

INTRODUZIONE AL DATA WAREHOUSING

Il contenuto del presente documento deriva da una sintesi di quello che è stato il capitolo introduttivo della mia Tesi per la Laurea Triennale (2004).

L'intento del documento è quello di fornire una visione panoramica generale ed introduttiva a chi si affaccia per la prima volta al data warehousing, fornendo anche qualche riflessione di carattere pratico, nonché gli spunti e le fonti per eventuali approfondimenti. Considerando il solo esiguo numero di pagine di cui si compone il documento, è facile intuire come questo non possa avere alcuna pretesa di coprire interamente l'argomento; mi auguro altresì che possa comunque introdurlo in maniera sufficientemente chiara.

Tutte le informazioni qui riportate possono essere utilizzate per qualsiasi finalità, non a scopo di lucro. La citazione del presente testo e del suo autore non è necessaria, ma sicuramente gradita.

Il solo fatto di essere un derivato del mio lavoro di Tesi dovrebbe garantire una buona attendibilità del documento; considerata comunque la natura con cui viene resa disponibile l'opera, non mi assumo alcuna responsabilità per le eventuali inesattezze in essa contenute, né tanto meno per eventuali conseguenze avverse derivanti dalla lettura o dall'utilizzo di nozioni contenute in questo documento.

Dopo questo doveroso preambolo, non mi rimane che augurarvi buona lettura.

Stefano Maraspin
<http://www.maraspin.net>

1 Caratteristiche Principali

Un datawarehouse è uno strumento che viene comunemente inserito tra quelli individuati dall'acronimo DSS (Decision Support System), ovvero sistemi progettati per aiutare il management nelle decisioni da prendersi su grosse moli di dati e in grado di fornire rapidamente informazioni, rapporti e analisi di varia natura.

Un'applicazione tipica di tali sistemi e' ad esempio l'OLAP (On Line Analytical Processing), che mette a disposizione del manager un ambiente di dati multidimensionale, nel quale e' possibile eseguire ricerche, aggregando i dati in suo possesso; un altro esempio e' costituito dal Data Mining, che applica tecniche di intelligenza artificiale ai grossi archivi, alla ricerca di informazioni utili, celate da grosse quantità di dati insignificanti.

In molti casi, l'infrastruttura alla base di tali sistemi DSS è solitamente un Data Warehouse, ovvero un "magazzino" virtuale, dove vengono memorizzate e integrati i dati provenienti da molteplici fonti.

Il concetto di Data Warehouse viene proposto per la prima volta da Bill Inmon, nella seconda metà degli anni ottanta, e viene definito come segue: *"il data warehouse è una collezione di dati, a supporto del processo decisionale manageriale orientata al soggetto, integrata, non volatile e dipendente dal tempo"*. [L1]

Ulteriori articoli sull'argomento compaiono subito dopo; viene così' definito, ad esempio sull'IBM System Journal, in un articolo ad opera di Barry Devlin: *"Un singolo, completo e consistente deposito di dati, ottenuti da diverse fonti e resi disponibili agli utenti finali, in maniera tale da poter essere immediatamente fruibili"*. [L2]

I primi esempi di architetture di questo tipo vengono presentate da IBM e Hewlett-Packard, rispettivamente nel 1991 e 1993. Da allora tecnologie e terminologie si sono sostanzialmente evolute, così' da ampliare ambiti di utilizzo e modalità di implementazione per simili strumenti; e' così oggi possibile individuare almeno tre tipologie per un data warehouse:

- Enterprise Data Warehouse
- Data Mart
- Operational Data Store

La prima di queste tipologie e' quella classica, inizialmente concepita da Bill Inmon e IBM, vale a dire una grande base di dati aziendale centralizzata, a supporto del processo decisionale; il data mart (che in molte fonti viene considerato come una parte integrante della tipologia di cui sopra), indica invece un modello dall'ambito di applicazione ridotto ("limited scope"); esso può ad esempio contenere i dati relativi alle vendite di un'azienda in una particolare regione territoriale, piuttosto che quelli legati ad una particolare linea di prodotti. Interessante citare a questo proposito il cosiddetto modello Multi-Tier, adottato da molte aziende e che predica la coesistenza di un Enterprise Data Warehouse e più Data Mart. La migrazione dei dati tra i due livelli può seguire un approccio top down (i dati passano dal data warehouse aziendale ai vari data mart) oppure bottom up (i dati sono caricati dai data mart verso il data warehouse aziendale).

I benefici del primo approccio derivano dal fatto che le operazioni di estrazione dai sistemi transizionali sono ridotte, in questo caso, a favore di una maggior consistenza delle informazioni contenute nei vari data mart; in molti casi tuttavia l'approccio Bottom Up consegue dalla condizione di esistenza di data mart, precedente a quella di un data warehouse; la scelta e' quindi obbligata in questo caso. [A1]

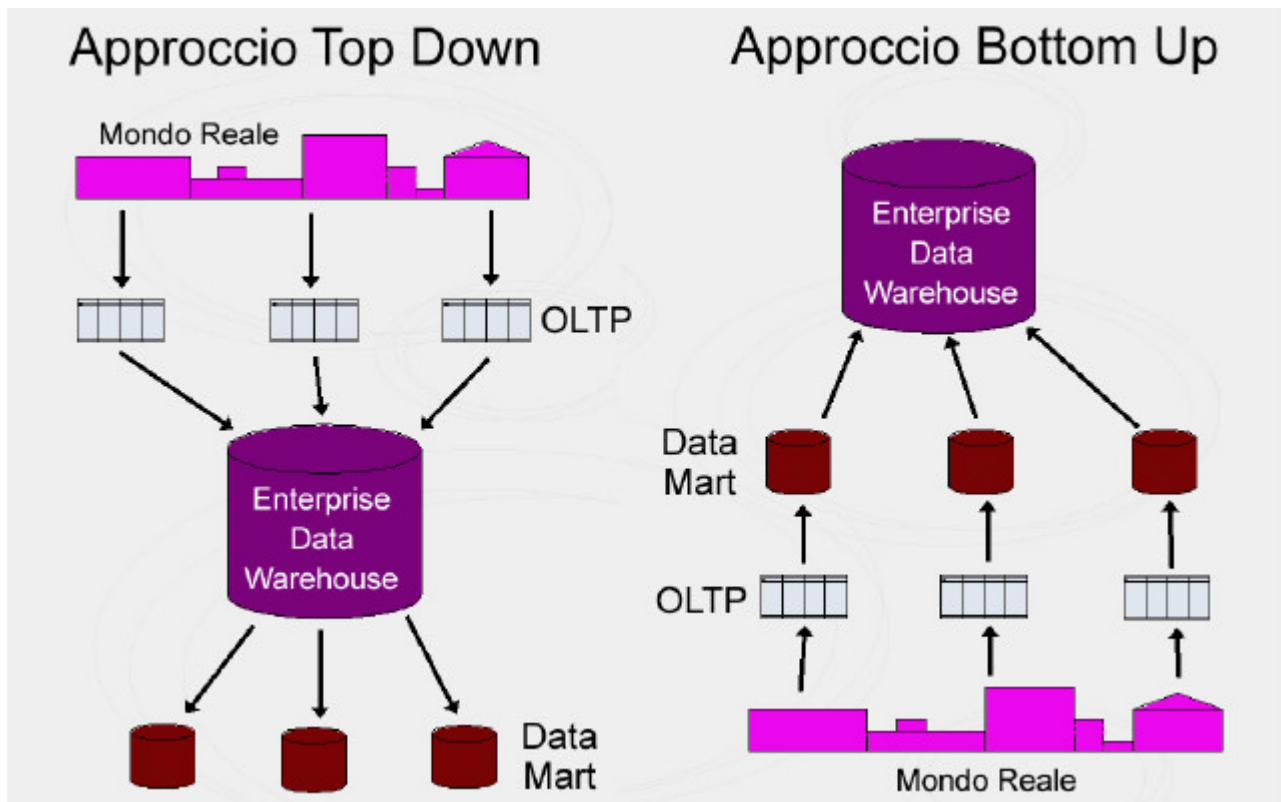


Fig 1 - Architettura Multi-Tiered per il Data Warehouse.

Per quanto riguarda l'ultima tipologia di data warehouse citata, possiamo invece dire che questo presenta nuovamente un "enterprise-wide-scope", ovvero un ambito di applicazione esteso a tutta l'azienda, con caratteristiche particolari tuttavia, per cio' che riguarda l'inserimento dei dati. Questi sono infatti inseriti nel database quasi in real-time, andando a ricordare quindi la frequenza di aggiornamento, tipica di un comune sistema OLTP.

A differenza di quest'ultimo, comunque, un Operational Data Store non viene aggiornato manualmente dagli utenti, ma sempre da molteplici fonti automatizzate; inoltre, in questo caso la base di dati e' gia' orientata alla disposizione, separazione logica e analisi dei soggetti, anzichè all'ottimizzazione delle operazioni che giornalmente sono più volte necessarie nella gestione di un'azienda o, nel nostro caso, di un ente di ricerca. Tali sistemi, proprio perché pensati per questo scopo, sono orientati alle singole operazioni (ottimizzazione degli inserimenti, aggiornamenti, ecc), piuttosto che alle analisi globali dei fenomeni e si dicono quindi di tipo operativo (On Line Transaction Processing).

Un qualsiasi data warehouse è invece orientato ai soggetti, sulla base dei quali potranno poi essere estrapolate informazioni rilevanti e quindi prese decisioni, ad esempio di tipo manageriale (o, nel nostro specifico caso, di tipo diagnostico) a lungo termine.

OLTP	OLAP
Mantiene i Dati Correnti	Mantiene Dati Storici
Ha un Alto Livello di Dettaglio	Presenta diversi Summarization Levels
I Dati sono dinamici	I dati sono Statici
Vi e' un alto livello di Throughput Transazionale	Vi e' un basso livello di Throughput Transazionale
Vi sono Pattern di utilizzo	Non vi sono schemi predefiniti
E' orientato alle applicazioni e alle singole transazioni	Orientato alle analisi e alla pianificazione strategica
Supporta le operazioni e le decisioni quotidiane	Supporta le decisioni a lungo termine
E' a servizio di un vasto numero di utenti operazionali	E' a servizio delle figure Manageriali

Fig 2 - Differenze tra l'OLTP e l'OLAP, applicazione tipica per un Data Warehouse.

La più importante delle caratteristiche di un data warehouse è probabilmente l'**integrazione**, che nasce dalla necessità di dare coerenza ai dati provenienti da diverse applicazioni, progettate per scopi diversi. Poiché i manager, per poter prendere le loro decisioni necessitano di ogni possibile fonte di dati, relativa (o meno) all'azienda, il problema da affrontare è quello di rendere questi dati accessibili ed omogenei in un unico ambiente. Questa operazione può non essere sempre immediata e possono altresì presentarsi alcuni problemi, legati ai diversi formati di memorizzazione dei dati, nei loro formati originali; un esempio immediato può esserci fornito dalle unità di misura: sistemi diversi possono gestire in maniera diversa le unità, per riferenziare una stessa misura; nel data warehouse bisognerà quindi decidere quale forma vogliamo tenere come valida e di conseguenza codificare i dati provenienti dalle altre applicazioni, prima dell'inserimento. Un altro esempio riguardante la necessità di integrazione, può essere legato alla nomenclatura; più applicazioni potrebbero rappresentare il medesimo strumento con diverse denominazioni e, in questo caso, sarebbe opportuno decidere quale sia la nomenclatura più adatta, da adottare universalmente.

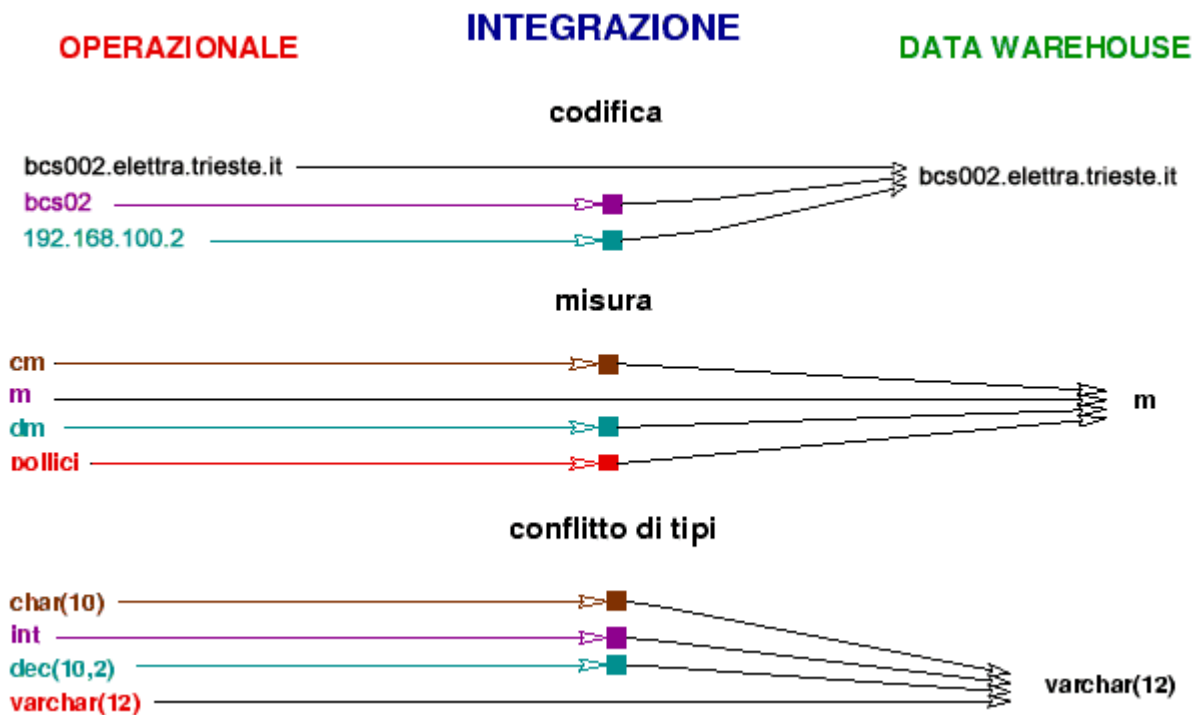


Fig 3 - Problemi nell'integrazione da diverse sorgenti di dati.

Un'altra caratteristica che un sistema per il data warehousing deve presentare, è la **non-volatilità**; i dati in esso contenuti non devono poter essere modificati dagli utenti, per mezzo di singole operazioni interattive (come avviene invece di norma in un sistema OLTP). Nel caso di un sistema di supporto alla diagnostica di un acceleratore, non è nel data warehouse che si andrà a modificare, ad esempio, la collocazione di uno strumento; in caso contrario, si potrebbe perdere ogni riferimento storico al fatto che lo strumento ha eventualmente cambiato posizione. Per questo motivo, i dati vengono caricati solitamente in massa ed in modalità batch, dagli appositi script aggiornanti; gli utenti finali possono elaborare, riassumere e visualizzare i dati, ma non modificarli.

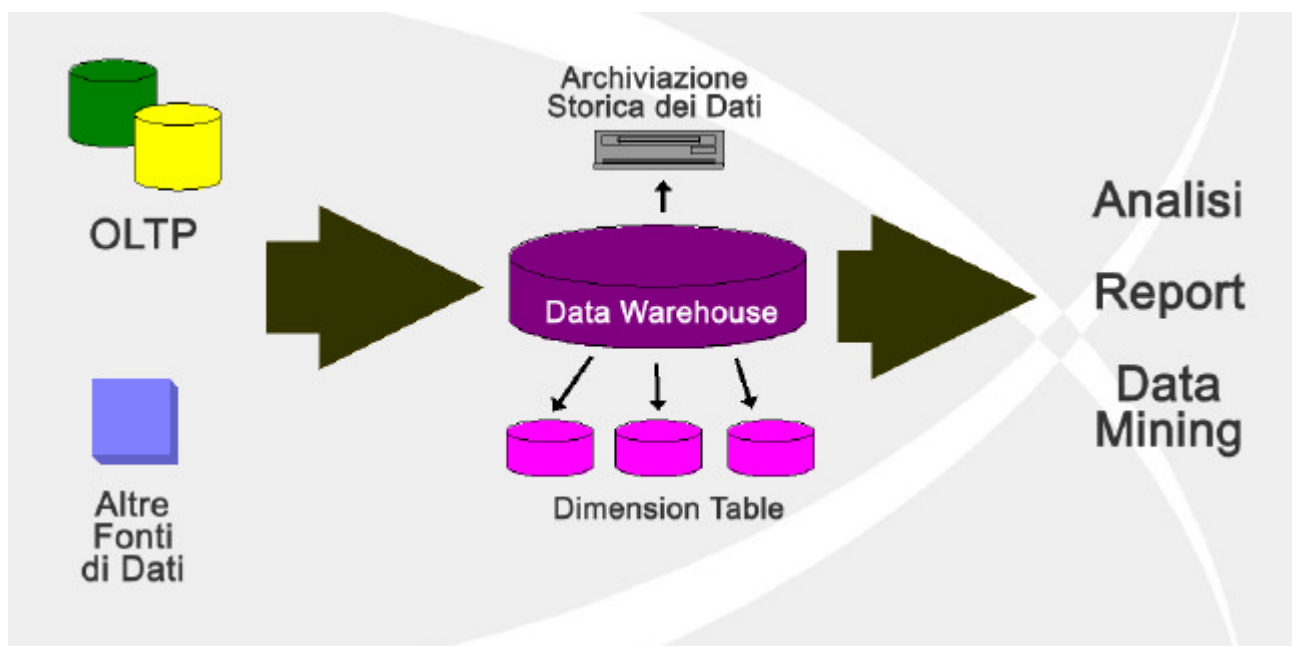


Fig 4 - Architettura Generica di un Sistema per il Data Warehousing

L'ultima caratteristica importante di un data warehouse è la **dipendenza dal tempo**. A differenza dei database dove le operazioni direttamente accessibili, di solito, sono quelle degli ultimi 60-90 giorni, in un data warehouse l'intervallo temporale si allarga fino ad arrivare a coprire un arco di 5-10 anni. In un ambiente operativo, il database contiene il "valore corrente" e questo dato può essere modificato solo perdendo ogni riferimento al dato precedente; in un data warehouse i dati sono mantenuti nella loro interezza, come se fosse scattata una foto istantanea (*snapshot*) del sistema, finendo così per tenere conto anche della storia dei vari soggetti coinvolti nel tempo. La struttura chiave di un sistema operativo inoltre può contenere degli elementi di tempo (anno, mese, data, ora , ...) o meno, mentre quella di un data warehouse **deve** sempre contenere qualche elemento di questo tipo.

Nella definizione per tale architettura, c'è inoltre da considerare che un data warehouse non è il solo insieme di dati strutturati, conforme alle regole appena descritte, ma piuttosto un sistema composto anche dalle applicazioni che servono per estrarre, analizzare e presentare i dati medesimi, al fine di ricavare le informazioni in essi presenti. Viene di seguito riportata una struttura tipica di un data warehouse, che si compone di diversi livelli di dettaglio (detti anche *summarization level*).



Fig 5 - Summarization Level di un Sistema per il Data Warehousing.

Partendo da sinistra, vediamo per primo un archivio di dati dettagliati, dove sono registrati soggetti che si riferiscono ad un tempo lontano; questi dati sono talvolta salvati su nastri, perché, essendo richiesti solo di rado, si considera accettabile un tempo di accesso più elevato. Poi troviamo i dati attuali ad un elevato livello di dettaglio: essi tengono conto di un periodo relativamente breve. In alcuni casi vi può poi essere la necessità di disporre di dati leggermente o altamente riassunti, vale a dire di dati riepilogativi, relativi a periodi di aggregazione e latenza più lunghi. Ogni livello di dettaglio viene ricavato a partire dal livello corrente, a cui giungono per mezzo di apposite procedure, che li ricavano dai sistemi OLTP. Una volta che i dati sono "invecchiati" passano automaticamente agli altri livelli di dettaglio. Direttamente connesso con tale aspetto, è un altro concetto piuttosto importante nel contesto di un sistema di data warehousing, ovvero la **granularità**, da intendersi inversamente proporzionale al livello di dettaglio con cui vengono salvati i dati: più questo è alto, più bassa è la granularità e viceversa.

L'importanza di tale aspetto deriva dal fatto che questa è direttamente legata al volume di dati salvato e, di conseguenza, alle prestazioni del sistema e alla necessità di risorse hardware. Bisogna ovviamente cercare di scegliere il giusto livello di granularità, per evitare di memorizzare dettagli che non verranno mai presi in considerazione o non registrarne altri di essenziali.

Una buona soluzione può essere quella di scegliere contemporaneamente più livelli di granularità, andando a registrare fisicamente dati di dettaglio diverso in tabelle diverse. È così possibile passare da una visione sintetica delle informazioni, ottenuta accedendo in un primo momento ai dati

altamente riassunti, ad una visione dettagliata, presa dalle tabelle a più bassa granularità, ottimizzando così il numero di accessi ai supporti magnetici e l'uso del DBMS; questa metodologia (che sarà ripresa in seguito), è comunemente detta **Drill Down**. Ovviamente il rovescio della medaglia di tale approccio si ha nelle complicazioni apportate dalla necessità di dover trasferire continuamente i dati tra i vari livelli.

Come si era accennato, i soggetti sono ora posti al centro dell'attenzione, per quanto riguarda le analisi; vengono rappresentati a livello logico nel data warehouse con una serie di tabelle collegate tra loro tramite relazioni e sono il fulcro delle operazioni di ricerca e confronto, eseguite dagli utenti del data warehouse. Vengono scelti in base al tipo di organizzazione aziendale ed al tipo di data warehouse che si intende progettare. Alcuni esempi di soggetti nei datawarehouse aziendali, sono i seguenti:

- clienti
- vendite
- prodotti
- reclami

A questo punto, la scelta di un modello logicamente rappresentativo per questi soggetti cade necessariamente tra i due candidati attualmente citati in letteratura [A1]: quello tabulare e quello dimensionale.

Il primo dei due ricorda molto un sistema transazionale (come ci viene suggerito anche dal nome) e si limita a raccogliere dati storici, derivati e denormalizzati dai sistemi OLTP in “grandi” tabelle riassuntive; il contributo pratico che tale modello può dare in alcuni contesti è notevole (si pensi a resoconti che rendono una presentazione numerico-testuale già formattata, recuperando le informazioni di partenza da differenti sistemi OLTP), ma non si presta certo ad analisi complesse su grosse moli di dati eterogenei; per questo motivo non verrà ulteriormente trattato in questa sede. Il modello dimensionale, invece, divenuto tra l'altro lo standard de facto per il data warehousing, rappresenta i dati con un cosiddetto “hypercube” (ipercubo), ovvero un array multidimensionale. I valori dello stesso rappresentano i fatti (attraverso misure numeriche), mentre le sue dimensioni costituiscono i parametri relativi alle misure stesse, come, per esempio, in un'analisi economica, i luoghi dove è avvenuta una vendita, piuttosto che il prodotto oggetto della stessa. [L4]

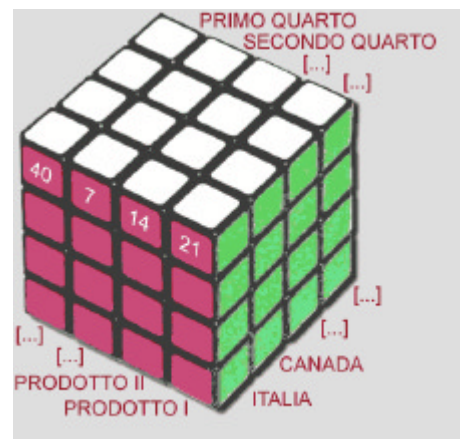
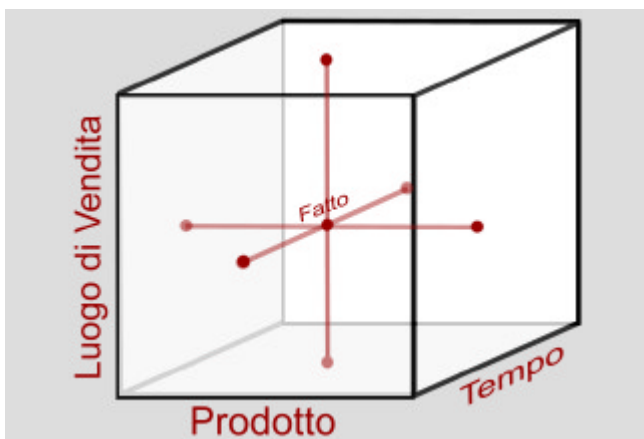


Fig 6a Hypercube Ideale Fig 6b Esempio di Hypercube

2 Progettazione di un Data Warehouse

Dato che una buona parte del data warehouse è costituito da una base di dati, simile a quelle comunemente utilizzate in ambito OLTP, anche in fase di progettazione possiamo seguire le fasi utilizzate nell'ambito del design dei database relazionali. Il primo stadio di questo processo viene chiamato "Design Concettuale" e mira a creare schemi Entità/Relazione a partire dai requisiti formulati dagli utenti (o per meglio dire dal management, trattandosi pur sempre di un sistema DSS). Il secondo passo è quello volto alla realizzazione del Design Logico, in cui si passa da uno schema E/R ad un modello tabulare o, più spesso dimensionale. In ultima istanza, si procede quindi al Design Fisico, andando a creare le tabelle, gli indici e tutte le altre eventuali strutture di supporto.

Durante tutta la fase di progettazione e il ciclo di vita del data warehouse stesso, sarà sempre bene ricordare che un sistema di questo tipo sarà probabilmente popolato da una grande quantità di dati e impiegato con funzionalità diverse, a seconda dei casi; per tale motivo, sarà opportuno eseguire un'analisi specifica e dettagliata, prima dell'implementazione, andando a anche considerare quegli aspetti intrinseci dell'ambiente in cui si andrà ad operare (hardware, sistema operativo, software a disposizione, operazioni più frequenti previste, ecc).

2.1 Design Concettuale

Come si può dedurre dalla propria denominazione, è questa una fase teorica e altamente astratta; non si considerano ancora i dettagli dell'implementazione fisica. Per individuare i soggetti del data warehouse e le relazioni che intercorrono tra essi, si utilizza spesso il modello Entità/Relazioni; tale modello trova molto spazio nella letteratura classica delle basi di dati e, per questo motivo non ci si soffermerà a lungo sull'argomento.

Vale la pena comunque ricordare il fatto che il modello E/R viene usato per fornire una descrizione ottimale dei requisiti, a partire dalle richieste degli utenti, identificando con chiarezza i soggetti rilevanti (entità), le proprietà di tali soggetti (attributi) e come questi interagiscono l'uno con l'altro (relazioni). [L6]

Mentre i diagrammi E/R sono stati tradizionalmente associati con modelli altamente normalizzati, come quelli utilizzati in ambito OLTP, gli stessi sono anche utili nell'ambito del design di un data warehouse dalla modellazione dimensionale; invece di andare alla ricerca di unità atomiche di informazioni (come entità ed attributi) e le relazioni tra essi, si identificano quelle che costituiranno le misure e quelli che costituiranno i soggetti dell'indagine (rispettivamente, i valori e le dimensioni del nostro hypercube).

2.2 Design Logico

Nel design logico, che rappresenta un passo molto importante nella realizzazione di un sistema per il data warehousing, si va a costruire uno schema (generalmente dimensionale), dove le informazioni rappresentate nel modello E/R sono correttamente ed efficientemente fatte corrispondere (mapping) ad oggetti del database, come tabelle, colonne, ecc. Questa parte del design è particolarmente importante, perché l'organizzazione che verrà data agli oggetti influenzerà notevolmente le prestazioni del sistema. [L7]

Alcuni esempi della "mappatura" che avviene a questo livello sono:

- Entità in Tabelle
- Relazioni in Chiavi Esterne
- Attributi in Colonne

- Identificatori Univoci in Chiavi Primarie
- Identificatori Univoci in Indici Unici

A tale riguardo, può essere interessante notare il fatto che il modello E/R porti spesso e volentieri a tabelle altamente normalizzate, condizione ottimale in un ambiente operativo, dove le query coinvolgono solitamente poche tuple; nell'ambito di un data warehouse invece, può capitare che la normalizzazione porti a tempi di risposta insostenibili, dovuti ai numerosi accessi ai dischi (a seguito dei join tra le diverse strutture di dati); può quindi prospettarsi la necessità di denormalizzare le tabelle e creare ridondanze (andando quindi incontro a tempi più lunghi e maggiori insidie in sede di caricamento/aggiornamento dei dati, a fronte di una maggiore velocità in sede di interrogazione in senso stretto).

Ovviamente avere dati ridondanti nel data warehouse porta ad uno spreco di risorse per quanto riguarda le dimensioni dei file, ma questo è il prezzo da pagare per ottenere migliori prestazioni nei tempi di risposta; quello che occorre cercare è quindi un compromesso tra la dimensione del data warehouse e le prestazioni del sistema.

E' chiaro che tali scelte vanno sempre prese in base alle premesse (dimensioni previste, livello di ridondanza apportato, ecc) e all'ambiente che si ha davanti (hardware, sistema operativo, ecc).

2.2.1 Snowflake Design

Il modello più utilizzato per il design logico di un data warehouse è probabilmente il cosiddetto schema a stella (star schema), e nella maggior parte dei casi, un caso specifico di questo, ovvero lo schema a fiocco di neve (snowflake schema). Esso rappresenta un modello dimensionale, composto da una "fact table" centrale e una serie di "dimension table" correlate, che possono poi essere a loro volta scomposte in tabelle sottodimensionali.

In termini di database relazionali, in uno Star Schema, la fact table contiene generalmente una o più misure (di cui almeno una contenente un'informazione temporale) e tutte le chiavi esterne, corrispondenti alle chiavi primarie delle varie dimension table.

La fact table e' quindi l'unica ad avere join multiple con altre tabelle, mentre l'unica relazione della chiave primaria di ciascuna dimension table, avviene appunto con la fact table stessa. A loro volta, schemi a stella di questo tipo, possono diventare schemi a "snowflake" (ovvero a fiocco di neve), con l'apporto di gerarchie sugli attributi, permettendo alle dimension table di essere normalizzate a loro volta e quindi di avere tabelle sottodimensionali.

In letteratura vi sono a dire il vero alcune discussioni circa l'utilizzo di tali tabelle sottodimensionali, poiche' la struttura risultante dal loro utilizzo potrebbe portare a rallentamenti nell'esecuzione dei molteplici join della fact table, con le diverse dimension table; in molti casi pero' tale approccio è vivamente consigliato, se non altro per la chiara separazione logica che offre sui soggetti (in alcuni casi la cosa si rivela addirittura necessaria, come ad esempio per i dati anagrafici); le query possono inoltre essere ottimizzate con opportuni accorgimenti (cosa che verra' fatta e trattata in seguito), cosi' da evitare i possibili effetti negativi di tale scelta, che puo' quindi essere apprezzata per la semplicita', intuitivita' ed estensivita' che offre. [A2]

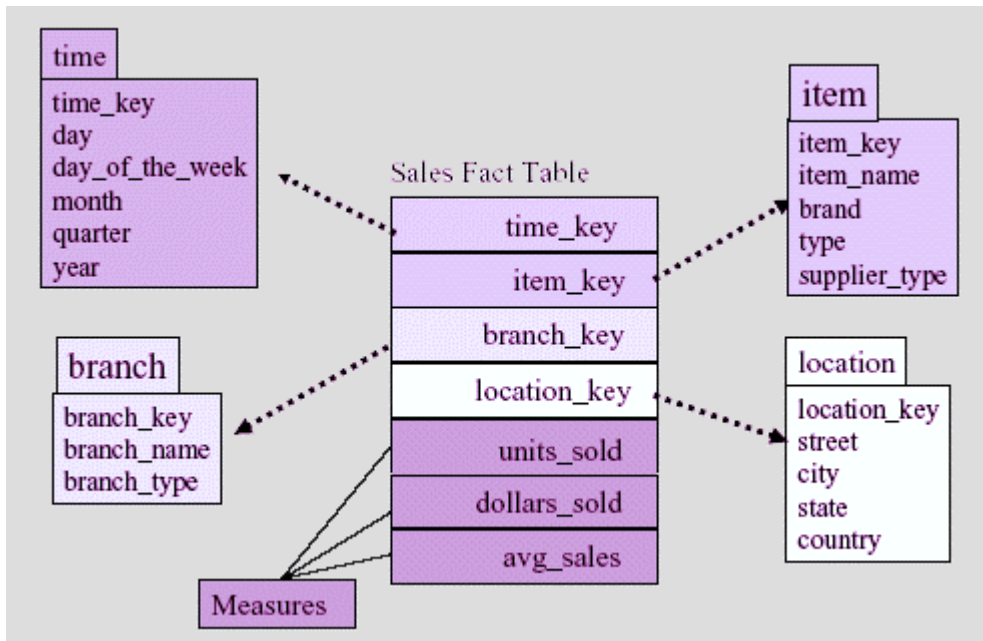


Fig 7 Esempio di Star Schema

Formalmente, possiamo quindi dire che una base di dati (D) segue uno schema a snowflake, se è aciclica e il suo join tree ha una radice designata (D0), in relazione univoca con ciascuna delle dimension table coinvolte.

Se D è uno schema a snowflake e l'altezza di JT(D) è uno, allora D è uno star schema.

Quando D ha uno schema a snowflake, D0 è chiamata la fact table, mentre Di's di altezza uno sono chiamate le dimension tables e i rimanenti schemi di relazione di altezza h>1 sono chiamate le subdimensional tables. [A2]

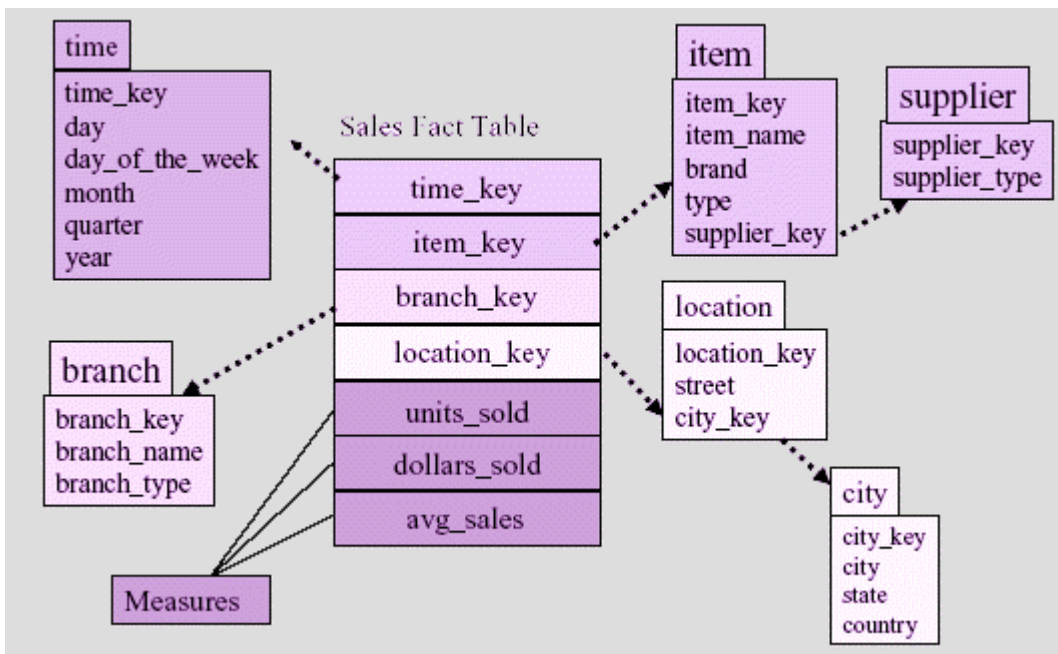


Fig 8 Esempio di Snowflake Schema

Un'ulteriore schema di design che si trova in letteratura per modelli dimensionali, è la cosiddetta fact-constellation, nella quale molteplici fact tables condividono le stesse tabelle dimensionali, viste come una “collezione di stelle” e quindi chiamata “galaxy schema” o “constellation schema”. Come viene indicato da Ralph Kimball, uno dei personaggi più attivi negli studi inerenti il data warehousing, senza alcun dubbio questo è l'unico approccio da seguire, quando vi sono più misure da registrare, che coinvolgono gli stessi soggetti. La duplicazione delle dimension table, porterebbe infatti ad ovvie e inutili inconsistenze e ingiustificate difficoltà gestionali/analitiche. [A3]

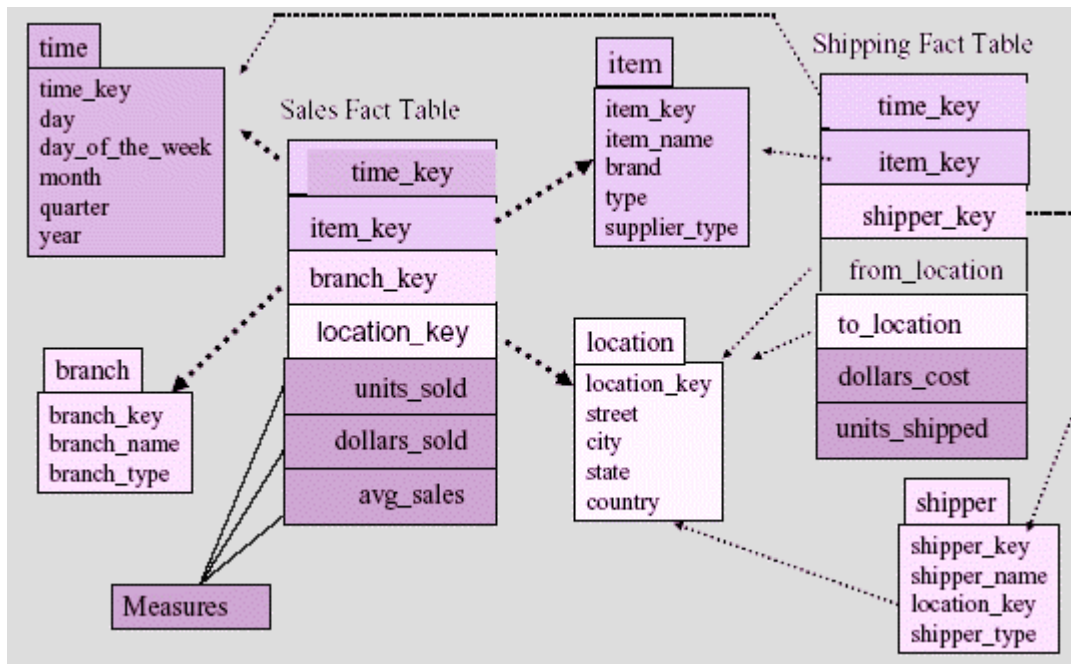


Fig 9 Esempio di Constellation Schema

Come si è potuto constatare, è sostanzialmente la disposizione degli oggetti coinvolti che cambia in questi modelli; ad ogni modo, le fact table contengono tipicamente i fatti (o le misure) e chiavi esterne, alle dimension table; le dimension table, conosciute anche con il nome di lookup table o reference table, contengono i dati relativamente statici del data warehouse e sono solitamente di tipo testuale. Vengono di seguito riportate alcune informazioni aggiuntive su questi importanti oggetti, che costituiscono solitamente la base di un data-warehouse.

2.2.2 Fact Table

Questa tabella contiene tutte le misure, ovvero gli elementi d'indagine che variano in continuazione nelle entità che stiamo considerando. Contiene praticamente una colonna per ogni dimensione del cubo di analisi, oltre che (eventualmente ...si vedano in seguito le factless fact table) una colonna “value” con il valore della misura e una o più referenze temporali.

La *chiave primaria* di una fact table è composta da tutte le *chiavi esterne* che la legano alle dimension table, oltre che dall'elemento temporale; se ne deduce che ogni record della fact table è individuato dai record delle dimension table: uno per ogni tabella dimensionale. Pensando allo star schema come ad un grafico multidimensionale, ciascun record della fact table si trova in un punto, le cui coordinate finite e discrete sono determinate da un elemento preso da ciascuna dimensione (si pensi all'ipercubo, visto in precedenza). [A4]

Come suggerisce R. Kimball, è bene far sì che queste chiavi esterne siano tutte chiavi “surrogate” (traduzione letterale dall'inglese), ovvero una chiavi artificiali, a sostituzione di eventuali chiavi naturali, usate in produzione.

Il motivo principale che porta a questa scelta sarà chiarito a breve (nel contesto delle slowly changing dimension); per il momento è comunque doveroso notare che l'utilizzo di chiavi surrogate (di tipo numerico) comporta generalmente un notevole risparmio di spazio, rispetto all'utilizzo, ad esempio di chiavi naturali, di tipo testuale.

Il rovescio della medaglia di tale approccio è dato dalla necessità di eseguire (in sede di inserimento dei dati) un "look up" nelle dimension table per ogni record inserito, alla ricerca di tali chiavi "imposte" (a sostituzioni di quelle naturali o cosiddette, "di produzione"); ottimizzando comunque le procedure batch per il popolamento del data warehouse (questo è quanto accadrà in seguito), si è comunque in grado di semplificare e velocizzare notevolmente il processo. [A5]

La fact table risulta quindi fortemente normalizzata, perchè contiene solamente la chiave primaria e i pochi attributi che variano nel tempo, che costituiscono le misure delle indagini; la cosa non è sempre vera invece nelle dimension table.

Di solito una fact table viene aggiornata giornalmente: per questa ragione, se l'arco di tempo che contiene è di qualche anno, il numero dei suoi record può raggiungere e superare qualche milione. Bisogna allora porre particolare attenzione nella progettazione di questa tabella, soprattutto nella scelta del tipo di campi che vogliamo salvare e degli indici che permetteranno l'accesso selezionato. Si prediligono solitamente i campi di dimensione ridotta (numerici) e gli indici di tipo bit map (analizzati tra poco).

2.2.3 Dimension Table

Le dimension table contengono le descrizioni delle cosiddette "dimensioni di mercato". Per esempio possiamo trovare in una dimension table la completa definizione di un prodotto con tutti i suoi attributi. I migliori attributi sono quelli testuali che possono essere usati come sorgente di restrizioni nelle query degli utenti o come intestazioni degli insiemi di risposta agli end-user.

La *chiave primaria* di una dimension table è composta da un solo attributo (a differenza di quella di una fact table che è composita) che si ripete come *chiave esterna* (surrogate key) nella fact table.

Non è possibile mettere direttamente in relazione tra loro due dimension table e spesso non ha neanche senso cercare di connettere due dimensioni, perchè riguardano soggetti logicamente diversi; la loro unione acquista significato solo attraverso il verificarsi di un fatto (e quindi attraverso fact table). [A4]

In una ricerca che coinvolge un modello dimensionale, è importante che per prima cosa vengano scandite le dimension table per determinare quali elementi soddisfino le caratteristiche selezionate e poi la fact table per ricavare i risultati cercati. Il DBMS deve essere forzato a lavorare in questo modo, per evitare di accedere alla tabella più numerosa finché non si sono determinate le condizioni di selezione dei record sulle tabelle meno popolate e quindi più velocemente interrogabili. Si dice in questo caso che viene applicata la tecnica dello **slice and dice**, ossia letteralmente dell'"*affettare e tagliare a dadi*" l'ipercubo dimensionale, intendendo che è possibile interrogare e visualizzare selettivamente alcune parti della fact table, selezionandole precedentemente in base ad un determinato range di valori di una o più dimensioni; tale operazione aumenta notevolmente la velocità di risposta del sistema, poichè le selezioni avverranno su tabelle meno densamente popolate.

Sempre per questioni di prestazioni, le dimension table sono talvolta denormalizzate (star schema), perchè sono quelle che subiscono più accessi in lettura e, quindi, sebbene i dati siano ridondanti, risultano più rapidi i tempi di risposta; inoltre il risparmio di spazio che si ottiene dalla loro normalizzazione è solitamente insignificante, data la differenza di dimensione tra i file che contengono la fact table e quelli che contengono le dimension table. Alcuni espedienti (vedi ad es. Join Index, in seguito) consentono tuttavia di ottimizzare le operazioni di tale tipo, permettendo quindi la coesistenza di prestazioni e pulizia nel design.

2.2.4 Factless Fact Table

Le factless fact table sono letteralmente da intendersi come fact table che non “immagazzinano” alcuna misura.

Un tipo di tabelle di questo tipo è quello preposto alla registrazione di eventi; un esempio può essere fornito da un sistema che tiene conto delle partecipazioni dei ricercatori nei progetti; non dev'essere inserita una misura, come ad esempio, le ore di lavoro eseguito o previsto, ma solo un record che attesti una partecipazione; ecco come in questo caso sia possibile avere una tabella contenente chiavi esterne di dimension table (progetti, ricercatori) ed eventualmente elementi temporali (anno in cui è partito il progetto, ad esempio), ma non vi è la necessità di registrare alcuna misura.

In alcuni casi (come nel nostro), si può tuttavia decidere di utilizzare un codice fittizio (es. 1) per denotare questo evento, riuscendo così a collocare tali dati, anche in una tabella inizialmente strutturata per l'accoglimento di misure; tale espediente rende inoltre il codice SQL prodotto per le interrogazioni più leggibile e ha impatto minimo sulle dimensioni del database.

Un altro tipo di factless fact table è chiamato “coverage table” e viene frequentemente usato quando le informazioni contenute nella fact table primaria non sono sufficienti. Si consideri l'esempio di una fact table, che tiene conto delle vendite di prodotti durante alcune promozioni; tale tabella risponde a molte domande interessanti, ma non è in grado di rispondere a domande del tipo “cosa NON è successo?”; per esempio, questa non è in grado di rispondere a domande del tipo: “Quali prodotti in promozione non sono stati venduti?”; questo avviene poiché la fact table registra solamente le vendite. La coverage table giunge quindi in aiuto in questo caso; un record viene inserito nella coverage table per ogni prodotto in promozione di ogni negozio. A questo punto, per rispondere alla domanda “Quali prodotti in promozione non sono stati venduti?” occorre un doppio passaggio; in primo luogo si cercano nella coverage table i prodotti in promozione in quel giorno, in quel negozio; in secondo luogo, si consulta la fact table principale, ricercando i prodotti che sono stati venduti. La risposta giunge quindi dalla disgiunzione degli insiemi risultato. [A6]

2.2.5 Slowly changing dimension

I valori di alcuni attributi dei record appartenenti ad una dimension table possono cambiare nel tempo; se il cambiamento è tanto rapido, da farlo salvare nella fact table, non ci sono problemi e ci si può attenere agli schemi presentati in precedenza; se tuttavia questi cambiamenti sono sporadici, ci si trova di fronte ad una cosiddetta slowly changing dimension.

Il metodo migliore per gestire tali eventualità deriva direttamente dall'utilizzo delle chiavi surrogate citate in precedenza e consiste nell'inserimento di un nuovo record in una dimension table, ogni qualvolta un attributo del soggetto interessato cambia.

È sicuramente l'approccio migliore, perché permette di navigare avanti ed indietro nella storia dell'elemento e come unico svantaggio comporta una maggior necessità di spazio su disco; considerando comunque che la cardinalità delle dimension table è generalmente di diversi ordini di grandezza inferiore rispetto alla fact table, è spesso possibile trascurare tale particolare. Interessante invece notare che non occorre aggiungere attributi di tempo alla tabella delle dimensioni, per tener conto di quando un elemento è cambiato; questa tecnica comporta infatti un partizionamento implicito e indiretto della fact table, nella quale tutti i record precedenti alla data di cambiamento di stato saranno riferiti alla versione precedente dell'elemento, mentre tutti i record correnti saranno legati alla versione corrente (nuovo record). Così attraverso la fact table sarà possibile risalire alla storia degli elementi stessi.

Nella pratica, si possono incontrare, a dire il vero, altri approcci a fronte di questo problema, ma questi derivano spesso da una cattiva progettazione di base (usando ad esempio chiavi naturali nelle dimension table, anziché le surrogate key viste in precedenza) e rischiano di comportare seri problemi nell'utilizzo del data warehouse. [A7]

Molteplici esempi di tali problematiche possono essere immediatamente individuati, quando ad esempio si utilizzano production key (con tale termine intendiamo gli appellativi con cui vengono identificati ad esempio i prodotti di un'azienda, in un sistema OLTP) come chiavi per le dimension table; eccone alcuni:

La produzione può decidere di modificare qualche dettaglio di un prodotto (soggetto), senza tuttavia cambiare la chiave; la storia del soggetto viene così persa [problema delle slowly changing dimension in senso stretto]

La produzione può decidere di riutilizzare chiavi dismesse da tempo, ma di cui il data warehouse aveva tenuto traccia, andando così a confondere soggetti diversi.

La produzione può decidere di ristrutturare il formato delle chiavi, ad esempio per risparmiare spazio; si rischia in questo caso di dover ri-aggiornare l'intera fact table, andando incontro a potenziali conflitti con dati aggregati provenienti da altri sistemi (es. dimensione/tipo dei campi – numerici/alfanumerici).

Risulta quindi immediato che l'esigua quantità di spazio risparmiata non basta assolutamente a giustificare un approccio di tale tipo, che anzi, in alcuni casi viene fortemente ed esplicitamente criticato, anche nella letteratura [A3].

2.3 Design Fisico

Il livello fisico costituisce la concretizzazione del livello logico e viene ottenuto andando a creare le tabelle del database (facendo quindi attenzione ai dettagli, come per esempio i tipi di campi utilizzati, l'allocazione dello spazio, il tipo di strutture dati da utilizzare, ecc), nonché aggiungendo le opportune chiavi, gli indici e le cosiddette “*Integrity Constraints*”, ad assicurare che la memorizzazione dei dati sia conforme al modello scelto. [L7]

Vengono così implementate le varie tabelle, cercando di allocare opportunamente lo spazio sui dischi, in base alle previste esigenze e impiegando identificatori univoci (spesso chiavi primarie), in grado di differenziare sempre record tra loro diversi, ma che riguardano lo stesso soggetto.

Le fact table si presenteranno altamente normalizzate e prive di campi testuali (essendo quelle maggiormente popolate, ciò assicura un cospicuo risparmio di spazio), mentre tutte le descrizioni per gli elementi dimensionali troveranno spazio nelle dimension table, composte per lo più da campi testuali appunto e talvolta denormalizzate.

Dopo aver realizzato tali tabelle, occorre procedere anche alla definizione di ulteriori strutture di supporto, a garanzia delle prestazioni e della consistenza della base di dati. In particolare può essere utile (o necessario) aggiungere indici, ulteriori restrizioni a garanzia dell'integrità ed eventualmente viste.

Le caratteristiche fisiche del modello dei dati e gli strumenti messi a disposizione del progettista (indici, integrity constraints, ecc..) dipendono in gran parte dal DBMS che intendiamo usare; in fase di progettazione di un data warehouse è quindi fondamentale anche la scelta del DBMS da utilizzare.

Alcune Integrity constraints di tipica implementazione, utilizzate per assicurare il rispetto delle regole associate alla base di dati e per prevenire l'inserimento di informazioni non valide nelle tabelle, sono ad esempio la constraint UNIQUE, la FOREIGN KEY e la condizione NOT NULL. [L7]

La prima di esse serve a garantire l'unicità dei valori di una certa colonna; ad esempio, nel nostro ambito di applicazione, in una tabella dimensionale, non vogliamo che vi siano due strumenti distinti con lo stesso nome. Ecco che allora possiamo adottare un indice di tipo UNIQUE, per evitare tale ipotesi.

Un altro esempio di constraint, è costituito dalla FOREIGN KEY, istituita per assicurare che due chiavi siano in relazione primaria-esterna. In uno schema a stella, una constraint di chiave esterna serve a validare le relazioni tra una fact table e le dimension tables.

Un ulteriore esempio di queste constraints è fornito dalla clausola NOT NULL, utilizzata per impedire l'inserimento di valori nulla in una certa colonna.

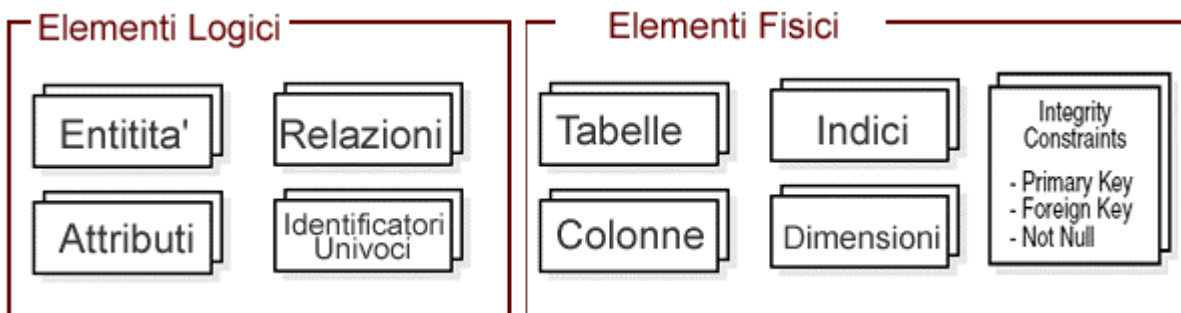


Fig 10 – Oggetti del Livello Logico e Oggetti del Livello Fisico

Le viste costituiscono invece presentazioni preimpostate di dati contenuti in una o più tabelle o altre viste; una vista prende l'input di una query e le tratta come fosse una tabella, senza richiedere tuttavia spazio aggiuntivo nel database ed evitando ridondanze.

Un ulteriore e importante strumento utilizzato nella conservazione dei dati nei data warehouse, è il partizionamento dei dati, studiato per migliorare la gestione di grandi quantità di informazioni e migliorare anche le performances del sistema, offrendo la possibilità di eseguire interrogazioni in parallelo sui dati.

Nei data warehouse tale approccio riguarda soprattutto quei campi legati ad elementi temporali, solitamente suddivisi in sotto-periodi, che divengono così più facilmente gestibili.

2.3.1 Utilizzo degli Indici

Per avere successo, un data warehouse deve permettere un accesso ai dati intuitivo, facile e soprattutto immediato; uno strumento molto utile per velocizzare le interrogazioni sono gli indici, strutture opzionali associati alle tabelle, operano proprio come gli indici di un libro o di un elenco telefonico, permettendo di recuperare le informazioni in modo rapido. [L5]

Per avere successo un data warehouse deve permettere un accesso ai dati intuitivo, facile e soprattutto immediato; e' per questo auspicabile che il DBMS supporti innanzitutto diversi tipi di indicizzazione, tra le quali, spiccano per importanza quelle che si basano su:

- Indici Bit Map
- Indici Join
- Indici B-Tree

I **bit map index** rappresentano una buona tecnica di indicizzazione per attributi con bassa cardinalita' del dominio; vengono costruiti a partire dai valori distinti di una colonna di una tabella e sono costituiti da un distinto vettore di bit, per ogni possibile valore V del dominio. Quando viene istanziata una query, vengono quindi considerati gli indici che soddisfano la condizione (bit posto a 1), combinandola con le altre clausole: in questo modo, se una query deve soddisfare delle condizioni poste su due o più attributi, le righe di risultato possono essere calcolate con l'ausilio di semplici operazioni logiche sulle sequenze di bit ottenute dalla scansione parallela degli indici.

Cust	Autore	Stato
C1	N	H
C2	S	M
C3	W	L
C4	W	H
C5	S	L
C6	W	L
C7	W	H

RowId	N	S	E	W
1	1	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	0	0	1
5	0	1	0	0
6	0	0	0	1
7	0	0	0	1

RowId	H	M	L
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
5	0	0	1
6	0	0	1
7	1	0	0

SELECT Cust FROM Progetti WHERE Autore = W AND Stato = L

Fig 11 – Esempio di Bitmap Index

Il vantaggio derivante dall'uso di indici bitmap cresce nelle colonne nelle quali la proporzione tra il numero di valori distinti e il numero totale di records e' inferiore al 1%. Un ottimo esempio di questo tipo e' dato, ad esempio, da un'ipotetica colonna "SESSO", dove a fronte di milioni di potenziali records, vi sono solamente due gradi di cardinalita' dei soggetti.

Un'indicizzazione di questo tipo puo' essere notevolmente piu' performante rispetto ad una ottenuta per mezzo dei B-tree, soprattutto quando la colonna considerata viene interrogata congiuntamente ad altre colonne indicizzate (cosa tipica, peraltro nel contesto di un datawarehouse). [W2] Contrariamente ad altri tipi di indici, i bitmap includono anche i record che hanno valori NULL; la cosa puo' essere utile in alcuni tipi di interrogazioni SQL, come ad esempio nel caso di operazioni che coinvolgono una funzione aggregata COUNT, dove un esempio di questo tipo e' dato dalla query: "SELECT COUNT(*) FROM employees".

Qualsiasi indice bitmap puo' essere usato per questa query, poiche' tutti i record sono indicizzati, inclusi quelli che hanno valori NULL; se cosi' non fosse stato (vedi nel caso dei B-Tree), l'ottimizzatore avrebbe dovuto usare come indici solamente le colonne che non potevano contenere valori NULL.

L'utilizzo di tali indici puo' quindi velocizzare le interrogazioni (si citano miglioramenti che superano anche l'8.5% in fatto di velocita' di esecuzione); vi sono tuttavia alcune considerazioni da fare in merito a tale tipologia di indici:

Nelle interrogazioni SQL contenenti una selezione "WHERE", non vi debbono ovviamente essere referenze a colonne che non sono contenute nell'indice (nell'ambito della selezione stessa)

Bisogna considerare che l'overhead derivante dall'aggiornamento degli indici bitmap e' sostanziale; tipicamente gli indici bitmap sono distrutti e ricostruiti nell'ambito delle normali procedure batch di manutenzione del database, che seguono l'importazione dei dati.

I **join index** sono sempre utili nell'ambito delle interrogazioni e sono direttamente correlati con la struttura a stella o a snowflake di un datawarehouse. Sono impiegati per indicizzare e quindi mettere in relazione immediata i valori delle dimensioni alle misure di una fact table e costituiscono in pratica un indice bitmap su due o piu' tabelle.

Un indice join e' un metodo efficiente (e salvaspazio) per ridurre il volume dei dati da sottoporre a join, andando ad effettuare preventivamente le selezioni sui records da correlare; per ogni valore nella colonna di una tabella, l'indice join tiene conto del numero di riga corrispondente, in una o piu' altre tabelle. [L7]

Locazione		
Key	Citta'	...
rid1	1	Stockholm ...
rid2	2	London ...
rid3	3	Paris ...

Prodotto		
Key	Nome	...
rid22	1	# 5 ...
rid23	2	Noah ...
rid24	3	Esempio ...

Vendite (fact table)				
	LKey	PKey	TKey	Qnt
rid4	1	1	1	5
rid5	1	2	1	7
rid6	1	3	1	4
rid7	2	1	1	8
rid8	2	2	1	3
rid9	2	3	1	5
rid10	3	1	1	20
rid11	3	2	1	10
rid12	3	3	1	30
rid13	1	1	2	10
rid14	1	2	2	9
rid15	1	3	2	7
rid16	2	1	2	5
rid17	2	2	2	10
rid18	2	3	2	8
rid19	3	1	2	20
rid20	3	2	2	50
rid21	3	3	2	30

Periodo		
Key	Mese	...
rid25	1	Jan ...
rid26	2	Feb ...
rid27	3	Mar ...
rid27	4	Apr ...

**JOIN INDEX
LOC.-PROD.**

LocK	PrdK	Rid
1	1	rid4
1	1	rid13
1	2	rid5
1	2	rid14
1	3	rid6
1	3	rid15
...		

Fig 12 - Esempio di Join Index

Gli **indici B-tree** sono infine i piu' efficienti quando si hanno colonne con un alto grado di cardinalita'; nell'ambito di un sistema per il data warehousing, i B-tree dovrebbero essere utilizzati solo per colonne "unique" o comunque altre colonne con cardinalita' molto alte.

Uno studio approfondito di tali strutture non e' argomento di tale elaborato; vale tuttavia la pena di ricordare che un B-tree (di ordine m) e' un albero con le seguenti caratteristiche:

- ogni nodo ha un numero di figli minore o uguale a m
- ogni nodo, eccetto la radice, ha un numero di figli maggiore o uguale a m/2
- la radice ha almeno 2 figli, a meno che non sia una foglia
- tutte le foglie sono allo stesso livello
- un nodo che non sia una foglia e che abbia k figli, contiene k-1 chiavi [L6]

In molti casi non dovrebbe essere comunque necessario indicizzare le colonne con dati ad alta cardinalita' nel contesto di un data warehouse (tipicamente le misure stesse), dato che le operazioni

di selezione primare dovrebbero riguardare le dimension table in primo luogo; e' pertanto consigliato utilizzare soprattutto indici di tipo bitmap, facendo uso di tali indici solo in alcuni casi specifici. [L7]

3.Considerazioni sull'Hardware

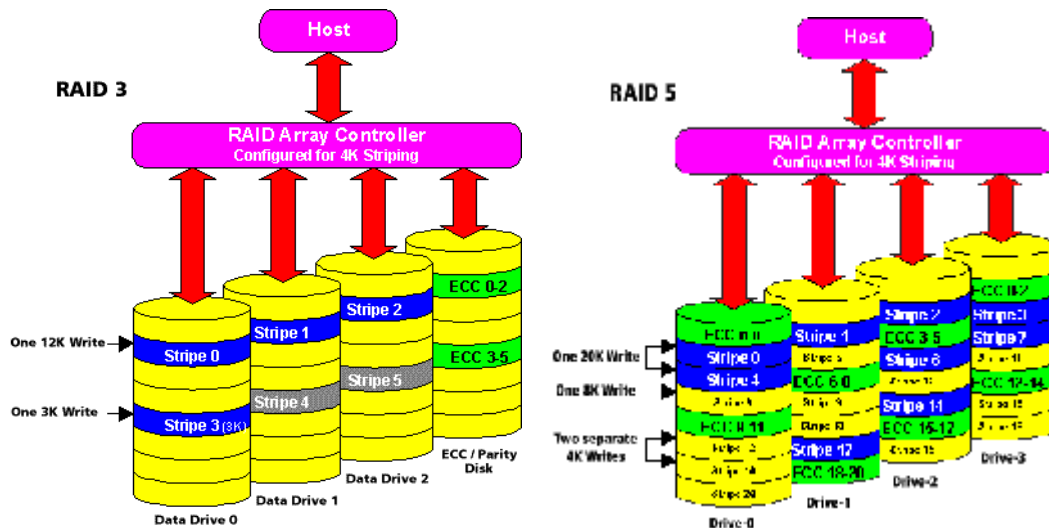
Data l'enorme quantita' di dati con cui hanno generalmente a che fare, i data warehouse risentono spesso delle performance dei dispositivi di I/O impiegati; una volta progettati gli schemi, bisogna quindi porre una certa attenzione anche per quanto riguarda la scelta di una piattaforma sulla quale far girare l'applicazione (o meglio l'insieme delle applicazioni che costituiscono il data warehouse).

Si fa spesso utilizzo dei cosiddetti sistemi RAID (Redundant Arrays of Inexpensive Disks), andando ad eseguire uno striping (distribuzione) dei dati su diversi dischi fisici, così da permettere il recupero delle informazioni in modo parallelo ed eliminare (o per lo meno limitare) gli effetti dei cosiddetti colli di bottiglia nell'I/O (si dice in questo caso che l'applicazione e' I/O bound). [L7]

Tale tecnica permette il recupero in parallelo di informazioni da dischi diversi e consente così di migliorare nettamente le prestazioni relative all'I/O; l'unico svantaggio che ne deriva, è dato dal fatto che non sarà piu' possibile individuare uno specifico file in un drive fisico, causando maggiori difficoltà in sede di recupero dei dati, a seguito di eventuali danni fisici su uno dei drive coinvolti; sono ovviamente previste soluzioni ridondanti, per evitare che il semplice danneggiamento di un disco pregiudichi il contenuto dell'intero array; ad ogni modo, anche se soltanto uno dei dischi dovesse guastarsi, l'intero RAID dovrebbe essere ricostruito, in seguito alla sostituzione della parte danneggiata. Per migliorare ulteriormente l'affidabilità di tale architettura è tuttavia possibile ricorrere al cosiddetto "hot spare" ("ricambio a caldo"): in caso di guasto di uno dei dischi, viene messo a disposizione un disco rigido aggiuntivo, in grado di sostituire immediatamente quello danneggiato.

Vi sono quindi diverse tipologie di RAID, che offrono lo striping e il mirroring (la ridondanza) dei dati a diversi livelli, cose entrambe importanti in questo contesto; nell'ambito dei data warehouse, una buona scelta potrebbe cadere sul RAID 3, in grado di offrire alte prestazioni, attraverso lo striping dei dati sui dischi disponibili (eccetto uno utilizzato esclusivamente per la registrazione dei bit di parità) oppure sul RAID 5, in grado di offrire un ottimo bilanciamento della ridondanza dei dati e un'elevata capacità (in questo caso non vi e' un solo disco preposto ad accogliere le informazioni di parità, che vengono invece divise tra i dischi coinvolti). La configurazione per il RAID 3 o il RAID 5 richiedono l'utilizzo di almeno 3 dischi rigidi, sebbene fonti autorevoli indicano come sia opportuno utilizzare almeno un disco per ogni processore installato (per facilitare il parallelismo nell'esecuzione dell'I/O).

La scelta tra i due avviene solitamente dopo aver considerato quali saranno le operazioni tipiche del sistema; il RAID 3 si presta meglio ad operazioni di lettura sequenziali lunghe di un processo unico, con stream unico, rivelandosi invece non adatto per I/O di processi multipli (lungi o corti). Il RAID 5, che invece utilizza lo striping del disco e la distribuzione dei dati di parità, impedisce il rallentamento I/O provocato dagli accessi costanti ad un'unica unità; e' quindi da preferirsi quando si presume che diversi processi possano essere eseguiti parallelamente. [L8]



4 Cenni sulle possibili applicazioni per un Data Warehouse

Un data warehouse nasce soprattutto come base di dati sulla quale fare ricerche e calcoli che coinvolgono grandi quantità di dati; i tipi di ricerca che si possono effettuare non sono predeterminati, ma è l'utente che di volta in volta sceglie cosa cercare: occorre quindi mettere a sua disposizione una struttura semplice che gli permetta di muoversi tra i dati, e strumenti potenti che lo aiutino nelle analisi. [L10]

Per quanto riguarda la ricerca e la navigazione tra i dati del warehouse, i livelli di summarization permettono operazioni naturali ed immediate come il *“Drill down”* ed il *“Roll up”*, mentre per le analisi su grandi quantitativi di dati troviamo veri e propri strumenti dedicati, per l'*OLAP* (On Line Analytical Processing) ed il **Data Mining**. [A9]

4.1 Drill down

L'operazione di drill down è l'operazione più antica su un data warehouse e permette sostanzialmente di partire da un livello di presentazione dei dati molto generico e scendere nei dettagli passo per passo. Nel nostro modello, gli attributi delle tabelle dimensionali giocano un ruolo molto importante a questo proposito; tali attributi sono testuali e sono la base per la selezione e il raggruppamento dei dati nei report finali. Supponendo che ci si trovi di fronte ad uno schema a fiocco di neve (snowflake), è possibile dapprima eseguire un'interrogazione che coinvolga il primo livello dimensionale, per poi proseguire nella selezione e individuare specifici record degli altri n livelli sottodimensionali. [A10]

Un esempio può rendere più chiaro questo concetto. Supponiamo di avere un'azienda e fare una ricerca che metta a confronto le vendite del prodotto A di quest'anno, rispetto a quello scorso...

Lo schema utilizzato prevede che vi siano:

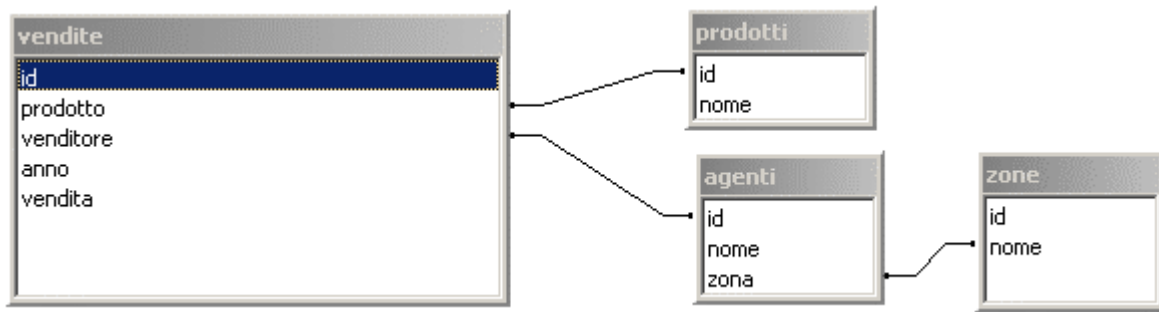


Table 1: Dati altamente riassunti.

- una fact table dove vengono registrate le vendite di quest'anno
- una dimension table con i dettagli dei singoli prodotti
- due tabelle gerarchicamente dipendenti, con gli Agenti e le Zone di Copertura

Prodotto	Regione	Quest'anno	Confronto
Prod. A	Est	110	12%
Prod. A	Ovest	179	-3%
Prod. A	Est	55	5%

Table 2: Dati altamente riassunti.

Questi dati riassuntivi evidenziano una perdita nelle vendite nelle regioni occidentali; a questo punto e' possibile cercare i motivi di queste perdite e scendere maggiormente nel dettaglio nella nostra ricerca aggiungendo il nome degli agenti che distribuiscono i prodotti nelle varie zone. Il risultato può essere quello di tab.

Prodotto	Regione	Agente	Quest'anno	Confronto
Prod. A	Est	X	52	21%
Prod. A	Ovest	Y	28	5%
Prod. A	Est	K	30	6%
Prod. A	Ovest	C	93	4%
Prod. A	Ovest	I	75	5%
Prod. A	Ovest	H	11	-15%
Prod. A	Est	Z	25	5%
Prod. A	Est	V	30	6%

Table 3: Dati più dettagliati dopo una singola operazione di drill down.

Già a questo livello di dettaglio le eventuali decisioni di chi sta facendo le ricerche sono più chiare.

4.2 Roll Up

Se l'operazione di drill down consiste nell'aggiunta progressiva di informazioni per la selezione, il rolling up permette invece di ottenere rappresentazioni più generali, a partire da analisi precise. Si procede con l'eliminazione progressiva dei parametri, utilizzati in fase di selezione dei dati. Nell'esempio precedente, in pratica, si potrebbe passare da una visione dettagliata dell'operato di ciascun agente, ad una più generale, in grado di mostrare le ripercussioni dell'operato dei singoli, a livello zonale o generale. L'operazione è in sintesi l'esatto opposto di quella di Drill Down. [A10]

4.3 On-Line Analytical Processing

Negli anni 80, Edgar Codd coniò il termine OLTP (On-Line Transaction Processing) e propose 12 criteri per l'individuazione di un sistema di questo tipo. La terminologia e i criteri furono ampiamente accettati come standard di database utilizzati per gestire i sistemi operazionali impiegati quotidianamente in una società.

Nel 1993, Codd pubblicò un altro white paper, commissionato dalla Arbor Software (ora Hyperion Solutions) e chiamato 'Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate'; sfortunatamente le regole che fornì questa volta non vennero accolte con molto entusiasmo, dato che sembravano formulate sulla base di slogan promozionali aziendali, piuttosto che da ponderate regole logico-matematiche.

Tali regole non riconobbero quindi alcun successo, cosa che invece ebbe il termine OLAP, che sembrava adatto un pò a tutti, per identificare quegli strumenti da utilizzarsi per semplificare il processo decisionale aziendale; il termine fu quindi impiegato e spesso abusato in diversi contesti, anche se, a dire il vero, non fu mai fornita una definizione formale, comunemente accettata di tale tecnologia.

Per risolvere tale problematica, l'OLAP Report propose, nel 1995 alcune regole atte a classificare un prodotto come OLAP; per non incorrere negli errori di Codd, si cercò di adottare una formula in grado di dare una definizione quanto più possibile semplice, facile da ricordare e indipendente dai venditori. La definizione risultante fu la cosiddetta *FASMI*, tuttora accettata e spesso citata in ambito di applicativi OLTP.

Volendo scomporre l'acronimo appena citato, abbiamo che per essere conforme a queste regole che lo rendano adatto all'utilizzo in ambito OLAP, un sistema dev'essere:

- **Fast:** il sistema deve riuscire a rispondere alle interrogazioni in media in cinque secondi; alle domande più facili deve dare dei risultati entro un secondo, mentre in nessun caso può superare i 20 secondi nel tempo di risposta. Alcune ricerche concludono infatti che gli utenti sono spesso tentati di comporre la tristemente combinata combinazione di tasti CTRL+ALT+CANC quando un applicativo rimane paralizzato per più di 30 secondi, molto spesso anche a seguito di speciali messaggi di avvertimento. Ad ogni modo, anche se l'utente dovesse sopportare tali tempi di caricamento, questi sarebbe probabilmente distratto da altri fattori nel frattempo, andando a spezzare la propria catena di pensiero e facendo quindi perdere di qualità alle analisi che sta conducendo. La velocità sia ben chiaro non è facile da ottenere quando si ha a che fare con grosse quantità di dati, soprattutto se sono richiesti calcoli on-the-fly e ad-hoc. Le aziende che propongono tali strumenti hanno cercato in molti di ottimizzare tali aspetti, andando anche ad agire sull'hardware delle piattaforme dove queste applicazioni devono girare; nessun prodotto si rivela attualmente ottimizzato e anzi, è questo un settore ancora in piena fase di ricerca e sviluppo.
- **Analysis:** il sistema deve riuscire a fare analisi statistiche in modo abbastanza semplice per l'utente finale. Come minimo il sistema OLAP deve fornire la possibilità di eseguire nuovi calcoli e fornire risposte a richieste specifiche particolari, anche come parte di un'analisi e di

restituire report sui dati in ogni modo desiderato dall'utente, senza che quest'ultimo debba preoccuparsi di scrivere linee di codice.

- **Shared:** il sistema deve fornire tutti i requisiti di sicurezza affinché ogni utente possa accedere ai dati di propria competenza in lettura e, se possibile avere un accesso ai dati in scrittura da parte di più utenti, deve essere in grado di gestire il locking e la concorrenza. Questa è uno degli aspetti più controversi riscontrabili nei prodotti attualmente in commercio che, considerando il “read-only” come unico approccio possibile, implementano politiche di sicurezza a dir poco semplicistiche. Anche i prodotti che vengono offerti con il supporto multi-utente la lettura/scrittura molte volte presentano modelli di sicurezza alquanto discutibili (un esempio ci è fornito in questo caso da Microsoft OLAP Service).
- **Multidimensional:** è forse il requisito più importante, l'essenza delle applicazioni OLAP. I sistemi OLAP devono fornire una vista concettuale multidimensionale dei dati.
- **Information:** è tutto ciò di cui necessita il sistema dovunque e comunque sia immagazzinato, proveniente dai dati dettagliati o aggregati.

Le tecnologie impiegate per ottenere queste caratteristiche “FASMI” sono molteplici e, tra esse, possiamo trovare, ad esempio le architetture client-server, l'orientazione agli oggetti, il calcolo parallelo, il multi-threading e in molti casi anche metodi proprietari ottimizzati per l'immagazzinamento e il trattamento dei dati.

Lo scopo di tali applicazioni è di fornire ai propri utenti le caratteristiche di analisi che si trovano in un foglio elettronico di livello avanzato, unitamente alla capacità di gestione dei dati e all'affidabilità di un database. [W4]

Gli strumenti OLAP devono quindi avere una buona integrazione anche con i sistemi che forniscono loro i dati; alcuni OLAP usano database multidimensionali proprietari fornito col prodotto, altri si interfacciano con database relazionali esistenti ed altri ancora sono una forma ibrida, che tiene alcuni dati in database multidimensionali ed accede ad altri direttamente sugli archivi relazionali.

Grazie al supporto di queste diverse forme di basi di dati, OLAP permette una vista dei dati che può andare oltre le due o tre dimensioni offerte dai fogli elettronici, pur fornendo gli strumenti e l'immediatezza che questi mettono a disposizione.

Il modello multidimensionale organizza i dati in termini delle dimensioni di mercato attuali dell'azienda. Per esempio le vendite possono essere classificate per prodotto, cliente, periodo storico, localizzazione, valore e quantità venduta. L'intersezione di tutte le dimensioni produce una cella come nei fogli elettronici a due dimensioni. Sebbene questo tipo di dati possa essere certamente immagazzinato in un database relazionale, l'SQL non è il modo naturale di estrarre informazione da una struttura multidimensionale.

Localizzare una cella in un database multidimensionale è facile per l'utente e per il calcolatore perché si conosce la posizione e non occorre ricorrere ad un indice, infatti le dimensioni ed i loro range sono noti, ciò che è ignoto è il dato contenuto nella cella, ma questo occupa uno spazio ed una posizione ben definiti in memoria. È questo il motivo che rende i database multidimensionali più facili da usare e aumenta notevolmente la resa nel manipolare dati multidimensionali in confronto a quella che si ha con i database relazionali. Il prezzo da pagare è la sparsity ossia la presenza di moltissime celle con contenuto nullo. Ciò accade per esempio perché in ciascun periodo ogni cliente compera una piccola porzione dei prodotti disponibili (se non addirittura niente). Nei modelli multidimensionali con molte dimensioni, la maggioranza delle celle non conterrà alcun dato. Perciò il prezzo della velocità è la memoria e viceversa.

Come nei fogli elettronici, ciascuna cella può essere calcolata da formule che richiamano altre celle. Nelle applicazioni di grandi dimensioni bisogna bilanciare bene l'uso delle risorse, perché la

maggior parte delle celle vengono calcolate a partire da altre, quindi bisogna scegliere un compromesso tra un calcolo in modalità batch, che salva i risultati in apposite tabelle (spreco di spazio), ed un calcolo in tempo reale, che fornisce i risultati al volo (spreco di tempo e CPU). Il vantaggio che si ottiene dalla prima soluzione è la disponibilità di informazione derivata accessibile in tempi brevissimi perché non deve essere calcolata on line ad ogni richiesta.

Gli utenti finali devono avere la possibilità di fare le analisi desiderate e navigare in tutte le dimensioni dell'applicazione senza restrizioni sulle funzionalità di calcolo o di report e con piccoli effetti sul rendimento del sistema.

Come si è detto, i tempi di risposta nelle interrogazioni sono imperativi nell'OLAP; l'abilità di manipolare intuitivamente i dati richiede il recupero rapido delle informazioni. Generalmente, più calcoli sono richiesti per produrre delle informazioni, più lento sarà il tempo di risposta. Per tale motivo, per mantenere i tempi di interrogazione contenuti, alcune informazioni, sulle quali avvengono molti accessi possono essere pre-calcolate (possono cioè essere dapprima elaborate e poi salvate nella base di dati nel loro nuovo formato, già aggregato). [W3][W4]

Un esempio di questo tipo di dati che si prestano al calcolo preventivo si ha con i dati riepilogativi, come per esempio le tabelle con le vendite di un'azienda mensili, semestrali o annuali.

Venditori diversi hanno approcci diversi sulla scelta delle variabili da pre-aggregare e sul modo di eseguire tali operazioni; è un'aspetto questo sul quale bisogna fare molta attenzione ...se è vero infatti che un'interrogazione su dati già aggregati sarà generalmente più veloce di una eseguita da dati grezzi, l'utilizzo massiccio ed eccessivo di questi valori precalcolati non solo contribuisce a far crescere a dismisura le dimensioni del database, ma anche i tempi che saranno necessari di volta in volta per eseguire le singole operazioni di aggregazione.

Inoltre, quando nuovi valori saranno aggiunti o modificati sulla base di dati, tali modifiche dovranno propagarsi anche sui dati pre-calcolati, che dipendono dalle nuove informazioni aggiunte; dato che tale processo richiede del tempo e, spesso gli aggiornamenti mantengono il database offline, tale operazione potrebbe dar luogo a fastidi per gli utenti.

Un metodo di classificazione generalmente utilizzato per classificare gli applicativi OLAP tiene in considerazione due aspetti importanti del loro operato, quali il tipo di memorizzazione dei dati adottato e il modo di elaborare i dati.

Memorizzazione dei Dati:

- Database Relazionali
- Database Multidimensionali
- Files Formato Client (in questo caso i dati da analizzare vengono estratti, in prima istanza, sul computer Client che ne eseguirà le analisi)

Modalità di Elaborazione dei Dati:

- Database Relazionali (utilizzo di SQL per le interrogazioni)
- Motori Multidimensionali (server dedicato, con opportune ottimizzazioni)
- Client (calcolo eseguito sulle CPU dei computer client)

Dalla combinazione delle categorie appena analizzate, otteniamo teoricamente 9 opzioni; alcune di esse non hanno tuttavia senso, dato che, per esempio, non ha alcun senso utilizzare un database multidimensionale per la memorizzazione dei dati e poi semplici query SQL per l'analisi degli stessi.

Per questo motivo, le architetture che hanno senso sono sei e vengono riportate di seguito, assieme con alcuni esempi di produttori che le adottano. [L10]

	DB Relazionali	DB Multidimensionali	File su Client
Query SQL	1 Microstrategy DSS		
Motori Multidimensionali	2 Oracle Express Informix	4 Microstrategy DSS Hyperion Essbase	
Client	3 Oracle Discoverer	5 Hyperion Enterprise	6 BrioQuery BusinessObjects

Fig 13 – Tipologie di OLAP

Sulla base di questa matrice, notiamo che con il termine ROLAP (Relational OLAP) vengono individuati i prodotti che si posizionano nelle caselle 1,2 e 3; il termine MOLAP (Multidimensional OLAP) indica i prodotti che si collocano nelle caselle 4 e 5; i prodotti chiamati HOLAP (Hybrid OLAP) si posizionano contemporaneamente nelle caselle 2 e 4; i prodotti chiamati DOLAP (Desktop OLAP) si posizionano nella casella 6.

Per quanto riguarda le applicazioni più comuni che è possibile effettuare con uno strumento orientato all'OLAP, oltre che il Roll Up e il Drill Down viste in precedenza, troviamo:

Slice: coincide con la fase di selezione di una dimensione di un cubo e della sua presentazione.

Dice: crea un “sub-cubo” del cubo corrente, selezionando una o più dimensioni e il range di valori da considerare all'interno di queste dimensioni.

Pivot: è l'operazione che sostanzialmente rimuove una dimensione da un cubo e la sostituisce con un'altra. Un esempio può essere utile in questo caso. Supponiamo che il proprietario di un sistema di e-commerce voglia vedere le vendite mensili totali per un grande numero di oggetti, in un foglio di calcolo. Se l'utente volesse sapere quali oggetti fossero stati venduti meglio in un particolare semestre, sarebbe particolarmente esoso in termini di tempo andare alla ricerca di tali informazioni tra le diverse pagine. Una tabella pivot permetterebbe invece di riorganizzare velocemente i dati e creare un sommario per ogni oggetto, nel semestre di riferimento. *[W4]*

4.4 Data Mining

Il termine Data Mining letteralmente significa “estrarre dati” e viene spesso citato in letteratura anche come Knowledge Discovery in Databases (scoperta della “conoscenza” dai dati contenuti nei database). *[A][L11][L12]*

Nonostante il mercato pulluli al momento di tutta una serie di nuovi prodotti e aziende specializzate in tale settore, il soggetto di tale materia ha in realtà radici piuttosto antiche, che lo legano ad una tradizione di ricerca e pratica che continua da non meno di 40 anni. Il nome “data mining” apparve per la prima volta all'inizio degli anni 60, con implementazioni che non esulavano dalla semplice analisi statistica; i pionieri in questo campo erano SAS, SPSS e IBM, tutte, tra l'altro compagnie che ancor oggi sono molto attive nell'ambito del Data Mining.

Le applicazioni che venivano sviluppate al tempo miravano a fornire strumenti in grado di eseguire le più comuni funzioni statistiche, come per esempio la correlazione, la regressione, il chi-square e la cross-tabulation; SAS e SPSS in particolare continuano ad offrire anche questi prodotti classici, ma il termine data mining ha oramai assunto una connotazione che lo lega maggiormente ad approcci che cercano anche di spiegare e soprattutto predire cosa stia dietro (o davanti) ai dati analizzati.

Alla fine degli anni 80 l'analisi statistica classica fu estesa e permeata da una serie di concetti nuovi, quali ad esempio “la logica fuzzy”, “il ragionamento euristico” e le “reti neurali”, individuati come facenti parte di quelli che erano i risultati degli studi sull'intelligenza artificiale (AI).

Al di là del successo ottenuto dall'AI in quegli anni, tali termini si prestavano bene all'utilizzo in ambito commerciale, dove venivano spesso e volentieri impiegati per promettere meraviglie ai potenziali clienti. Gli ambiti d'applicazione di tali tecniche risultarono tuttavia limitati ed erano spesso richiesti grandi sforzi per codificare ed inserire conoscenza ed esperienza umane in un sistema.

E' solo verso la fine degli anni novanta che si iniziò ad avere un approccio migliore con queste nuove tecnologie e si riuscì a concretizzare qualche buona idea, probabilmente anche grazie ai nuovi strumenti che venivano offerti (uno su tutti il Data Warehouse). Attualmente vi sono diversi modi di descrivere il data mining; vengono di seguito riportate alcune definizioni ritenute significative:

*"Il **Data Mining** è il processo di scoperta di nuove correlazioni, strutture e tendenze significative, setacciando grandi moli di dati depositate in archivi, utilizzando sia tecnologie di riconoscimento di modelli che tecniche statistiche e matematiche"*
(Gartner Group Inc) [W2]

*"Il **Data Mining** è un processo di scoperta di conoscenza che consiste nell'estrazione di informazioni precedentemente sconosciute ed utilizzabili da database di grandi dimensioni"*
(Meta Group) [W6]

*"Il **Data Mining** è il processo di esplorazione ed analisi, con strumenti automatici e semi-automatici, di grandi quantità di dati al fine di scoprire strutture e regole significative"*
(Michael Berry, Gordon Linoff "Mastering Data Mining") [L11]

Andando ad estrarre elementi comuni a queste definizioni, possiamo quindi dire che il Data Mining consiste nell'uso di una varietà di tecniche per identificare *pepite* di informazione, utili a livello decisionale, ma sommerse da grandi quantità di dati ed estrarre queste pepite in modo tale che possano essere poste in uso in aree quali il DSS, la previsione e la stima.

I dati spesso sono così voluminosi che non se ne può fare un uso diretto ed è l'informazione nascosta quella utile. [I2]

Giunti a questo punto, qualora non fosse ancora chiara la differenza tra OLAP e Data Mining, possiamo citare a riguardo il fatto che quest'ultimo ha come obiettivo principale la scoperta di "patterns" nei dati, mentre l'OLAP si limitava a presentare o visualizzare dati preventivamente analizzati.

È importante notare che nel data mining è il computer che si occupa di trovare modelli dei dati, identificandone regole e caratteristiche che li legano. Il processo di analisi parte da un insieme limitato di dati e, usando una certa metodologia, cerca di sviluppare una rappresentazione ottimale della struttura dei dati; durante questa fase il processo acquisisce conoscenza. Una volta che tale conoscenza è stata acquisita, questa può essere estesa ad un insieme più vasto di dati basandosi sull'assunzione che il largo insieme di dati ha una struttura simile a quello più semplice.

Le fasi che portano dall'insieme dei dati grezzo all'estrazione della conoscenza possono essere riassunte nei cinque punti, individuati inizialmente da Usama Fayyad [L13] e ora comunemente accettati:

- **Selezione:** selezione o segmentazione dei dati secondo alcuni criteri;
- **Preprocessing:** "pulizia" dei dati da certe informazioni ritenute inutili e che possono rallentare le future interrogazioni. In questa fase, inoltre, i dati possono essere trasformati per evitare eventuali inconsistenze dovute al fatto che dati simili possono provenire da sorgenti diverse e quindi con metadati leggermente diversi (ad esempio in un database il sesso di una persona può essere salvato come "m" o "f" ed in un altro come 0 o 1);

- **Trasformazione:** i dati non sono semplicemente trasferiti da un archivio ad uno nuovo, ma sono trasformati in modo tale che sia possibile anche aggiungere informazione a questi, come per esempio informazioni demografiche comunemente usate nella ricerca di mercato. Quindi i dati vengono resi “usabili e navigabili”;
- **Data Mining** (in senso stretto): questo stadio si occupa di estrarre dei modelli dai dati. Un modello può essere definito come: dato un insieme di fatti (i dati) F , un linguaggio L ed alcune misure di certezza C , un modello è una dichiarazione S nel linguaggio L che descrive le relazioni che esistono tra i dati di un sottoinsieme G di F con una certezza c tale che S sia più semplice in qualche modo della enumerazione dei fatti contenuti in G .
- **Interpretazione e valutazione:** i modelli identificati dal sistema vengono interpretati cosicché la conoscenza che se ne acquisisce può essere di supporto alle decisioni, quali ad esempio la previsione e classificazione dei compiti, il riassunto dei contenuti di un database o la spiegazione dei fenomeni osservati.

Il modo migliore per parlare del data mining è quindi parlare direttamente su ciò che fa; procediamo quindi nell'analisi di alcuni brevi esempi...

Iniziamo con un'operazione di Clustering, che può coincidere ad esempio con la collocazione in categorie individuate di una grande massa di clienti, apparentemente omogenea, che li rende indistinguibili tra loro.

In questo tipo di operazioni, solitamente è sufficiente fornire al sistema il numero di categorie nei quali si desidera avere suddivisi i propri clienti e lui si preoccuperà di determinare le categorie più opportune e collocare i diversi clienti in queste, secondo i dati legati a ciascun record.

Già questo semplice esempio, costituisce un caso di "undirected data mining", dove l'utente non ha un archivio ordinato preesistente e spera che sia il tool per il Data Mining a rivelare qualche struttura utile. Per avere buone probabilità di successo, in questo contesto, i record forniti in input a questa procedura di cluster, dovrebbero essere idealmente altamente descrittivi, preferibilmente contenendo informazioni sia demografiche che comportamentali per ogni cliente. Gli algoritmi per il Clustering lavorano bene con tutti i tipi di dati, inclusi quelli numerici e testuali e possono essere usati in questi contesti elementi di statistica, ragionamento mnemonico, reti neurali e alberi di decisione. [LII]

Un esempio di classificazione si ha quando il sistema esamina un dato cliente e provvede ad inserirlo nella categoria di appartenenza (o cluster) più adatta, in base alle caratteristiche che questo presenta; un altro esempio di questo tipo può essere fornito, in ambito medico, dalle diagnosi.

In entrambi i casi, una descrizione del cliente o del paziente sono forniti all'algoritmo che si occupa della classificazione. Il classificatore determina il cluster al quale cliente (o paziente) si avvicina di più, in base alle caratteristiche che presenta.

Vedendo le cose in questo modo, si può intuire come l'attività di clustering condotta in precedenza possa essere un primo passo naturale, verso l'attività di classificazione, molto utile in molti ambienti di data warehouse.

Una classificazione è in fondo già di per se una decisione e già con questi banali esempi possiamo ricavare risultati importanti: possiamo decidere se un dato cliente è tra quelli che porta profitto all'azienda o se un paziente si rivela adatto per essere sottoposto ad un certo trattamento.

Le tecniche che possono essere usate per la classificazione includono anche qui i metodi statistici classici, il ragionamento mnemonico, le reti neurali e gli alberi decisionali, ma anche gli algoritmi genetici e le link analysis.

La stima e la predizione sono due attività simili che generalmente comportano misure numeriche come risultato; per esempio possiamo desiderare conoscere, su un set di clienti, individuato come “medi”, il livello di indebitamento medio corrente (stima) o futuro (predizione).

Anche tali tecniche possono portare alla classificazione; si pensi ad esempio al caso in cui volessimo stabilire che tutto i clienti con piu' di 100,000 euro di debiti debbano essere considerati come appartenenti ad una categoria “a rischio di credito”. Le stime numeriche offerte da questo tipo

di applicazioni hanno il vantaggio addizionale che i candidati possono venire ordinati; per esempio, potremmo avere un budget limitato per una campagna pubblicitaria e desiderare di offrire un'offerta promozionale solo ai primi 10,000 clienti, ordinati secondo i previsti profitti che potrebbero portare all'azienda.

Le tecniche utilizzate in questo caso possono anche qui variare dalla statistica, alle reti neurali per l'analisi numerica, alle tecniche di classificazione viste in precedenza, nei casi di predizione di un risultato discreto.

Il raggruppamento per affinità è un tipo speciale di partizionamento (clustering) e mira ad identificare gli eventi che avvengono simultaneamente. Un esempio di questo tipo si può avere ad esempio quando si vuol capire quali prodotti siano venduti assieme, nello stesso tempo.

Dal punto di vista del data processing, questo si presenta come un problema, poiché in un ambiente di vendita ci sono migliaia di prodotti diversi.

Non ha senso numerare tutte le combinazioni degli oggetti venduti assieme, a due a due; tale lista potrebbe raggiungere presto dimensioni astronomiche; la cosa utile è invece trovare combinazioni significative, come ad esempio scoprire che la vendita di una lattina di cola, avviene spesso in concomitanza con l'acquisto di "Frozen Pasta Dinners."

Tecniche specifiche per il raggruppamento secondo le affinità includono statistiche standard, ragionamento su base mnemonica, link analysis, e strumenti di analisi specifica del mercato.
[L11][L12][L13]

5 Le Interfacce Utente

Anche se ovviamente il tipo di applicazione da eseguire sui dati determina in modo imprescindibile il tipo di interfaccia utente più adatta per le applicazioni sul data warehouse, si può notare come ultimamente sempre più sistemi supportino l'analisi e la presentazione dei dati su interfacce web.

A seguito di tale tendenza, sono anche progrediti gli studi a riguardo, volti a fornire alcune linee guida nella realizzazione di applicativi efficaci per l'analisi dei dati.

Come sempre, quando ci troviamo di fronte ad una finestra sul world wide web, una prima condizione imposta è quella della velocità di esecuzione; davanti all'attesa di caricamento di una pagina, dopo alcuni secondi, l'utente è generalmente portato a credere che qualcosa sia andato storto nella sua connessione ed è fortemente tentato ad azionare il reload della pagina, andando incontro così a tempi di risposta allungati ulteriormente dalla nuova richiesta, che ha annullato quella precedente.

Tutte le operazioni di caricamento dovrebbero pertanto mantenersi sotto i 10 secondi e, qualora vi fossero eccezioni o comunque si prospettasse la possibilità di avere tempi di attesa superiori ai 5 secondi, sarebbe opportuno che l'utente fosse avvisato di quanto sta accadendo, così da tamponare l'irrefrenabile tentazione di agire con un "reload" della pagina richiesta.

Un'altro consiglio relativo alla realizzazione delle interfacce web riguarda la presentazione dei contenuti e l'organizzazione degli stessi. È ovviamente opportuno presentare sempre tutte le azioni possibili per l'utente, magari aumentando la chiarezza delle stesse, suddividendole in categorie e aggiungendo sempre funzionalità di aiuto.

È invece vivamente sconsigliato l'utilizzo di finestre pop-up, banners e tutti quegli espedienti intrusivi che stanno prendendo piede di questi tempi, a supporto di aggressive campagne pubblicitarie condotte online.

Infatti, a causa delle tendenze sempre più invasive che queste presentano, vengono inevitabilmente percepite dagli utenti come semplici fastidi e quindi molto spesso ignorate, anche se magari contenenti informazioni importanti; il designer deve quindi giudicare ad ogni evenienza se l'utilizzo di tali fonti di distrazioni è giustificato dalla particolare situazione in cui si trova (ad esempio, in

presenza di un tempo di caricamento lungo, un popup che avvisa di quanto sta accadendo potrebbe essere ammissibile).

Sempre per ridurre fattori di distrazione. in caso di presentazioni dei contenuti personalizzate o salvataggio delle preferenze, è importante che gli utenti non debbano eseguire autenticazioni più di una volta per sessione; sarebbe anzi opportuno anche tenere traccia delle diverse sessioni aperte ed eventualmente riprendere quelle concluse prematuramente, ad esempio a causa di errori di rete.

In ultima istanza e ad un livello strettamente tecnico, le interfacce devono fare utilizzo di tecnologie quanto più possibile standard, così da incontrare la compatibilità anche con i sistemi più vecchi; una o due esperienze con un sito che richiedono di scaricare un particolare plugin o che causano errori nell'esecuzione di Javascript, sono spesso sufficienti per far sì che l'utente collochi l'applicazione nella sua lista dei luoghi "da evitare". [A8]

BIBLIOGRAFIA

Libri

- [L1] Building the Data Warehouse by W. Inmon (John Wiley and Sons, 1993)
- [L2] Data Warehouse from Architecture to Implementation by B. Devlin (Addison Wesley Longman, 1996)
- [L3] The Data Warehouse Lifecycle Toolkit by R. Kimball (John Wiley and Sons, 1998)
- [L4] The Data Warehouse Toolkit (second edition) by R. Kimball (John Wiley and Sons, 2002)
- [L5] Database Systems (third edition) by T. Connolly (Pearson Addison Wesley, 2001)
- [L6] Fundamentals of Database Systems (third edition) by R. Elmasri & S. Navathe (Pearson Addison Wesley, 1999)
- [L7] Oracle9i Data Warehousing Guide by P. Lane (Oracle Corporation, 2002)
- [L8] Architettura dei computer, un approccio strutturato by A. Tanenbaum.. (UTET, 2000)
- [L9] Using the Data Warehouse by W. Inmon (John Wiley and Sons, 1994)
- [L10] I Sistemi di Supporto Alle Decisioni: Offerta, Domanda, Applicazioni by A. De Toni, G. Nassimbeni, S. Tonchia (Francoangeli, 2000)
- [L11] Mastering Data Mining by J. Berry & S. Linoff (John Wiley and Sons, 1999)
- [L12] Data Mining by P. Adriaans & D. Zantinge (Addison Wesley, 1996)
- [L13] The First International Conference on Knowledge Discovery & Data Mining by Usama Fayyad, Ramasamy, Uthurusamy (Amer Assn for Artificial, August 1995)

Articoli

- [A1] Modeling the Data Warehouse and Data Mart by Paul Winsberg (InfoDB Vol. 10, Number 3)
- [A2] Why is the Snowflake Schema a Good Data Warehouse Design? by M. Levene & G. Loizou (Information Systems, Volume 28 , Issue 3, 2003)
- [A3] What Not to Do: Dimensional modeling mistakes to guard against by Ralph Kimball (Intelligent Enterprise Magazine, October 24, 2001)
- [A4] Fact Tables and Dimension Tables by R. Kimball (Ralph Kimball Associates: Articles)
- [A5] Keep Control over Record Identifiers by generating new keys for the data warehouse by R. Kimball (Ralph Kimball Associates: Articles)
- [A6] Factless Fact Tables by R. Kimball (DBMS, September 1996)
- [A7] *Slowly Changing Dimensions* by R. Kimball (DBMS, April 1996)
- [A8] Designing the User Interface by R. Kimball (Intelligent Enterprise Systems, September 14, 1999, Volume 2 - Number 13)
- [A9] The New DSS: Data Warehouses, OLAP, MDD and KDD by P. Gray & H. Watson (AIS Americas Conference, 1996)
- [A10] Drilling Down, Up, and Across by R. Kimball (DBMS - March 1996)

Risorse World Wide Web [al 05/2004]

- [W1] Ralph Kimball Associates: Articles - <http://www.rkimball.com/html/articles.html>
- [W2] Oracle Daily Feature, Aug 5 2002, Bitmap Indexes - <http://otn.oracle.com/products/oracle9i/daily/aug05.html>
- [W3] Americas Conference on Information Systems – <http://hsb.baylor.edu/ramsower/acis/>
- [W4] The Olap Report – <http://www.olapreport.com>
- [W5] Gartner Group Inc - <http://www3.gartner.com/Init>
- [W6] Meta Group - <http://www.metagroup.com/us/home.do>

Glossario

Ad Hoc Query – Un'interrogazione che non può essere determinata, prima del momento in cui è effettivamente lanciata.

Aggregazione – Un tipo di derivazione, dove il valore dei dati proviene dai dati di uno stesso soggetto.

Data Warehouse Centralizzato – Un Data Warehouse dove una singola base di dati si trova al servizio di diverse unità aziendali, con un modello di dati unico, che copre tutte le esigenze dei diversi dipartimento.

Data Cleansing -- The process of removing errors and resolving inconsistencies in source data before loading the data into a target environment.

Data Mart -- A type of data warehouse designed to meet the needs of a specific group of users such as a single department or part of an organization. Typically a data mart focuses on a single subject area such as sales data. Data marts may or may not be designed to fit into a broader enterprise data warehouse design.

Data Mining -- A process of analyzing large amounts of data to identify hidden relationships, patterns, and relationships. This is often called "discovery-driven" data analysis.

Modello -- A logical map that represents the inherent properties of the data independent of software, hardware or machine performance considerations. The model shows data elements grouped into records, as well as the associations between those records.

Data Warehouse -- A subject oriented, integrated, time-variant, non-volatile collection of data in support of management's decision making process. A repository of consistent historical data that can be easily accessed and manipulated for decision support.

Database -- A collection of data which are logically related.

Decision Support System (DSS) -- Systems which allow decision makers in organizations to access data relevant to the decisions they are required to make.

Drill Down -- A method of exploring detailed data that was used in creating a summary level of data. Drill down levels depend on the granularity of the data in the data warehouse.

Executive Information System (EIS) -- Tools programmed to provide canned reports or briefing books to top-level executives. They offer strong reporting and drill-down capabilities. Today these tools allow ad-hoc querying against a multi-dimensional view of data, and most offer analytical applications along functional lines such as sales or financial analysis.

Estrazione -- The process of copying a subset of data from a source to a target environment.

Middleware -- A communications layer that allows applications to interact across hardware and network environments.

On-Line Analytical Processing (OLAP) -- Processing that supports the analysis of business trends and projections. It is also known as Multi-Dimensional Analysis.

OLTP (Operational Applications) -- Applications which support the daily operations of the enterprise. Usually included in this class of applications are Order Entry, Accounts Payable, Accounts Receivable, etc.

Query -- A request for information from the Data Warehouse posed by the user or tool operated by the user.

Relational Database Management System (RDBMS) -- A database system built around the relational model based on tables, columns and views.

Star Schema -- A modeling scheme that has a single object in the middle connected to a number of objects around it radially - hence the name star. A fact such as sales, compensation, payment, or invoices is qualified by one or more dimensions such as by month, by product, by geographical region. The fact is represented by a fact table and the dimensions are represented by dimension tables.