InterData

Programma di ricerca (cofinanziato dal MURST, esercizio 1997)

Metodologie e tecnologie per la gestione di dati e processi su reti Internet e Intranet

Il sistema MOMIS per l'integrazione di sorgenti strutturate e semistrutturate

Sonia Bergamaschi, Domenico Beneventano, Maurizio Vincini

T3-R08 **10 febbraio 1999**

Sommario

In questo Rapporto viene presentata l'architettura del sistema MOMIS (Mediator envirOnment for Multiple Information Sources) [12, 13, 14, 15], basato sulla proposta I^3 (Intelligent Information of Integration) [50], rivolto all'integrazione di sorgenti di dati testuali strutturati e semistrutturati. L'obiettivo che ci proponiamo è quello di fornire un *accesso integrato* a sorgenti informative eterogenee, mettendo in condizioni l'utente generico di porre una singola query ed ottenere in'unica risposta unificata dei dati provenienti dalle singole sorgenti. Nel sistema introduciamo un modello ed un linguaggio di definizione comune, derivato dallo standard ODMG-93, utile agli scopi di integrazione. Questo linguaggio, ODL_{I^3}, è stato proposto ed illustrato nel rapporto precedente [10].

Tema	Tema 3: Integrazione di informazioni estratte da siti WWW
Codice	T3-R08
Data	10 febbraio 1999
Tipo di prodotto	Rapporto tecnico
Unità responsabile	MO
Unità coinvolte	MO
Autori	Sonia Bergamaschi, Domenico Beneventano, Maurizio Vincini
Autore da contattare	Sonia Bergamaschi Dipartimento di Scienze dell'Ingegneria Università degli Studi di Modena e Reggio Emilia Via Campi 213/B, 41100 Modena, Italia sonia@dsi.unimo.it

Il sistema MOMIS per l'integrazione di sorgenti strutturate e semistrutturate

Sonia Bergamaschi Domenico Beneventano Maurizio Vincini

Sommario

In questo Rapporto viene presentata l'architettura del sistema MOMIS (Mediator environment for Multiple Information Sources) [12, 13, 14, 15], basato sulla proposta I^3 (Intelligent Information of Integration) [50], rivolto all'integrazione di sorgenti di dati testuali strutturati e semistrutturati. L'obiettivo che ci proponiamo è quello di fornire un *accesso integrato* a sorgenti informative eterogenee, mettendo in condizioni l'utente generico di porre una singola query ed ottenere in'unica risposta unificata dei dati provenienti dalle singole sorgenti. Nel sistema introduciamo un modello ed un linguaggio di definizione comune, derivato dallo standard ODMG-93, utile agli scopi di integrazione. Questo linguaggio, ODL_{I^3}, è stato proposto ed illustrato nel rapporto precedente [10].

Il processo di integrazione è basato sull'analisi delle descrizioni degli schemi di ciascuna sorgente, descritti secondo il modello proposto per sorgenti strutturate e nella forma di *object patterns* nel caso semistrutturato. L'approccio ha il proprio fondamento teorico sulle tecniche: *Logiche descrittive* e *clustering*, che vengono utilizzate per costruire la visione integrata delle informazioni e per le operazioni di query processing per il reperimento delle informazioni.

In sezione 1 viene introdotta la nostra proposta; in sezione 2 viene fornita la descrizione dell'architettura. La sezione 3 introduce un esempio utilizzato come riferimento per le sezioni successive, nelle quali viene completata l'illustrazione del nostro approccio. In particolare, in sezione 4 viene descritto il modello comune dei dati, in sezione 5 le estensioni alla logica descrittiva OLCD, nelle sezioni 6,7,8,9 la generazione del Thesaurus comune, l'analisi delle affinità, dei cluster e la costruzione dello schema comune di mediatore, rispettivamente. La sezione 10 si occupa di descrivere il processo di esecuzione delle query all'interno del sistema MOMIS ed infine nella sezione 11 la nostra proposta viene confrontata con altri progetti presenti nel mondo della ricerca.

1 Introduzione

La continua crescita di informazioni accessibili (sia sulla rete Internet, sia all'interno di una azienda) ha portato all'esigenza di costruire sistemi che aiutino il loro reperimento, ovvero che si preoccupino di ritrovare le informazioni (e questo può essere realizzato attraverso i cosiddetti motori di ricerca) ma che diano anche una visione integrata di queste, in modo da eliminare le incongruenze e le ridondanze necessariamente presenti. Per questo scopo è stato introdotto da Wiederhold [67] il concetto di *mediatore*, un modulo il cui compito principale è quello di reperire ed integrare informazioni da una molteplicià di sorgenti eterogenee. I problemi che devono essere affrontati in questo ambito sono principalmente dovuti ad eterogeneità strutturali e applicative, nonché alla mancanza di una ontologia comune, che porta a differenze semantiche tra le fonti di informazione. A loro volta, queste differenze semantiche possono originare diversi tipi di conflitti, che vanno dalle semplici incongruenze nell'uso dei nomi (quando nomi differenti sono utilizzati in sorgenti diverse per identificare gli stessi concetti) a conflitti strutturali (quando modelli diversi sono utilizzati per rappresentare le stesse informazioni). In letteratura sono stati presentati diversi sistemi diretti all'integrazione di database convenzionali, come pure applicabili all'integrazione di dati semistrutturati. Di questi, seguendo quanto esposto in [18], si può realizzare una categorizzazione sulla base del diverso tipo di approccio utilizzato, distinguendoli tra *semantici* e *strutturali*. Per quanto riguarda l'approccio *strutturale* (e tra questi l'esempio piú importante è senza dubbio costituito dal progetto TSIMMIS, già ampiamente descritto nel rapporto INTERDATA [11]) si possono sottolineare i seguenti punti caratterizzanti:

- utilizzo di un modello self-describing per trattare tutti i singoli oggetti presenti nel sistema, sopperendo all'eventuale mancanza degli schemi concettuali delle diverse sorgenti;
- inserimento delle informazioni semantiche in modo esplicito attraverso l'utilizzo di regole dichiarative (ed in particolare, in TSIMMIS, attraverso le MSL rule);
- utilizzo di un linguaggio self-describing che facilita l'integrazione anche e soprattutto di dati semi-strutturati;
- l'assenza degli schemi concettuali non permette, in caso di database di grandi dimensioni, una ottimizzazione semantica delle interrogazioni;
- sono eseguibili solo le interrogazioni predefinite dall'operatore (per le quali viene preventivamente memorizzato un piano di accesso), limitando in questo modo la libertà di richieste all'utente del sistema.

Altri progetti, e fra questi si inserisce MOMIS, seguono invece un approccio che si può definire *semantico*, e che è caratterizzato da diverse peculiarità:

- per ogni sorgente sono a disposizione dei metadati, codificati nello schema concettuale;
- nello schema concettuale sono pure presenti le informazioni semantiche, che possono essere utilmente sfruttate sia nella fase di integrazione delle sorgenti, sia in quella di ottimizzazione delle interrogazioni;
- deve essere presente nel sistema, per poter descrivere in modo uniforme tutte le informazioni che devono essere condivise, un modello di dati comune;
- viene realizzata una unificazione (parziale o totale) degli schemi concettuali (realizzata in modo automatico o manuale), per arrivare alla definizione di uno schema globale.

Diverse sono le motivazioni che possono portare alla adozione di un approccio di questo tipo, unito all'utilizzo di un modello ad oggetti:

1. la presenza di una schema globale permette all'utente di formulare qualsiasi interrogazione che sia consistente con lo schema, e le informazioni semantiche in esso comprese possono contribuire ad una eventuale ottimizzazione di queste interrogazioni;
2. la natura stessa degli schemi che utilizzano i modelli ad oggetti, attraverso l'uso delle primitive di generalizzazione e di aggregazione, permette la riorganizzazione delle conoscenze estensionali;
3. l'adozione di una semantica *type as a set* per gli schemi permette di controllarne la consistenza, facendo riferimento alle loro descrizioni;
4. ampi sforzi sono sono già stati realizzati per lo sviluppo di standard rivolti agli oggetti: CORBA [66] per lo scambio di oggetti attraverso sistemi diversi; ODMG-93 [36] (e con esso i modelli ODM e ODL per la descrizione degli schemi, e OQL come linguaggio di interrogazione) per gli object-oriented database;

5. l'adozione di una semantica di *mondo aperto* permette il superamento delle problematiche legate all'uso di un convenzionale modello ad oggetti per la descrizione di dati semistrutturati: gli oggetti di una classe condividono una struttura minima comune (che è quindi la descrizione della classe stessa), ma possono avere ulteriori proprietà non esplicitamente comprese nella struttura della classe di appartenenza.

1.1 Problematiche da affrontare

Pur avendo a disposizione gli schemi concettuali delle varie sorgenti, non è certamente un compito semplice individuare i concetti comuni ad esse, le relazioni che possono legarli, né tantomeno è banale realizzare una loro coerente integrazione. Tralasciando per un momento le differenze dei sistemi fisici (alle quali dovrebbero pensare i moduli wrapper) i problemi che si è dovuto risolvere, o con i quali occorre giungere a compromessi, sono (a livello di mediazione, ovvero di integrazione delle informazioni) essenzialmente di due tipi:

1. problemi ontologici;
2. problemi semantici.

1.1.1 Problemi ontologici

In questo ambito di ricerca, per ontologia si intende "l'insieme dei termini e delle relazioni usate in un dominio, che denotano concetti ed oggetti". Con ontologia quindi ci si riferisce a quell'insieme di termini che, in un particolare dominio applicativo, denotano in modo univoco una particolare conoscenza e fra i quali non esiste ambiguità poiché sono condivisi dall'intera comunità di utenti del dominio applicativo stesso. Non è certamente l'obiettivo di questo rapporto dare una descrizione esaustiva di ontologia e dei problemi che essa comporta (ancorché ristretti al campo dell'integrazione delle informazioni), ma ci limitiamo a riportare una semplice classificazione delle ontologie (mutuata da Guarino [59, 58]), per inquadrare l'ambiente in cui ci si muove. I livelli di ontologia (e dunque le problematiche ad essi associate) sono essenzialmente quattro:

1. *top-level ontology*: descrivono concetti molto generali come spazio, tempo, evento, azione..., che sono quindi indipendenti da un particolare problema o dominio: si considera ragionevole, almeno in teoria, che anche comunità separate di utenti condividano la stessa top-level ontology;
2. *domain e task ontology*: descrivono, rispettivamente, il vocabolario relativo a un generico dominio (come può essere un dominio medico, o automobilistico) o a un generico obiettivo (come la diagnostica, o le vendite), dando una specializzazione dei termini introdotti nelle top-level ontology;
3. *application ontology*: descrivono concetti che dipendono sia da un particolare dominio che da un particolare obiettivo.

Come ipotesi semplificativa di questo progetto, si è considerato di muoversi all'interno delle *domain ontology*, ipotizzando quindi che tutte le fonti informative condividano almeno i concetti fondamentali (ed i termini con cui identificarli). A tale scopo, come si vedrà nel seguito, si potrebbe utilizzare sistemi lessicali, quali, ad esempio, WordNet [47, 55].

1.2 Problemi semantici

Pur ipotizzando che anche sorgenti diverse condividano una visione similare del problema da modellare, e quindi un insieme di concetti comuni, niente ci dice che i diversi sistemi usino esattamente gli stessi vocaboli per rappresentare questi concetti, né tantomeno le stesse strutture dati. Poiché infatti le diverse sorgenti sono state progettate e modellate da persone differenti, è molto improbabile che queste persone condividano la stessa "concettualizzazione" del mondo esterno, ovvero non esiste nella realtà una semantica univoca a cui chiunque possa riferirsi.
Se la persona P1 disegna una fonte di informazioni (per esempio DB1) e un'altra persona P2 disegna la stessa fonte DB2, le due basi di dati avranno sicuramente differenze semantiche: per esempio, le coppie sposate possono essere rappresentate in DB1 usando degli oggetti della classe COPPIE, con attributi MARITO e MOGLIE, mentre in DB2 potrebbe esserci una classe PERSONA con un attributo SPOSA.
Come riportato in [62] la causa principale delle differenze semantiche si può identificare nelle diverse concettualizzazioni del mondo esterno che persone distinte possono avere, ma non è l'unica. Le differenze nei sistemi di DBMS possono portare all'uso di differenti modelli per la rappresentazione della porzione di mondo in questione: partendo così dalla stessa concettualizzazione, determinate relazioni tra concetti avranno strutture diverse a seconda che siano realizzate attraverso un modello relazionale, o ad oggetti.
L'obiettivo dell'integratore, che è quello di fornire un accesso integrato ad un insieme di sorgenti, si traduce allora nel non facile compito di identificare i concetti comuni all'interno di queste sorgenti e risolvere le differenze semantiche che possono essere presenti tra di loro. Possiamo classificare queste contraddizioni semantiche in tre gruppi principali:

1. **eterogeneità tra le classi di oggetti**: benché due classi in due differenti sorgenti rappresentino lo stesso concetto nello stesso contesto, possono usare nomi diversi per gli stessi attributi, per i metodi, oppure avere gli stessi attributi con domini di valori diversi o ancora (dove questo è permesso) avere regole differenti su questi valori;

2. **eterogeneità tra le strutture delle classi**: comprendono le differenze nei criteri di specializzazione, nelle strutture per realizzare una aggregazione, ed anche le *discrepanze schematiche*, quando cioé valori di attributi sono invece parte dei metadati in un altro schema (come può essere l'attributo SESSO in uno schema, presente invece nell'altro implicitamente attraverso la divisione della classe PERSONE in MASCHI e FEMMINE);

3. **eterogeneità nelle istanze delle classi**: ad esempio, l'uso di diverse unità di misura per i domini di un attributo, o la presenza/assenza di valori nulli.

Parallelamente a tutto questo, è però il caso di sottolineare la possibilità di sfruttare adeguatamente queste differenze semantiche per arricchire il nostro sistema: analizzando a fondo queste differenze, e le loro motivazioni, si può arrivare al cosiddetto **arricchimento semantico**, ovvero all'aggiungere esplicitamente ai dati tutte quelle informazioni che erano originariamente presenti solo come metadati negli schemi, dunque in un formato non interrogabile.

1.3 L'approccio

Vediamo ora una sommaria descrizione del nostro approccio.

La prima ipotesi che è stata fatta, per restringere il campo applicativo del sistema da progettare (e di conseguenza per restringere il campo dei problemi a cui dare risposta) è di avere a che fare con sorgenti di dati testuali strutturati e semistrutturati, come possono essere basi di

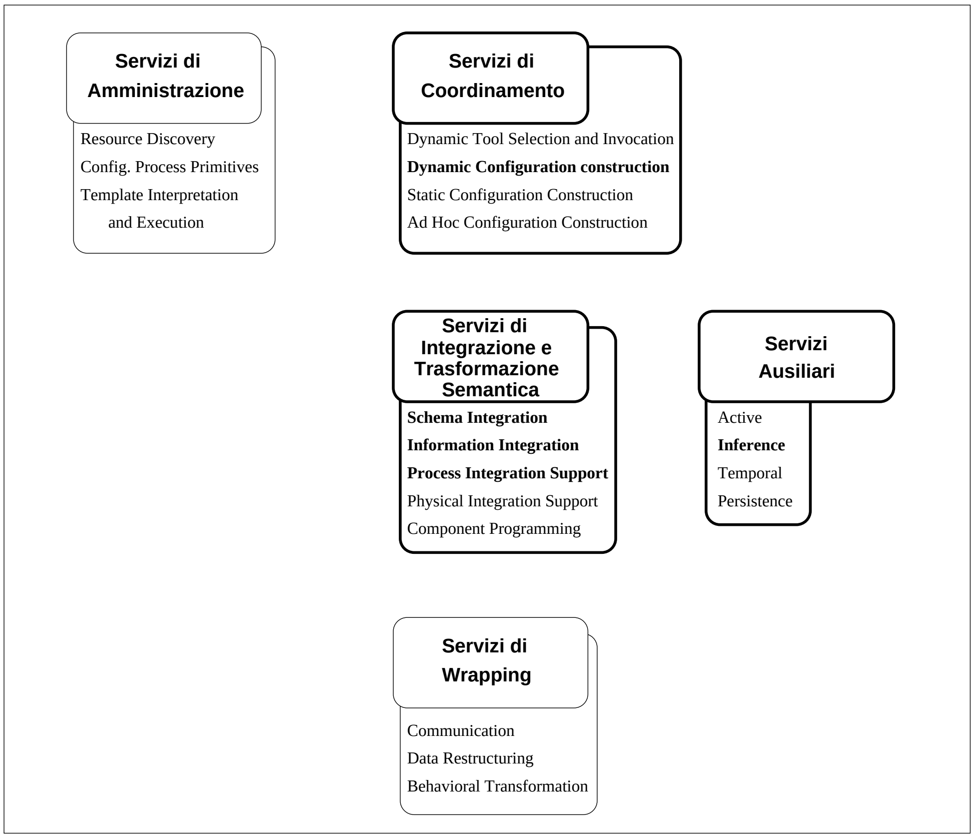

Figura 1: Servizi I^3 presenti nel mediatore

dati relazionali, ad oggetti e file di testo, pagine html. L'approccio architetturale scelto è stato quello proposto dal progetto ARPA-I^3 [50], simile ad altri progetti tra i quali SIMS [4, 5] ed Information Manifold [51, 53]. Per una descrizione dettagliata dello stato dell'arte della ricerca nell'ambito dell'integrazione si veda [10] L'architettura consta di 3 livelli principali:

1. utente: attraverso un'interfaccia grafica l'utente pone delle query su uno schema globale e riceve un'unica risposta, come se stesse interrogando un'unica sorgente di informazioni;

2. mediatore: il mediatore gestisce l'interrogazione dell'utente, combinando, integrando ed eventualmente arricchendo i dati ricevuti dai wrapper, ma usando un modello (e quindi un linguaggio di interrogazione) comune a tutte le fonti;

3. wrapper: ogni wrapper gestisce una singola sorgente, convertendo le richieste del mediatore in una forma comprensibile dalla sorgente, e le informazioni da essa estratte nel modello usato dal mediatore.

Facendo riferimento ai servizi descritti nella proposta ARPA-I^3, l'architettura del mediatore che si è progettato comprente le seguenti funzionalità (si veda la Figura 1):

- servizi di Coordinamento: sul modello di facilitatori e mediatori, il sistema sarà in grado, in presenza di una interrogazione, di individuare automaticamente tutte le sorgenti che ne saranno interessate, ed eventualmente di scomporre la richiesta in un insieme di sottointerrogazioni diverse da inviare alle differenti fonti di informazione;

- servizi di Integrazione e Trasformazione Semantica: saranno forniti dal mediatore servizi che facilitino l'integrazione sia degli schemi che delle informazioni, nonchè funzionalità di supporto al processo di integrazione (come può essere la Query Decomposition);
- servizi Ausiliari: sono utilizzate tecniche di Inferenza per realizzare, all'interno del mediatore, una fase di ottimizzazione delle interrogazioni.

All'interno del sistema sono stati adottati un modello comune dei dati (ODM$_{I^3}$) ed un linguaggio di definizione comune (ODL$_{I^3}$) per descrivere l'integrazione delle informazioni. ODM$_{I^3}$ e ODL$_{I^3}$ sono definiti come sottoinsiemi dei corrispondenti modello e linguaggio nello standard ODMG-93 [36], seguendo la proposta di standard per un linguaggio di mediatore sviluppata dal gruppo di lavoro I^3/POB [26]. In piú, ODL$_{I^3}$ introduce nuovi costruttori per il supporto al processo di integrazione. MOMIS si appoggia alla logica descrittiva OLCD (Object Language with Complements allowing Descriptive cycles [6, 16]), come linguaggio interno e ad ODB-Tools [9, 7, 1] come tool di supporto per il ragionamento e l'inferenza per l'integrazione di sorgenti e l'ottimizzazione di query. D'altra parte, per supportare il processo semiautomatico di integrazione degli schemi, viene utilizzato ARTEMIS (Analysis of Requirements: Tool Environment for Multiple Information Systems) [31, 32], un tool basato sulle tecniche di clustering gerarchico.

Nel processo di integrazione viene dapprima costruito un Thesaurus comune (Common Thesaurus) di relazioni terminologiche (*terminological relationships*) basate sulla descrizione delle sorgenti in linguaggio ODL$_{I^3}$. Il thesaurus rappresenta il punto di partenza per la seconda fase, dove le tecniche di clustering sono impiegate per identificare le classi che descrivono informazioni simili in sorgenti differenti e fonderle in una descrizione globale ed integrata delle sorgenti analizzate. Gli aspetti salienti ed innovativi della nostra proposta riguardano i seguenti aspetti:

- Modellazione di schema (che chiameremo *object patterns*) per le sorgenti semistrutturate, investigando nello stesso modo che in [48]. Sono presentate e discusse le estensioni al linguaggio ODL$_{I^3}$ ed alla corrispondente DL OLCD per permettere l'introduzione degli *object patterns* nei processi di integrazione e di query processing.
- Estensione delle procedure di clustering basate sulle tecniche di affinitá per permettere il trattamento degli *object patterns.*
- Query processing e ottimizzazione usando la conoscenza racchiusa nello schema integrato risultante. In quest'ambito viene inoltre descritto il ruolo di ODB-Tools relativamente alla decomposizione ed alla ottimizzazione delle query globali.

2 L'architettura di MOMIS

MOMIS é stato progettato per fornire un accesso integrato ad informazioni eterogenee memorizzate sia in database di tipo tradizionale (e.g. relazionali, object-oriented) o file system, sia in sorgenti di tipo semistrutturato.

Seguendo l'architettura di riferimento I^3 [50], in MOMIS si possono distinguere quattro componenti principali (come si puó vedere da Fig. 2):

1. *Wrapper*: posti al di sopra di ciascuna sorgente, sono i moduli che rappresentano l'interfaccia tra il mediatore vero e proprio e le sorgenti locali di dati. La loro funzione è duplice:
    - in fase di integrazione, forniscono la descrizione delle informazioni in essa contenute. Questa descrizione viene fornita attraverso il linguaggio ODL$_{I^3}$ (descritto in Sezione 4);

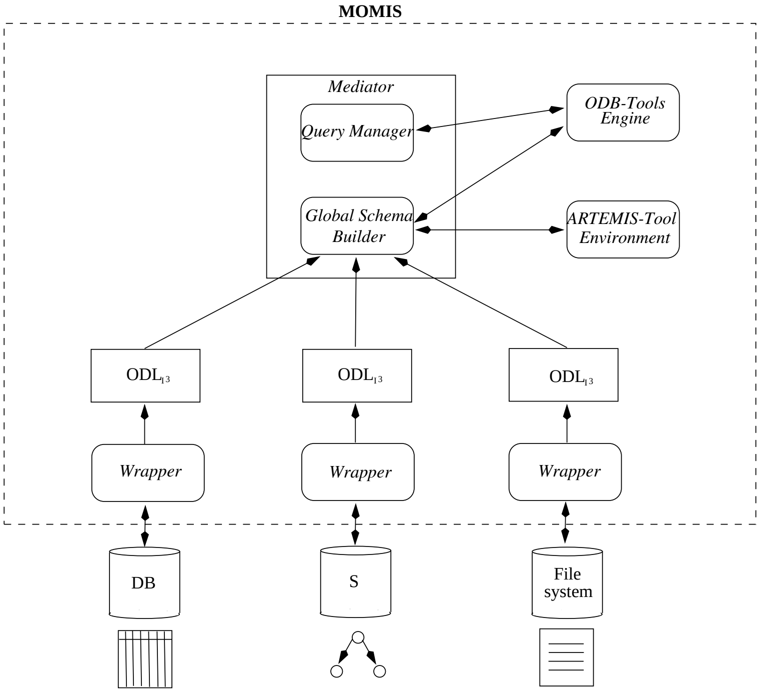

Figura 2: Architettura del sistema MOMIS

- in fase di query processing, traducono la query ricevuta dal mediatore (espressa quindi nel linguaggio comune di interrogazione OQL$_{I^3}$, che sarà definito in analogia al linguaggio OQL) in una interrogazione comprensibile (e realizzabile) dalla sorgente stessa. Devono inoltre esportare i dati ricevuti in risposta all'interrogazione, presentandoli al mediatore attraverso il modello comune di dati utilizzato dal sistema;

2. *mediatore*: è il cuore del sistema, ed è composto da due moduli distinti.

   - *Global Schema Builder (GSB)*: è il modulo di integrazione degli schemi locali, che, partendo dalle descrizioni delle sorgenti espresse attraverso il linguaggio ODL$_{I^3}$, genera un unico schema globale da presentare all'utente. Questa fase di integrazione, realizzata in modo semi-automatico con l'interazione del progettista del sistema, fa uso degli ODB-Tools (sfruttati dal modulo software **SIM**$_1$) e delle tecniche di clustering di ARTEMIS;
   - *Query Manager (QM)*: è il modulo di gestione delle interrogazioni. In particolare, genera le query in linguaggio OQL$_{I^3}$ da inviare ai wrapper partendo dalla singola query formulata dall'utente sullo schema globale. Servendosi delle tecniche di Logica Descrittiva, il QM genera automaticamente la traduzione della query sottomessa nelle corrispondenti sub-query delle singole sorgenti.

3. *ODB-Tools*, un tool basato sulla logica descrittiva OLCD [6, 16] che compie la validazione di schemi e l'ottimizzazione di query [9, 7, 1].

4. *ARTEMIS-Tool Environment*, un tool che compie analisi e clustering di schemi [29].

Il principale scopo che ci si è preposti con MOMIS è la realizzazione di un sistema di mediazione che, a differenza di molti altri progetti analizzati, contribuisca a realizzare, oltre alla fase di query processing, una reale integrazione delle sorgenti.

Processo di Integrazione. L'integrazione delle sorgenti informative strutturate e semistrutturate viene compiuta in modo semi-automatico, utilizzando le descrizioni degli schemi locali in linguaggio ODL_{I^3} e combinando le tecniche di logica descrittiva e di clustering. Le attivitá compiute sono le seguenti:

1. *Estrazione di relazioni terminologiche*, grazie al supporto di ODB-Tools. Durante questo passo viene costruito un Thesaurus Comune di *relazioni terminologiche*. Le relazioni terminologiche esprimono la conoscenza di inter-schema su sorgenti diverse e corrispondono alle asserzioni intensionali utilizzate in [35]. Le relazioni terminologiche sono derivate in modo semi-automatico a partire dalle descrizioni degli schemi in ODL_{I^3}, attraverso l'analisi strutturale e di contesto delle classi coinvolte, utilizzando ODB-Tools e le tecniche di logica descrittiva. L'estrazione delle relazioni terminologiche sará discussa in Sezione 6.
2. *Affinity-based clustering di classi ODL_{I^3}*, con il supporto dell'ambiente ARTEMIS-Tool, le relazioni terminologiche contenute nel Thesaurus vengono utilizzate per valutare il livello di affinitá tra le classi ODL_{I^3} in modo di identificare le informazioni da essere integrate a livello globale. A tal fine, ARTEMIS calcola i coefficienti che misurano il livello di affinitá delle classi ODL_{I^3} basandosi sia sui nomi che sugli attributi. Le classi ODL_{I^3} con maggiore affinitá vengono raggruppate utilizzando le tecniche di clustering [42].
3. *Costruzione dello schema globale di mediatore*, con il supporto di ODB-Tools i cluster di classi ODL_{I^3} affini sono analizzate per costruire lo schema globale del Mediatore. Per ciascun cluster viene definita una classe globale ODL_{I^3} che rappresenta tutte le classi che sono riferite al cluster, e che é caratterizzata dall'unione dei loro attributi. L'insieme delle classi globali definite costituisce lo schema globale di Mediatore che deve essere usato per porre le query alle sorgenti locali integrate. In questa fase OLCD e ODB-Tools sono utilizzati per realizzare una generazione semi-automatica delle classi globali.

Query processing e ottimizzazione. Quando l'utente pone una query sullo schema globale, MOMIS analizza la query e produce un insieme di sub-query che saranno inviate a ciascuna sorgente informativa coinvolta. In accordo ad altri approcci proposti in quest'ambito [4, 5], il processo consiste di due attivitá principali:

1. *Ottimizzazione Semantica*. Il Mediatore agisce sulla query sfruttando la tecnica di ottimizzazione semantica supportata da ODB-Tools in modo da ridurre il costo del query access plan. L'ottimizzazione é basata sull'inferenza logica a partire dalla conoscenza contenuta nei vincoli di integritá dello schema globale.
2. *Formulazione del piano di accesso*. Dopo aver ottenuto la query ottimizzata, viene generato l'insieme di sub-query relative alle sorgenti coinvolte. A questo scopo, il Mediatore utilizza l'associazione tra classi globali e locali per esprimere la query globale in termini degli schemi locali.

L'Ottimizzazione Semantica e il query plan sará descritto in Sezione 10.

Pur non essendo centrale per il nostro sistema, vogliamo descrivere il processo che rende disponibile al Mediatore gli schemi sorgenti in linguaggio ODL_{I^3}. Per quanto riguarda i database strutturati convenzionali, la descrizione dello schema é sempre disponibile e puó quindi essere tradotto automaticamente in linguaggio ODL_{I^3} da uno specifico wrapper. Diverso é il discorso delle sorgenti semistrutturate, dove, a priori, non é definito alcuno schema. Per questa ragione risulta necessaria una procedura di estrazione dello schema per le sorgenti semistrutturate (e.g., nella forma di dataguides [48]). In MOMIS introduciamo perció il concetto di *object pattern* per rappresentare a livello intensionale la struttura degli oggetti di una sorgente semistrutturata. Questo processo verrá discusso nella sezione seguente.

2.1 Semistructured data and object patterns

L'incremento nel numero e nella dimensione delle sorgenti informative che in questi anni si sono rese disponibili all'utente generico e, a volte, casuale, è stato accompagnato da un'esplosione nella varietà dei formati in cui i dati vengono rappresentati. Nella maggior parte dei casi, purtroppo, sebbene si possa riconoscere nei dati una sorta di struttura, questa è talmente irregolare da non poter essere facilmente riconducibile nè ai consolidati modelli relazionali [64] e neppure a quelli più ricchi quali quelli ad oggetti [36]. In genere si parla di questi dati in termini di dati semistrutturati (semistructured data), come confermato da diversi interventi presenti in letteratura [2, 23]. A causa della natura frammentata dei dati, l'enfasi degli interventi viene posta sull'importanza di derivare una rappresentazione coincisa (indicata brevemente come "summary") della struttura in modo tale da fornire all'utente che si trova ad interagire con le informazioni un'idea della struttura e del contesto riguardante la sorgente dei dati. Questa possibilità di giungere alla definizione di una struttura per dati semistrutturati facilita la formulazione di query e potrebbe essere usata per la fase di ottimizzazione delle query stesse.

La caratteristica di base dei dati semistrutturati risiede nella loro natura che può essere definita "auto-descrittiva" ("self-describing"). Questo significa che le informazioni, che generalmente vengono associate allo schema, sono direttamente specificate nei dati. Come detto, in letteratura vari sono i modelli proposti per la rappresentazione dei dati semistrutturati [23, 25, 69]. In accordo a questi modelli, la nostra proposta prevede di rappresentare sorgenti semistrutturate come grafi ad albero etichettati, dove i dati semistrutturati possono rappresentare sia i nodi che le etichette sugli archi. Infatti, supponiamo di utilizzare sorgenti semistrutturate conformi al modello OEM [3, 69], la cui definizione è la seguente:

Definizione 1 (Oggetto Semistrutturato) *Un* oggetto semistrutturato, *indicato con so, è una tripla nella forma* $\langle \mathtt{id}, \mathtt{label}, \mathtt{value} \rangle$ *dove:*

- `id` *è l'idenficatore dell'oggetto;*
- `label` *è una stringa che descrive ciò che l'oggetto rappresenta;*
- `value` *è il valore dell'oggetto, che può essere atomico o complesso.*

Per quanto concerne i valori atomici, essi possono essere *integer*, *real*, *string*, *image*, mentre un valore complesso è un insieme di oggetti semistrutturati, cioè un insieme di coppie (*id,label*). In analogia a questa caratterizzazione, definiamo *oggetto semistrutturato atomico* un oggetto a valore atomico, e *oggetto semistrutturato complesso* un oggetto il cui valore è complesso.

Per esempio,⟨`16, name, "Ann Red"`⟩ rappresenta un oggetto semistrutturato atomico con valore di tipo stringa, mentre l'oggetto ⟨`8, exam,` {`(24, date), (25, type), (26, outcome)`}⟩ rappresenta un oggetto semistrutturato complesso il cui valore è dato da tre oggetti atomici. L'oggetto complesso può quindi essere pensato come il padre di tutti gli oggetti che definiscono il suo valore (detti oggetti figli). In generale, un dato oggetto può avere uno o più genitori: nella nostra notazione indichiamo che un oggetto so' è un oggetto figlio di un altro oggetto so tramite $so \rightarrow so'$ e indichiamo con $label(so)$ l'etichetta di so. Perciò, intuitivamente, una sorgente semistrutturata S può essere vista come un grafo orientato, in cui i nodi rappresentano gli oggetti semistrutturati e gli archi orientati rappresentano la relazione tra un oggetto complesso e tutti gli oggetti atomici che ne individuano il suo valore.

Un esempio di sorgente semistrutturata è rappresentato in Fig. 2.1, dove sono presenti informazioni relative al Dipartimento di Cardiologia (`Cardiology Department`) di un dato ospedale.

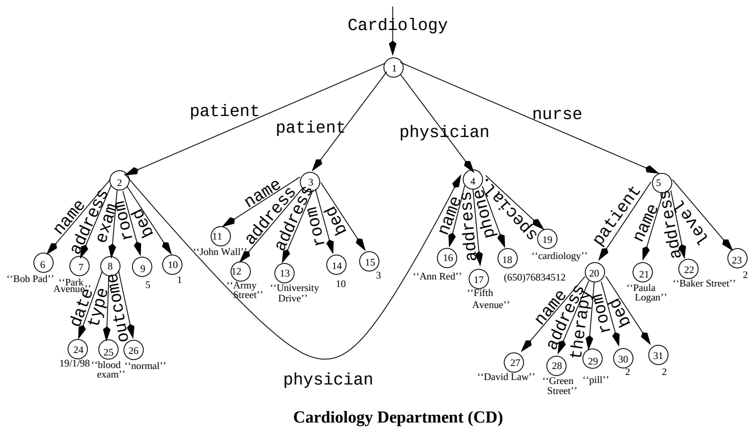

Figura 3: Cardiology Department (CD)

Nei modelli di dati semistrutturati, le etichette sono il più possibile esplicative e possono essere utilizzate per raggruppare gli oggetti assegnando la stessa etichetta ad oggetti correlati. Per questa ragione, data una sorgente semistrutturata S, vogliamo scoprire i tipi differenti di oggetti memorizzati attraverso un approccio basato sui pattern che tenga conto della semantica di mondo aperto caratteristico delle logiche descrittive. L'approccio si puó riassumere brevemente nel seguente modo: tutti gli oggetti complessi so di S vengono partizionati in base alle loro etichette e valori in insiemi disgiunti, indicati come set_l, in modo tale che tutti gli oggetti che fanno riferimento allo stesso insieme abbiano la stessa etichetta l. Successivamente ad ogni insieme associamo un *object pattern*. Formalmente, un *object pattern* é definito nel seguente modo:

Definizione 2 (Object pattern) *Poni set_l un insieme di oggetti definito sulla sorgente semistrutturata S. L'*object pattern *dell'insieme set_l é una coppia nella forma $\langle l,A \rangle$, dove l é l'etichetta degli oggetti correlati all'insieme set_l, ed $A = \bigcup$ label(so') tale che esiste almeno un oggetto $so \in set_l$ con $so \rightarrow so'$.*

Partendo da questa definizione, é facile capire come un *object pattern* sia rappresentativo di tutti i diversi oggetti che descrivono lo stesso concetto in una data sorgente semistrutturata. Piú specificatamente, l indica il concetto e A le proprietá (dette anche attributi) che caratterizzano il concetto all'interno della sorgente. Poiché gli oggetti semistrutturati possono essere eterogenei, a volte le etichette nell'insieme A di un *object pattern* possono essere definite solo per alcuni oggetti dell'insieme set_l e non per tutti: chiameremo queste particolari etichette come "optional" e verranno indicate con il simbolo "*".

In Fig. 2.1 sono indicati tutti gli *object pattern* della sorgente semistrutturata considerata come esempio. Sono stati definiti quattro *object pattern*: `Patient` che contiene informazioni riguardo ai pazienti; `Physician` e `Nurse` contenenti informazioni riguardo al personale medico; `Exam` contenente informazioni riguardo la `date`, il `type` e `outcome` di un esame clinico.

Come detto la descrizione degli *object pattern* segue la semantica di mondo aperto *open world semantics* tipica dell'approccio seguito in logica descrittiva [17, 20, 28, 68]. Gli oggetti

Patient-pattern = (`Patient`,{`name`,`address`,`exam*`,`room`,`bed`,`therapy*`,`physician*`})
Physician-pattern = (`Physician`,{`name`,`address`,`phone`,`specialization`})
Nurse-pattern = (`Nurse`,{`name`,`address`,`level`,`patient`})
Exam-pattern = (`Exam`,{`date`,`type`,`outcome`})

Figura 4: Gli object pattern della sorgente semistrutturata di esempio

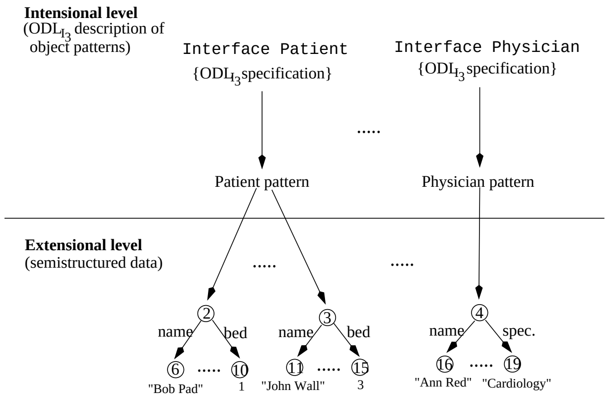

Figura 5: Livelli di astrazione degli oggetti semistrutturati

di un pattern condividono una struttura di minimo rappresentata dalle proprietá non opzionali, ma possono avere altre proprietá addizionali che caratterizzano il singolo oggetto (o un numero ridotto di oggetti) che chiamiamo optional. In questo modo, gli oggetti di una sorgente semistrutturata possono evolversi ed aggiungere nuove proprietá, pur rimanendo istanze valide dell'*object pattern* corrispondente e quindi mantenendo efficaci le query sull'*object pattern*.

Il nostro approccio si pone quindi l'obiettivo di distinguere, all'interno di una sorgente semistrutturata, la parte relativa alla struttura da quella dei dati: il livello intensionale contiene ciascuna descrizione degli *object pattern* secondo la descrizione ODL$_{I^3}$. Il livello estensionale contiene invece gli oggetti semistrutturati (vedi Fig. 5).

3 Esempio di riferimento

In questo capitolo verrà utilizzato, per meglio chiarire tutti i passaggi che vengono effettuati sia nella fase di integrazione che in quella di query processing, il seguente esempio di riferimento riguardante un dominio applicativo di un'organizzazione medica.

Si consideri un ambiente ospedaliero ed in particolare i Dipartimenti di cardiologia (`Cardiology`) e quello di terapia intensiva (`Intensive Care`), dove, ovviamente, c'é l'esigenza di condividere le informazioni riguardo i propri pazienti. Il dipartimento `Cardiology` (brevemente indicato come `CD`) contiene oggetti di tipo semistrutturato relativi a pazienti con malattie di tipo ischemico, cardiocircolatorio ed ipertensione, ed informazioni riguardo il personale medico afferente il dipartimento. In Fig. 2.1 é riportata una porzione di esempio dei dati: abbiamo un oggetto complesso radice con quattro figli anch'essi di tipo complesso, due pazienti, un medico ed una nurse. (Ovviamente nell'applicazione reale ci sono molti pazienti, medici e nurse). Ciascun `Patient` é dotato di oggetti atomici `name`, `address`, `room`, e `bed`. `Physician` contiene a sua volta quattro oggetti atomici (i.e., `name`, `address`, `phone`, e `specialization`), mentre `Nurse` contiene come valori tre oggetti atomici (i.e., `name`, `address`, e `level`) ed uno complesso (i.e., il `patient` di cui é responsabile).

Intensive Care Department (ID)

```
Patient(code, first_name, last_name, address, test, doctor_id)
Medical_Staff(id, first_name, last_name, phone, address, availability, position)
Test(number, type, date, laboratory, result)
Dis_Patient(code, date, note)
```

Figura 6: Intensive Care Department (ID)

Il dipartimento `Intensive Care` (`ID`) memorizza le proprie informazioni in un database relazionale relative a: pazienti la cui diagnosi riguarda trauma o infarto e sullo staff medico. Il database contiene quattro relazioni: `Patient`, `Medical_Staff`, `Test` e `Dis_Patient` (vedere Fig. 6). Le informazioni personali relative ai pazienti sono mantenute nella relazione `Patient`. `Dis_Patient` é un sottotipo di `Patient` e contiene informazioni riguardo ai pazienti dimessi. Ciascun "clinician", cioé il personale medico quali dottori, infermieri, farmacisti (che costituiscono la relazione `Medical_Staff`) é caratterizzato da `first_name`, `last_name`, `phone`, `address`, `availability` e `position`. La relazione `Test` é usata per mantenere informazioni riguardanti tutti gli esami dei pazienti.

4 Il linguaggio ODL$_{I^3}$

Allo scopo di facilitare la comunicazione tra i wrapper ed il Mediatore, introduciamo un linguaggio di definizione di dati di alto livello, chiamato ODL$_{I^3}$.

In accordo alle raccomandazioni della proposta di standardizzazione per linguaggi di mediazione [27], secondo la quale i diversi sistemi di mediazione avrebbero potuto supportare sorgenti con modelli complessi, come quelli ad oggetti, e sorgenti molto più semplici, come file di strutture, ed al diffondersi del modello di dati ad oggetti (e il suo standard ODMG-93), ODL$_{I^3}$ é una estensione al linguaggio standard ODL che supporta le necessitá del nostro sistema di integrazione intelligente di informazioni.

Principali caratteristiche del linguaggio sono:

- supporto di sorgenti strutturate (database relazionali, ad oggetti, e file system) e semistrutturate;
- descrizione di ciascuna fonte di informazioni secondo il modello comune dei dati, e quindi utilizzare il concetto di *classe*, indipendentemente dal modello originale utilizzato. Sarà compito del wrapper provvedere alla traduzione dal modello originale di descrizione all'ODL$_{I^3}$, aggiungendo eventualmente in questa descrizione tutte le informazioni necessarie al mediatore (il nome e il tipo della sorgente,...);
- dichiarazione di regole di integrità (*if then rule*), definite sia sugli schemi locali (e magari da questi ricevute), che riferite allo schema globale, e quindi inserite dal progettista del mediatore;
- dichiarazione di regole di mediazione, o *mapping rule*, utilizzate per specificare l'accoppiamento tra i concetti globali e i concetti locali originali;
- utilizzo della *semantica di mondo aperto*, che permette alle classi descritte di cambiare formato (magari aggiungendo attributi agli oggetti) senza necessariamente cambiarne la descrizione;

- traduzione automatica e trasparente all'utente delle descrizioni nella logica descrittiva OLCD, con conseguente possibilitá di utilizzarne le capacità nei controlli di consistenza e nell'ottimizzazione semantica delle interrogazioni;
- introduzione dell'operatore di *unione* (`union`), che permette l'espressione di strutture dati in alternativa nella definizione di una classe;
- introduzione del costruttore *optional* (*), specificato dopo il nome di un attributo per indicare la sua natura opzionale;
- dichiarazione di relazioni terminologiche, che permettono di specificare relazioni di sinonimia (SYN), ipernimia (BT), iponomia (NT) e relazione associativa (RT) tra due tipi.

Utilizzando questo linguaggio, il wrapper compie la traduzione delle classi da integrare e le fornisce al mediatore: da sottolineare che le descrizioni ricevute rappresentano tutte e sole le classi che una determinata sorgente vuole mettere a disposizione del sistema, e quindi interrogabili. Non è quindi detto che lo schema locale ricevuto dal mediatore rappresenti l'intera sorgente, bensì ne descrive il sottoinsieme di informazioni visibili da un utente del mediatore, esterno quindi alla sorgente stessa.
La sintassi del linguaggio ODL_{I^3}è riportata, in BNF, nell'Appendice A.

Come detto, object pattern e schemi sorgenti sono tradotti in descrizioni ODL_{I^3}: per gli object pattern la traduzione é diretta. In particolare, dato un a pattern $\langle l, A\rangle$ o una relazione di una sorgente relazionale abbiamo che: i) un nome di classe ODL_{I^3} corrisponde a l o al nome della relazione, e ii) per ciascuna etichetta $l' \in A$ o attributo della relazione, viene definito un attributo nella corrispondente classe ODL_{I^3}. Come esempio mostriamo le descrizioni delle classi `CD.Patient` e `ID.Medical_Staff`, rimandando invece all'Appendice B per l'intera descrizione dell'esempio di riferimento di Sezione 3:

```
interface Patient

 ( source semistructured Cardiology_Department )
{  attribute string    name;
   attribute string    address;
   attribute set<Exam> exam*;
   attribute integer   room;
   attribute integer   bed;
   attribute string    therapy*;
   attribute set<Physician> physician*;
   };

interface Medical_Staff
 ( source relational Intensive_Care
   extent Medical_Staffers
   key id )
{  attribute string  id;
   attribute string  first_name;
   attribute string  last_name;
   attribute integer phone;
   attribute string  address;
   attribute string  availability;
```

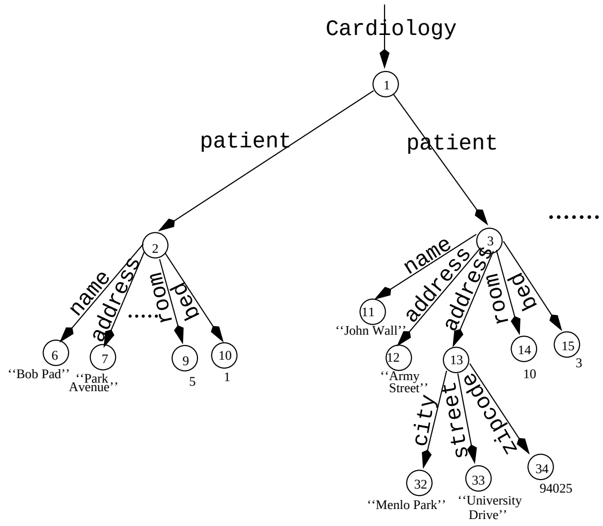

Figura 7: Un esempio di oggetti fortemente eterogenei

```
   attribute string  position; };
```

Un caso particolare avviene quando un oggetto "fortemente" eterogeneo é definito nella sorgente semistrutturata. In riferimento alla nostra sorgente semistrutturata CD di Fig. 2.1, consideriamo l'oggetto:
$so = \langle \mathtt{2}, \mathtt{patient}, \{(\mathtt{6}, \mathtt{name}), (\mathtt{7}, \mathtt{address}), (\mathtt{8}, \mathtt{exam}), (\mathtt{9}, \mathtt{room}), (\mathtt{10}, \mathtt{bed})\}\rangle$.

Supponiamo che l'oggetto relativo al paziente `3` sia presente avendo un oggetto `address` non-atomico, cioé che `address` contenga come valori tre oggetti atomici (`city, street, zipcode`) (vedere Fig. 7). In questo caso, l'estrazione del pattern dovrebbe essere leggermente modificata. In particolare viene estratto un nuovo pattern per address, `address-pattern` = $(\mathtt{address}, \{\mathtt{city}, \mathtt{street}, \mathtt{zipcode}\})$. Per comprendere e gestire anche questo tipo di eterogeneitá, in ODL_{I^3} definiamo il costruttore **union** per le classi. La definizione in ODL_{I^3} dei pattern `Patient` e `address` é mostrata Fig. 8.

```
interface Address
 ( source semistructured Cardiology_Department )
{  attribute string city;
   attribute string street;
   attribute string zipcode; };
 union
{  string; };

interface Patient
 ( source semistructured Cardiology_Department )
{  attribute string   name;
   attribute Address  address;
   .............................. };
```

Figura 8: Un esempio di tipo unione

La semantica del costruttore di union e di optional viene descritta nella prossima sezione, utilizzando la logica descrittiva OLCD.

5 Estensioni al formalismo OLCD per l'integrazione

Il formalismo OLCD é stato introdotto in [16, 6]. L'estensione principale ad OLCD che proponiamo riguarda il costruttore di *union* ($\sqcup$). L'operatore *union* ($\sqcup$) puó essere usato per rappresentare la semantica dell'operatore di `union` del linguaggio ODL$_{I^3}$. É stato formalizzato in [6], con il significato di unione di tutte le possibili istanze di un tipo.

Per esempio, il pattern `Address` di Fig. 8 viene tradotto in OLCD come segue:

$$\sigma_V(\texttt{Address}) \quad = \quad [\ \texttt{city} : \texttt{String}, \texttt{street} : \texttt{String}, \texttt{zipcode} : \texttt{String}\ ]$$

L'operatore di *union* ($\sqcup$) é anche utile per definire nel nostro formalismo il concetto di attributo opzionale. Infatti un attributo opzionale specifica che il dato per una specifica istanza puó esistere oppure essere assente. In OLCD descriviamo questo concetto come union tra il dominio dell'attributo e il tipo *bottom* ($\perp$).

Nel nostro esempio il pattern `Patient` puó essere rappresentato come segue: [1]:

$$\begin{array}{lll} \sigma_P(\texttt{Patient}) & = \triangle\ [& \texttt{name} : \texttt{String}, \texttt{address} : \texttt{String}, \texttt{exam} : \{\texttt{Exam}\}, \\ & & \texttt{room} : \texttt{Integer}, \texttt{bed} : \texttt{Integer}, \\ & & \texttt{therapy} : \texttt{String} \sqcup \perp\ , \texttt{physician} : \{\texttt{Physician}\}\ ] \end{array}$$

OLCDfornisce, utilizzando la semantica dei tipi virtuali e del *type as set* per la descrizione dei tipi, tecniche di ragionamento rilevanti: calcolo della relazione di *subsumption* tra tipi (i.e. relazioni "isa" implicata dalla descrizione dei tipi), decidere l'*equivalence* tra tipi e rilevazione di tipi *incoerenti* (i.e., sempre empty).

Per comprendere l'utilizzo della subsumption nel contesto di attributi optional, supponiamo di considerare i tipi valore `A` e `B`,

$$\sigma_V(\texttt{A}) \quad = \quad [\ \texttt{att1} : \texttt{String}, \texttt{att2} : \texttt{String}\ ]$$
$$\sigma_V(\texttt{B}) \quad = \quad [\ \texttt{att1} : \texttt{String}, \texttt{att2} : \texttt{String} \sqcup \perp\ ]$$

Calcolando la subsumption tra i tipi `A` e `B` otteniamo che `B` subsume `A` (A isa B) anche se non é stato dichiarato esplicitamente.

6 Estrazione di Relazioni Terminologiche

Lo scopo di questa fase è la costruzione di un Thesaurus di *relazioni terminologiche* che rappresenti la conoscenza a disposizione sulle classi da integrare (ovvero sui nomi delle classi, sugli attributi) e che sarà la base per il calcolo di affinità tra le classi stesse. Definiamo quindi un modello di rappresentazione delle classi. Sia $S = \{S_1, S_2, \ldots, S_N\}$ un insieme di schemi di N sorgenti eterogenee che devono essere integrate. Come richiesto dall'ODL$_{I^3}$, ogni schema sorgente S_i è composto da un insieme di *classi*: una classe $c_{ji} \in S_i$ è caratterizzata da un nome e da un insieme di attributi, $c_{ji} = \langle n_{c_{ji}}, A(c_{ji})\rangle$. A sua volta, ogni attributo $a_h \in A(c_{ji})$, con $h = 1, \ldots, n$, è definito da una coppia $a_h = \langle n_h, d_h\rangle$, dove n_h è il nome e d_h è il dominio associato ad a_h. Si è inoltre ipotizzato che, per identificare in modo univoco all'interno del mediatore un nome (sia esso di attributo, sia esso di classe), sia rispettivamente necessaria la coppia *nome_sorgente,nome classe* e *nome_sorgente,nome_attributo*. Si parlerà inoltre genericamente di *termine* t_i indicando con esso un nome di classe o di attributo.

Le relazioni che si possono definire all'interno del Thesaurus sono le seguenti:

- SYN (synonym-of): definita tra due termini t_i e t_j, con $t_i \neq t_j$, che sono considerati sinonimi, ovvero che possono essere interscambiati nelle sorgenti, identificando lo stesso

[1] σ_P e σ_V introducono il tipo primitivo e virtuale rispettivamente

concetto del mondo reale. Un esempio di relazione SYN nel nostro esempio è ⟨`ID.Test` SYN `CD.Exam`⟩. SYN é simmetrica, cioé, t_i SYN $t_j \Rightarrow t_j$ SYN t_i.

- BT (broader-term): definita tra due termini t_i e t_j tali che t_i ha un significato più ampio, più generale di t_j. Un caso di BT, nel nostro esempio, può essere ⟨`ID.Medical_Staff` BT `CD.Nurse`⟩.

- NT (narrower-term): concettualmente è la stessa relazione espressa con una BT, intesa dall'altro punto di vista, dunque t_i BT $t_j \rightarrow t_j$ NT t_i. Lo stesso esempio potrebbe infatti essere ⟨`ID.Medical_Staff` NT `CD.Nurse`⟩.

- RT (related-term): definita tra due termini t_i e t_j che sono generalmente usati nello stesso contesto, tra i quali esiste comunque un legame generico. Per esempio, possiamo avere la seguente ⟨`CD.Patient` RT `CD.Exam`⟩. La relazione é simmetrica.

La scoperta di relazioni terminologiche presenti all'interno degli schemi è un processo semi-automatico, caratterizzato dalla interazione tra il progettista del sistema e gli ODB-Tools. Lo sforzo fatto in questa fase è stato diretto a limitare il più possibile l'intervento dell'operatore, al fine di aumentare la porzione di definizione del Thesaurus realizzabile in modo realmente automatico. L'intero processo che porta, partendo dalle descrizioni degli schemi in ODL$_{I^3}$, alla definizione del un Thesaurus comune si articola in quattro passi.

1. **Estrazione automatica di relazioni dagli schemi sorgenti.** Sfruttando le informazioni semantiche presenti negli schemi strutturati (sia basati sui modelli ad oggetti, sia relazionali) può essere identificato in modo automatico un insieme di relazioni terminologiche. In particolare durante questa preliminare fase di analisi, si puó automaticamente estrarre:

   (a) **schemi ad oggetti**:
   - relazioni BT e NT derivate dalle gerarchie di generalizzazione;
   - relazioni RT derivate dalle gerarchie di aggregazione;

   (b) **schemi relazionali**:
   - relazioni BT e NT derivate dalle gerarchie di generalizzazione (foreign key definita su primary key in entrambe le relazioni: `ID.Dis_Patient` and `ID.Patient`);
   - relazioni RT derivate dalle foreign key.

   Esempio 1 Considerando l'esempio di riferimento, le relazioni terminologiche automaticamente estratte sono le seguenti:

   ⟨`ID.Patient` RT `ID.Medical_Staff`⟩
   ⟨`ID.Patient` RT `ID.Test`⟩
   ⟨`CD.Patient` RT `CD.Physician`⟩
   ⟨`CD.Patient` RT `CD.Exam`⟩
   ⟨`ID.Patient` BT `ID.Dis_Patient`⟩
   ⟨`CD.Nurse` RT `CD.Patient`⟩

2. **Revisione/Integrazione delle relazioni.** Interagendo con il modulo, il progettista deve inserire tutte le relazioni terminologiche che non sono state estratte nel passo precedente, ma che devono comunque essere presenti per pervenire ad una esatta integrazione delle sorgenti. In particolare, possono essere inserite relazioni che coinvolgono classi appartenenti

a schemi diversi, come pure relazioni che si riferiscono a nomi di attributi. Da sottolineare che, durante questa fase, tutte le relazioni inserite dal progettista hanno carattere esclusivamente *terminologico*, escludendo quindi qualunque considerazione sulle estensioni. È quindi una fase che, nei prossimi sviluppi del progetto, potrebbe facilmente essere coadiuvata dall'uso di un dizionario che evidenzi eventuali termini sinonimi, o correlati tra loro. Un approccio potrebbe essere quello di rivolgersi a sistemi lessicali (vedere, per esempio WordNet [47, 55]), che prevedono relazioni terminologiche tra termini a priori.

In generale, relazioni terminologiche esplicite possono correlare classi ODL_{I^3} le cui descrizioni presentano conflitti semantici rispetto alle relazioni di subsumption/generalizzazione ed equivalenza, poiché queste ultime specificano relazione di supertipo/sottotipo e di uguaglianza tra tipi delle classi coinvolte.

Per esempio, supponiamo che una relazione di SYN sia definita tra due classi aventi la stessa struttura (per esempio `ID.Test` e `CD.Exam`). Per rendere compatibile questa relazione terminologica con una relazione di equivalenza per ODB-Tools é necessario uniformare la descrizione di entrambe le classi. Il risultato della modifica per le classi esaminate diventa:

```
interface Test
( ... )
{ attribute string  type;
  attribute integer date;
  attribute string  laboratory;
  attribute string  result;
  attribute string  outcome; };

interface Exam ( ... )
{ attribute integer date;
  attribute string  type;
  attribute string  outcome;
  attribute string  laboratory;
  attribute string  result; };
```

Lo stesso tipo di problematica si evidenzia per la relazione di BT tra `ID.Patient` e `ID.Dis_Patient`. Tradurre questa relazione terminologica nella corrispondente relazione di subsumption implica la modifica delle descrizioni nel seguente modo:

```
interface Patient
(...)
{ attribute string  code;
  attribute string  first_name;
  attribute string  last_name;
  attribute string  address;
  attribute string  test;
  attribute string  doctor_id; };

interface Dis_Patient
( ...  )
{ attribute string  code;
  attribute integer date;
```

```
  attribute string  note;
  attribute string  position;
  attribute string  first_name;
  attribute string  last_name;
  attribute string  address;
  attribute string  test;
  attribute string doctor_id;
};
```

Infine quando viene asserita una nuova relazione di RT, ad esempio tra `CD.Patient` e `ID.Test`, un nuovo attributo di aggregazione viene inserito nel seguente modo:

```
interface Patient
( ... )
{  attribute string          name;
   attribute string          address;
   attribute set<Exam>       exam*;
   attribute integer         room;
   attribute integer         bed;
   attribute string          therapy*;
   attribute set<Physician>  physician*;
   attribute Test            newRT1};

interface Test
( ... )
{  attribute string  type;
   attribute integer date;
   attribute string  laboratory;
   attribute string  result;
};
```

Queste trasformazioni conducono alla generazione automatica di uno "schema virtuale" contenente, per ciascuno schema locale, le descrizioni modificate delle classi; tale schema viene utilizzato da ODB-Tools per eseguire ulteriori inferenze di relazioni per arricchire il Thesaurus.

Esempio 1 Nel nostro esempio, supponiamo che il progettista inserisca le seguenti relazioni terminologiche relative sia a classi che ad attributi:

⟨`ID.Medical_Staff` BT `CD.Physician`⟩
⟨`ID.Medical_Staff` BT `CD.Nurse`⟩
⟨`ID.Test` SYN `CD.Exam`⟩
⟨`ID.Patient` SYN `CD.Patient`⟩
⟨`CD.Patient.physician` BT `ID.Patient.doctor_id`⟩
⟨`CD.Patient.name` BT `ID.Patient.first_name`⟩
⟨`CD.Patient.name` BT `ID.Patient.last_name`⟩
⟨`CD.Nurse.level` SYN `ID.Medical_Staff.position`⟩
⟨`CD.Exam.outcome` SYN `ID.Test.result`⟩

3. **Validazione delle relazioni.** In questa fase, ODB-Tools viene utilizzato per validare le relazioni terminologiche del Thesaurus definite tra due attributi. La validazione è basata sul controllo di compatibilità dei domini associati agli attributi. In questo modo le relazioni terminologiche vengono distinte in *valide* e *invalide*. In particolare, dati i due attributi $a_t = \langle n_t, d_t \rangle$ e $a_q = \langle n_q, d_q \rangle$ coinvolti in una relazione, verranno eseguiti i seguenti controlli:

   - $\langle n_t$ SYN $n_q \rangle$: la relazione è considerata valida se d_t e d_q sono equivalenti, oppure uno è la specializzzazione dell'altro;
   - $\langle n_t$ BT $n_q \rangle$: la relazione è considerata valida se d_t contiene od è equivalente a d_q;
   - $\langle n_t$ NT $n_q \rangle$: la relazione è considerata valida se d_t è contenuto in, od è equivalente a d_q;

   Nel caso in cui il dominio dell'attributo contiene il connettore di unione, una relazione viene considerata valida quando almeno un dominio risulta compatibile. Questa fase di validazione è compiuta utilizzando ODB-Tools per il calcolo della relazione di subsumption e equivalence.

   Esempio 2 Con riferimento al Thesaurus definito nell'esempio 1, riportiamo l'output della fase di validazione: per ciascuna relazione il flag di controllo [1] indica una relazione valida, mentre il flag [0] una invalida.

⟨`CD.Patient.physician` BT `ID.Patient.doctor_id`⟩	[0]
⟨`CD.Patient.name` BT `ID.Patient.first_name`⟩	[1]
⟨`CD.Patient.name` BT `ID.Patient.last_name`⟩	[1]
⟨`CD.Nurse.level` SYN `ID.Medical_Staff.position`⟩	[0]
⟨`CD.Exam.outcome` SYN `ID.Test.result`⟩	[1]

4. **Inferenza di nuove relazioni.** In questa fase vengono inferite nuove relazioni terminologiche utilizzando lo "schema virtuale" definito nella fase di revisione/integrazione e calcolando le nuove relazioni di subsumption e aggregazione. Queste nuove relazioni semantiche vengono tradotte nelle corrispondenti relazioni terminologiche arricchendo il Thesaurus che ora contiene sia le relazioni esplicite che quelle inferite. Il risultato ottenuto viene chiamato Common Thesaurus.

   Esempio 3 Rispetto al nostro esempio, le relazioni terminologiche inferite sono:

   ⟨`ID.Patient` RT `CD.Physician`⟩
   ⟨`ID.Patient` RT `CD.Exam`⟩
   ⟨`CD.Patient` RT `ID.Medical_Staff`⟩
   ⟨`CD.Patient` RT `ID.Test`⟩
   ⟨`ID.Dis_Patient` RT `ID.Medical_Staff`⟩
   ⟨`ID.Dis_Patient` RT `CD.Exam`⟩
   ⟨`ID.Dis_Patient` RT `ID.Test`⟩
   ⟨`ID.Dis_Patient` RT `ID.Medical_Staff`⟩
   ⟨`ID.Dis_Patient` RT `CD.Physician`⟩

   La rappresentazione grafica del Common Thesaurus relativo ai Dipartimenti `Cardiology` and `Intensive Care` è riportata in Fig. 9, dove le linee continue rappresentano le relazioni esplicite/estratte, le linee tratteggiate quelle inferite, e le etichette indicano il tipo di relazione terminologica.[2]

[2] Per semplità di rappresentazione sono riportate solamente le relazioni tra nomi di classi.

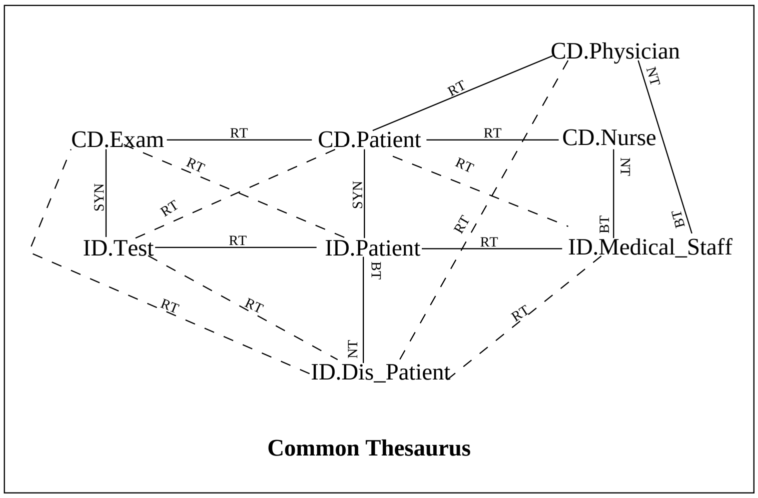

Figura 9: Common Thesaurus relativo ai Dipartimenti `Cardiology` e `Intensive Care`

7 Analisi di Affinità delle classi ODL$_{I^3}$

Per realizzare l'integrazione degli schemi ODL$_{I^3}$ delle differenti sorgenti in uno schema globale, abbiamo bisogno di tecniche per l'identificazione delle classi che descrivono le stesse informazioni (o informazioni semanticamente equivalenti), e che sono localizzate all'interno di sorgenti diverse. A questo scopo, le classi ODL$_{I^3}$ sono analizzate e raffrontate attraverso il concetto di *affinità*, che ci permette di determinare il livello di *similarità* tra classi.

Questa attività è compiuta con il supporto dell'ambiente di ARTEMIS. ARTEMIS è stato costruito per l'integrazione semi-automatica di database eterogenei di tipo strutturato [34]. Nell'ambito del progetto MOMIS, le potenzialità di ARTEMIS sono state estese per permettere l'analisi di affinità di descrizioni espresse in linguaggio ODL$_{I^3}$ e gestire i dati semistrutturati.

Le classi ODL$_{I^3}$ sono analizzate e valutate attraverso un *affinity coefficient* che permette di determinare il livello di similarità tra classi contenute in sorgenti diverse. In particolare, per le classi vengono analizzate le relazioni che esistono tra i loro nomi (attraverso il *Name Affinity Coefficient*) e tra i loro attributi (per mezzo dello *Structural affinity Coefficient*), per arrivare ad un valore globale denominato *Global Affinity Coefficient*.

La valutazione dei coefficienti di affinità si basa sulle relazioni terminologiche memorizzate nel Thesaurus. A questo scopo, il Thesaurus viene organizzato in una struttura simile alle Associative Networks [44], dove i nodi (ciascuno dei quali rappresenta genericamente un termine, sia esso il nome di una classe o il nome di un attributo) sono uniti attraverso relazioni terminologiche. A loro volta, tutte le relazioni presenti in questa rete sono percorribili in entrambi i sensi (dunque anche le BT e NT): due termini sono quindi affini se esiste un percorso che li unisce, formato da qualsivoglia relazioni. Per dare una valutazione numerica della affinità tra due termini, a ogni tipo di relazione viene associato un peso (denominato *strength* e denotato da $\sigma_{\Re}$), che sarà tanto maggiore quanto più questo tipo di relazione contribuisce a legare due termini (sarà quindi $\sigma_{syn} \geq \sigma_{bt/nt} \geq \sigma_{rt}$). In questa sezione si userà $\sigma_{ij_{\Re}}$ per denotare il peso della relazione terminologica $\Re$ definita tra i termini t_i e t_j. Nel nostro esempio, e nelle sperimentazioni precedentemente realizzate presso l'Università di Milano, si è adottato $\sigma_{syn} = 1$, $\sigma_{bt} = \sigma_{nt} = 0.8$ e $\sigma_{rt} = 0.5$.

Definizione 3 (Affinity Function) Presi due termini, t_i e t_j, possono essere presenti nel Thesaurus zero o più cammini che li uniscono, formati da relazioni. Ad ognuno di questi cammini corrisponde naturalmente un valore, dato dal prodotto dei pesi delle relazioni in esso coinvolte. La Funzione di Affinità $A_{thes}(t_i,t_j)$ tra due termini, t_i e t_j, restituisce il valore maggiore tra questi, corrispondente al cammino più *stringente*, che unisce questi termini (che non sempre coincide col cammino più breve), definito come segue:

$$A_{thes}(t_i,t_j) = \begin{cases} 1 & \text{se } t_i = t_j \\ \sigma_{i1_{\Re}} \cdot \sigma_{12_{\Re}} \cdot \ldots \cdot \sigma_{(k-1)j_{\Re}} & \text{se } t_i \rightarrow^k t_j \\ 0 & \text{in tutti gli altri casi} \end{cases}$$

dove la notazione $t_i \rightarrow^k t_j$ denota appunto il più *stringente* tra questi cammini di lunghezza k, con $k \geq 1$, tra t_i e t_j nel Thesaurus.

Il livello di Affinità tra termini dipende quindi dalla lunghezza del cammino che li unisce, ma pure dal tipo delle relazioni coinvolte in questo cammino (e quindi dal loro peso). Per ogni coppia di termini, sarà necessariamente $A_{thes} \in [0,1]$. La Affinità tra due temini sarà 0 se non esiste alcun cammino che li unisce, 1 se i due termini coincidono.

Esempio 2 Si consideri il Thesaurus illustrato in Figura 9. Esiste un cammino tra le classi `CD.Physician` e `ID.Patient`:

$$\texttt{CD.Physician} \rightarrow^{rt} \texttt{CD.Patient} \rightarrow^{syn} \texttt{ID.Patient}$$

Da questo si deduce che $A_{thes}(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5 \cdot 1 = 0.5$.

Definizione 4 (Termini Affini) Due termini t_i, t_j si dicono *affini*, e si denotano con $t_i \sim t_j$, se la loro Funzione dei Affinità restituisce un valore maggiore o uguale ad un predefinito valore di soglia $\alpha > 0$, cioè:

$$t_i \sim t_j \leftrightarrow A_{thes}(t_i,t_j) \geq \alpha$$

Per esempio, si supponga $\alpha = 0.4$.
In questo caso, dal momento che $A_{thes}(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5$, possiamo concludere che $\texttt{CD.Physician} \sim \texttt{ID.Patient}$.

7.1 Coefficienti di Affinità

In questo paragrafo, vengono date le definizioni dei coefficienti *Name Affinity Coefficient*, *Structural Affinity Coefficient* e *Global Affinity Coefficient* facendo riferimento a due classi ODL_{I^3} c e c' appartenenti rispettivamente alle sorgenti S e S'.

Definizione 5 (Name Affinity Coefficient) Misura la affinità di due classi calcolata rispetto ai loro nomi. Il *Name Affinity Coefficient* di due classi c e c' denotato da $NA(c,c')$, è la misura della affinità tra i loro nomi, n_c e $n_{c'}$, calcolata come segue:

$$NA(c,c') = \begin{cases} A_{thes}(n_c, n_{c'}) & \text{se } n_c \sim n_{c'} \\ 0 & \text{in tutti gli altri casi} \end{cases}$$

Esempio 3 Si considerino le classi `CD.Physician` e `ID.Patient`. Si avrà che $NA(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5$

Definizione 6 (Structural Affinity coefficient) Lo Structural Affinity Coefficient di due classi c e c', scritto $SA(c,c')$, è la misura dell'affinità dei loro attributi, calcolata come segue:

$$SA(c,c') = \frac{2\cdot \mid \{(a_t,a_q) \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\} \mid}{\mid A(c) \mid + \mid A(c') \mid} \cdot F_c$$

$$\begin{aligned} F_c &= \tfrac{|\{x \in C| \ flag(x)=1 \ \}|}{|C|} \\ \mathrm{C} &= \{(a_t,a_q) \mid a_t \in A(c), a_q \in A(c'),\, n_t \sim n_q\} \end{aligned}$$

dove C è l'insieme delle coppie di attributi validabili (ovvero delle coppie coinvolte in relazioni che possono essere validate attraverso un controllo sui domini) e $flag(x) = 1$ sta per un risultato positivo della suddetta validazione.

Lo Structural Affinity Coefficient è valutato utilizzando la funzione di Dice [30], e raffinato da un fattore di controllo F_c; essa restituisce un valore compreso nell'intervallo [0,1] proporzionale al numero di attributi *affini* tra le classi considerate.
Rispetto alla definizione originaria di SA precedentemente sviluppata (si veda [31]), è stata proposta in questa tesi l'estensione realizzata dal fattore di controllo F_c. Il termine F_c realizza un controllo sui domini degli attributi coinvolti nella relazione da esaminare (il controllo è quello esposto nel terzo passo della Sezione 6), permettendo quindi di non limitare la computazione di questo coefficiente alla sola analisi dei nomi che identificano gli attributi (analisi terminologica) e di estenderla alla considerazione dei domini che caratterizzano questi attributi. In pratica, una relazione che coinvolge attributi viene pesata in modo maggiore o minore nel calcolo del coefficiente a seconda che questa relazione trovi o meno riscontro anche nei tipi dei domini, e non solo nei nomi degli attributi. Il termine F_c va quindi a rifinire il coefficiente SA, moltiplicando la prima parte di questo per un termine compreso tra 0 e 1: in particolare, F_c è il rapporto tra numero di relazioni validabili memorizzate nel Thesaurus tra attributi delle due classi, e numero di queste relazioni che sono state validate.
In questo modo, maggiore sarà il numero di attributi affini tra le due classi, e maggiore il numero di controlli positivi, più alto risulterà il valore dello *Structural Affinity Coefficient*.

Per la valutazione della Structural Affinity, gli attributi *optional* delle classi ODL_{I^3} che rappresentano degli object patterns debbono essere analizzati in modo specifico ed appropriato. A seconda della tipologia di attributi considerati, sono possibili diverse opzioni per il calcolo del coefficiente $SA()$:

1. *All attribute-based.* Con questa opzione, gli attributi *optional* sono trattati alla stregua degli altri e sono sempre presi in considerazione nella valutazione dell'affinità tra classi ODL_{I^3} che descrivono object patterns.

2. *Common attribute-based.* Con questa opzione, gli attributi *optional* non vengono considerati nella valutazione dell'affinità.

3. *Threshold-based.* Con questa opzione, gli attributi *optional* vengono considerati nella valutazione dell'affinità solamente nel caso in cui sono comuni ad una percentuale prefissata di oggetti contenuti negli object pattern.

La terza opzione è la più complessa da applicare, in quanto richiede di fissare un valore di soglia che molto spesso dipende dal singolo object pattern considerato o dalla sorgente. Ciascuna opzione modifica il valore di affinità risultante: a parità di coppie di attributi affini, la seconda opzione fornisce un valore di affinità maggiore rispetto alla prima opzione (infatti, scegliendo l'opzione 2, nel denominatore della formula del $SA()$ sono considerati un numero inferiore di attributi). D'altra parte, se la maggior parte degli attributi di una classe sono opzionali allora la scelta dell'opzione 1 risulta migliore per il calcolo dell'affinità. In definitiva, la scelta tra le prime due opzioni è legata al singolo dominio analizzato: nel nostro esempio applicheremo entrambe le opzioni discutendo i valori ottenuti.

Esempio 4 Consideriamo le classi `CD.Physician` e `ID.Patient`. Applicando l'opzione *all attribute-based*, abbiamo che

$$SA(\texttt{CD.Physician}, \texttt{ID.Patient}) = \frac{2 \cdot 2}{4 + 6} = 0.4$$

dovuti alle seguenti affinità: `CD.name`$\sim${`ID.first_name`,`ID.last_name`} e `CD.address`$\sim$`ID.address`.

Definizione 7 (Global Affinity Coefficient) Il Global Affinity Coefficient di due classi c e c', denotato da $GA(c, c')$, è la misura della loro affinità calcolata come la somma pesata degli *Name Affinity Coefficient* e *Structural Affinity Coefficient*:

$$GA(c, c') = \begin{cases} w_{NA} \cdot NA(c, c') + w_{SA} \cdot SA(c, c') & \text{se } NA(c, c') \neq 0 \\ 0 & \text{in tutti gli altri casi} \end{cases}$$

dove i pesi w_{NA} e w_{SA}, con $w_{NA}, w_{SA} \in [0, 1]$ e $w_{NA} + w_{SA} = 1$, sono stati introdotti per dare al progettista la possibilità di variare caso per caso l'importanza dovuta ad ognuno dei due coefficienti rispetto all'altro. Nel nostro esempio di riferimento, abbiamo considerato ugualmente rilevanti ai fini dell'integrazione i due coefficienti, ponendo quindi $w_{NA} = w_{SA} = 0.5$.
Durante il calcolo di questo coefficiente globale, è comunque data implicitamente una maggiore rilevanza al *Name Affinity coefficient*, e quindi ai nomi delle classi stesse: per classi i cui nomi non hanno nulla in comune non è neppure valutata la affinità rispetto ai loro attributi, e conseguentemente il loro GA risulterà nullo.

Esempio 5 Considerando i Name e Structural Affinity coefficients dell'Esempio 3, ed Esempio 4, rispettivavemte, il Global Affinity coefficient tra le classi `CD.Physician` e `ID.Patient` è calcolato nel seguente modo:

$$GA(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5 \cdot 0.5 + 0.5 \cdot 0.4 = 0.45$$

7.2 Considerazioni sul processo di affinità

La correttezza del processo di affinità dipende sia dall'affidabilità delle relazioni terminologiche, sia dai parametri (i.e., strengths, weights, thresholds) che intervengono nel calcolo dei coefficienti. Come si può facilmente intuire, il processo di affinità contiene intrinsecamente una parte di soggettività, dovuta al fatto che la conoscenza specifica e l'esperienza del progettista giocano un ruolo fondamentale nella costruzione del Common Thesaurus e nella scelta dei parametri. Per rendere la fase di affinità più obiettiva, in MOMIS abbiamo introdotto funzionalità di tipo interattivo. In particolare, la costruzione e la validazione del Common Thesaurus in ODB-Tools è un processo interattivo, ed il progettista può aggiungere terminological relationships addizionali tipiche del dominio applicativo in esame. ODB-Tools valida tutte le relazioni contenute nel Thesaurus e ne inferisce di nuove, mantenendo il Common Thesaurus consistente e corretto. Il processo di valutazione dell'affinità in ARTEMIS è pure interattivo e basato sui pesi (modificabili), per permettere al designer di modificare in modo appropriato i parametri e validare le scelte

Procedura di Hierarchical Clustering
/* **Input:** K classi da analizzare */

1. Calcola tutte le coppie di Global Affinity coefficients $GA(c, c')$.
2. Crea un cluster per ogni classe.
3. **Repeat**
   Scegli la coppia c_h, c_k di cluster correnti
   con i coefficienti di affinita' maggiori in M, $M[h,k] = max_{i,j} M[i,j]$
   Forma un nuovo cluster unendo c_h, c_k;
   Aggiorna M cancellando le righe e le colonne corrispondenti a c_h e c_k;
   Definisci una nuova riga e una nuova colonna per il nuovo cluster.

   until rango di M e' maggiore di 1.

end procedure.

Figura 10: Procedura di clustering

compiute durante tutti gli step del processo. L'approccio di affinità utilizzato in ARTEMIS è stato sperimentato su insiemi diversi di schemi concettuali di database per selezionare i valori di default più appropriati, che sono quelli risulati più soddisfacenti nella maggior parte dei casi.

Per una discussione più approfondita delle sperimentazioni compiute su schemi forniti dalla Pubblica Amministrazione Italiana, si può fare riferimento a [31, 34]. Considerazioni di tipo più generali sull'utilizzo e sulla definizione delle tecniche semi-automatiche basate sui pesi per l'analisi di schemi concettuali si possono trovare in [33].

8 Generazione dei Cluster di classi ODL$_{I^3}$

Per l'identificazione degli insiemi di classi ODL$_{I^3}$ aventi elevata affinità negli schemi considerati sono utilizzate tecniche di clustering, attraverso le quali le classi sono automaticamente classificate in gruppi caratterizzati da differenti livelli di affinità, formando un albero [42].

Questa procedura di clustering (vedi Figura 10) utilizza una matrice M di rango k, con k uguale al numero totale di classi ODL$_{I^3}$ che devono essere analizzate. In ogni casella $M[h,k]$ della matrice è rappresentato il coefficiente globale $GA()$ riferito alle classi c_h e c_k. La procedura di clustering è iterativa e comincia allocando per ogni classe un singolo cluster: successivamente, ad ogni iterazione, i due cluster tra i quali sussiste il $GA()$ di valore massimo nella matrice M sono uniti. M è così aggiornata dopo ogni operazione di fusione tra cluster, cancellando le righe e le colonne corrispondenti ai cluster unificati, e inserendo una nuova riga ed una nuova colonna che rappresenti il nuovo cluster determinato. Vengono quindi calcolati i coefficienti $GA()$ tra questo cluster aggiunto e tutti quelli già presenti nella matrice: in particolare, viene mantenuto il valore $GA()$ massimo tra i due che erano stati già calcolati tra i cluster rimossi ed il corrispondente cluster col quale si vuole determinare il nuovo valore del coefficiente globale. La procedura termina quando tutte le classi appartengono ad un unico cluster, ossia è definito completamente l'albero di affinitá.

La Fig. 11 mostra l'albero di affinitá risultante dalla clusterizzazione del nostro insieme di classi ODL$_{I^3}$. In particolare, la Fig. 11(a) illustra l'albero di affinitá ottenuto utilizzando l'opzione "all attribute-based" nel calcolo del coefficiente della Structural Affinity, mentre la Fig. 11(b) illustra l'albero ottenuto applicando l'opzione "common attribute-based".

Dopo aver costruito l'albero di affinitá, il problema è quello di selezionare i cluster più

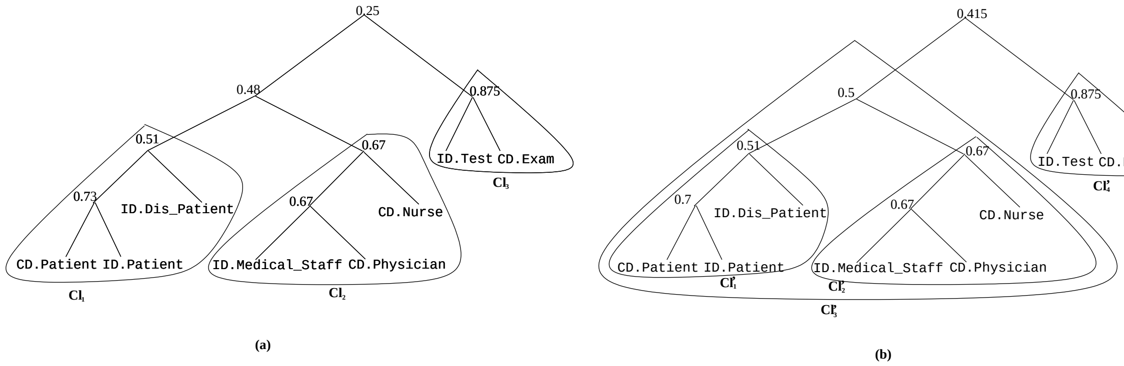

Figura 11: Affinity trees for the `Cardiology` and `Intensive Care` data sources

appropriati per la definizione delle classi nello schema globale. La procedura di selezione dei cluster, in ARTEMIS, viene mantenuta interattiva attraverso la modifica del valore di soglia. Il progettista specifica il valore della soglia T ed i cluster caratterizzati da un valore di $GA()$ superiore o uguale a T sono selezionati e proposti. Per alti valori di T si ottengono cluster piccoli e molto omogenei tra loro. Diminuendo il valore di T, i cluster ottenuti contengono più classi e sono più eterogenei. Nel tool il valore di default di T viene posto pari a 0.5. Tale valore può essere modificato dinamicamente, sulla base della specifica applicazione in esame.

Con riferimento alla Fig. 11, i cluster sono selezionati utilizzando i valori di default. Come si può vedere dall'albero di affinità, il numero di attributi considerati durante il processo di calcolo della Structural Affinity modifica il numero di cluster ottenuti. Nel primo caso (Fig. 11(a)), l'albero dei cluster contiene: $\mathbf{Cl_1}$, che rappresenta le informazioni riguardanti i pazienti; $\mathbf{Cl_2}$ che rappresenta le informazioni riguardanti lo staff medico e $\mathbf{Cl_3}$ che rappresenta le informazioni riguardanti gli esami clinici. Nel secondo caso (Fig. 11(b)), i cluster $\mathbf{Cl'_1}$ e $\mathbf{Cl'_2}$ sono ancora presenti, (e rappresentano ancora informazioni riguardanti i pazienti e lo staff medico rispettivamente). In più, un ulteriore cluster è stato formato e selezionato nell'albero di affinità, chiamato $\mathbf{Cl'_3}$, che unisce il cluster $\mathbf{Cl'_1}$ che il $\mathbf{Cl'_2}$. Questo accade in quanto il pattern `Patient` include alcuni attributi optional. Per questa ragione, applicando l'opzione 2., la Structural Affinity tra la classe ODL$_{I_3}$ che rappresenta il pattern `Patient` e le altre classi ODL$_{I_3}$ risulta più alta rispetto alla Structural Affinity ottenuta utilizzando l'opzione 1. (essendo il numero di coppie di attributi con affinità uguale nei due casi).

Pensiamo che entrambe queste opzioni siano valide in contesti applicativi diversi: per mantenere la flessibilità e l'approccio general-purpose, ARTEMIS permette di calcolare l'albero di affinità utilizzando entrambe le opzioni (1. e 2.) e lasciando al designer la scelta più opportuna.

9 Costruzione dello Schema Globale di mediatore

In questa sezione viene presentato il processo che porta alla definizione dello *Schema Globale* del Mediatore a partire dai cluster precedentemente determinati, ovvero della visione dei dati che sarà presentata all'utente in fase di Query Processing.

La prima fase di questo processo viene realizzata automaticamente e genera, per ogni cluster, una $global\_class_i$ rappresentativa di tutte le classi che fanno parte del cluster (ovvero una classe che costituisca una visione unificata di queste classi). Sia Cl_i un cluster determinato nella fase precedente: ad esso viene associata la $global\_class_i$, alle quale corrisponderà quindi un insieme di attributi globali. La fase di determinazione degli attributi è realizzabile in modo automatico, basandosi sulle relazioni tra attributi memorizzate nel Thesaurus (vedi Sezione 6) e seguendo i seguenti criteri:

- ad ogni *global_class*$_i$ è associata l'unione degli attributi di tutte le classi appartenenti al cluster Cl_i dal quale è stata generata;
- all'interno dell'unione degli attributi sono identificati tutti gli insiemi di termini definiti sinonimi, e ne viene riportato solo uno tra essi (rimuovendo quindi tutti gli altri);
- all'interno dell'unione degli attributi sono identificati tutti gli insiemi di termini legati da relazioni di specializzazione (tra i quali erano quindi state definite relazioni di BT e NT) e vengono riorganizzati all'interno di gerarchie: per ognuna di queste gerarchie è mantenuto solamente il termine più generale (che quindi ne sta a capo e ne può essere considerato il rappresentante) mentre sono rimossi tutti gli altri.

Per esempio, per il cluster $\texttt{Cl}_1$ di Fig. 11 si ottiene il seguente insieme di attributi globali:

$$A = (\texttt{name, code, address, exam*, room, bed, therapy*, date, note, physician*})$$

Per meglio completare la definizione delle classi ODL$_{I^3}$ globali occorre specificare ulteriori informazioni: in particolare MOMIS richiede al designer di fornire, in modo interattivo, informazioni riguardanti:

1. *global_class*$_i$ name
   il tool propone un insieme di nomi candidati per la classe globale, utilizzando le terminological relationships definite nel Common Thesaurus per le classi contenute in $\texttt{Cl}_\texttt{i}$. Basandosi su questi candidati, il designer può decidere il nome più appropriato per *global_class*$_i$. Nel caso in cui i nomi candidati non sono significativi, è comunque possibile assegnare nomi diversi. Nel nostro esempio possiamo deciderediassegnare il nome di `Hospital_Patient` alla classe globale definita in corrispondenza del cluster $\texttt{Cl}_1$.

   Perciò, rispetto a $\texttt{Cl}_1$ otteniamo la classe globale:

   Hospital_Patient = (`Hospital_Patient`,{`name,code,address,`
   `exam*,room,bed,therapy*,date,`
   `note,physician*`})

2. *mappings* tra gli attributi globali di *global_class*$_i$ ed i corrispondenti attributi delle classi in $\texttt{Cl}_\texttt{i}$

   In particulare, se un attributo globale è ottenuto da più di un attributo di una stessa classe in $\texttt{Cl}_\texttt{i}$, il designer deve specificare il tipo di *correspondence* che deve essere definita nell'attributo globale. In MOMIS vengono fornite le seguenti opzioni:

   - *and* correspondence
     specifica che un attributo globale corrisponde alla concatenazione degli attributi della classe $c_h \in \texttt{Cl}_\texttt{i}$.
     Per esempio, l'attributo globale `name` di `Hospital_Patient` corrisponde alla concatenazione degli attributi `first_name` e `last_name` della classe `ID.Patient in` $\texttt{Cl}_1$. Specificando l'*and* correspondence per l'attributo `name`, il progettista afferma che entrambi i valori degli attributi `first_name` e `last_name` debbono essere considerati quando è coinvolta la classe `ID.Patient`.

- *union* correspondence
  specifica che l'attributo globale corrisponde ad almeno uno degli attributi specificati della classe $c_h \in \mathtt{Cl_i}$;

3. *default values* che vengono assegnati agli attributi globali in corrispondenza della classe $c_h \in \mathtt{Cl_i}$. Un valore di default di un attributo globale è sempre verificato quando viene valutato nella classe c_h.

4. *new attributes* per la classe globale.

   Per esempio, un nuovo attributo `dept` può essere aggiunto nella classe globale `Hospital_Patient`. Questo attributo è utilizzato per mantenere informazioni legate al dipartimento in cui un paziente viene ricoverato.

Un esempio di definizione di un classe globale in linguaggio ODL$_{I^3}$ è mostrato in Fig. 12 per la classe `Hospital_Patient`. Come si può vedere da questa figura, per ciascun attributo sono definite regole di mapping, in modo da specificare sia informazioni sulla corrispondenza tra attributo globale ed il corrispondente nelle classi locali associate, sia su eventuali valori di default o di null (mancanza della corrispondenza). Per esempio, per l'attributo globale `name`, la regola di mapping specifica gli attributi che debbono essere considerati in ciascuna classe locale del cluster $\mathtt{Cl_1}$. Nel caso specifico, viene definita una *and* correspondence per la classe `ID.Patient`. Una diversa mapping rule è definita per l'attributo globale `dept` per specificare il valore da essere associato con `dept` per le istanze di `CD.Patient`, `ID.Patient`, e `ID.Dis_Patient`.

Come scritto precedentemente, la *union* correspondance risulta utile quando un attributo globale corrisponde a due o più attributi di una classe di una sorgente, in base al valore di un attributo terzo chiamato *tag attribute*. Per esempio, supponendo di avere definito un cluster globale per la classe `automobile` e che in una classe di una sorgente viene memorizzato il prezzo delle auto nelle due valute di Lire Italiane e US Dollari e si abbia `country` come attributo tag. Il progettista può definire una *union* correspondance tra gli attributi `Italian_price` e `US_price` introducendo una rule che specifichi il mapping locale basato sul valore dell'attributo `country`. Utilizzando la sintassi delle mapping rule, la dichiarazione è la seguente:

```
...
 attribute integer price
    mapping rule (S.car.Italian_price union
                  S.car.US_price on Rule1),
                 ...
...
 rule Rule1 { case of S.car.country:
              ``Italy'' : S.car.Italian_price;
              ``US''    : S.car.US_price; }
```

Come si può vedere, nella definizione di un attributo globale non viene specificato alcun dominio: questo significa che MOMIS accetta i domini definiti nelle sorgenti locali e, in risposta ad una query, sono visualizzati i dati nei formati specifici delle sorgenti locali.

Tra gli sviluppi futuri del sistema, seguendo quanto suggerito in [46, 40], un nostro obiettivo è quello di sviluppare una libreria aperta di funzioni che contiene le principali operazioni di mapping e che il progettista può eseguire, può personalizzare ed estendere secondo le proprie necessità.

La descrizione in linguaggio ODL$_{I^3}$ della classe globale *global_class$_i$* viene tradotta in una mapping table persistente. Come esempio, la mapping table per la classe `Hospital_Patient` è mostrata in Fig. 13.

```
interface Hospital_Patient
{ attribute name
     mapping_rule (ID.Patient.first_name and
                  ID.Patient.last_name),
                  CD.Patient.name;
     ...
  attribute physician
     mapping_rule CD.Patient.physician,
                  Id.Patient.doctor_id
  attribute dept
     mapping_rule CD.Patient = 'Cardiology',
                  ID.Patient = 'Intensive Care',
                  ID.Dis_Patient = 'Intensive Care'
}
```

Figura 12: Esempio di classi globali in ODL_{I^3}

Hospital_Patient	code	name	exam	...	physician	dept
CD.Patient	null	name	exam	...	physician	'Cardiology'
ID.Patient	code	first_name *and* last_name	test	...	doctor_id	'Intensive Care'
ID.Dis_Patient	code	null	null	...	ID.Patient. doctor_id	'Intensive Care'

Figura 13: `Hospital_Patient` mapping table

10 Query Processing e Optimization

In questa sezione verrà descritto il processo che, sfruttando le informazioni memorizzate nelle mapping table, realizza la Query Reformulation. Quando un utente generico invia una query al sistema MOMIS, il Query Manager produce un insieme di sottoquery che dovranno essere spedite alle diverse fonti di informazione per dare risposta alla query originale.

Seguendo l'impostazione simile ad altri approcci *semantici* [4, 5], questo processo consiste di due fasi distinte:

- semantic optimization
- query plan formulation

10.1 Semantic Optimization

In questa fase il Query Manager processa la query data utilizzando le tecniche di ottimizzazione semantica supportate da ODB-Tools [8, 9, 7, 1] allo scopo di ridurre il costo del piano di accesso della query. La query viene sostituita da una nuova che incorpora ogni possibile restrizione che non è presente nella query originale ma è logicamente implicata dalla query basata sullo schema globale. La trasformazione è basata sulle inferenze logiche basate sulla conoscenza del contesto (in particolare sugli integrity constraint) dello schema globale di mediatore.

Consideriamo, ad esempio, la query: "Seleziona i nomi dei pazienti il cui esame ha come risultato 'Heart risk' ".

```
Q1:
select name
```

```
from   Hospital_Patient
where  exam.result = ‘Heart risk’
```

Supponiamo ora che nel dominio `Hospital` esista una relazione tra il risultato di un esame e il dipartimento in cui il paziente viene ricoverato; ad esempio che tutti i pazienti il cui esame ha il valore ‘Heart risk’ sono ricoverati nel dipartimento ‘Cardiology’. In base a questo requisito il progettista può definire il seguente vincolo di integrità relativo alla classe `Hospital_Patient` dello schema globale:
`rule R1 forall X in Hospital_Patient: (X.exam.result = ‘Heart risk’) then X.dept = ‘’Cardiology’;`

Il Query Manager, utilizzando il query optimizer di ODB-Tools, esegue l’espansione semantica della query applicando la rule `R1`. La query risultante è la seguente:

```
Q1:
select name
from   Hospital_Patient
where  exam.result = ‘Heart risk’
and    dept = ‘Cardiology’
```

L’espansione semantica è compiuta al fine di aggiungere più fattori booleani possibile nella “where clause”: questo processo rende il query plan formulation più costoso (poichè deve essere tradotta una query più pesante per ciascuna sorgente di interesse), ma il recupero dei dati sulla singola sorgente può essere enormemente ridotto se sono presenti indici sui predicati aggiunti in fase di ottimizzazione.

10.2 Query Plan Formulation

Dopo aver prodotto la query ottimizzata, il Query Manager genera un insieme di sottoquery destinate alle sorgenti locali. Per ciascuna sorgente informativa coinvolta, il Query Manager deve tradurre la query ottimizzata in termini dei corrispondenti schemi locali, utilizzando la mapping table associata a ciascuna classe globale coinvolta nella query. Allo scopo di ottenere ciascuna sottoquery, il Query Manager controlla e traduce ciascun fattore booleano nella clausola `where`.
In particolare, l’algoritmo di *Check* (mostrato in Fig. 14) è stato pensato per l’eliminazione dalla lista di classi da interrogare quelle nelle quali si è già sicuri, basandosi sulle informazioni memorizzate nella mapping table, di non trovare dati *utili* o comunque *verificabili*. Per eliminare le fonti che fornirebbero dati non *utili*, ci si basa sui valori di default della mapping table: se il valore definito di default per un attributo in una classe locale è diverso dal valore specificato (in **and**) nella clausola `where`, la classe non sarà interrogata.
Per eliminare invece le sorgenti che fornirebbero dati non *verificabili* ci si basa sui valori nulli presenti nella mapping table: in particolare, se è stata specificata una condizione nella clausola `where` (sempre in **and** con le altre condizioni espresse nella clausola) su un attributo globale per il quale non esiste un attributo locale corrispettivo nella classe che si sta analizzando, si è scelto di non procedere all’interrogazione della classe, non potendo essere verificata quella condizione. È stata quindi fatta una scelta drastica, per mantenere la sicurezza di ricevere solamente dati completamente verificati, ma sarebbero comunque possibili scelte differenti: tra queste, si potrebbe ad esempio ipotizzare l’esistenza di un livello di *credibilità* delle risposte, che l’utente dovrebbe scegliere, che esprima la percentuale delle condizioni che si vogliono siano assolutamente verificate dai dati ricevuti in risposta all’interrogazione.

Algoritmo di Controllo ed Eliminazione
/* **Input:** Cluster Cl_j e k classi appartenenti al cluster */

begin procedure:
  for each classe $c_h \in Cl_j$ **do**
    for each fattore boolean **do**
      if (attributo globale $\in$ fattore boolean) corrisponde a:
      1. valore nullo in c_h: nessuna query locale viene generata per c_h;
      2. valore di default in c_h: **if** valore specificato nel fattore boolean factor
         e' diverso da quello di default
         then nessuna query locale viene generata per c_h;
end procedure.

Figura 14: Algoritmo di Controllo ed Eliminazione

Con riferimento al nostro esempio, lo step 1 dell'algoritmo di *Check* esclude la classe `Dis_Patient` e lo step 2 la classe `Patient` della sorgente `ID`. Conseguentemente, solo la seguente sottoquery per la sorgente `CD` viene inviata:

```
select R.name
from   Patient R
where  exists X in R.exam : X.outcome = 'Heart risk'
```

In tal modo, una effettiva ottimizzazione viene compiuta poichè solamente una sorgente locale deve essere acceduta, anzichè due.
Se gli integrity constraint sono forniti anche per le sorgenti locali, è possibile iterare il procedimento eseguendo l'ottimizzazione semantica della sottoquery prima di inviarla alla sorgente locale. Per esempio, se la seguente rule viene definita per la sorgente `CD`:
`rule R2 forall X in Patient:  (exists X.exam.outcome = 'Heart risk') then X.room > 20 ;`
`R2` può essere utilizzata per ottimizzare la sottoquery, dando origine alla seguente query:

```
select R.name
from   Patient R
where  exists X in R.exam : X.outcome = 'Heart risk'
and    R.room > 20
```

Questa trasformazione può essere utile nel caso in cui sia disponibile un indice per l'attributo `room`.

Come altro esempio della query formulation sullo schema globale, consideriamo il caso di rappresentazione eterogenea dell'attributo address e supponiamo di ricercare il nome di tutti i pazienti con un determinato indirizzo. La query risultante è la `Q2`:

```
Q2: select name
from Hospital_Patient
where address.street = 'Army Street'
```

Il processo di reformulation della query `Q2` porta alla generazione di due sottoquery per la sorgente `CD` (in quanto occorre tener conto del costruttore di union sulll'attributo address) e una sottoquery per il dipartimento `Intensive Care`:

```
CD:
select R.name
from   Patient R
where  R.address = ‘Army Street’

CD:
select R.name
from   Patient R
where  R.address.street = ‘Army Street’

ID:
select P.first_name, P.last_name
from   Patient P
where  P.address = ‘Army Street’
```

11 Confronto con altri lavori

Lavori correlati al progetto MOMIS sono identificati nell'area dei semistructured data e dell'integrazione di informazioni eterogenee.

Semistructured data. La problematica della modellizzazione dei semistructured data è stata particolarmente investigata in letteratura. In particolare, una survey inerente ai problemi del semistructured data modeling querying è stata presentata in [23]. Due modelli simili per semistructured data sono stati proposti in [25, 69], e sono basati un grafo etichettato a radice, in cui i nodi sono gli oggetti (i dati) e gli archi le etichette. Nel modello presentato in [25], l'informazione risiede solamente nelle etichette, mentre nel modello "Object Exchange Model" (OEM) propopsto da Papakonstantinou et. al. in [69], l'informazione risiede anche nei nodi.

Nel primo modello, gli archi sono etichettati sia tramite i tipi base piú comuni quali integer, real, string, sia tramite nomi (chiamati *symbols*) che corrispondono ai nomi degli attributi delle relazioni o delle classi. Nel secondo modello abbiamo: *i)* i nodi foglia sono etichettati dai dati; *ii)* i nodi interni non sono etichettati con dati significativi; *iii)* gli archi sono etichettati solo da symbols (nomi di attributi). È possibile definire funzioni di mapping tra i due modelli: introducendo nuovi archi, un grafo OEM può essere convertito nel corrispondente modello proposto in [25].

L'altra problematica che è stata fortemente studiata riguarda il tentativo di aggiungere struttura a dati semistrutturati: questa prospettiva è vicina alla nostra proposta di object pattern. In particolare, in [48] viene proposta la nozione di *dataguide* come una "loose description of the structure of the data" memorizzati nella sorgente informativa. Una proposta per inferire struttura in dati semistrutturati è stata presentata in [57], dove gli autori utilizzano un modello dei dati basato su grafi derivato da [25, 69]. In questo modelli, la struttura dei dati è una gerarchia di tipi che viene ottenuta tramite un particolare algoritmo che misura la distanza (*jump*) tra i tipi, basandosi sull'importanza relativa di alcuni attributi rispetto agli altri in un insieme elevato di dati. In questo modo vine costruita una "reasonably small approximation" dei tipi che rappresentano la vasta e irregolare casistica dei dati presenti nella sorgente, riuscendo a distinguere gli attributi principali ("core" attribute) dagli altri meno significativi. In [24] viene proposta una nuova nozione di schema a grafo appropriata per rappresentazioni database tramite grafi etichettati con radice. Il principale utilizzo della struttura estratta da una sorgente semistrutturata è stato presentato per la query optimization. Infatti, l'esistenza di un cammino nella struttura semplifica la valutazione della query limitando la ricerca sui soli dati rilevanti. La nostra attenzione verso i dati semistrutturati è stata rivolta verso l'utilizzo di una struttura nella forma di object patterns per la rappresentazione di dati semistrutturati per utilizzarla come supporto all'integrazione di dati.

Heterogeneous information integration. In quest'area sono stati sviluppati molti progetti basati sull'architettura a mediatore [4, 5, 21, 37, 39, 56, 61].
MOMIS è basato su di un'architettura a mediatore e segue il 'semantic approach'.

Seguento la classificazione dei sistemi di integrazione proposti da Hull [49], MOMIS si pone nel gruppo di quelli che seguono il "virtual approach". Virtual approach è stato proposto all'inizio degli anni '80 nei modelli multidatabase [54, 41]. Più recentemente, alcuni sistemi sono stati sviluppati basandosi sulla logica descrittiva [4, 53], tra i quali CLASSIC [20]. Tutti gli approcci virtuali sono basati sul modello di query decomposition, inviando sottoquery ai database sorgenti e unificando le risposte sui dati selezionati. Sistemi più recenti basati sulla logica descrittiva sono focalizzati fondamentalmente sulle query congiuntive (i.e. quelle esprimibili utilizzando la selezione, proiezione e join), beneficiando della *Open World Assumption*. Riguardo allo schema, viene utilizzato un approccio "top-down": viene creato uno schema globale che comprende tutte le informazioni rilevanti delle singole sorgenti, ed i dati delle sorgenti sono espressi come viste dello schema globale [65].

Riferendoci sempre alla classificazione proposta da Hull, MOMIS può essere posizionato nella categoria delle "read-only views", e cioè di quei sistemi il cui compito è quello di supportare una vista integrata dei dati presenti in database diversi con accessi di tipo read-only. I progetti più simili al nostro sono: GARLIC, SIMS, Information Manifold e Infomaster.

Il progetto GARLIC [56, 61] fornisce un'architettura con wrapper complessi che forniscono la descrizione delle sorgenti in linguaggio OO (chiamato GDL) e poi viene definito manualmente uno schema globale che unifica la visione delle sorgenti locali tramite gli oggetti Garlic Complex Objects.
Il progetto SIMS [4, 5] propone di creare la definizione dello schema globale utilizzando la logica descrittiva (i.e. il linguaggio LOOM) per descrivere le sorgenti informative. L'uso dello schema globale permette ad entrambi i progetti GARLIC e SIMS di supportare query di qualsivolgia natura basate sullo schema invece di quelle predefinite dal sistema.

Information Manifold Systems [51, 53], alla stregua del progetto MOMIS, fornisce un mediatore e una query manager independente dalla sorgente. Lo schema di input di Information Manifold System è dato da un'insieme di descrizione delle sorgenti; così, data una query, il sistema è in grado di creare un query plan in grado di rispondere alla query utilizzando le sorgenti locali. Gli algoritmi che descrivono le tecniche di decisione per la scelta delle informazioni utili e per generare il query plan sono forniti in [53, 52]. Lo schema globale di input è completamente modellato dall'utente, a differenza del nostro approccio in cui l'integrazione viene guidata dal sistema.

Infomaster System [46, 40] fornisce l'accesso integrato a sorgenti informative eterogenee e distribuite, mostrando all'utente il sistema come fosse centralizzato ed omogeneo. Il sistema è basato su di uno schema globale, modellato completamente dall'utente e dal modulo di query processing che determina dinamicamente ed efficientemente il piano di accesso utilizzando regole di traduzione che armonizzano le sorgenti eterogenee.

D'altra parte, molti progetti sono basati sull'approccio 'structural' [21, 37, 39]. Il progetto TSIMMIS [37, 45] segue l'approccio 'structural' ed utilizza un modello self-describing (OEM) per rappresentare gli oggetti ed una tecnica di pattern matching per eseguire un insieme predefinito di query basate su query template. La conoscenza semantica viene codificata nelle regole di mediazione scritte in linguaggio MSL (Mediator Specification Language) che permettono l'integrazione delle sorgenti a livello di mediatore. Sebbene la generalità e l'immediatezza di OEM e MSL rendono questo approccio un buon candidato per l'integrazione di sorgenti informative eterogenee e semistrutturate di notevoli dimensioni, si ha un aumento notevole dei costi di sviluppo nel caso di inserimento dinamico di altre sorgenti nel sistema. Infatti, in TSIMMIS, le nuove sorgenti non solo sono collegate allo specifico wrapper (che quindi deve essere specificato e/o sviluppato) ma il mediatore deve ridefinire le regole MSL e ricompilarle; l'amministratore del sistema deve infine specificare come il sistema utilizza la nuova sorgente [65]. In MOMIS questa attività viene assistita dal sistema.

Un'altra area di ricerca è focalizzata sulla identificazione e risoluzione delle inconsistenze semantiche tra gli schemi, basandosi sulla definizione di corrispondenza tra elementi di schemi. In [22] viene presentato il modello Summary Schema Model (SSM) per fornire il supporto all'idenficazione delle similarità semantiche. Utilizzando le tassonomie presenti, SSM viene usato come descrizione di alto livello e coincisa di tutti i dati disponibili nella sorgente. Due entità sono semanticamente correlate se sono mappate nello stesso concetto della tassonomia.

In [43] viene definita una tecnica basata sulla fuzzy logic per derivare le terminological relationships tra gli elementi dello schema.

Altri approcci considerano gli attributi ed i domini degli elementi dello schema [19, 38]. In [19], relazioni, attributi, ed integrity constraints presenti nello schema sorgente sono espressi utilizzando un linguaggio di tipo logico per unificare la conoscenza di tutti gli elementi nella sorgente.

Infine, alcuni gruppi di ricerca hanno proposto di considerare le architetture federate per la risoluzione delle eterogeneità semantiche, utilizzando la definizione di schema federato data in [60, 63].

A Il linguaggio di descrizione ODL$_{I^3}$

L'appendice riporta la descrizione della BNF del linguaggio ODL$_{I^3}$. Viene descritta la parte si sintassi relativa alle estensioni compiute rispetto alla grammatica ODL dello standard ODMG-93.

⟨interface_dcl⟩	::=	⟨interface_header⟩ **{**[⟨interface_body⟩]
		[**union** ⟨interface_body⟩]**};**
⟨interface_header⟩	::=	**interface** ⟨identifier⟩
		[⟨inheritance_spec⟩]
		[⟨type_property_list⟩]
⟨inheritance_spec⟩	::=	: ⟨scoped_name⟩ [,⟨inheritance_spec⟩]

Parte relativa alla specificazione delle classi, in cui occorre specificare il tipo e il nome della sorgente informativa.

⟨type_property_list⟩	::=	**(** [⟨source_spec⟩] [⟨extent_spec⟩] [⟨key_spec⟩] [⟨f_key_spec⟩] **)**
⟨source_spec⟩	::=	**source** ⟨source_type⟩ ⟨source_name⟩
⟨source_type⟩	::=	**relational** | **nfrelational** | **object** | **file** | **semistructured**
⟨source_name⟩	::=	⟨identifier⟩
⟨extent_spec⟩	::=	**extent** ⟨extent_list⟩
⟨extent_list⟩	::=	⟨string⟩ | ⟨string⟩,⟨extent_list⟩
⟨key_spec⟩	::=	**key**[**s**] ⟨key_list⟩
⟨f_key_spec⟩	::=	**foreign_key** (⟨f_key_list⟩) references ⟨identifier⟩[,⟨f_key_spec⟩]
...		

Regole di definizione del mapping tra attributi della classe globale dello scshema di mediatore ed icorrispondenti nelle sorgenti locali.

⟨attr_dcl⟩	::=	**[readonly] attribute**
		[⟨domain_type⟩] ⟨attribute_name⟩ **[*]**
		[⟨fixed_array_size⟩] [⟨mapping_rule_dcl⟩]
⟨mapping_rule_dcl⟩	::=	**mapping_rule** ⟨rule_list⟩
⟨rule_list⟩	::=	⟨rule⟩ | ⟨rule⟩**,**⟨rule_list⟩
⟨rule⟩	::=	⟨local_attr_name⟩ | **‘**⟨identifier⟩**’**
		⟨and_expression⟩ | ⟨union_expression⟩
⟨and_expression⟩	::=	**(** ⟨local_attr_name⟩ **and** ⟨and_list⟩ **)**
⟨and_list⟩	::=	⟨local_attr_name⟩ | ⟨local_attr_name⟩ **and** ⟨and_list⟩
⟨union_expression⟩	::=	**(** ⟨local_attr_name⟩ **union** ⟨union_list⟩ **on** ⟨identifier⟩ **)**
⟨union_list⟩	::=	⟨local_attr_name⟩ | ⟨local_attr_name⟩ **union** ⟨union_list⟩
⟨local_attr_name⟩	::=	⟨source_name⟩**.**⟨class_name⟩**.**⟨attribute_name⟩

...

Terminological relationships usate per la definizione del Common Thesaurus.

⟨relationships_list⟩	::=	⟨relationship_dcl⟩**;** | ⟨relationship_dcl⟩**;** ⟨relationships_list⟩
⟨relationships_dcl⟩	::=	⟨local_attr_name⟩ ⟨relationship_type⟩ ⟨local_attr_name⟩
⟨relationship_type⟩	::=	**SYN** | **BT** | **NT** | **RT**

...

Definizione degli OLCD integrity constraint: le regole sono definite tramite la logica di *if then* e sono valide per ogni istanza del database.

Definizione delle regole di *or* e *union*.

⟨rule_list⟩	::=	⟨rule_dcl⟩**;** | ⟨rule_dcl⟩**;** ⟨rule_list⟩
⟨rule_dcl⟩	::=	**rule** ⟨identifier⟩ ⟨rule_spec⟩
⟨rule_spec⟩	::=	⟨rule_pre⟩ **then** ⟨rule_post⟩ | **{** ⟨case_dcl⟩ **}**
⟨rule_pre⟩	::=	⟨forall⟩ ⟨identifier⟩ **in** ⟨identifier⟩ **:** ⟨rule_body_list⟩
⟨rule_post⟩	::=	⟨rule_body_list⟩
⟨case_dcl⟩	::=	**case of** ⟨identifier⟩ **:** ⟨case_list⟩
⟨case_list⟩	::=	⟨case_spec⟩ | ⟨case_spec⟩ ⟨case_list⟩
⟨case_spec⟩	::=	⟨identifier⟩ **:** ⟨identifier⟩ **;**
⟨rule_body_list⟩	::=	**(** ⟨rule_body_list⟩ **)** | ⟨rule_body⟩ |
		⟨rule_body_list⟩ **and** ⟨rule_body⟩ |
		⟨rule_body_list⟩ **and (** ⟨rule_body_list⟩ **)**
⟨rule_body⟩	::=	⟨dotted_name⟩ ⟨rule_const_op⟩ ⟨literal_value⟩ |
		⟨dotted_name⟩ ⟨rule_const_op⟩ ⟨rule_cast⟩ ⟨literal_value⟩ |
		⟨dotted_name⟩ **in** ⟨dotted_name⟩ |
		⟨forall⟩ ⟨identifier⟩ **in** ⟨dotted_name⟩ **:** ⟨rule_body_list⟩ |
		exists ⟨identifier⟩ **in** ⟨dotted_name⟩ **:** ⟨rule_body_list⟩
⟨rule_const_op⟩	::=	$= \mid \geq \mid \leq \mid > \mid <$
⟨rule_cast⟩	::=	**(**⟨simple_type_spec⟩**)**
⟨dotted_name⟩	::=	⟨identifier⟩ | ⟨identifier⟩**.**⟨dotted_name⟩
⟨forall⟩	::=	**for all** | **forall**

B Sorgenti di esempio in linguaggio ODL$_{I^3}$

Di seguito é riportata la descrizione, attraverso il linguaggio ODL$_{I^3}$, dell'esempio di riferimento di Sezione 3.

```
Cardiology_Department (CD):
```

```
interface Patient
 ( source semistructured
      Cardiology_Department )
{  attribute string        name;
   attribute string        address;
   attribute set<Exam>     exam*;
   attribute integer       room;
   attribute integer       bed;
   attribute string        therapy*;
   attribute set<Physician> physician*; };
```

```
interface Physician
 ( source semistructured
      Cardiology_Department )
{  attribute string  name;
   attribute string  address;
   attribute integer phone;
   attribute string specialization;};
```

```
interface Nurse
 ( source semistructured
      Cardiology_Department )
{  attribute string      name;
   attribute string      address;
   attribute integer     level;
   attribute set<Patient> patient; };
```

```
interface Exam
 ( source semistructured
      Cardiology_Department )
{  attribute integer date;
   attribute string  type;
   attribute string  outcome; };
```

Intensive_Care_Department (ID):

```
interface Patient
( source relational Intensive_Care
  extent Patients
  key  code
  foreign_key(test) references Test,
  foreign_key(doctor_id) references
  Medical_Staff )
{ attribute string  code;
  attribute string  first_name;
  attribute string  last_name;
  attribute string  address;
  attribute string  test;
  attribute string  doctor_id; };
```

```
interface Medical_Staff
( source relational Intensive_Care
  extent Medical_Staffers
  key id )
{ attribute string  id;
  attribute string  first_name;
  attribute string  last_name;
  attribute integer phone;
  attribute string  address;
  attribute string  availability;
  attribute string  position; };
```

```
interface Test
( source relational Intensive_Care
  extent Tests
  key number )
{ attribute integer number;
  attribute string  type;
  attribute integer date;
  attribute string  laboratory;
  attribute string  result; };
```

```
interface Dis_Patient
( source relational Intensive_Care
  extent Dis_Patients
  key code
  foreign_key(code) references Patient)
{ attribute string  code;
  attribute integer date;
  attribute string  note; };
```

Riferimenti bibliografici

[1] Odb-tools Project. Available at http://sparc20.dsi.unimo.it/index.html.

[2] S. Abiteboul. Querying semi-structured data. In *Proceedings of the International Conference on Database Theory, ICDT97*, pages 1–18, Athens, Greece, 1997.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semistructured data. *Journal of Digital Libraries*, 1(1), 1996.

[4] Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[5] Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.

[6] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.

[7] D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODL-tools: a description logics based tool for schema validation and semantic query optimization in object oriented databases. In *Sesto Convegno AIIA - Roma*, 1997.

[8] Domenico Beneventano, Sonia Bergamaschi, and Claudio Sartori. Semantic query optimization by subsumption in OODB. In H. Christiansen, H. L. Larsen, and T. Andreasen, editors, *Flexible Query Answering Systems*, volume 62 of *Datalogiske Skrifter - ISSN 0109-9799*, Roskilde, Denmark, 1996.

[9] Domenico Beneventano, Sonia Bergamaschi, Claudio Sartori, and Maurizio Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. In *Int. Conference on Data Engineering - ICDE97*, 1997. http://sparc20.dsi.unimo.it.

[10] S. Bergamaschi, D. Beneventano, S. Castano, and M. Vincini. Integrazione di informazione: il linguaggio odl_i^3 e la logica descrittiva olcd. Technical Report INTERDATA T3-R03, MURST 40%, 1998. http://bsing.ing.unibs.it/ deantone/interdata_tema3/.

[11] S. Bergamaschi, D. Beneventano, and M. Vincini. Stato dell'arte della ricerca nell'ambito dell'estrazione/integrazione intelligente di informazioni. Technical Report INTERDATA T3-R01, MURST 40%, 1998. http://bsing.ing.unibs.it/ deantone/interdata_tema3/.

[12] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. Exploiting schema knowledge for the integration of heterogeneous sources. In *Sesto Convegno Nazionale su Sistemi Evoluti per Basi di Dati - SEBD98, Ancona*, pages 103–122, 1998.

[13] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. An intelligent approach to information integration. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, june 1998.

[14] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, and M. Vincini. A semantic approach to information integration: the momis project. In *Sesto Convegno della Associazione Italiana per l'Intelligenza Artificiale AI*IA98, Padova*, pages 78–82, 1998.

[15] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.

[16] S. Bergamaschi and B. Nebel. Acquisition and validation of complex object database schemata supporting multiple inheritance. *Journal of Applied Intelligence*, 4:185–203, 1994.

[17] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Trans. on Database Systems*, 17(3):385–422, September 1992.

[18] Sonia Bergamaschi. Extraction of informations from highly heterogeneous source of textual data. In *First International Workshop CIA-97 COOPERATIVE INFORMATION AGENTS - DAI meets Database Systems, University of Kiel, Kiel, Germany*, 1997.

[19] J.M. Blanco, A. Illarramendi, and A. Goni. Building a federated relational database system: An approach using a knowledge-based system. *Int. Journal of Intelligent and Cooperative Information Systems*, 3(4):415–455, 1994.

[20] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *SIGMOD*, pages 58–67, Portland, Oregon, 1989.

[21] Y.J. Breibart et al. Database integration in a distributed heterogeneous database system. In *Proc. 2nd Intl IEEE Conf. on Data Engineering, Los Angeles, CA*, 1986.

[22] M.W. Bright, A.R. Hurson, and S. Pakzad. Automated resolution of semantic heterogeneity in multidatabases. *ACM Transactions on Database Systems*, 19(2):212–253, 1994.

[23] P. Buneman. Semistructured data. In *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, pages 117–121, Tucson, Arizona, 1997.

[24] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. Adding structure to unstructured data. In *Proc. of ICDT 1997*, pages 336–350, Delphi, Greece, 1997.

[25] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. of the ACM SIGMOD International Conference*, pages 505–516. ACM Press, 1996.

[26] P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard. Technical report, University of Maryland, 1996. ftp://ftp.umiacs.umd.edu /pub/ONRrept/ medmodel96.ps.

[27] P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard, April 1996. Available at ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps.

[28] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of Int. Conference on Deductive and Object-Oriented Databases*, 1995.

[29] S. Castano and V. De Antonellis. Deriving global conceptual views from multiple information sources. In *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, 1997.

[30] S. Castano and V. De Antonellis. Engineering a library of reusable conceptual components. *Information and Software Technology*, 39:65–76, 1997.

[31] S. Castano and V. De Antonellis. Semantic dictionary design for database interoperability. In *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, 1997.

[32] S. Castano and V. De Antonellis. A discovery-based approach to database ontology design. *Distributed and Parallel Databases, Special Issue on Ontologies and Databases*, 1998.

[33] S. Castano, V. De Antonellis, M.G. Fugini, and B. Pernici. Conceptual schema analysis: Techniques and applications. *accepted for publication on ACM Transactions on Database Systems*, 1997.

[34] S. Castano, V. De Antonellis, and S. De Capitani Di Vimercat. Global viewing of heterogeneous data sources. Technical Report 98-08, Dip. Elettronica e Informazione, Politecnico di Milano, Milano, Italy, 1998.

[35] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[36] R. G. G. Cattell, editor. *The Object Database Standard: ODMG93.* Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[37] S. Chawathe, Garcia Molina, H., J. Hammer, K.Ireland, Y. Papakostantinou, J.Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ Conference, Tokyo, Japan*, 1994. ftp://db.stanford.edu /pub/chawathe/1994/ tsimmis-overview.ps.

[38] C. Clifton, E. Housman, and A. Rosenthal. Experience with a combined approach to attribute-matching across heterogeneous databases. In *Proc. of IFIP DS-7 Data Semantics Conf.*, 1997.

[39] U. Dayal and H. Hwuang. View definition and generalization for databse integration in a multidatabase system. In *Proc. IEEE Workshop on Object-Oriented DBMS - Asilomar, CA*, 1986.

[40] O. M. Duschka and M. R. Genesereth. Infomaster - an information integration tool. In *Proceedings of the International Workshop "Intelligent Information Integration" during the 21st German Annual Conference on Artificial Intelligence, KI-97*, 1997.

[41] Thomas et al. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22(3):237–266, 1990.

[42] B. Everitt. *Computer-Aided Database Design: the DATAID Project.* Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[43] P. Fankhauser, M. Kracker, and E.J. Neuhold. Semantic vs. structural resemblance of classes. *SIGMOD RECORD*, 20(4):59–63, 1991.

[44] N.V. Findler, editor. *Associative Networks.* Academic Press, 1979.

[45] H. Garcia-Molina et al. The TSIMMIS approach to mediation: Data models and languages. In *NGITS workshop*, 1995. ftp://db.stanford.edu /pub/garcia/1995/tisimmis-models-languages.ps.

[46] M. R. Genesereth, A. M. Keller, and O. Duschka. Infomaster: An information integration system. In *Proceedings of 1997 ACM SIGMOD Conference*, 1997.

[47] J. Gilarranz, J. Gonzalo, and F. Verdejo. Using the eurowordnet multilingual semantic database. In *Proc. of AAAI-96 Spring Symposium Cross-Language Text and Speech Retrieval*, 1996.

[48] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured data. In *23rd VLDB Int. Conf.*, 1997.

[49] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *ACM Symp. on Principles of Database Systems*, pages 51–61, 1997.

[50] R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.

[51] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *In Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous*, 1995.

[52] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query-answering algorithms for information agents. In *AAAI/IAAI*, volume 1, pages 40–47, 1996.

[53] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB 1996*, pages 251–262, 1996.

[54] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22:267–293, 1990.

[55] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[56] M.J.Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams, and E.L. Wimmers. Object exchange across heterogeneous information sources. Technical report, Stanford University, 1994.

[57] S. Nestorov, S. Abiteboul, and R. Motwani. Inferring structure in semistructured data. *SIGMOD Record*, 26(4):39–43, 1997.

[58] N.Guarino. Understanding, building, and using ontologies. A commentary to 'Using Explicit Ontologies in KBS Development', by van Heijst, Schreiber, and Wielinga.

[59] N.Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. Technical report, Summer School on Information Extraction, Frascati, Italy, July 1997.

[60] M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta. A methodology for integration of heterogeneous databases. *IEEE Trans. on Knowledge and Data Engineering*, 6(6):920–933, 1994.

[61] M.T. Roth and P. Scharz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, March 1995.

[62] F. Saltor and E. Rodriguez. On intelligent access to heterogeneous information. In *Proceedings of the 4th KRDB Workshop*, Athens, Greece, August 1997.

[63] A.P. Sheth and J.P. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), 1990.

[64] J. D Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Maryland - USA, 1989.

[65] J. D. Ullman. Information integration using logical views. In *Intl. Conf on Database Theory*, pages 19–40, 1997.

[66] AA. VV. The common object request broker: Architecture and specification. Technical report, Object Request Broker Task Force, 1993. Revision 1.2, Draft 29, December.

[67] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.

[68] W. A. Woods and J. G. Schmolze. The KL-ONE family. In F. W. Lehman, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992. Pubblished as a Special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2-9.

[69] Y.Papakonstantinou, H.Garcia-Molina, and J.Widom. Object exchange across heterogeneous information sources. In *Proc. of ICDE95*, Taipei, Taiwan, 1995.