Creazione di una vista globale d'impresa con il sistema MOMIS basato su Description Logics

D. Beneventano[1,2], **S. Bergamaschi**[1,2], **A. Corni**[1,2] and **M. Vincini**[1]
(1) Università di Modena e Reggio Emilia (2) CSITE-CNR Bologna
DSI - Via Campi 213/B V.le Risorgimento, 2
41100 Modena 40136 Bologna
e-mail: dbeneventano@deis.unibo.it, [sonia,corni,vincini]@dsi.unimo.it

Sommario

Sviluppare strumenti intelligenti per l'integrazione di informazioni provenienti da sorgenti eterogenee all'interno di un'impresa è un argomento di forte interesse in ambito di ricerca. In questo articolo proponiamo tecniche basate su strumenti *intelligenti* per l'estrazione e l'integrazione di informazioni provenienti da sorgenti strutturate e semistrutturate fornite dal sistema MOMIS. Per la descrizione delle sorgenti presenteremo e utilizzeremo il linguaggio object-oriented ODL$_{I^3}$ derivato dallo standard ODMG. Le sorgenti descritte in ODL$_{I^3}$ vengono elaborate in modo da creare un *thesaurus* delle informazioni condivise tra le sorgenti. L'integrazione delle sorgenti viene poi effettuata in modo semiautomatico elaborando le informazioni che descrivono le sorgenti con tecniche basate su *Description Logics* e tecniche di *clustering* generando uno *Schema globale* che permette la visione integrata virtuale delle sorgenti.

1 Introduzione

Nel corso degli ultimi anni le imprese si sono via via dotate di sistemi per l'archiviazione delle informazioni costruendo sistemi informativi contenenti dati correlati tra loro ma spesso ridondanti, eterogenei e non sempre consistenti.
D'altra parte l'esplosione delle reti di calcolatori ha accelerato l'esigenza di condivisione e del reperimento delle informazioni presenti sui singoli sistemi di archiviazione dati, giungendo ad una visione integrata in modo da eliminare incongruenze e ridondanze. L'esigenza è quella di poter accedere *comodamente* a tutte le informazioni aziendali distribuite in diverse basi di dati, oppure poter costruire applicazioni che utilizzano in tempo reale tali informazioni come se si lavorasse su di un'unica base di dati. 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 questo lavoro proponiamo un approccio intelligente all'integrazione delle informazioni che supporta sia sorgenti informative strutturate che semistrutturate. Mostreremo anche, attraverso un esempio, come tale approccio può essere applicato in un impresa manifatturiera producendo una vista globale virtuale delle informazioni contenute nel sistema informativo aziendale.
Allo scopo di presentare le informazioni estratte ed integrate attraverso un linguaggio comune viene introdotto un linguaggio object-oriented, chiamato ODL$_{I^3}$, derivato dallo standard ODMG [1], che utilizza la *Description Logics* OLCD (Object Language with Complements allowing Descriptive cycles) [2, 3].

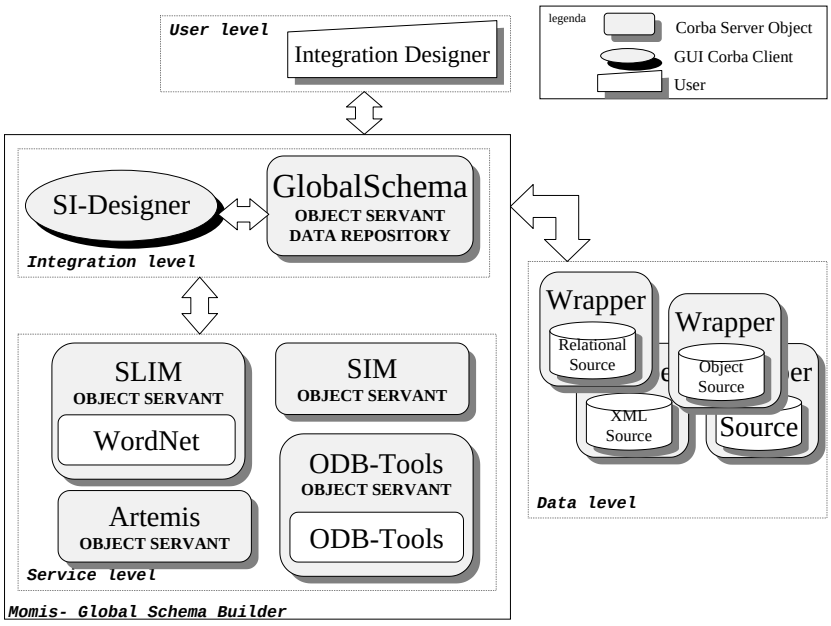

Figura 1: Architettura del sistema MOMIS

Le sorgenti da integrare vengono descritte attraverso il linguaggio ODL$_{I^3}$ e, utilizzando le tecniche di inferenza delle *Description Logics* (DL$_S$), viene definita un'ontologia comune sotto forma di Common Thesaurus. Dopo questa fase iniziale il Thesaurus viene arricchito aggiungendo relazioni interschema ottenute utilizzando il sistema lessicale WordNet [4, 5]. Partendo dal Common Thesaurus l'integrazione di informazioni viene realizzata in modo semi-automatico utilizzando tecniche di clustering e inferenze di OLCD, giungendo alla definizione di uno schema globale che contiene la visione d'insieme delle sorgenti integrate. Questo schema viene utilizzato dagli utenti esterni che, tramite un'architettura a mediatore [6], possono richiedere il reperimento di informazioni presenti su qualunque delle singole sorgenti interessate al processo di integrazione.
Le tecniche descritte sono implementate nel sistema MOMIS [7] (Mediator envirOnment for Multiple Information Sources), inserito all'interno del progetto MURST INTERDATA [8]. Come altri sistemi proposti in letteratura (vedere sezione 8), MOMIS segue un *approccio semantico* proponendo l'integrazione a partire dagli schemi concettuali ed utilizzando il *mediatore* per l'esecuzione delle query. A differenza di altri progetti simili, MOMIS fornisce un effettivo supporto al progettista.

2 L'architettura di MOMIS

MOMIS è 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 [6], in MOMIS si possono distinguere quattro componenti principali per la fase di intergrazione delle sorgenti (Fig. 1):

1. *Wrapper*: posti al di sopra di ciascuna sorgente, sono i moduli che rappresentano l'interfaccia tra il mediatore 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};
- in fase di query processing, traducono la query ricevuta dal mediatore (espressa quindi nel linguaggio comune di interrogazione OQL_{I^3}, definito a partire dal linguaggio OQL) in una interrogazione comprensibile 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: 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 e delle tecniche di clustering;
   - 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 di tecniche delle *Description Logics*, il QM genera automaticamente la traduzione della query sottomessa nelle corrispondenti sub-query delle singole sorgenti.
3. *ODB-Tools Engine*, un tool basato su OLCD [2, 3] che compie la validazione di schemi e l'ottimizzazione di query [9, 10, 11].
4. *ARTEMIS-Tool Environment*, un tool basato sulle tecniche di clustering *affinity-based* che compie l'analisi ed il clustering delle classi ODL_{I^3} [12].

3 Estrazione di informazione da sorgenti eterogenee

Il primo passo nell'estrazione di informazioni è la costruzione di una rappresentazione semanticamente ricca delle sorgenti da integrare attraverso un modello di dati comune. Nell'approccio semantico ció viene realizzato considerando lo schema concettuale delle singole sorgenti.
Per quanto riguarda le sorgenti strutturate convenzionali (database relazionali, object-oriented, file) la descrizione è disponibile e puó essere automaticamente convertita in un modello di dati comune (si veda ad esempio [13]).
Le sorgenti informative semistrutturate (oggi lo standard de facto per sorgenti dati semistrutturate è XML [14]) non definiscono un proprio schema, in quanto, per loro natura sono *autodescrittive*, il che significa che le informazioni sulla struttura della sorgente sono specificate nei dati stessi. In letteratura sono stati proposti modelli per la rappresentazione dei dati semistrutturati [15, 16, 17]. In accordo a questi modelli, la nostra proposta prevede di rappresentare sorgenti semistrutturate come grafi ad albero etichettati, dove i dati semistrutturati sono rappresentati dai nodi e dalle etichette sugli archi. La figura 2 mostra un esempio di sorgente semistrutturata che contiene informazioni riguardanti il magazzino di un'impresa (assumiamo che la sorgente semistrutturata sia conforme con il modello OEM [18, 17] di cui riprenderemo la terminologia).
Un oggetto semistrutturato può essere visto come una tripla del tipo: $(\texttt{id}, \texttt{etichetta}, \texttt{valore})$ dove `id` è l'identificatore, `etichetta` è la stringa che descrive il tipo di oggetto rappresentato e `valore` è il valore dell'oggetto che può essere sia atomico che complesso. I valori atomici possono essere integer, real, string, image, mentre i valori complessi sono insiemi di oggetti semistrutturati, definiti come coppie $(id, etichetta)$.

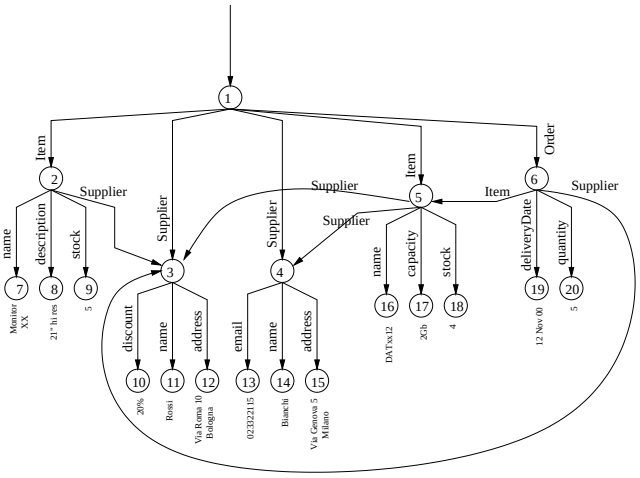

Figura 2: Esempio di magazzino per il commercio (*Magazzino Semistrutturato* - MS)

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 DL_S. 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*.
Data una etichetta l che qualifica un certo numero di oggetti, ogn'uno di questi oggetti può essere complesso e avere come attributi uno o più oggetti a loro volta qualificati da delle etichette. L'object pattern associato a l è dato dall'unione delle etichette che qualificano almeno uno degli oggetti che compongono gli oggetti complessi qualificati da l. Si tratta di un concetto difficile da esprimere a parole ma abbastanza semplice a livello intuitivo. In figura 3 presentiamo la definizione formale di *object pattern*.

Posto set_l un insieme di oggetti definito sulla sorgente semistrutturata S, l'*object pattern* dell'insieme set_l è una coppia nella forma $\langle$l,A$\rangle$, dove l è l'etichetta degli oggetti correlati all'insieme set_l, ed $A = \bigcup$ etichetta(so') tale che esiste almeno un oggetto $so \in \mathrm{set}_l$ con $so \rightarrow so'$.

Figura 3: Definizione formale di *Object Pattern*

Ad esempio, gli *object pattern* per gli oggetti della sorgente di figura 2 sono mostrati in figura 4, dove sono definiti tre *object pattern* diversi: `Item` contenente la descrizione dei prodotti, `Supplier` per le informazioni dei fornitori e `Order` che definisce gli ordini.

Item-pattern	= (Item, { name, Supplier, stock, capacity*, description*})
Supplier-pattern	= (Supplier, { name, address, email*, discount*})
Order-pattern	= (Order, { Supplier, Item, deliveryDate, quantity})

Figura 4: Object patterns per la sorgente semistrutturata

È facile vedere 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, può capitare che le etichette nell'insieme A di un *object pattern* siano definite solo per alcuni oggetti dell'insieme set_l e non per tutti: chiameremo queste particolari etichette come "optional" e le indicheremo con il simbolo "*".
Come detto la descrizione degli *object pattern* segue la semantica di mondo aperto tipica dell'approccio seguito nelle DL_S [19, 20, 21, 22]. Gli oggetti 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*.

3.1 Esempio di riferimento

Introduciamo ora un esempio applicativo che sará utilizzato nel lavoro per illustrare l'approccio seguito. Consideriamo il caso di un'azienda manifatturiera nel campo dei personal computer che ha acquisito due nuove divisioni e che si vuole dotare di un controllo centralizzato della produzione, in particolare ci focalizziamo sulla gestione delle scorte nei vari magazzini.

```
Part(id, name, description, width, height, weight)
Location(building, cell, width, height, wCapacity, partId,
quantity)
exp_Part( SELECT id, name, description, SUM(quantity) as
availability
    FROM Part, Location
    WHERE id = partId )
```

Figura 5: Esempio di magazzino automatico, *relazionale* (MR)

Una delle due nuove divisioni acquisite gestisce i prodotti grazie ad una base di dati semistrutturata (già vista in figura 2) che chiameremo MS. Tale divisione è caratterizzata dalla capacità di gestire ordini a fornitori. L'altra divisione è caratterizzata dal possedere un magazzino automatizzato che è gestito utilizzando una base di dati relazionale che chiameremo MR. In figura 5 è descritto parte del database relazionale per la gestione di tale magazzino automatizzato.
Una terza sorgente, che chiameremo MO, è una base di dati ad oggetti che mantiene la lista di tutti i prodotti trattati dalla multinazionale che ha acquisito i magazzini *MS* e *MR*. Tale database è descritto dal seguente semplice schema:

```
interface Product(  source objectDatabase MO
      extent ProdoctList ){
   attribute string    name;
   attribute string    description;
   attribute integer   price };
```

Nel seguito dell'articolo vedremo come queste sorgenti saranno integrate.

3.2 Il linguaggio ODL_{I^3}

Allo scopo di supportare l'integrazione semantica di informazioni introduciamo un linguaggio object-oriented, chiamato ODL_{I^3}, per la rappresentazione di schemi concettuali e *object pattern*. In accordo alle

raccomandazioni di ODMG e *I³/POB*, ODL$_{I^3}$ riprende le specifiche del linguaggio ODL introducendo alcune estensioni utili per l'integrazione di informazioni e per modellare dati semistrutturati.

Costruttore Union : indicato come `union`, viene introdotto per esprimere come dominio di un attributo due o più strutture dati.

Costruttore Optional : indicato come "*", viene introdotto per gli attributi per specificare l'opzionalitá.

Relazioni Terminologiche : esprimono la conoscenza interschema riguardo gli schemi delle sorgenti in esame. Sono di tipo estensionale [23] e possono essere espresse per classi ed attributi.

- 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 concetto del mondo reale. Un esempio di relazione SYN è dato da (`MS.stock` SYN `MR.availability`).
- BT (broader-term): definita tra due termini t_i e t_j tali che t_i ha un significato più generale di t_j. Con riferimento all'esempio, si puó scrivere (`MO.Product` BT `MR.Part`) e (`MO.Product` BT `MS.Item`). Ciò significa che tutti i prodotti in MS e MR sono descritti anche in MO. La relazione BT non è simmetrica. La relazione opposta a BT è NT (Narrower Term) e si ha: $(t_i \text{ BT } t_j) \Leftrightarrow (t_j \text{ NT } t_i)$.
- RT (related-term): definita tra due termini t_i e t_j che sono generalmente usati nello stesso contesto, tra i quali esiste un legame generico. Per esempio si puó avere la seguente relazione (`Item` RT `Supplier`).

Regole Sono introdotte due tipi di regole: *if then*, per esprimere in forma dichiarativa i vincoli di integritá intra ed inter schema, e regole di *mapping* per esprimere le relazioni esistenti tra lo schema integrato e gli schemi sorgenti.

Vediamo ad esempio come parte delle descrizioni dei magazzini `MS` e `MR` sono tradotte in ODL$_{I^3}$.

```
interface Item(                    interface Part(
      source semistructured MS           source relational MR
      extent OrdersWarehouse  ){         extent AutomatizedWarehouse
   attribute string        name;         key id ){
   attribute set<Supplier> supplier;  attribute string    id;
   attribute integer       stock;     attribute string    name;
   attribute string        capacity*; attribute string    description;
   attribute string        description*;attribute integer  availability;
   };                                 };
```

3.3 OLCD: un formalismo per oggetti complessi e regole di integrità

In questa sezione daremo una descrizione intuitiva di OLCD (per una trattazione formale, vedere [2]). OLCD è un'estensione dell'ODL (*object description language*, da non confondersi con ODL-ODMG) presentato in [3] per modellare oggetti complessi. OLCD contempla un *sistema di tipi base*: *string*, *boolean*, *integer*, *real*; e primitive per la definizione di nuovi tipi: *tuple*, *set* e *class* che permettono di dichiarare tipi complessi o classi di oggetti. I tipi classi di oggetti (i.e. le *classi*) definiscono insiemi di *oggetti con un identificatore ed un valore*, mentre i tipi valore definiscono insiemi di valori anche complessi finitamente annidati. È definito l'operatore *intersezione* per creare intersezioni tra tipi esistenti; questo operatore permette di modellare l'ereditarietà semplice o multipla. L'operatore unione ($\sqcup$) permette l'*unione* tra tipi esistenti e la modellazione della generalizzazione. Ai tipi cosìdefiniti può essere

assegnato un nome. Un tipo con nome può essere: (1) *primitivo* quando la sua descrizione esprime condizioni necessarie e quindi la sua estensione è fornita dall'utente oppure (2) *virtuale* quando la sua descrizione esprime condizioni necessarie e sufficienti e quindi la sua estension è *calcolata*.
Le estensioni a ODL introdotte in OLCD sono: *tipi path*, *regole di integrità* e operatore *union*. La prima estensione è stata introdotta per trattare in modo semplice e flessibile strutture annidate. I *path*, che sono essenzialmente sequenze di attributi, supportano il concetto principale dei linguaggi di interrogazione object-oriented consentendo la navigazione tra le strutture dati definite nelle classi [24]. In particolare, come in [24], vengono forniti *quantified path* per navigare attraverso insiemi di tipi. I quantificatori sono esistenziali e universali e possono apparire più di una volta nello stesso *path*.
La seconda estensione permette di esprimere in modo dichiarativo vincoli di integrità nella forma di regole *if then* definite sul dominio in cui la parte antecedente e la parte conseguente seguono il formalismo del linguaggio.
L'operatore $\sqcup$ dell'OLCD può essere usato per rappresentare la semantica dell'operatore *union* dell'ODL$_{I^3}$. Questo è stato formalizzato in [2] dove $\sqcup$ esprime l'unione di tutti i possibili attributi dei tipi coinvolti.
L'operatore $\sqcup$ è utile anche per definire in OLCD attributi opzionali, lo si può fare definendo l'unione tra un generico attributo e e l'operatore di *tipo indefinito* ($\uparrow$).
Per esempio il *pattern* Item del magazzino semistrutturato è rappresentato in OLCD in questo modo[1]:

σ_P(Item) = $\triangle$ ([name : String] $\sqcap$ [supplier : {Supplier}] $\sqcap$ [stock : Integer] $\sqcap$
([capacity : String] $\sqcup$ capacity$\uparrow$) $\sqcap$
([description : String] $\sqcup$ description$\uparrow$))

Le DL$_S$ (e quindi OLCD) consentono di applicare tecniche di *ragionamento*: calcolo delle relazioni di *sussunzione* tra i tipi (le relazioni *isa*), calcolo dell'*equivalenza* tra tipi e verifica dei tipi *incoerenti* (i tipi sempre vuoti).
Vediamo un esempio di sussunzione nell'ambito degli attributi opzionali: consideriamo i seguenti tipi valore A e B:

σ_V(A) = [att1 : String, att2 : String]
σ_V(B) = [att1 : String $\sqcap$ ((att2 : String) $\sqcup$ att2$\uparrow$)]

Grazie al calcolo della sussunzione tra questi tipi scopriamo che B sussume A (A isa B) anche se questo non è stato esplicitamente dichiarato.
È stato sviluppato un sistema chiamato ODB-Tools [25, 10, 11], basato su OLCD che realizza le citate tecniche di ragionamento. È anche disponibile via internet all'indirizzo [11].

4 Tecniche di ragionamento sulle sorgenti mediante OLCD

Per sviluppare tecniche intelligenti per l'integrazione semantica è molto importante avere a disposizione un'ontologia condivisa delle sorgenti coinvolte. L'ontologia fornisce un vocabolario di riferimento su cui identificare l'eterogeneitá e conseguentemente risolverla. Per ottenere l'ontologia condivisa noi proponiamo di utilizzare le tecniche di ragionamento delle DL$_S$ e il sistema lessicale WordNet per costruire un *Common Thesaurus* di relazioni terminologiche a partire dagli schemi delle sorgenti descritte secondo il linguaggio ODL$_{I^3}$. L'attivitá si svolge secondo i seguenti passi.

4.1 Estrazione di relazioni terminologiche

Utilizzando ODB-Tools viene costruito un *Common Thesaurus* di *relazioni terminologiche*. Le relazioni terminologiche esprimono la conoscenza di inter-schema su sorgenti diverse e corrispondono alle asserzioni intensionali utilizzate in [23]. 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

[1] σ_P and σ_V introducono rispettivamente tipi primitivi e tipi virtuali

coinvolte. Grazie alle tecniche delle DL_S, la determinazione di relazioni terminologiche dagli schemi sorgenti in MOMIS è un'attività semi-automatica che consiste nei seguenti passi:

1. **Estrazione automatica di relazioni dalla descrizione delle sorgenti**.
   Utilizzando ODB-Tools su schemi semanticamente ricchi è possibile estrarre un insieme iniziale di relazioni BT, NT e RT. In particolare, traducendo ODL_{I^3} in descrizioni OLCD ODB-Tools estrae relazioni BT e NT direttamente dalle gerarchie di generalizzazione, ed estrae relazioni RT dalle gerarchie di aggregazione. Altre relazioni di tipo RT sono estratte in sorgenti relazionali dalle definizioni di *chiavi esterne.* Nelle sorgenti semistrutturate sono estratte relazioni RT dalla struttura della sorgente stessa, in particolare da come sono *collegati* tra loro gli oggetti.

   Esempio 1 Consideriamo gli schemi `MS` e `MR`. Le relazioni terminologiche estratte automaticamente da ODB-Tools sono:
   (`MR.Part` RT `MR.Location`)
   (`MS.Item` RT `MS.Supplier`)
   (`MS.Order` RT `MS.Item`)
   (`MS.Order` RT `MS.Supplier`)

2. **Revisione/integrazione delle relazioni**.
   Il progettista dovrà inserire ulteriori relazioni terminologiche che completano quelle estratte automaticamente. Ad esempio, può accadere che in alcune sorgenti (in particolare relazionali e semistrutturate) non siano dichiarate esplicitamente gerarchie di generalizzazione.

   Un aiuto per *scoprire* tali relazioni è fornito da sistemi lessicali (come WordNet). In questo caso sinonimi, relazioni tra termini più generici o più specifici e termini correlati possono essere proposti al progettista basandosi sulle analoghe relazioni dell'ontologia impiegata. Ad esempio la relazione (`MR.Part` SYN `MS.Item`) può derivare da un suggerimento di WordNet, infatti in tale ontologia *Part* appare come sinonimo di *Item* (nel senso *numero 2*).

   Il progettista dovrà inserire anche le relazioni terminologiche che legano tra loro concetti correlati tra sorgenti diverse. Si tratta di relazioni che descrivono il *dominio* delle sorgenti.

   Esempio 2 Supponiamo che in questa fase vengano definite le seguenti relazioni che correlano classi ed attributi delle sorgenti `MS`, `MR` e `MO`:

   (`MS.Item` SYN `MR.Part`)
   (`MS.Item.name` SYN `MR.Part.name`)
   (`MS.Item.description` SYN `MR.Part.description`)
   (`MS.Item.stock` SYN `MR.Part.availability`)
   (`MO.Product` BT `MS.Item`)

3. **Validazione delle relazioni**.
   La validazione è basata sulla *compatibilità dei domini* associati agli attributi.

   Dati due attributi[2] $a_t = (n_t, d_t)$ e $a_q = (n_q, d_q)$ sono svolte le seguenti verifiche:

   - se $(n_t$ SYN $n_q)$: la relazione è considerata valida se d_t e d_q sono equivalenti oppure un dominio è una specializzazione dell'altro.
   - se $(n_t$ BT $n_q)$: la relazione è considerata valida se d_t contiene o è equivalente a d_q.

[2] Assumiamo che un attributo a sia definito come una coppia $a = (n, d)$ dove n è il nome dell'attributo e d è il dominio esplicitato come *tupla di tipi semplici.*

- se $(n_t$ NT $n_q)$: la relazione è considerata valida se d_t è contenuto o è equivalente a d_q.

Quando un dominio è definito tramite l'operatore `union` la relazione è valida quando almeno un dominio è compatibile. Tale controllo è effettuato da ODB-Tools tramite calcolo della sussunzione e calcolo delle equivalenze.

4. **Calcolo di nuove relazioni**.
In questo passo sono utilizzate le capacità di calcolo inferenziale di ODB-Tools. A partire dallo *schema virtuale* definito grazie alla fase di revisione/integrazione degli schemi vengono calcolate tutte le possibili relazioni di sussunzione e aggregazione. Tali relazioni sono poi tradotte in relazioni terminologiche che arricchiscono il thesaurus. Il thesaurus contente sia le relazioni esplicite (definite dal progettista) che quelle calcolate prende il nome di *Common Thesaurus*. In figura 6 è riportato il *Common Thesaurus* nel caso del nostro esempio. Sono presenti linee piene e tratteggiate che rappresentano, rispettivamente, relazioni esplicite e calcolate.

Figura 6: Esempio di Common Thesaurus

5 Clustering basato sull'affinità delle classi ODL$_{I^3}$

La fase di clustering permette di ottenere lo schema globale integrato partendo dalla conoscenza presente nel *Common Thesaurus*. Vengono identificate in modo semi automatico classi semanticamente simili raggruppate in cluster. Per questo scopo viene calcolato un insieme di *coefficienti di affinità* (i.e. valori numerici compresi in [0,1]) per ciascuna coppia di classi basandosi sulle relazioni (validate) del *Common Thesaurus*. I coefficienti di affinitá definiscono il grado di relazione semantica tra due classi sulla base del loro nome (*Name Affinity coefficient*) e dei loro attributi (Structural Affinity coefficient). La combinazione lineare di questi coefficienti definisce infine il *Global Affinity coefficient*. Il *Global Affinity coefficient* viene utilizzato da un algoritmo di clustering gerarchico [26] per classificare le classi sulla base della loro affinitá. Il risultato finale della procedura di clustering è l'*albero di affinitá* nel quale le classi ODL$_{I^3}$ rappresentano le foglie e ciascun nodo definisce il grado di affinitá delle classi sottostanti. Le procedure di clustering sono fornite dal tool ARTEMIS [27, 28]. In figura 7 si vede il semplice cluster generato per il nostro esempio dei magazzini.

6 Costruzione dello schema globale

Lo schema globale costituisce il punto di accesso ai dati delle sorgenti integrate da parte degli utenti esterni. Le classi dello schema sono definite a partire dai cluster presenti nell'*albero di affinità*: la generazione di tali classi (denominate g_{c_i}) è interattiva e supportata dal modulo Global Schema Builder.

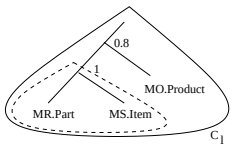

Figura 7: Esempio di semplice cluster gerarchico

Dato un cluster Cl_i di classi, MOMIS genera automaticamente gli attributi della singola classe globale come unione degli attributi delle classi del cluster. Gli attributi aventi un coefficiente di affinità non nullo e validato sono *unificati* in un unico attributo.
MOMIS richiede al progettista di inserire interattivamente le seguenti informazioni:

- *Nome della classe globale*
- *Mapping*
  si tratta di definire le corrispondenze tra gli attributi della classe globale e gli attributi delle singole classi aggregate. Poichè si tratta di sorgenti eterogenee MOMIS mette a disposizione i costruttori:
  - *and* per indicare, ad esempio, che l'attributo `nome` della classe globale corrisponde alla concatenazione degli attributi `cognome` e `nome` di una data classe sorgente.
  - *union* (con significato simile alla *union* del linguaggio C) per risolvere il problema in presenza di attributi incompatibili, ad esempio il campo `data` potrebbe essere specificato come struct di interi o come campo `longinteger`. La *union* permette di trattare entrambi i casi.
- *Valori di default*
  Si tratta dei valori associati agli attributi non sempre presenti nelle varie sorgenti.
- *Nuovi attributi*
  Si tratta di attributi definiti solo nella classe globale che non hanno corrispondenza nelle classi locali. Sono utili per il query processing.

```
interface Item {
  attribute name mapping_rule
      MO.Product.name, MS.Item.name, MR.Part.name;
  attribute description mapping_rule
      MO.Product.description, MS.Item.description,
      MR.Part.description;
  attribute stock  mapping_rule
      MS.Item.stock, MR.Part.availability;
  attribute price  mapping_rule
      MO.Product.price;
  attribute type  mapping_rule
      MO.Product = "multinational",
      MS.Item = "supplier warehouse",
      MR.Part = "automatized warehouse";
}
```

Figura 8: Dichiarazione della classe globale `Item`

In figura 8 è riportato un esempio di definizione di una classe globale attraverso il linguaggio ODL_{I^3}. Per la sintassi completa si veda [29]. Tale dichiarazione genera la seguente *mapping table* che sarà utilizzata in seguito dal *Query Manager* per tradurre una query globale in sotto-query eseguibili dalle sorgenti.

Source	name	description	stock	price	type
MS.Item	name	description	stock	null	'supplier warehouse'
MR.Part	name	description	availability	null	'automatized warehouse'
MO.Product	name	description	null	price	'multinational'

In cui potrebbero apparire, ad esempio, i seguenti dati:

Source	name	description	stock	price	type
MS	Monitor XX	20" Hi Res	5	null	'supplier warehouse'
MS	DATxx12	null	4	null	'supplier warehouse'
MR	Fd14	Floppy dsk	99	null	'automatized warehouse'
MR	Hdxx7	Hard dsk	8	null	'automatized warehouse'
MO	CPU700	CPU 500Mhz	null	500	'multinational'
MO	VC16	Video card	null	133	'multinational'

7 Esecuzione e ottimizzazione delle Query

Vediamo brevemente come MOMIS ottimizza e risponde alle interrogazioni. Quando un utente invia una query a MOMIS il modulo *Query Manager* (QM) produce un insieme di sotto-query da inviare alle sorgenti coinvolte dalla query. Ciò avviene attraverso le fasi di *ottimizzazione semantica* e *formulazione del piano di interrogazione*.
Nella fase di Ottimizzazione semantica sono utilizzate le tecniche di ottimizzazione semantica fornite da ODB-Tools. Grazie a tale strumento viene generata una query equivalente in cui sono esplicitate tutte le condizioni applicabili alla query in grado di rendere l'accesso più selettivo (e quindi più efficiente) derivanti dalla conoscenza delle sorgenti (regole di integrità e struttura dello schema). L'ottimizzazione avviene aggiungendo alla clausola `where` della query tutte le possibili condizioni applicabili ad essa [9]. Quindi, utilizzando la tavola di mapping associata ad ogni classe globale, il QM riformula la query adattandola ai singoli schemi locali. Per ottenere le query locali il QM verifica e controlla ogni fattore presente nella clausola `where` della query. MOMIS cerca di generare le query per le sorgenti strettamente necessarie alla corretta esecuzione della query globale. Cerca cioè di prevedere quali sorgenti, se interrogate, non ritornerebbero nessuna istanza. Ciò è possibile confrontando i fattori booleani della query con i valori (di default) specificati nella mapping table. Vediamo un semplice esempio:

```
SELECT name FROM Item WHERE price=133;
```

Il QM riconosce che l'unica sorgente ad avere `price` non *null* è `MO.Product` e quindi l'unica sotto-query generata sarà:

```
SELECT Item.name FROM MO.Product WHERE price=133;
```

8 Confronto con altri lavori

Lavori correlati al progetto MOMIS sono identificati nell'area dei semistructured data e dell'integrazione di informazioni eterogenee.
Dati semistrutturati. La problematica della modellazione dei semistructured data è stata particolarmente investigata in letteratura. In particolare, un *survey* inerente ai problemi del *semistructured data modeling querying* è stato presentato in [15]. Due modelli simili per semistructured data sono stati proposti in [16,

17], e sono basati un grafo etichettato, in cui i nodi sono gli oggetti (i dati) e gli archi le etichette. Nel modello presentato in [16], l'informazione risiede solamente nelle etichette, mentre nel modello "Object Exchange Model" (OEM) proposto da Papakonstantinou et. al. in [17], l'informazione risiede anche nei nodi.
Anche la prospettiva di aggiungere struttura a dati semistrutturati, visione vicina alla nostra proposta di *object pattern*, è stata studiata in letteratura. In particolare, in [30] viene proposta la nozione di *dataguide* come una *descrizione lasca dei dati* memorizzati nella sorgente informativa. In [31] viene presentata una nuova nozione di schema per dati semistrutturati utilizzando un modello dati con grafo etichettato proposto in [16, 17]. Un altro approccio è stato proposto in [32], dove l'utilizzo di tecniche euristiche e regole di classificazione vengono utilizzate per estrarre una gerarchia di tipi dai dati semistrutturati.
In questo articolo abbiamo approfondito l'aspetto della traduzione degli *object pattern* in *Description Logics*, piuttosto che occuparci degli algoritmi di estrazione; le tecniche proposte nei lavori citati possono essere applicate per l'estrazione di object pattern.
Integrazione di informazioni eterogenee. In quest'area sono stati sviluppati molti progetti basati sull'architettura a mediatore [33, 34, 35, 36, 37, 38, 13].
Seguendo la classificazione dei sistemi di integrazione proposti da Hull [39], MOMIS è fra i sistemi che seguono il *virtual approach*. *Virtual approach* è stato proposto all'inizio degli anni '80 nei modelli multidatabase [40, 41]. Più recentemente, alcuni sistemi sono stati sviluppati basandosi sulle DL_S [33, 42], 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 sulle DL_S 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 [43].
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 a MOMIS sono: GARLIC, SIMS, Information Manifold e Infomaster.
Il progetto GARLIC [38, 13] fornisce un'architettura con wrapper complessi che forniscono la descrizione delle sorgenti in linguaggio OO (chiamato GDL) e viene quindi definito manualmente uno schema globale che unifica la visione delle sorgenti locali tramite gli oggetti Garlic Complex Objects.
Il progetto SIMS [33, 34] propone di creare la definizione dello schema globale utilizzando le *Description Logics* (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 qualsivoglia natura basate sullo schema invece di quelle predefinite dal sistema.
Information Manifold Systems [44, 42], alla stregua del progetto MOMIS, fornisce un mediatore e una query manager indipendente 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 [42, 45]. 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, 47] 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.
Un'altra proposta basata sulle DL_S e sulle tecniche di *reasoning* è descritta in [48], dove gli autori propongono un *declarative approach* utilizzando *intermodel assertions* per definire interrelazioni tra concetti

presenti in sorgenti differenti. Le *intermodel assertions* possono essere definite sia a livello intensionale che estensionale. Nella proposta la definizione dello schema globale (*Enterprise Model*) viene attuata manualmente dal progettista.

9 Conclusioni

In questo articolo è stato presentato un approccio *intelligente* per l'estrazione e l'integrazione di informazioni a partire da sorgenti dati diverse di un'azienda. Si tratta un approccio semantico basato su di un componente di *Description Logics* (ODB-Tools), un modulo per la generazione di cluster (ARTEMIS) ed un linguaggio d'*interfaccia* ODL_{I^3}. La generazione di uno schema globale per il mediatore viene proposta come un processo semi-automatico. Utilizzando lo schema globale, il modulo Query Manager genera il piano di accesso per l'esecuzione delle query distribuite sui singoli sistemi. Il Query Manager si avvale di tecniche di ottimizzazione semantica per ridurre la complessitá ed i tempi di recupero delle informazioni e fornire all'utente una risposta.

Riferimenti bibliografici

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

[2] 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.

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

[4] 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.

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

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

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

[8] MOMIS staff. Momis (mediator environment for multiple information sources) project part of the interdata project. Available at http://sparc20.dsi.unimo.it/Momis.

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

[11] ODB-Tools staff. Odb-tools Project. Available at http://sparc20.dsi.unimo.it.

[12] 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.

[13] 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, 1997.

[14] The World Wide Web Consortium (W3C). Extensible markup language (xml). Available at http://www.w3.org/XML/.

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

[16] 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.

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

[18] 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.

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

[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] 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.

[22] 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.

[23] 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.

[24] E. Bertino, M. Negri, G. Pelagatti, and L. Sbattella. Object-oriented query languages: The notion and the issues. *IEEE Trans. Knowl. and Data Engineering*, 4(3):223–236, June 1992.

[25] D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. Odb-tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. In *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997.

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

[27] 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.

[28] S. Castano, V. De Antonellis, M.G. Fugini, and B. Pernici. Conceptual schema analysis: Techniques and applications. *ACM Transactions on Database Systems*, 23(3):286–332, 1998.

[29] 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/.

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

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

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

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

[42] 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.

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

[44] 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.

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

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

[47] 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.

[48] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. A schema analysis and recociliation tool environment for heterogeneous databases. In *Proceedings of the Sixt International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13, 1998.