D2I

Integrazione, Warehousing e Mining di sorgenti eterogenee

Programma di ricerca (cofinanziato dal MURST, esercizio 2000)

Analisi dello stato dell'arte sulle architetture di data warehouse

G. Greco, S. Greco, P. Rullo

D2.R1 **30 marzo 2001**

Sommario

La realtà economica e sociale è caratterizzata da un'elevata competitività in tutti i suoi settori. La caduta delle barriere economiche e l'evoluzione dei sistemi di automazione hanno favorito l'integrazione e la qualità dei servizi. D'altro canto, le esigenze e le abitudini del mercato evolvono in modo continuo e rapido; questo fattore induce le aziende a riprogettare i processi aziendali per minimizzare i tempi ed i costi produttivi dei servizi offerti. I sistemi informativi tradizionali non sono in grado di rispondere in maniera efficace alle nuove esigenze che si sono venute determinando. Infatti questi sistemi sono stati concepiti in maniera disorganizzata e senza alcuna capacità di integrazione; da ciò deriva un'evidente diseconomicità nella gestione delle informazioni dovuta a varie ragioni, tra cui l' incapacità di sfruttare le potenzialità informative dei sistemi disponibili e la necessità di verificare e mantenere vincoli tra dati memorizzati su sorgenti informative differenti.

Tutto ciò ha reso indispensabile la progettazione di nuovi strumenti capaci di elaborare informazioni provenienti da più basi di dati. Le due principali risposte a tale esigenza sono state le Federazioni di basi di dati e i Data Warehouse (DW) di cui ci occuperemo in questo lavoro.

Tema	Tema 2: Progettazione e interrogazione di Data Warehouse
Codice	D2.R1
Data	30 marzo 2001
Tipo di prodotto	Rapporto tecnico
Numero di pagine	21
Unità responsabile	CS
Unità coinvolte	CS, BO
Autore da contattare	Sergio Greco Dipartimento di Elettronica, Informatica e Sistemistica Università della Calabria 87030 Rende, Italia greco@deis.unical.it

Analisi dello stato dell'arte sulle architetture di data warehouse

G. Greco, S. Greco, P. Rullo

30 marzo 2001

Abstract

La realtà economica e sociale è caratterizzata da un'elevata competitività in tutti i suoi settori. La caduta delle barriere economiche e l'evoluzione dei sistemi di automazione hanno favorito l'integrazione e la qualità dei servizi. D'altro canto, le esigenze e le abitudini del mercato evolvono in modo continuo e rapido; questo fattore induce le aziende a riprogettare i processi aziendali per minimizzare i tempi ed i costi produttivi dei servizi offerti. I sistemi informativi tradizionali non sono in grado di rispondere in maniera efficace alle nuove esigenze che si sono venute determinando. Infatti questi sistemi sono stati concepiti in maniera disorganizzata e senza alcuna capacità di integrazione; da ciò deriva un'evidente diseconomicità nella gestione delle informazioni dovuta a varie ragioni, tra cui l' incapacità di sfruttare le potenzialità informative dei sistemi disponibili e la necessità di verificare e mantenere vincoli tra dati memorizzati su sorgenti informative differenti.

Tutto ciò ha reso indispensabile la progettazione di nuovi strumenti capaci di elaborare informazioni provenienti da più basi di dati. Le due principali risposte a tale esigenza sono state le Federazioni di basi di dati e i Data Warehouse (DW) di cui ci occuperemo in questo lavoro.

1 Introduction

Un DW è una collezione integrata e persistente di dati aziendali, orientata al supporto alle decisioni, che viene costruita per favorire le attività di analisi. In un DW convergono, da un insieme di sorgenti informative esterne (basi di dati operazionali), tutte le informazioni utili ai processi decisionali aziendali. Esso è mantenuto fisicamente separato dalle altre sorgenti informative, per diversi motivi. Innanzitutto, i sistemi operazionali sono ottimizzati per processi transazionali che richiedono lettura e/o modifica di pochi dati alla volta con tempi di risposta strettissimi, mentre un DW deve favorire attività analitiche di supporto ai processi decisionali. Tali attività richiedono l'accesso a quantità di dati anche molto grandi per produrre risultati sintetici in tempi relativamente ristretti. Esse vengono effettuate utilizzando applicazioni con caratteristiche opportune, che vanno sotto il nome di OLAP (On-Line Analytical Processing).

A differenza di un sistema di basi di dati federate, un DW richiede la materializzazione dei dati precedentemente memorizzati nelle basi di dati operazionali; inoltre, i dati sono immagazzinati in serie storiche. L'approccio generale per la costruzione di un DW prevede:

- estrazione, traduzione ed opportuno filtraggio delle informazioni di ciascuna sorgente di interesse;
- fusione di informazioni correlate provenienti da sorgenti diverse;
- memorizzazione, in un deposito opportuno, di tali informazioni.

Nel suo complesso, una architettura di Data Warehouse (DWA) consiste di una serie di elementi funzionali tra di loro interconnessi utili nel rappresentare la struttura complessiva

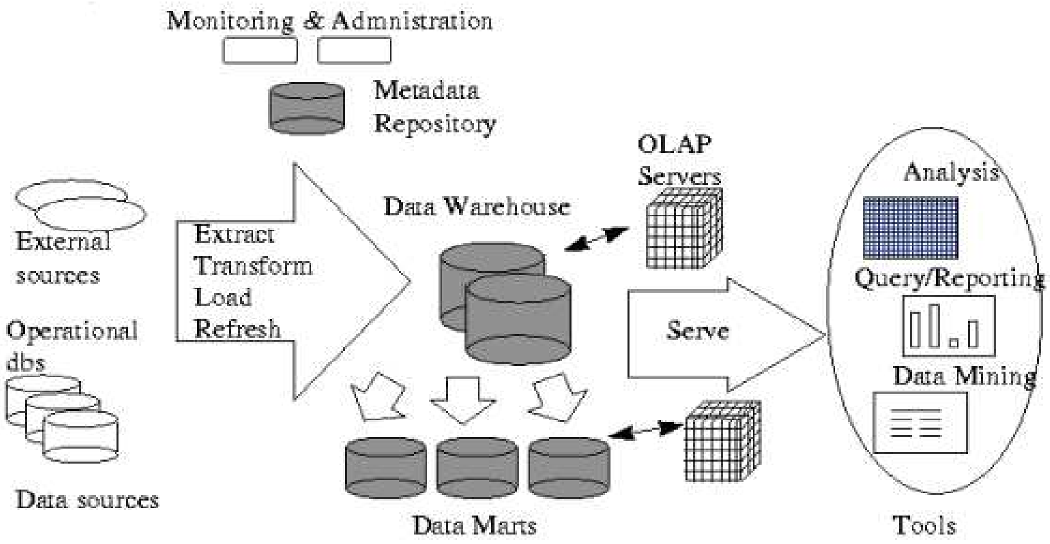

Figure 1: Architettura di Data Warehouse

dei dati, nello stabilire metodi di processamento degli stessi nonché nel definire la forma di presentazione da offrire ad un utente finale.

Nella Figura 1, possiamo vedere uno schema tipico che include strumenti per l'estrazione di dati da svariati database operazionali e da sorgenti esterne, strumenti per la pulizia, la trasformazione e l'integrazione di questi dati; per il carico nel warehouse e per il periodico refresh dovuto alla necessità di riflettere eventuali cambiamenti occorsi nelle sorgenti. In aggiunta al principale warehouse, possono coesistere diversi data marts specifici per un determinato settore applicativo-decisionale. I dati nel warehouse e nei data marts sono memorizzati e gestiti da uno o più warehouse servers, che presentano viste multidimensionali dei dati oltre che una varietà di front end tools. In fine, esiste un repository per memorizzare e gestire i metadati, e altri strumenti per il monitoraggio e l'amministrazione del sistema complessivo. In termini logici si può dire che risulta possibile individuare tre gruppi di componenti principali:

- Back End: sistemi di gestione dei processi di estrazione, filtraggio, integrazione e trasformazione dei dati relativi agli ambienti operativi;
- Sistema di gestione dei dati (DBMS): contiene i dati del DW;
- Front End: comprende tutte le funzioni che consentono agli utenti l'accesso ai dati.

Passando all'implementazione fisica, il warehouse può essere centralizzato oppure distribuito in modo da bilanciare il carico e per favorire la scalabilità; in una tale architettura, tipicamente il metadata repository è replicato in ogni frammento del warehouse e il sistema viene amministrato centralmente. Una alternativa, implementata come espediente quando è troppo costoso costruire una singola struttura logicamente integrata è la realizzazione di una federazione di warehouses o data marts, aventi ognuno il proprio repository e una amministrazione decentrata. In questo lavoro saranno presentate le classiche architetture concettuali di data warehouse in ordine di complessità crescente [6] insieme ad alcune varianti come le architetture federate e schemi di architetture fisiche parallele.

2 Architettura logica Architettura ad un Livello

L'idea di base di questa architettura è che ogni dato sia memorizzato una ed una sola volta, non facendo distinzione alcuna tra le varie tipologie di dato; la conseguenza è che anche eventuali dati

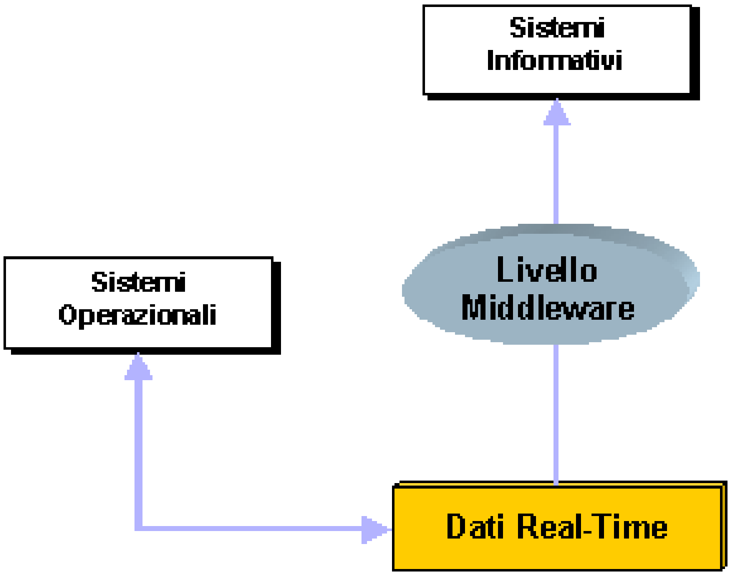

Figure 2: Architettura ad un livello

derivati risultano trattati alla stessa stregua dei dati real-time da cui derivano. L'architettura ad un livello consente la minimizzazione del volume dei dati memorizzati, ed evita il problema della sincronizzazione dei dati ripetuti. Gli svantaggi di tale approccio sono però altrettanto evidenti. Il principale è la contesa che si attua sul dato tra sistemi operazionali e sistemi decisionali, degradando le prestazioni di entrambi. Un altro punto debole è che l'approccio ad un livello non fornisce un aiuto nel definire come i dati distribuiti possano essere implementati ne come utenti distribuiti su diverse aree geografiche possano accedere alla singola copia esistente dei dati. E' altresì chiaro che il supporto dato ad applicazioni decisionali da questa architettura è veramente minimo. Le organizzazioni che possono trarre vantaggio da una architettura ad un livello sono quelle dove il volume dei dati è particolarmente ampio e dove c'è una esigenza limitata di analisi su dati aggregati. Queste caratteristiche sono spesso riflesse nell'approccio che la compagnia ha adottato nella memorizzazione dei dati, che in generale è centralizzata e mainframe oriented. L'approccio ad un livello porta inevitabilmente alla creazione di un Data Warehouse Virtuale, che evita la creazione di un data base di riconciliazione e di databases di dati derivati (Architetture a due e a tre livelli). Tale approccio ha avuto una risonanza notevole in quanto promette tempi di sviluppo rapidissimi, evitando la replicazione dei dati. L'accesso ai databases operazionali avviene attraverso l'uso di un "middleware" che ha lo scopo di presentare i dati all'utente come fossero strutturati in funzione di un approccio di Data Warehouse [22]. Il mito del VDW sta però rapidamente scomparendo, in quanto studi più approfonditi [1, 7] hanno dimostrato che molte delle attività di pianificazione più pesanti (identificazione dei data sources, trasformazione dei dati, etc.) non sono evitabili in un approccio virtuale, che quindi non risulta essere così rapido come si pensava. Inoltre mentre è possibile almeno in parte ottenere con questa architettura le caratteristiche di tematicità ed integrazione, non è possibile ottenere storicizzazione, salvo che quella già espressa dagli ambienti operativi e permanenza dei dati, in quanto i dati operativi all'occorrenza vengono modificati. Inoltre il tempo di accesso per l'utente finale non può essere sotto controllo, in quanto questo dipende di volta in volta dal tipo di query richiesta, dal tempo richiesto per riconciliare i diversi formati dei databases, dal carico della rete e così via.

3 Architettura a Due Livelli

L'architettura a due livelli permette di separare le esigenze dei sistemi operazionali e dei sistemi decisionali attraverso la separazione tra dati real-time e dati derivati, come mostrato in Figura 3. Il livello inferiore (real-time data) è usato dalle applicazioni operazionali sia in lettura che in scrittura, mentre il livello superiore è usato esclusivamente da applicazioni decisionali in lettura. Va notato che i dati derivati possono essere una semplice copia dei dati operativi o possono essere calcolati da questi attraverso procedure di estrazione.

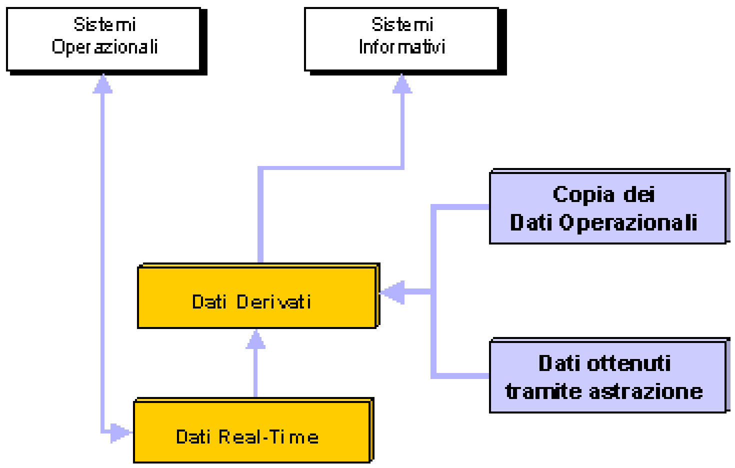

Figure 3: Architettura a due livelli

L'approccio a due livelli risolve immediatamente il problema della concorrenza tra applicazioni operative e decisionali. Un ulteriore beneficio deriva dalla possibilità di fornire a differenti utenti differenti derivazioni degli stessi dati operativi. Uno dei problemi che questo tipo di architettura introduce è l'alto livello di duplicazione dei dati, in molti casi incontrollato. La principale ragione di questo problema può essere vista facilmente osservando la Figura 4 per enfatizzare che ognuno dei due livelli è in realtà composto da molteplici insiemi di dati sovrapposti. Inoltre, non vi è una relazione uno a uno tra gli insiemi presenti nei livelli real-time

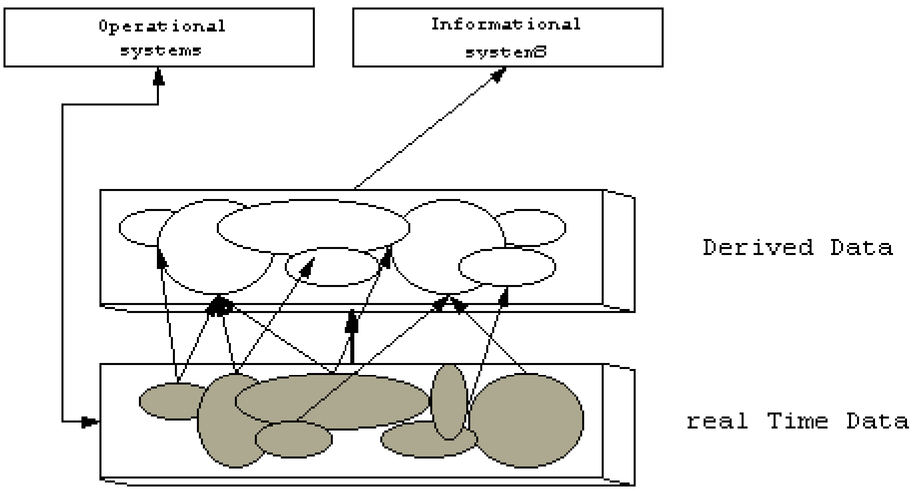

Figure 4: Duplicazione dei dati in architettura a due livelli

data e gli insiemi presenti nel livello dei dati derivati. Anche se questi problemi sono rilevanti, questa architettura è allo stato attuale delle cose la più usata e un notevole sforzo è stato fatto per sistematizzare l'implementazione basata su tale approccio [17, ?]. L'ambiente in cui tale approccio ha ragione di essere adottato con successo, è in tutti quei casi in cui vi è omogeneità tra gli ambienti operativi e dove vi è un numero limitato di piattaforme hardware e software. Questo perché sotto questi requisiti il processo di derivazione risulta più facile da controllare.

3.1 Data Marts

Quando una classica architettura a due livelli si rivolge ad uno specifico problema, soddisfacendo specifiche esigenze di un'area limitata ed omogenea di utenti si suole parlare di architettura di Data Marts. La sola differenza tra un datawarehouse ed un data mart è difatti l'ambito applicativo. In alcuni casi, i data marts sono costruiti in una via isolata, e successivamente integrati in un warehouse corporativo più grande, ma possono anche essere implementati con un approccio duale.

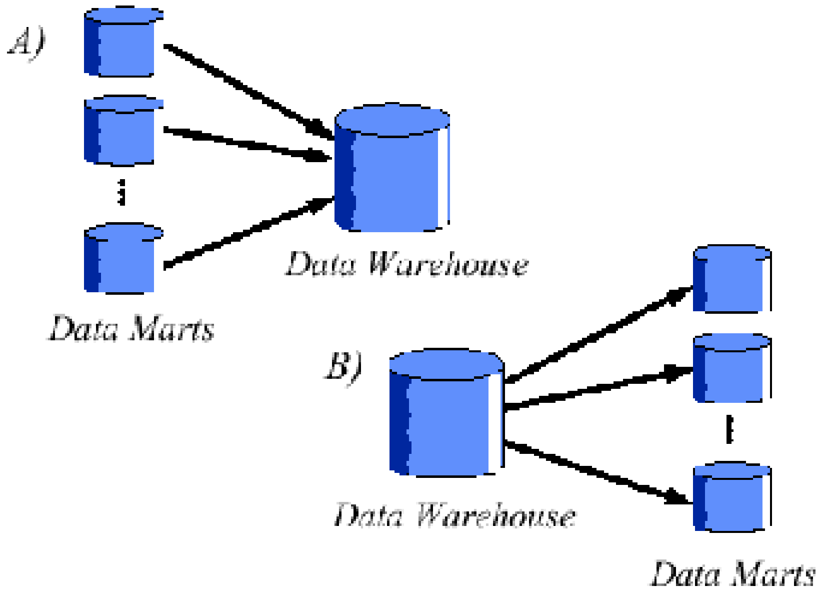

Figure 5: DW e Data Marts

Essendo un Data Mart fortemente orientato alle esigenze informative da soddisfare, vengono meno alcuni vincoli fondamentali di un Corporate Data Warehouse come mantenere i dati ad un forte livello di dettaglio per poter soddisfare richieste non previste e garantire che il significato di ogni dato sia univoco e valido per l'intera organizzazione. Molti miti si sono formati attorno all'idea di Data Mart, alcuni di essi vanno decisamente sfatati [31, 7]:

1. I Data Marts sono di piccole dimensioni. Il basarsi sulle dimensioni per determinare se si è implementato un Data Warehouse o un Data Mart è sciocco. Una definizione molto in voga è che un database con meno di 50 Gbytes è un Data Mart mentre un database con più di 50 Gbytes è un Data Warehouse. Un Data Mart come già detto è pensato per una specifica richiesta di un gruppo di utenti omogeneo, e in funzione di ciò non ha vincoli dimensionali ne deve necessariamente aggregare i dati su cui opera. Non a caso molti Data Marts implementati contengono centinaia di Gbytes di dati di dettaglio.

2. I Data Marts sono meno complessi di un Data Warehouse e più veloci da sviluppare. Anche se un Data Mart è intrinsecamente meno complesso di un Data Warehouse in quanto si focalizza su uno specifico aspetto del business, non va dimenticato che i problemi riguardanti l'acquisizione e l'integrazione dei dati da molteplici data sources rimangono quasi totalmente

invariati.

3. I Data Marts sono scalabili in un Data Warehouse. Spesso la proliferazione incontrollata di Data Marts indipendenti porta ad una possibile incoerenza tra i dati (diversi Data Marts non trattano sempre informazioni disgiunte) con la conseguente impossibilità di essere scalati a formare un Data Warehouse, senza ridisegnarne il modello dati.

4. I Data Marts sono più flessibili rispetto ai Data Warehouse. Questa credenza è del tutto priva di fondamento anche se molto diffusa. Un Data Mart infatti proprio per la sua specificità non è in genere in grado di soddisfare richieste non previste o di supportare con modifiche minime l'aggiunta di ulteriori data sources.

4 Architettura a Tre Livelli

L'architettura a tre livelli nasce dall'osservare che la trasformazione dei dati real-time in dati derivati non richiede un solo passo come avviene nell'architettura a due livelli ma bensì due passi che sono la rinconciliazione dei dati provenienti dai diversi data sources e la derivazione dei dati richiesti dall'utente non dai dati real-time, ma dai dati riconciliati. Questo porta all'architettura di Figura 6.

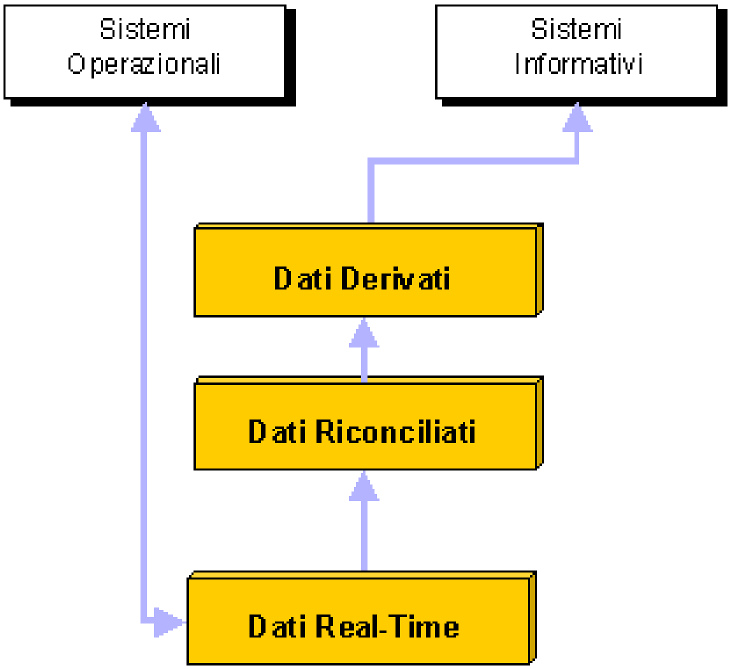

Figure 6: Architettura a tre livelli

4.1 Architettura Logica

All'architettura logica descritta corrisponde l'architettura concettuale a tre livelli mostrata in Figura 7. Si osserva una architettura caratterizzata dai seguenti livelli di astrazione dei dati:

- Dati Operazionali (o real-time): rappresentano lo stato corrente del business. Sono i dati al livello più basso di dettaglio e sono creati, manipolati e acceduti dai sistemi operazionali (sorgenti esterne).
- Dati Riconciliati: questi dati provengono da processi di razionalizzazione e integrazione dei dati operazionali provenienti dalle sorgenti esterne.

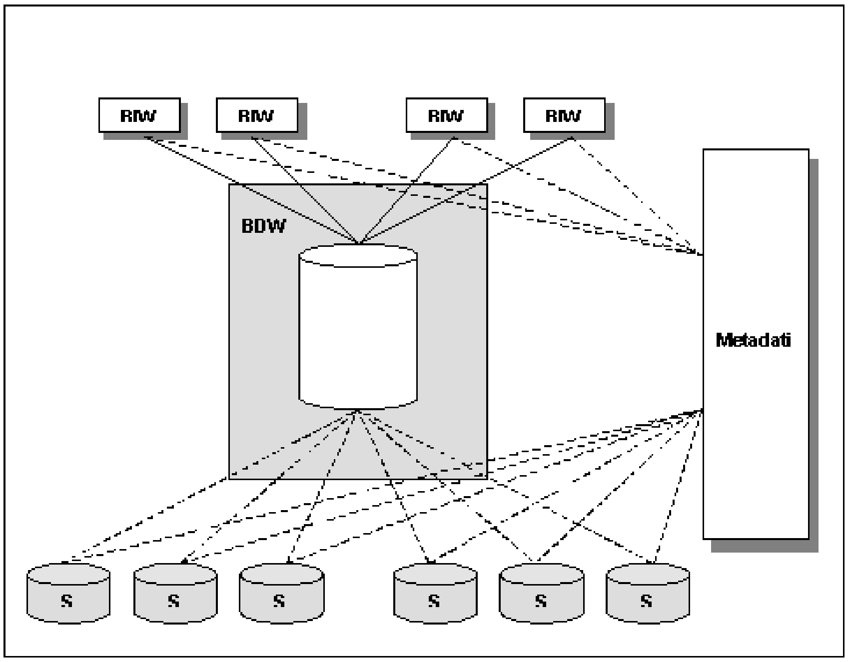

Figure 7: Architettura logica

- Dati Derivati: sono dati che nascono da processi di trasformazione dei dati riconciliati. Possono esistere sia ad un elevato livello di dettaglio sia in forma aggregata. Questi dati sono acceduti esclusivamente in lettura da applicazioni per il supporto alle decisioni.

Un sistema di DW a tre livelli ha pertanto quattro componenti fondamentali:

- I Sistemi operazionali, che gestiscono i dati operazionali con gli schemi e le applicazioni corrispondenti;

- Il BDW (Business Data Warehouse), che gestisce i dati riconciliati con i rispettivi schemi e le rispettive applicazioni;

- Alcuni BIW (Business Information Warehouse) o Data Mart che gestiscono i dati derivati con gli schemi e le applicazioni corrispondenti. I sistemi e le applicazioni che, tipicamente, si collocano in questo contesto sono: Sistemi di Reportistica; Sistemi di Analisi; Sistemi DSS (Decisional Support Systems); Applicazioni di Data Mining; Applicazioni OLAP(On-Line Analytical Processing);

  Questo ambiente, a differenza del BDW, è fortemente distribuito su macchine client. Da un unico BDW centralizzato possono essere popolati molteplici BIWs distribuiti, ognuno soddisfacente le particolari esigenze di un singolo utente o di gruppi di utenti (es. dipartimenti o divisioni).

- Il Gestore dei Metadati che gestiscono le informazioni relative alle sorgenti, agli schemi ed alle interazioni tra i tre livelli del DW.

Con il termine data warehouse si intenderà nel seguito l'unione del BDW e dei BIWs, [12, **?**] usano sostanzialmente la stessa definizione di data warehouse, mentre in [14] tale termine va a coprire tutti i dati che hanno importanza per una impresa, inclusi i dati operazionali, visti però in un'ottica informativa e non transazionale.

Nelle sezioni seguenti analizzeremo in dettaglio ciascuna delle varie componenti ora elencate.

4.2 Metadati

I metadati sono una descrizione dei dati presenti nel DW e delle operazioni effettuate su di essi. Questi dati possono essere classificati in Metadati build-time, di controllo e usage. I Metadati Build-time sono creati e usati durante la progettazione e la costruzione dei DB e dei processi operazionali o informativi; essi forniscono una descrizione dei dati di business e delle funzioni che operano su di essi. I Metadati di Controllo sono creati e mantenuti durante il ciclo di vita dei sistemi operazionali e del DW. Si possono suddividere ulteriormente in Metadati Currency e Metadati di Accesso. I primi forniscono informazioni di tipo temporale (ad es. l'ultima volta che è stata modificata una tabella) e sono gestiti dalle applicazioni che aggiornano fisicamente i dati di business. I secondi vengono impiegati per gestire le operazioni di accesso sicuro ai dati e per indagare sulla frequenza di utilizzazione dei dati stessi. Quest'informazione permette di conoscere il grado di utilità di un dato e consente di stabilire se è bene mantenerlo nel DW. I Metadati Usage traggono origine dai Metadati build-time ai quali sono molto simili come contenuto ma, differentemente da questi, hanno una struttura maggiormente orientata all'accesso ed alla facilità di comprensione da parte dell'utente finale. Contengono informazioni che possono aiutare l'utente a sapere cosa c'è nel DW, cosa indica, dov'è fisicamente collocato, come accedervi e come usarlo. Questi dati assumono rilevanza solo quando l'esplosione del volume dei dati crea seri problemi di orientamento. Come per i dati di business anche per i metadati bisogna pensare ad un processo di storicizzazione. La velocità di variazione di questi dipende dal gruppo di appartenenza, infatti le variazioni relative ai metadati build-time sono lente e ben pianificate, al contrario, quelle relative ai dati derivati sono veloci in quanto devono seguire le variazioni dei BIW.

4.3 Business Information Warehouses

Con il termine business information warehouses (BIWs) si intendono sistemi di reportistica, sistemi di analisi, sistemi DSS, applicazioni di data mining etc. Questo ambiente a differenza del BDW, è fortemente distribuito tipicamente su sistemi client-server o su workstations. Da un unico BDW centralizzato vengono allora popolati molteplici BIWs distribuiti, ognuno soddisfacente le particolari esigenze di un singolo utente o di gruppi di utenti (es. dipartimenti o divisioni). Va notato che è inoltre possibile popolare alcuni BIWs da altri BIWs. I sistemi autorizzati a tale scopo vengono detti staging BIWs. Risulta chiaro dalle definizioni data in precedenza che un sistema BIW deve essere ottimizzato in termini di tempo di risposta alle queries.

4.4 Il Livello dei dati riconciliati

La presenza di un livello di riconciliazione crea un modello di dati comune per l'intera organizzazione. L'attività di riconciliazione elimina le inconsistenze e le irregolarità dai dati operazionali, integrando questi in modo razionale. Una volta costruiti i dati riconciliati, quelli derivati vengono ricavati da questi piuttosto che dai dati operazionali. Apparentemente, il principale difetto dell'architettura a tre livelli consiste, ancora una volta, nella duplicazione dei dati nel livello riconciliato. Tuttavia l'introduzione di un terzo livello non incrementa in maniera significativa la richiesta di risorse per la memorizzazione; infatti varie stime suggeriscono che:

- nei dati operazionali ciascuna informazione è replicata diverse volte;
- qualora non esista un deposito comune per i dati utilizzati da più sistemi di supporto alle decisioni, nel livello dei dati derivati le informazioni comuni devono essere comunque replicate per ciascuna applicazione che le utilizzi.

Si sottolinea che la presenza di un livello di dati riconciliati comporta i seguenti vantaggi:

- in un'architettura a tre livelli il costo necessario per ottenere i dati derivati è ridotto rispetto alle altre due architetture. Infatti, le operazioni di riconciliazione vengono eseguite una sola volta e non devono essere ripetute per ciascun insieme di dati derivati;

- i dati riconciliati sono particolarmente adatti per la reingegnerizzazione delle applicazioni operazionali; spesso alcune informazioni, come i dati storici, sono state memorizzate nei dati operazionali soltanto perch non esisteva nessun'altra possibilità; la presenza di un livello di dati riconciliati consente di trasferire questi dati dal livello operazionale a quello riconciliato; in tal modo si ottiene una semplificazione dei sistemi operazionali;

- può accadere che alcuni dati derivati non possano essere estratti in maniera semplice dai dati operazionali (per esempio perch l'estrazione è costosa o perch il processo di estrazione richiede la sospensione di particolari processi transazionali). In un'architettura a due livelli le informazioni difficili da ottenere devono essere memorizzate per ciascun insieme di dati derivati che necessitano di queste, dal momento che non vi è nessun'altra struttura in cui inserirle; ne consegue un aumento significativo del volume dei dati derivati. In un'architettura a tre livelli tali informazioni vengono memorizzate una sola volta nel livello riconciliato.

4.5 On-Line Analytical Processing (OLAP)

Con il termine OLAP si identifica un insieme di tecniche di analisi di dati, generalmente provenienti da database o data warehouse, progettate per supportare il processo decisionale. Per stabile se una applicazione possa essere definita OLAP, si può dire che essa deve innanzitutto consentire una visione multidimensionale dei dati, e deve inoltre rendere semplice ed efficiente l'analisi dei dati. Un sistema OLAP può essere realizzato utilizzando o un DBMS relazionale (Relational OLAP, ROLAP) o un server multidimensionale (Multidimensional OLAP, MOLAP). Quest'ultimo memorizza i dati multidimensionali in strutture fisiche specializzate.

L'operazione effettuata è detta analisi Slice-and-Dice. Definendo una gerarchia di carattere geografico, si potrebbero rappresentare i dati a differenti livelli di dettaglio (ad es. vendite aggregate per città, regione, nazione e cosı̀ via). Quest'ultimo tipo di analisi è detta "Drill". Generalmente le informazioni sono presentate in report o briefings ovvero documenti che contengono i dati in formato facilmente leggibile. Un report può contenere, ad esempio, istogrammi, grafici, tabelle semplici o incrociate, in cui sono visualizzate le variabili di interesse in funzione delle dimensioni scelte, lasciando all'utente la libertà di determinare le variabili da rappresentare ed, eventualmente, le dimensioni da cui farle dipendere. Una volta costruito un report, è possibile effettuare le operazioni di analisi quali quelle già viste (drill, slice-and-dice), ed altre ancora. Queste modalità di analisi, presuppongono quindi una opportuna gestione e memorizzazione delle informazioni. Nell'ambito della progettazione di applicazioni OLAP, sono da tenere presenti i seguenti punti:

- la progettazione di una struttura di memorizzazione delle informazioni adeguata al tipo di analisi propria dell'OLAP (struttura dati multidimensionale);

- la progettazione di un canale di comunicazione tra la sorgente dei dati (Data Warehouse) e il nodo di sviluppo dell'applicazione (base dati multidimensionale);

- la progettazione dei moduli di report e briefing di rappresentazione delle informazioni (analisi).

Gli strumenti OLAP si possono dividere in tre categorie:

- Soluzioni End to End che prevedono un database multidimensionale (non relazionale MOLAP) proprietario dal lato server e dei tools di analisi dal lato client. Questi prodotti necessitano di un impegno non indifferente nel configurare il database multidimensionale e nel predisporre opportune viste da presentare all'utente.

- Alla seconda categoria appartengono quei prodotti che memorizzano i dati in database relazionali (ROLAP), presentandoli ai tools di analisi come fossero memorizzati in un database multidimensionale. Per questi prodotti valgono le osservazioni fatte per i prodotti MOLAP del punto precedente.

- La terza categoria dei prodotti OLAP, sviluppatasi di recente, propone tools molto più snelli e facili sia da configurare che da utilizzare. Questi tools, una volta estratti i dati da database eventualmente remoti, li organizzano in un database multidimensionale dinamico residente nel client dell'utente che ha effettuato la query. Quest'approccio evita i costosi database multidimensionali dal lato server dei prodotti MOLAP e il complesso livello dei metadati dei prodotti ROLAP. Ovviamente il prezzo da pagare a fronte dei vantaggi sopra elencati, è la minore scalabilità che questi prodotti presentano.

5 Livello dei dati riconciliati con struttura gerarchica

Nel caso di ambienti applicativi che comprendono molti sistemi operazionali eterogenei e complessi, una struttura "piatta" dei dati riconciliati si rivela in realtà inadeguata. In tali casi, infatti, lo schema del livello riconciliato, ottenuto integrando gli schemi delle basi di dati operazionali, conterrebbe un gran numero ed una grande varietà di oggetti, diventando eccessivamente complesso per poter essere utilizzato. Di conseguenza, la generazione dei dati derivati dai dati riconciliati diventerebbe un'operazione molto complessa. É importante sottolineare che il problema è correlato alla complessità delle sorgenti informative e non all'architettura a tre livelli; analoghe difficoltà sorgerebbero, infatti, anche qualora venissero adottate architetture ad uno o a due livelli.

L'approccio proposto, basato sulla nozione di clustering, non prevede l'integrazione contemporanea di tutti gli schemi operazionali in un unico schema globale piatto; al contrario, gli schemi operazionali sono raggruppati in cluster omogenei (sulla base di una opportuna metrica) e successivamente, per ciascun cluster, vengono integrati gli schemi che lo compongono. Questo processo può essere reiterato più volte sugli schemi rappresentativi dei cluster ottenuti, ottenendo in tal modo una struttura gerarchica dei dati denominata repository dei dati. Questo è una sorta di catalogo delle basi di dati, capace di descrivere i dati e gli schemi corrispondenti coinvolti a vari livelli di astrazione. Notiamo che i dati riconciliati del livello di astrazione più basso sono materializzati, mentre i dati relativi agli altri livelli di astrazione sono rappresentati come di viste. Seguendo l'approccio descritto, l'architettura a tre livelli con struttura gerarchica dei dati riconciliati può essere rappresentata come mostrato in Figura 8. In tutte le architetture di Data Warehouse presentate nella letteratura, i metadati, che sono fondamentali tanto per una derivazione efficiente dei dati quanto per un loro utilizzo appropriato nei processi decisionali, sono derivati ed aggiornati separatamente dagli schemi del data warehouse; pertanto, spesso, essi non sono correlati e consistenti con gli schemi corrispondenti (ad esempio, possono essere non aggiornati). Dal momento che, generalmente, un repository dei dati non colleziona soltanto gli schemi delle basi di dati coinvolte, ma può anche memorizzare i metadati ad essi correlati, ne deriva che l'approccio proposto ricava i metadati durante la generazione dei dati riconciliati. Ogniqualvolta il repository dei dati (e, conseguentemente, il livello dei dati riconciliati) viene aggiornato, anche i metadati corrispondenti vengono aggiornati. Ciò rappresenta un secondo vantaggio dell'architettura proposta, che è capace di mantenere correlati e consistenti i dati riconciliati, i metadati e i dati derivati. Per quanto concerne il confronto tra uno schema riconciliato piatto ed uno gerarchico, valgono le seguenti considerazioni:

- Gli schemi piatti sono casi particolari di quelli gerarchici e corrispondono al caso in cui le

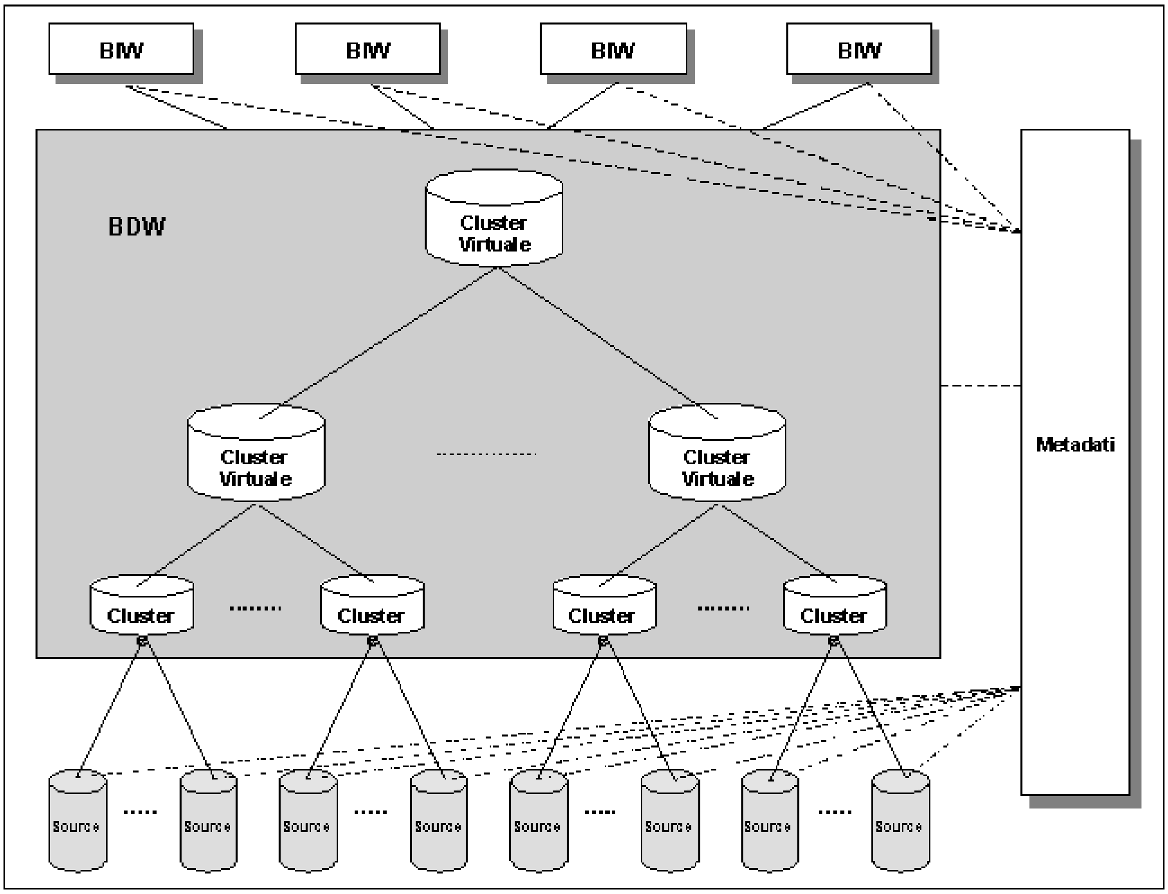

Figure 8: Struttura gerarchica

basi di dati operazionali vengono tutte raggruppate in un unico cluster e su di questo non viene eseguita alcuna astrazione.

- L'architettura gerarchica è particolarmente adatta per una costruzione incrementale del data warehouse. Inoltre, a differenza delle architetture piatte, la rimozione o la modifica di uno schema operazionale comporta l'aggiornamento degli schemi dei soli cluster in cui questo è coinvolto; in questo modo si evita la modifica dell'intero schema globale riconciliato.
- Quando il numero di basi di dati operazionali coinvolte è elevato è presumibile che in uno schema piatto vengano rappresentati centinaia di oggetti; conseguentemente, l'analisi di un tale schema risulta oltremodo difficile. Al contrario, in uno schema gerarchico, i vari sottoschemi che rappresentano i cluster hanno una chiara semantica.
- Utilizzando una struttura gerarchica per i dati riconciliati, la progettazione dei BIW (cioè dei Data Mart) è relativamente più semplice dal momento che ciascun Data Mart è relativo a un gruppo di sorgenti di dati che, nel complesso, sono situate in un sottoalbero del repository dei dati con radice in un determinato schema di livello K; pertanto il progettista può concentrarsi su un sottoinsieme ristretto di oggetti esaminato a livello di astrazione appropriato.
- Il costo del popolamento di un DW con struttura gerarchica dei dati riconciliati richiede la "costruzione" delle viste che rappresentano i livelli superiori della gerarchia. Tuttavia,

l'esecuzione delle viste non è un processo computazionalmente costoso, dal momento che esse sono eseguite da un DBMS unico centralizzato (che memorizza il livello dei dati materializzati e, in generale, il BDW) e coinvolgono, tipicamente, un insieme ristretto di dati (cioè quello corrispondente al cluster su cui la vista è stata definita).

Dalle suddette osservazioni si deduce che, pagando un prezzo limitato in termini di spazio e di tempo di calcolo richiesti, si ottiene un'architettura che mantiene tutti i pregi delle architetture classiche a tre livelli, superando quei problemi che nascono in presenza di un gran numero i sorgenti i dati.

6 Business Data Warehouse e architettura fisica

Il business data warehouse è la realizzazione fisica del livello dei dati riconciliati, implementato attraverso un ambiente relazionale in terza forma normale. Vista l'enorme mole di dati che un data warehouse deve essere in grado di trattare, la scelta obbligata sembra essere quella di un database parallelo [3]. Per capire meglio questo punto ricordiamo la natura particolare del data warehouse. W.H.Inmon lo ha descritto come "una collezione non-volatile, tempo-variante di supporto alle decisioni manageriali". Questo vuol dire che una volta che un dato è aggiunto esso non cambierà: ossia i dati sono inseriti una volta mentre durante le normali operazioni, richieste tipiche sono le letture e solo raramente updates. La conseguenza è che il caricamento e la lettura di dati possono essere resi più efficienti da una implementazione parallela. Questa soluzione non è però ovvia come puó sembrare; infatti i diversi modi in cui il BDW e i BIWs vengono usati può avere influenza sull'applicabilità della tecnologia parallela nei due livelli. Quando si vaglia l'uso di databases paralleli nell'implementazione del BDW vanno tenuti presenti i seguenti aspetti :

- Il BDW è di grandi dimensioni (centinaia di Gbytes è oggi una dimensione comune) e sostanzialmente normalizzato.
- La struttura dei dati nel BDW può essere molto variabile nelle fasi preliminari di sviluppo (2-3) anni.
- Il BDW è aggiornato regolarmente attraverso l'inserimento di record. Le performance di un database nell'inserimento di records è un parametro di progettazione fondamentale.
- Il processo di popolamento dei BIWs dal BDW è fortemente standardizzato.

Esistono essenzialmente tre schemi architetturali: symmetrical multi-processing (SMP), massive parallel processing (MPP), and non-uniform memory access (NUMA). Compareremo ora i vantaggi e gli svantaggi di ciascuna di tali architetture.

SMP In questo schema osserviamo un livello basilare di parallelismo; diversi processori vengono usati per realizzare i diversi tasks, attraverso la condivisione di memoria e dischi. Il problema essenziale di questo approccio è la scalabilità; infatti tutte le attività passano attraverso il bus del sistema. In aggiunta, una singola memoria globale limita la quantità di dati che risulta possibile processare. É piuttosto raro che un sistema SMP possa presentarsi con più di otto-sedici processori senza un costo esagerato per la progettazione del bus.

MPP In questa alternativa, diversi nodi sono collegati tra di loro per formare un solo sistema; ogni nodo risulta un mini sistema SMP completo di memoria e disco. Di conseguenza non è raro vedere sistemi con più di 64 processori cosi interconnessi. L'intero processo è distribuito attraverso il sistema; questo è tuttavia un potenziale punto debole; infatti, per distribuire efficientemente l'applicazione uno sviluppatore deve non solo ottimizzare il codice per tale architettura

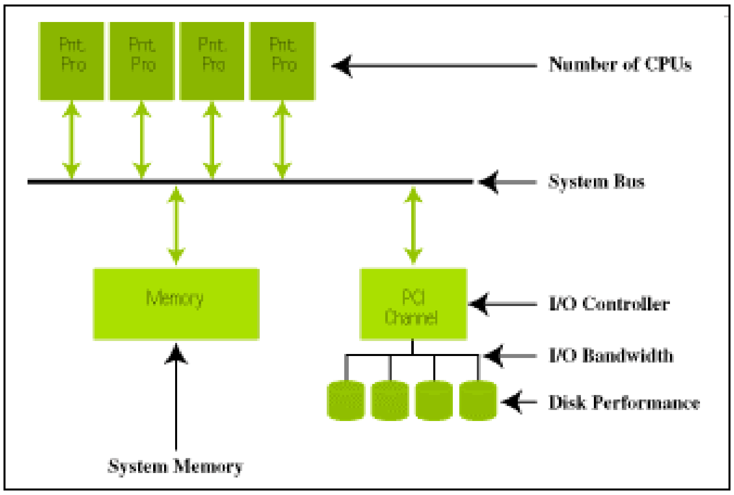

Figure 9: SMP

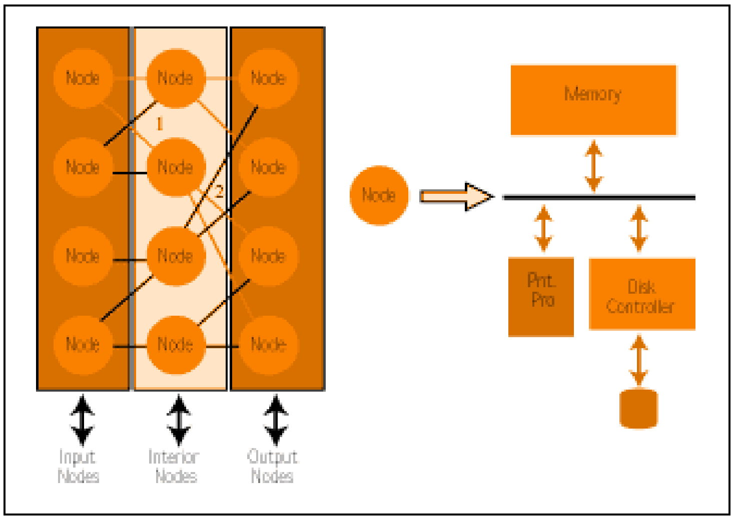

Figure 10: MPP

ma per il particolare numero di processori che si intende usare. Inoltre anche i dati risultano distribuiti; per cui se un processore richiede dati residenti su un altro nodo, deve necessariamente attuarsi un trasferimento. Di conseguenza al crescere del numero di nodi cresce anche la probabilità del verificarsi di un tale evento; per cui l'overhead del parallelismo in questi sistemi cresce spesso più rapidamente dei benefici apportati, senza considerare la difficoltà nell'inserimento di un nuovo nodo che causa la riconfigurazione dell'intero sistema.

NUMA Tali sistemi sono composti di blocchi elementari (SBB) simili ai nodi MPP, ma con la differenza che la memoria in ogni blocco è vista come una parte di una memoria globale; inoltre ogni SBB ha un accesso diretto ai dischi, per cui il recupero dei dati ha un costo uniforme indipendente dal blocco che richiede tale operazione.

Le considerazioni appena svolte portano alla conclusione che database paralleli ottimizzati per accessi read-only, con particolare enfasi su operazioni di scansione completa di tabelle, hanno benefici relativi nel livello BDW. Questo perché il processo di estrazione dati dal BDW è fondamentalmente standardizzato. Avremo inoltre che l'uso di architetture SMP o MPP può presentare difficoltà durante la fase iniziale di crescita (2-3 anni); infatti, l'ottimizzazione delle

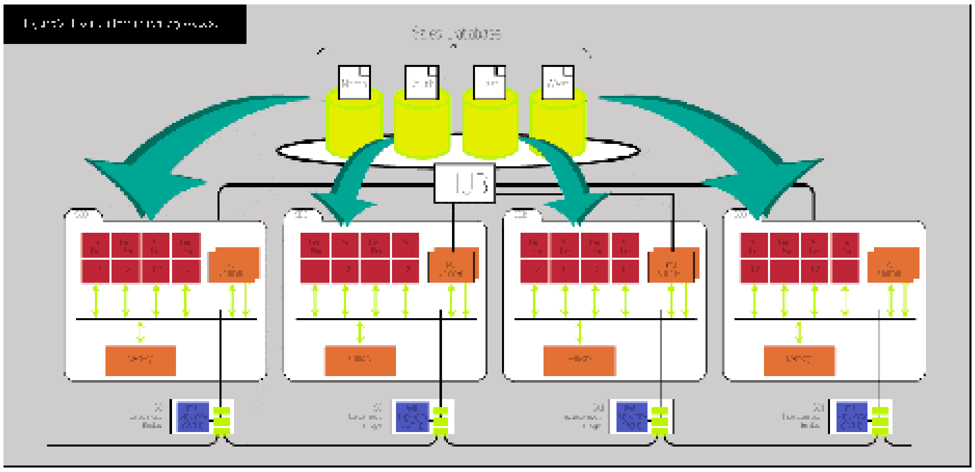

Figure 11: NUMA

performance in tali architetture è fortemente dipendente da come i dati sono distribuiti nei vari dischi. La scelta di come distribuire i dati va fatta in considerazione del contenuto e dell'uso che si andrà a fare dei dati. Tali fattori non sono completamente noti nella fase preliminare di crescita del BDW, caratterizzata da numerosi cambiamenti della struttura dati. In conclusione avremo che i databases paralleli possono aumentare significativamente le performance del BDW, e diventano praticamente l'unica scelta possibile all'aumentare delle dimensioni del warehouse. Va tenuto comunque presente che l'introduzione della tecnologia parallela nella fase di crescita del BDW può portare ad un grosso overhead dovuto a difficoltà di tuning. Una scelta più oculata sembra essere quella di un ambiente DBMS tradizionale per poi migrare successivamente ad una architettura parallela. L'uso della tecnologia parallela gioca un ruolo significativo anche nell'implementazione dei BIWs. Questo è dovuto alla capacità dei databases paralleli di trattare queries su tabelle molto grandi senza l'esigenza di indici predefiniti su tutte le possibili chiavi di ricerca. Le considerazioni fatte per l'implementazione parallela del BDW valgono anche in questo caso, tenendo però presente che :

- I BIWs devono essere ottimizzati nei tempi di risposta alle queries.
- Il popolamento dei BIWs non richiede le complesse procedure necessarie invece nel popolamento del BDW. Come risultato avremo che databases paralleli ottimizzati per le queries piuttosto che per gli aggiornamenti non pongono problemi in tale livello.
- Il tuning richiesto per databases paralleli che supportino i vari BIWs è relativamente semplice rispetto al tuning del BDW, in quanto ogni BIW è ristretto ad un particolare tipologia di dati.

7 Architetture federate

L'architetura federata di un data warehouse può essere considerata come la risposta naturale a molti casi reali in cui una organizzazione si trova a gestire diversi data warehouse o data marts. Un sistema federato (FDBS) è formato da diversi database di sistema; esso provvede all'accesso integrato alle varie componenti che cooperano al fine di produrre risposte consolidate alle queries. Molte architetture FDBS sono state proposte per sostenere tale accesso integrato; due di esse sono oggetto della nostra attenzione.

7.1 Architettura federata a 5 livelli

L'architettura di [29], detta dei cinque livelli, è molto generale e ingloba la maggior parte delle architetture precedentemente proposte. In questo schema tre tipi di modelli per i dati (data models) sono usati: primo, ciascuno componente database può avere il suo modello nativo; secondo, un data-model canonico (CDM); terzo, uno schema esterno che può essere definito in modo diverso per ciascun utente. Inoltre, tre meccanismi di base sono usati per definire gli schemi a livelli diversi a partire dagli schemi nel livello immediatamente precedente; vale a dire: traduzione di schemi (tra il CDM ed altro modelli), integrazione di schemi (tutti definiti con il CDM), e definizione di schemi derivati.

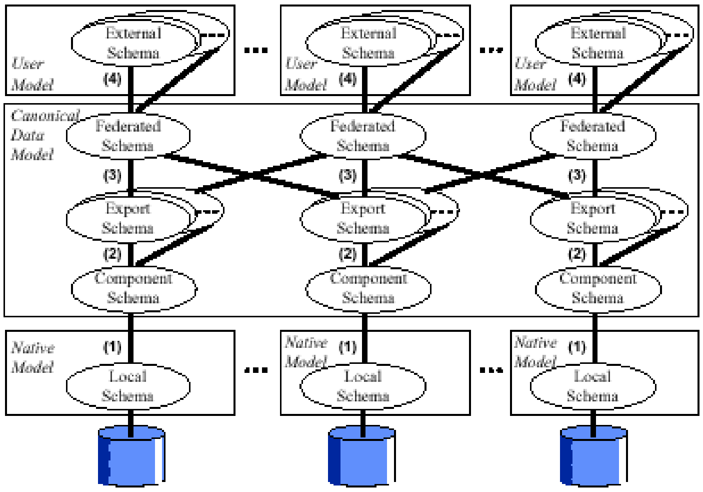

Figure 12: 5 livelli

Come mostra la Figura 12, ciascuna componente del sistema ha il suo schema locale definito in un modello nativo. Gli schemi locali sono tradotti per ottenere gli schema di componente definiti usando il CDM. Ciascuno database negozia quale di sui dati rendere accessibile, perciò, da ogni schema di componente diversi schemi di esportazione possono essere definiti usando un meccanismo di derivazione. Gli schemi esportati sono integrati per formare uno schema federato, da cui possono essere definiti schemi esterni usando un qualunque meccanismo di traduzione.

7.2 Architettura federata a 8 livelli

L'architettura ad otto livelli, definita da [28], è una estensione di quella a cinque livelli con tre nuovi livelli di schema che interessano tre situazioni non perfettamente gestite dall'architettura di base: federazioni di molti databases; schemi esterni in modelli dell'utente diversi dal CDM e semantiche multiple al livello dello schema federato. Perciò, risultano definiti i seguenti ulteriori livelli: lo schema negoziabile, lo schema dell'utente esterno e lo schema applicazione.

8 Integrated Data Warehouse Model

La ricerca nelle basi di dati eterogenee, come visto, ha fornito i metodi per integrare le basi di dati disparate in una singola architettura di unificazione: il modello federated della base di dati. Ma tale modello ha alcuni limiti; infatti lo schema federated non-è attuato, il che significa che le domande dovranno essere valutate nelle diverse basi di dati, con conseguente tempo di reazione più lento. Inoltre i dati dalle sorgenti esterne non sono integrati all'interno dello schema federated. E' stato di conseguenza proposto di estendere l'architettura federated per includere un data warehouse modellistico come vista materializzata dello schema federato sottostante.

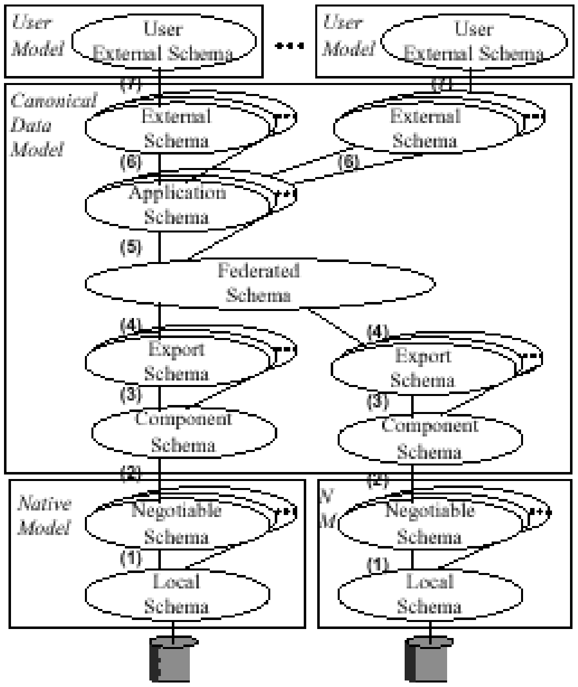

Figure 13: 8 livelli

In più, sono impiegate le tecniche di manutenzione di vista per aggiornare il data warehouse in coseguenza di cambiamenti nelle sorgenti operative. Questo metodo fornisce come vantaggi un'architettura unificata che lega il magazzino di dati alle basi di dati eterogenee multiple, la possibilità di fornire l'accesso flessibile ai dati correnti che risiedono nelle sorgenti di dati ed una facilità di manutenzione contro qualsiasi cambiamento allo schema nella sorgente o nel magazzino di dati. Il modello IDWM integra un data warehouse in una arcitettura federata strettamente accoppiata. In particolare le sorgenti di dati ed i dati operativi immagazzinano le forme la parte inferiore e la parte superiore rispettivamente, mentre lo strato dell'interfaccia fornisce i metodi di accesso fra i due. Quando l'accesso ai dati correnti è richiesto, è richiamato dalle sorgenti di dati operative accedendo tramite lo strato dell'interfaccia. Nel modello, una base di dati componente (CDB) si riferisce ad una sorgente di dati operativa. I dati che sono di interesse in primo luogo sono rappresentati uniformemente attraverso lo schema dell'esportazione, vengono poi integrati eliminando eventuali inconsistenze fornendo un schema dei dati integrati 14. La distinzione fra tale schema e lo schema del magazzino di dati è che il primo fornisce una vista costante e riconciliata di tutti i dati che esistono nell'impresa compresi i dati esterni, mentre il secondo è un sottoinsieme adeguato a i bisogni personali di un utente. Lo strato dell'interfaccia fornisce i metodi di accesso alle basi di dati componenti ed al magazzino di dati. IDWM non presuppone alcun modello specifico di dati adottato dalle sorgenti di dati ma per gli scopi dei meccanismi di sviluppo e dell'illustrazione, é utile considerare le basi di dati componenti ed il magazzino di dati come relazionali. Diamo risalto che questi presupposti in alcun modo non rendono il modello meno generico. IDWM considera il magazzino come una vista attuata sopra le basi di dati componenti. Per effettuare le viste il modello usa il metodo rinviato per la propagazione dei cambiamenti incrementali. Questo metodo permette che le tabelle attuate siano fuori sincronizzazione per un periodo di tempo, il che è tollerabile nel piano d'azione d' immagazzinamento. Ma il risparmio di tempi supplementari è guadagnato ammucchiando parecchi aggiornamenti insieme e mantenendo la transazione e la comunicazione ambientali al minimo. Il metodo impiegato in questo modello consiste nell'applicare un insieme di regole di propagazione ad una vista attuata. Tali regole computano le tuple che devono essere cancellate

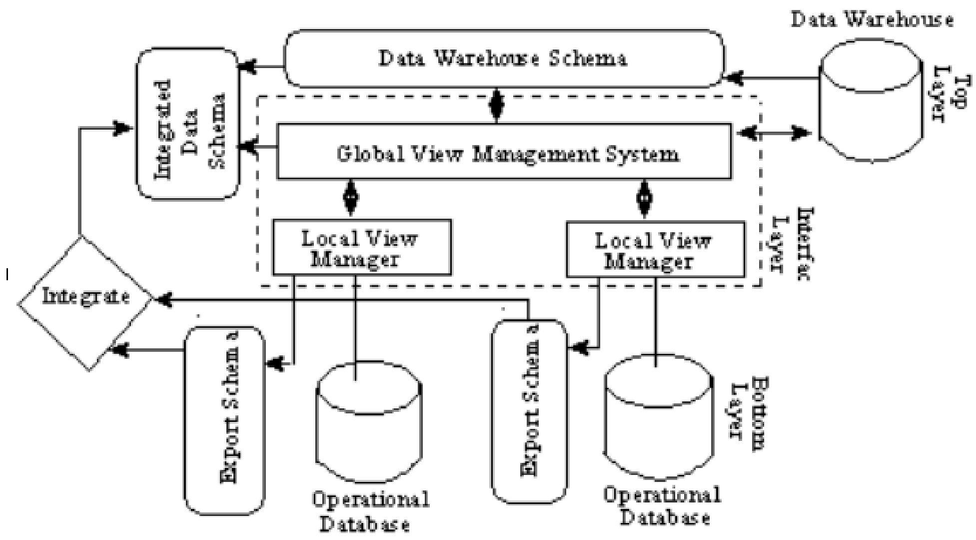

Figure 14: IDWM

ed inserite nella vista attuata accedendo alle tabelle della base ed ai relativi libri macchina di transazione, senza riferire le viste attuate nel magazzino di dati. Il sistema di gestione globale di vista (GVMS) e lo strumento locale del gestore di vista (LVM) sono i due meccanismi usati. LVM fornisce l'interfaccia fra base di dati operativa e magazzino di dati attraverso il GVMS. In più, trasmette periodicamente l'aggiornamento dalla sorgente operativa al magazzino. GVMS fornisce le funzionalità simili al magazzino di dati alle basi di dati operative. In più, assiste la richiesta di dati dal magazzino e coordina gli LVM durante gli aggiornamenti periodici.

9 POPOLAMENTO DI UN DATA WAREHOUSE

Il popolamento è sicuramente uno degli aspetti più difficoltosi e dispendiosi della realizzazione di un DW. Il processo di popolamento può essere suddiviso in tre fasi fondamentali: la cattura dei dati, la trasformazione dei dati catturati e l'applicazione delle modifiche al DW.

9.1 Cattura dei dati

Il processo di cattura consiste nell'interagire con le sorgenti per ottenere una copia dei dati contenuti in esse, o la lista delle modifiche occorse. Si hanno due tipologie di cattura:

- Cattura statica: fornisce una copia dei dati operazionali in un determinato istante di tempo. Questo tipo di cattura è utilizzata quando un insieme di dati è caricato per la prima volta nel BDW.

- Cattura incrementale: a differenza della cattura statica, fornisce soltanto l'insieme dei dati che sono stati aggiunti, modificati o cancellati in un certo intervallo di tempo. Se da una parte il volume dei dati è sensibilmente minore rispetto alla cattura statica, dall'altra si hanno degli algoritmi di maggiore complessità.

I dati operazionali possono essere classificati (da un punto di vista temporale) nelle seguenti categorie:

- Dati transienti o semiperiodici memorizzati a stati: La caratteristica di questi dati è l'impossibilità di conoscere quanto a lungo un record esisterà nel sistema. Nel caso di dati transienti ogni transazione su un dato sovrascriverà le informazioni precedenti. Nel caso dei dati semiperiodici i records non sono sovrascritti, ma le informazioni storiche sono mantenute soltanto parzialmente;

- Dati periodici memorizzati a stati o a eventi: Per questi tipi di dati è mantenuta la storia completa di ogni record nei sistemi operazionali.

9.2 Tecniche per la cattura dei dati

Nel caso di cattura statica la sola tecnica utilizzabile coincide con l'estrazione statica dei dati. Nel caso di cattura incrementale, le tecniche utilizzabili si dividono in tecniche di cattura immediata, che catturano i cambiamenti nel momento stesso in cui si verificano, e tecniche di cattura differita, che vengono applicate ad intervalli di tempo predefiniti e pertanto cattureranno le modifiche occorse in quel particolare intervallo di tempo. Segue una breve descrizione delle varie tecniche di cattura.

Static Capture Il processo di cattura statica consiste nel copiare tutti i dati operazionali o un loro sottoinsieme. L'unico problema che si incontra nell'implementare un processo di cattura statica è dovuto all'eterogeneità dei dati provenienti da sorgenti diverse come: DB relazionali, reticolari, gerarchici, flat-files, fogli di calcolo, pagine HTML ecc. L'applicazione di cattura, infatti, deve presentare in output i dati in un unico semplice formato.

Application-assisted Capture In questo approccio le applicazioni operazionali hanno il compito di fornire un insieme delle modifiche in maniera continua e non volatile. Ovviamente tali applicazioni dovranno essere state pensate e realizzate per fornire queste funzionalità. proprio quest'ultimo il punto debole di tale tecnica in quanto, se da un lato è possibile progettare i nuovi sistemi operazionali in modo che comprendano tali funzionalità, dall'altro è spesso impossibile o dispendioso modificare le vecchie applicazioni.

Triggered Capture In questo caso la responsabilità della cattura dei dati è affidata ai DBMS, ammesso che questi ultimi offrano la possibilità di implementare i trigger. Un trigger può essere pensato come una "store-procedure" attivabile al verificarsi di un evento. Nel caso della cattura di dati, si definiranno dei triggers corrispondenti alle modifiche di interesse occorse nei record dei DB. L'attivazione di un trigger causerà una transazione il cui scopo è quello di registrare l'evento che ha determinato la modifica o il record modificato.

Log Capture Le modifiche occorse nei dati di un DBMS sono mantenute in un file di log al fine di poter effettuare operazioni di backup e recovery. Questo log-file può essere utilizzato per catturare le modifiche di interesse da migrare nel warehouse. Ovviamente si assume che il DBMS mantenga un log-file che contenga informazioni sufficienti per supportare il processo di cattura.

Timestamp-based Capture Tale tecnica è applicabile soltanto se ai dati operazionali di interesse è associato un timestamp che identifichi l'istante temporale in cui tali records sono stati aggiornati inseriti o cancellati. Con queste assunzioni risulta semplice catturare le modifiche occorse tra due istanti di tempo. Le applicazioni che gestiscono i dati, o i sistemi su cui esse risiedono, devono farsi carico della gestione dei timestamps.

File Comparison Questo approccio, usato come alternativa qualora le tecniche precedenti non siano applicabili, prevede la comparazione di due immagini di dati operazionali per determinare le modifiche avvenute.

9.3 Processo e Funzioni di Trasformazione

Tale processo si colloca tra la cattura dei dati e il loro caricamento nel BDW. Il suo scopo è quello di definire la relazione tra i records in input ed in output attraverso l'applicazione di un certo numero di funzioni di trasformazione che possono essere raggruppate in sei categorie:

- Funzioni di selezione;
- Funzioni di separazione/concatenazione;
- Funzioni di normalizzazione/denormalizzazione;
- Funzioni di aggregazione;
- Funzioni di conversione;
- Funzioni di arricchimento.

Le prime quattro funzioni operano a livello di record mentre le ultime due a livello di attributi.

9.4 Applicazione delle modifiche ai dati riconciliati

I dati in output del processo di trasformazione vengono inseriti nel DW in due diverse circostanze: nel processo di creazione del BDW e nel mantenimento del BDW.

Le tecniche di applicazione delle modifiche sono le seguenti:

- Load: l'insieme dei dati da inserire viene caricato nel BDW sostituendo, eventualmente ce ne fossero, quelli preesistenti. Questa tecnica è usata per il popolamento iniziale del BDW.
- Append: l'insieme dei dati da inserire viene aggiunto ai dati preesistenti. Pertanto è possibile che si verifichi una duplicazione di records. Anche questa tecnica, come la precedente, è usata nella fase di popolamento iniziale del BDW.
- Merge Distruttivo: attraverso questa tecnica di cattura l'insieme dei dati da inserire viene unito con l'insieme dei dati esistenti. Se la chiave di un record da inserire coincide con la chiave di un record esistente, quest'ultimo viene sovrascritto. Il Merge Distruttivo può essere usato in fase di mantenimento del BDW.
- Merge Costruttivo: simile alla tecnica precedente, con la differenza che un nuovo dato catturato è sempre aggiunto come nuovo record. Se esiste già un record con la stessa chiave, questo non viene sovrascritto, ma opportunamente marcato per indicare che nel sistema esiste un record più aggiornato con la stessa chiave. Anche questa tecnica può essere applicata in fase di mantenimento del BDW.

References

[1] E. Brethenoux, H. Dresner, k. Strnge, and J. Block. Data Warehouse, Data Mining and Business Intelligence : The Hype Stops Here, *Gartner Group Strategic Analisys Report*, October 1997.

[2] C. Batini and M. Lenzerini, A methodology for data schema integration in the entity relationship model, *IEEE Transactions on Software Engineering*,10(6):650–664, 1984.

[3] C.J. Bontempo and C.M. Saracco. Database Management : Principles and Products. *Prentice-Hall PTR, Englewood Cliffs*, NJ,1996.

[4] S. Castano and V. De Antonellis, Semantic dictionary design for database interoperability, In *Proc. of International Conference on Data Engineering (ICDE'97)*, pages 43–54, Birmingham, United Kingdom, 1997, IEEE Computer Society.

[5] S. Chaudhuri and U. Dayal, An overview of data warehousing and olap technology, *ACM SIGMOD RECORD*, 26(1):65–74, 1997.

[6] B. Devlin. Data Warehouse, from Architecture to Implementation. *Addison-Wesley*, 1997.

[7] Gartner Group. So Many BuzzWords. *Gartner Group Conference Presentation*, November 1997.

[8] M. Golfarelli, D. Maio, and S. Rizzi, Conceptual design of data warehouses from e/r schemes, In Proc. of the *Hawaii International Conference on System Sciences*, Kona, Hawaii, USA, 1998.

[9] M. Golfarelli, D. Maio, S. Rizzi. The Dimensional Fact Model: a Conceptual Model for Data Warehouses. *International Journal of Cooperative Information Systems*, vol. 7, n. 23, pp. 215-247, 1998.

[10] M. Golfarelli, S. Rizzi. A Methodological Framework for Data Warehouse Design. Proceedings *ACM DOLAP'98*, Washington, pp. 3-9, 1998

[11] W.H. Inmon, What is a data warehouse?, *Prism Tech. Topic*, 1(1), 1997.

[12] W. H. Inmon. Building the Data Warehouse. *QED Information Sciences*, Wellesley, MA, 1992.

[13] W.H. Inmon and R.D. Hackathorn. Using the Data Warehouse. *Wiley*, New York, 1994.

[14] IBM. Information Warehouse : An Introduction. GC26-4876-00, *IBM Corporation*, 1991.

[15] A.K. Jain and R.C. Dubes, Algorithms for Clustering Data, *Prentice Hall*, Englewood Cliffs, 1988.

[16] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis, Fundamentals of Data Warehouses, *Springer*, 2000.

[17] R. Kaula. Integrating Decision Support Systems in Organizations : A Three-level Framework. *Industrial Mamagement & Data Systems*, No. 4, pages 8-14, 1994.

[18] L. Kaufman and P.J. Rousseeuw, Findings Groups in Data: an Introduction to Cluster Analysis, *John Wiley & Sons*, New York, 1990.

[19] J.A. Larson, S.B. Navathe, and R. Elmastri, A theory of attribute equivalence in databases with application to schema integration, *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[20] K.Orr, Understanding data warehousing, *American Programmer*, 8(5):2–7, 1995.

[21] K. Orr. Understanding Data Warehousing. *American Programmer*, 8(5) :2-7, 1995.

[22] William H. Inmon. What is a Data Warehouse ?. *Prism Tech Topic*, Volume 1 No. 1, 1997.

[23] L. Palopoli, L. Pontieri, G. Terracina, and D. Ursino, Intensional and extensional integration and abstraction of heterogeneous databases, *Data & Knowledge Engineering*, Forthcoming.

[24] L. Palopoli, D. Saccà, G. Terracina, and D. Ursino, A unified graph-based framework for deriving nominal interscheme properties, type conflicts and object cluster similarities, Proc. of Fourth *IFCIS Conference on Cooperative Information Systems (CoopIS'99)*, pages 34–45, Edinburgh, United Kingdom, 1999. IEEE Computer Society.

[25] L. Palopoli, D. Saccà, G. Terracina, and D. Ursino, Semi-automatic extraction of hyponymies and overlappings from heterogeneous database schemes, In Proc. of International *Conference on Database and Expert Systems Applications (DEXA 2000)*, Greenwich (England), United Kingdom, 2000, Forthcoming.

[26] L. Palopoli, D. Saccà, and D. Ursino, An automatic technique for detecting type conflicts in database schemes, Proc. of ACM *Conference on Information and Knowledge Management (CIKM'98)*, pages 306–313, Bethesda, Maryland, USA, 1998. ACM Press.

[27] L. Palopoli, G. Terracina, and D. Ursino, A semi-automatic tool for constructing access systems to pre-existing, distributed databases, Proc. of Challenges of Symposium on Advances in *Databases and Information Systems (ADBIS-DASFAA 2000)*, Prague, Czech Republic, 2000, Forthcoming.

[28] Saltor, F., Campderrich, B., Rodrguez, E., Rodrguez, L.: On Schema Levels for Federated DB Systems. In Yetongnon & Hariri (eds.): Proc. of the *ISCA Intl. Conf. on Parallel and Distributed Computing Systems (Dijon, 1996)*, ISCA, 766-771

[29] Sheth, A., Larson, J.: Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys*, Vol. 22, No. 3 (Sep. 1990)

[30] S. Spaccapietra and C. Parent, View integration: A step forward in solving structural conflicts, *IEEE Transactions on Knowledge and Data Engineering*, 6(2):258–274, 1994.

[31] K. Strange. Four Myths About Data Marts. Gartner Group Monthly Research Review, October 1997.

[32] G. Terracina and D. Ursino, A study on the interaction between interscheme property extraction and type conflict resolution, Proc. of *International Database Engineering and Applications Symposium (IDEAS '00)*, Yokohama, Japan, 2000. Forthcoming.

[33] J. Widom, Research problems in data warehousing, Proc. of *International Conference on Information and Knowledge Management (CIKM'98)*, pages 25–30, Baltimore, Maryland, USA,1995. ACM Press.