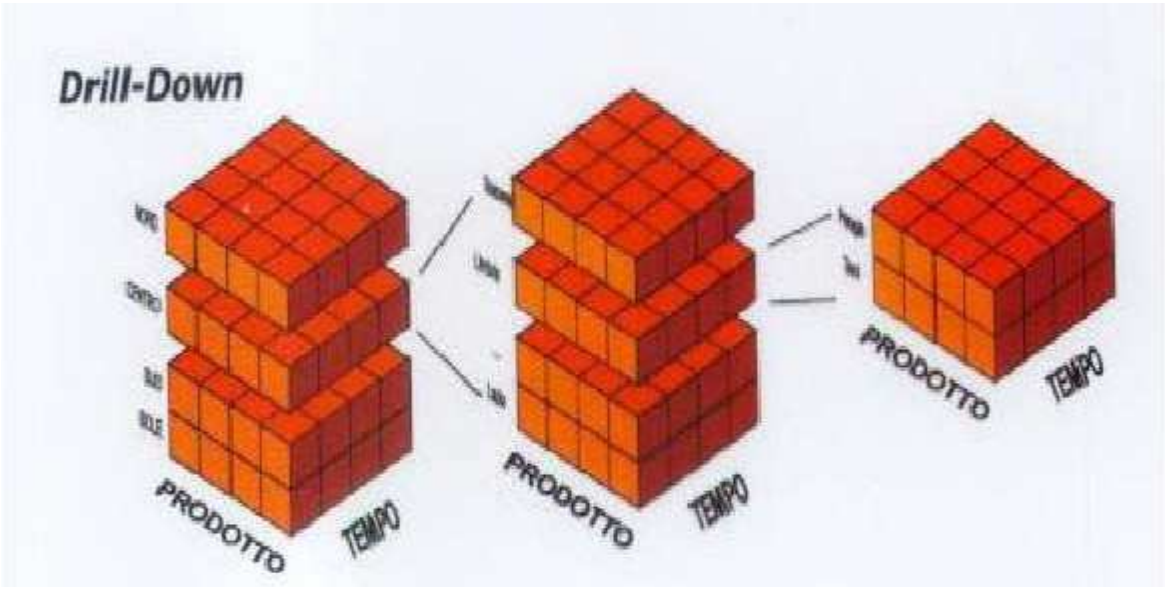


figura 3.2 – Drill-Down [3]

3.2.2 Rotation

Questa operazione permette di scambiare gli assi di visualizzazione delle dimensioni: è possibile spostare la dimensione Prodotto dall'asse X di una ipotetica visione cartesiana all'asse Y o quello Z. Nel caso di un cubo multidimensionale è possibile scambiare una dimensione visualizzata con una non in primo piano ("Dimension Swap" o "Pivoting") e combinare questa operazione con la somma lungo una o più dimensioni.

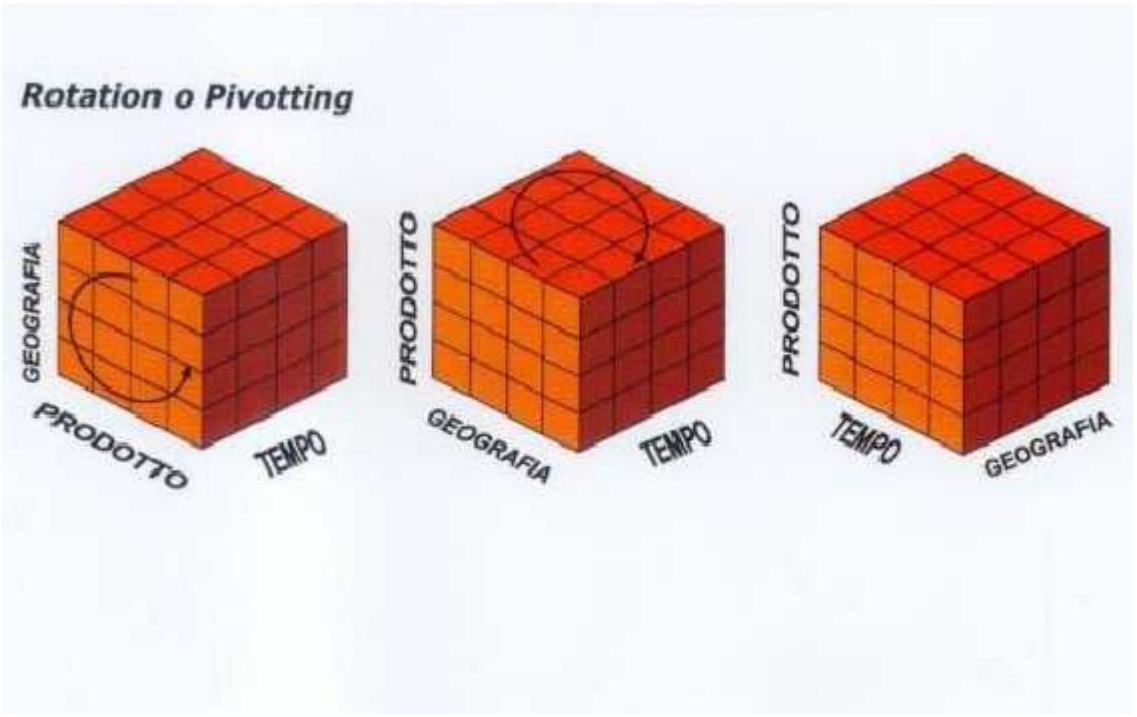


figura 3.3 – Rotation o Pivoting [3]

3.2.3 Slice & Dice

Letteralmente il termine significa “affettare e fare a cubetti” e rende perfettamente l'idea di cosa queste operazioni significhino: come vediamo in figura l'operazione di slice permette di solare una “fetta” di dati dal cubo multidimensionale. Ad esempio da sinistra a destra abbiamo isolato le vendite nel centro Italia, le vendite del solo prodotto 2, le vendite di marzo 1999.

Naturalmente è possibile combinare due slices ritagliate da due dimensioni diverse ottenendo ad esempio le vendite nel centro Italia del solo prodotto 2 oppure le vendite di marzo 1999 del prodotto 2. Combinando le tre slices otterremo un vero e proprio “Dice”: le vendite nel centro Italia del solo prodotto 2 durante marzo 1999.

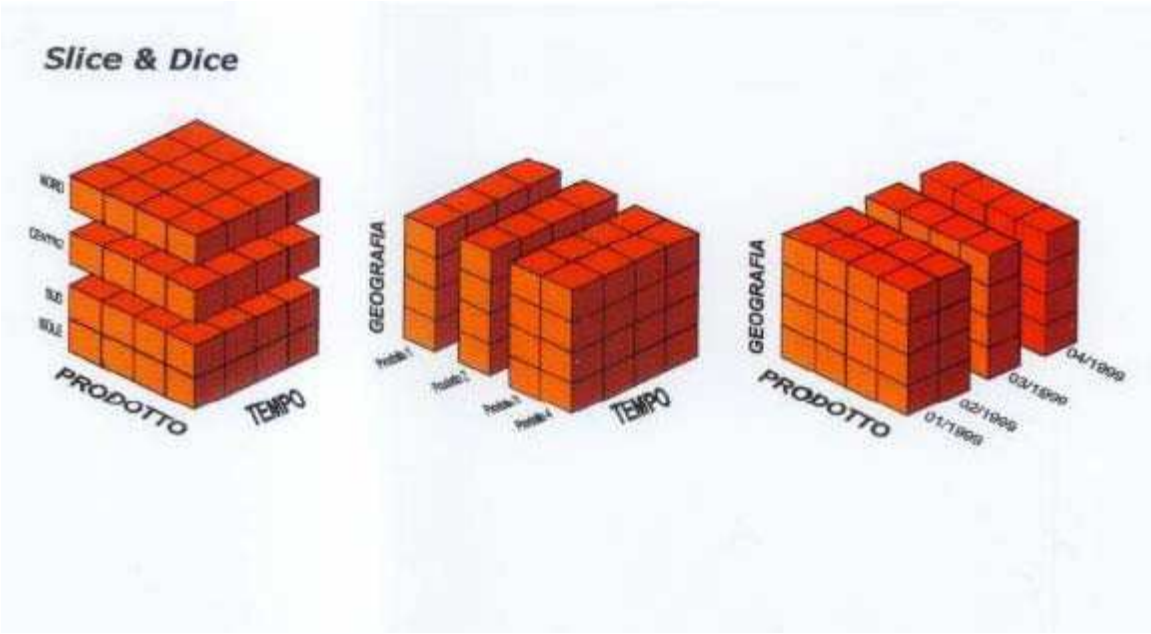


figura 3.4 – Slice & Dice

3.3 MONDRIAN: IL MOTORE OLAP

Per la scelta del motore OLAP, il mercato dell'open source al momento presenta come unica soluzione per l'analisi OLAP il software Mondrian.

Mondrian e' un motore per On Line Analytical Processing (OLAP) sviluppato in Java.

Implementa le funzionalita' indispensabili all'analisi dati (aggregazione, drilldown/through, slicing, dicing) ed e' in grado di eseguire query espresse nello standard MDX leggendo i dati da un RDBMS e presentando i risultati in forma multidimensionale per mezzo di una API Java.

La connessione alla base di dati di Data Warehouse avviene via JDBC, il che rende indipendente Mondrian dal particolare RDBMS utilizzato. Lo schema multidimensionale

della base dati può essere sia star che snowflake, e la sua descrizione viene fornita al motore sotto forma di file XML.

Mondrian e' progettato per delegare al DBMS tutte le funzionalita' che questo e' in grado di eseguire al meglio, in particolare l'aggregazione e l'utilizzo, ove consentito, di materialized views per ottimizzare la velocita' di risposta. Le raffinate strategie di caching consentono buone prestazioni in termini di velocita' di esecuzione.

In realtà però Mondrian non è una vera applicazione server, ma un'applicazione servlet (potremmo intenderla come un modulo aggiuntivo di un'applicazione server preesistente) che si appoggia ad un web server basato sulla tecnologia JSP (JavaServer Pages).

L'architettura di Mondrian è riassunta in figura 3.5 (tratto da http://mondrian.pentaho.org).

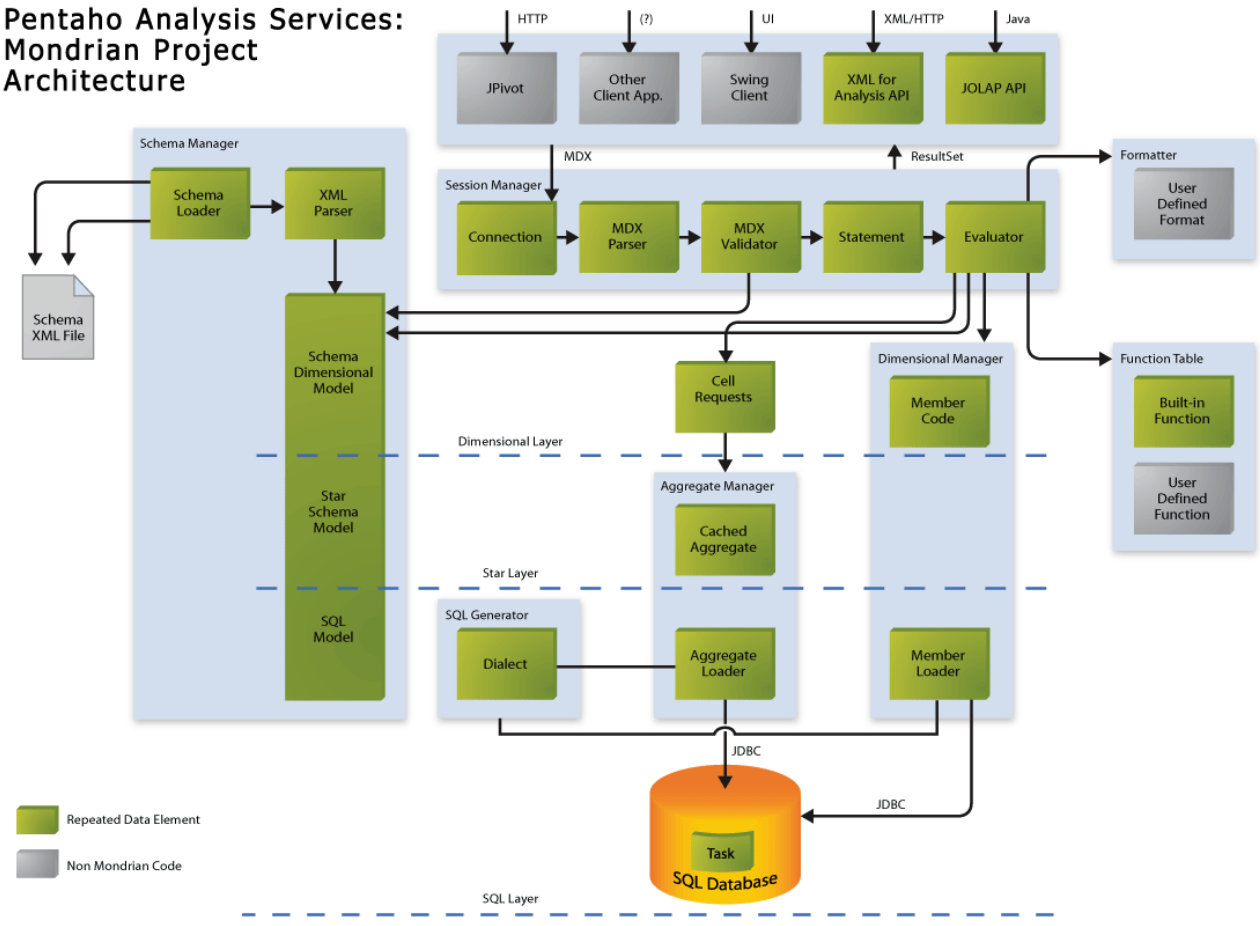


figura 3.5 – l'architettura di Mondrian

In breve, allo strato più basso abbiamo il collegamento al DBMS che viene caricato, modellato e ottimizzato attraverso procedure di caching dallo strato superiore in base alle definizioni fornite dallo schema XML.

Ancora più alto troviamo il motore MDX che si occupa di validare ed eseguire le query basate sul medesimo linguaggio divenuto ormai standard per le interrogazioni OLAP.
Infine è possibile interfacciare il Mondrian a client OLAP come JRubik e Jpivot (http://jpivot.sourceforge.net); sono inoltre disponibili API (*Application programming interface*) XMLA e JOLAP.

3.3.1 Setup di MONDRIAN

Come prima accennato occorre creare lo schema multidimensionale per Mondrian attraverso un foglio XML.
Ho scelto di definire separatamente ogni dimensione; la definizione avviene mediante una struttura ad albero. Ogni dimensione può avere uno o più gerarchie (scelta di come suddividere una dimensione) che a loro volta contengono dei sottolivelli (suddivisioni); inoltre l'ultimo livello (la foglia) può contenere dei campi descrittivi (proprietà).
In figura 3.6 viene mostrata ad esempio l'implementazione della dimensione relativa ai clienti.

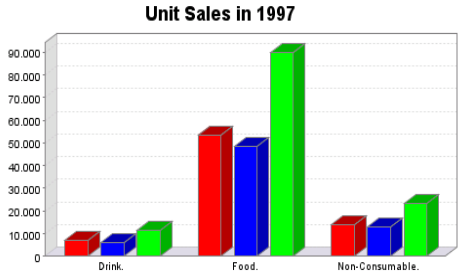
```
 <Dimension name="Clienti">
    <Hierarchy name="ClientiTipo" caption="Clienti per Tipo" hasAll="true"
allMemberName="tutti i clienti" primaryKey="id_cliente">
      <Table name="clienti"/>
      <Level name="Tipo" column="tipo" uniqueMembers="true"/>
      <Level name="Clienti" column="id_cliente" uniqueMembers="true">
        <Property name="Nome" column="nome"/>
        <Property name="Grandezza" column="grandezza"/>
      </Level>
    </Hierarchy>
    <Hierarchy name="Clienti" caption="Clienti Lista" hasAll="true"
allMemberName="tutti i clienti" primaryKey="id_cliente">
      <Table name="clienti"/>
      <Level name="Clienti" column="id_cliente" uniqueMembers="true">
        <Property name="Nome" column="nome"/>
        <Property name="Grandezza" column="grandezza"/>
      </Level>
    </Hierarchy>
  </Dimension>
```

figura 3.6 – La codifica della dimensione dei clienti

Una volta definite le dimensioni in maniera simile ho definito il cubo OLAP (figura 3.7).
Infine in figura 3.8 si illustra un esempio di navigazione attraverso il client JPivot; i dati visualizzati non fanno riferimento al data mart da me creato per via della presenza di dati sensibili. E' riportato di conseguenza un esempio che fa uso di dati demo presenti nel pacchetto distribuito con JPivot.

```
<Cube name="Vendite">
  <Table name="vendite"/>

  <Dimension name="Documenti">
       <Hierarchy hasAll="true" allMemberName="tutti">
         <Level name="Tipo Documento" column="tipo_documento" uniqueMembers="true"/>
              <Level name="Numero Documento" column="id_documento"
uniqueMembers="true"/>
       </Hierarchy>
  </Dimension>

  <DimensionUsage name="Clienti" source="Clienti" foreignKey="id_cliente"/>
  <DimensionUsage name="Zone" source="Zone" foreignKey="id_zona"/>
  <DimensionUsage name="Tempo" source="Tempo" foreignKey="id_tempo"/>
  <DimensionUsage name="Prodotti" source="Prodotti" foreignKey="id_prodotto"/>
  <DimensionUsage name="Agenti" source="Agenti" foreignKey="id_agente"/>

  <Measure name="Quantita" column="quantita" aggregator="sum" formatString="#,###.##"/>
  <Measure name="Importo" column="ammontare" aggregator="sum" formatString="#,###.##"/>
  <Measure name="Importo Ivato" column="ammontare_ivato" aggregator="sum"
formatString="#,###.##"/>
  <Measure name="provvigione" caption="Provvigione media" column="provvigione"
aggregator="avg" formatString="#,###.##"/>
  <CalculatedMember name="ammontareMenoProvvigioni"  caption="ammontare - provvigioni"
dimension="Measures">
       <Formula>[Measures].[ammontare] * (1 - [Measures].[provvigione]</Formula>
  </CalculatedMember>

</Cube>
```

figura 3.7 – definizione del cubo OLAP



	Measures		
Product	Unit Sales	Store Cost	Store Sales
-All Products	266.773	225.627,23	565.238,13
+Drink	24.597	19.477,23	48.836,21
-Food	191.940	163.270,72	409.035,59
+Baked Goods	7.870	6.564,09	16.455,43
+Baking Goods	20.245	15.370,61	38.670,41
+Breakfast Foods	3.317	2.756,80	6.941,46
+Canned Foods	19.026	15.894,53	39.774,34
+Canned Products	1.812	1.317,13	3.314,52
+Dairy	12.885	12.228,85	30.508,85
+Deli	12.037	10.108,87	25.318,93
+Eggs	4.132	3.684,90	9.200,76
+Frozen Foods	26.655	22.030,66	55.207,50
+Meat	1.714	1.465,42	3.669,89
+Produce	37.792	32.831,33	82.248,42
+Seafood	1.764	1.520,70	3.809,14
+Snack Foods	30.545	26.963,34	67.609,82
+Snacks	6.884	5.827,58	14.550,05
+Starchy Foods	5.262	4.705,91	11.756,07
+Non-Consumable	50.236	42.879,28	107.366,33

figura 3.8 – esempio di navigazione OLAP con il client JPivot

3.4 BUSINESS INTELLIGENCE OPEN SOURCE

Di recente in questi ultimi due anni, sono disponibili sul mercato alcune soluzioni open source per la business intelligence.

Pur non essendo ancora in grado di rivaleggiare con prodotti commerciali, è importante notare come lo sviluppo sia in costante crescita auspicando quindi un futuro promettente. Si fa inoltre presente che alcune di queste soluzioni sono già utilizzate ad oggi da alcune grosse aziende e pubbliche amministrazioni con buoni risultati. Analizzarle in tutte le loro funzionalità sarebbe troppo oneroso e mi limito quindi a riportare una breve descrizione di quelle da me provate. Tutte le piattaforme sono basata su tecnologia Java EE[13] e l'utilizzo dal lato utente prevede un'interfaccia web.

- *JasperSoft BI suite* (http://www.jaspersoft.com)
  Proveniente dagli stessi sviluppatori di JasperReports, famoso motore di reporting open source.
  Rende disponibile un framework per l'automatismo dei report nonché la loro schedulazione per l'invio automatico. Fornisce inoltre una componente ETL (con il modulo JasperETL basato su Talend) e di analisi OLAP (basate su Mondrian e Jpivot). E' un prodotto ancora giovane, ma promettente.
- *SpagoBI* (http://spagobi.eng.it)
  piattaforma di business intelligence realizzata in Italia da Engineering Ingegneria Informatica S.p.A. Tra i clienti utilizzatori troviamo la regione dell'Emilia Romagna e la regione Veneto, diverse aziende ospedaliere, ministeri e altre pubbliche amministrazioni.
  La suite possiede funzionalità ETL, reporting, OLAP, dashboard, data mining, gestione metadati.
  Risulta essere a mio parere una delle suite più complete tra quelle provate.
- *Pentaho (http://www.pentaho.org*)
  Tra gli sviluppatori spiccano figure che hanno lavorato per soluzioni concorrenti commerciali quali Business Object e Hyperion.
  Offre funzionalità ETL, reporting, OLAP, dashboard, data mining, gestione metadati e workflow.

13 Java Enterprise Edition. Per informazioni: http://java.sun.com/javaee/

CONCLUSIONI

Le soluzioni open source presenti nel mercato, permettono oggi di realizzare con costi ridotti, interessanti applicazioni nell'ambito della business analysis rendendo disponibili tecnologie che una volta erano utilizzate solo da aziende di grosse dimensioni, anche a realtà più piccole.

Tali soluzioni ancora non riescono forse a tener testa alle concorrenti versioni commerciali, ma la maggior parte di esse sono comunque da considerarsi prodotti con una certa maturità e fruibili senza limitazioni nell'uso.

In questo documento ho focalizzato l'attenzione sulla costruzione di un data mart al fine di effettuare navigazioni multidimensionali, ma un sistema OLAP fornisce anche altri vantaggi.

Avere a disposizione un database de-normalizzato e formato da poche tabelle permette di realizzare facilmente query anche complesse, senza ricorrere all'utilizzo di numerose join e interrogazioni complesse caratteristiche dei sistemi OLTP.

Inoltre in linea generale i tempi di esecuzione delle query stesse si riducono drasticamente.

Sempre per il fatto che il numero di tabelle che entrano in gioco in un data mart sono veramente poche, è possibile eseguire estrazioni dati anche da utenti non esperti del settore IT ed è possibile evitare di imparare il linguaggio SQL utilizzando software con funzionalità QBE (Query By Example) che fanno uso di strumenti grafici e user-friendly.

Anche la realizzazione di report che prima risultavano complessi, diventa molto più semplice, nonché si migliora la velocità di esecuzione degli stessi.

La creazione di un buon data warehouse è sicuramente alla base dei processi di business intelligence perché permette l'integrazione dei dati aziendali (EII[14]), requisito fondamentale per l'integrazione delle applicazioni (EAI[15]).

Le persone coinvolte nei processi di business intelligence utilizzano applicazioni software ed altre tecnologie per raccogliere, immagazzinare, analizzare e distribuire le informazioni.

Nella letteratura la business intelligence viene citata come il processo di "trasformazione di dati e informazioni in conoscenza", infatti tale conoscenza permettere alle persone di prendere decisioni strategiche fornendo informazioni precise, aggiornate e significative nel contesto di riferimento.

---

14 Enterprise Information Integration

15 Enterprise Application Integration

RIFERIMENTI

BIBLIOGRAFICI

[1] W.H. Inmon, Building The Data Warehouse. Third Edition. *Wiley*, 2003

[2] M. Golfarelli, S. Rizzi. Data Warehouse: teoria e pratica della progettazione. *Graw Hill*, 2006

[3] S. Dulli, V. Favero. Modelli e Strutture per il Data Warehousing. *Franco Angeli*, 2000

[4] R. Kimball, The Data Warehouse Toolkit. Second Edition. *Wiley*, 2002

[5] S. Dulli, M. De Angelis. Il Data Warehouse al Centro del Sistema Informativo *Franco Angeli*, 2000

[6] R. Kimball, The Data Warehouse Lifecycle Toolkit. *Wiley*, 1996

WEBGRAFICI

- http://www.di.uniba.it/~malerba/activities/datalight/attivita.htm
- http://it.wikipedia.org/wiki/OLAP
- http://it.wikipedia.org/wiki/Business_intelligence
- http://www.mokabyte.it/2004/11/intdati.htm
- http://openskills.info/topic.php?ID=70
- http://benchw.sourceforge.net

ALTRO MATERIALE

- Lucidi del corso di Sistemi Informativi Aziendali dell'università di scienze statistiche di Padova. Prof. Susi Dulli, 2006

RINGRAZIAMENTI

Si ringrazia di cuore la professoressa Susi Dulli per l'infinita pazienza e la disponibilità senza la quale sarebbe stato impossibile realizzare questa relazione.

Grazie inoltre alla Valente Pali Precompressi spa per l'opportunità lavorativa offerta che si è rilevata una forte esperienza costruttiva.

Un ringraziamento sentito e doveroso ai miei genitori per avermi sempre sostenuto e appoggiato in ogni mia decisione.

Infine ringrazio i miei fratelli Davide e Gianluca per l'affetto ricevuto, a Silvia per avere pazientemente controllato gli errori di ortografia e a tutti gli amici che con la loro presenza si dimostrano parte integrante e significativa della mia vita.