Parole chiave:
P2P
Peer Data Management Systems
Mapping semantici
Condivisione di dati

RINGRAZIAMENTI

Desidero ringraziare la Professoressa Sonia Bergamaschi per il prezioso aiuto fornitomi durante lo svolgimento della tesi, il dott. ing. Francesco Guerra e l'ing. Daniele Montanari per la loro disponibilità.

Un sincero ringraziamento va alla mia famiglia per il loro costante sostegno durante il percorso di studi e a tutti i miei amici, in particolare al Drinking Team (Manuel, Tiziano, Elis, Giacomo, Eddy, Bigna, Loris, Guido, Lory, Luca, Fulgido e Roberto), a Lele e al "quasi sindaco" Paolo Negro per tutti quei momenti in cui non mi hanno fatto pensare allo studio.

Non potrei non ringraziare tutti i miei compagni di corso e di studio, tra cui Matteo, Matte, Matteooooo!!!, Mareo e lo Squatter, Elena, Mauri, Enri, Riva, Tania, Robby, From, Musso e Fabiana con i quali ho condiviso gioie e fatiche in questi anni di studio.

Desidero infine ringraziare alcune persone, fatti o eventi, senza i quali questi tre anni non sarebbero stati gli stessi: Pietro e Martina, le avventure con la mitica Audi 80 dell'Elena, le fermate alla Lasa Metalli, Biagio Antonacci e Laura Pausini, le file lungo il Canaletto, il criceto di Matteo, i Francis Petra e la loro hit "Datti Fuoco", il Genio e i "Vignolesi", il bar Juve, i senatori del CICALA, tutti quelli che "li puoi trovare sempre in BSI", il Prete, il Papà e i suoi amici, Sorriso, Limite, il tipo "Scusatesehoilcollopiccolo" e i sosia di Giuliano Ferrara e Giacomo Leopardi.

A tutti un enorme GRAZIE!!!

Andrea Galavotti

INDICE

CAPITOLO 3

CAPITOLO 4

INDICE DELLE FIGURE

INTRODUZIONE

Il Peer-to-Peer (P2P) consiste di una rete aperta e finita di elementi distribuiti (come PC, notebook, PDA, e altri ancora), chiamati *peer* (pari), i quali possono scambiare dati e servizi con un insieme di altri peer, denominati *acqaintance* (conoscenti o vicini), tipicamente senza l'intervento di alcun server centrale, senza alcun registro globale degli appartenenti alla rete o la possibilità di usufruire di servizi globali, cioè senza la presenza di un gestore o di un controllo globale delle risorse. In tali sistemi, gruppi di computer condividono le proprie risorse (file, spazio di memorizzazione, cicli di computazione macchina) in modo da gestire in maniera non costosa delle operazioni che normalmente richiederebbero di server potenti e con grandi possibilità di memorizzazione delle informazioni. La loro architettura è completamente diversa da quelle classiche client-server: nei sistemi P2P un peer richiede un servizio ad una altro peer, che a sua volta può richiedere dati ad un terzo nodo, ovvero ogni peer può essere client e server a seconda che richieda un servizio o lo soddisfi, in quanto un'applicazione su un nodo interagisce con tutte le altre con le quali è in connessione.

Dalla loro prima introduzione (verso la fine degli anni '90), questi sistemi si sono affermati molto rapidamente, grazie alle caratteristiche che li contraddistinguono dai tradizionali sistemi client-server: adattamento, assenza di un'amministrazione centrale, disponibilità dei dati attraverso le replica degli

stessi, possibilità di condividere una grande quantità di risorse. Tra i sistemi esistenti Napster[4] e Gnutella [1], ma in generale tutti quelli che permettono la condivisione di file (in particolare musicali, come gli mp3), sono i più popolari e quelli che hanno diffuso il paradigma P2P. In realtà esistono molte altre applicazioni diverse da quelle del file sharing, come ad esempio applicazioni di computazione (Seti@Home [8]) e di collaborazione (Groove [2]).

Tuttavia, le dimensioni degli attuali sistemi P2P, la loro natura "aperta e dinamica" e la mancanza di un controllo centralizzato pongono delle sfide soprattutto relativamente alle prestazioni e alla sicurezza.

Nell'ambito delle Basi di Dati, i principali aspetti di ricerca sono relativi alla gestione e alla ricerca di dati in reti P2P (i quali sono male implementati nella maggior parte dei sistemi P2P oggi disponibili; in Napster, ad esempio, la ricerca dei dati si basa sulla conoscenza di due soli campi: "artista" e "titolo").

Tali dati devono essere gestiti attraverso appositi sistemi (definiti Peer Data Management Systems - PDMS) che permettono la condivisione di dati eterogenei in modo distribuito e scalabile.

Lo scopo di questa tesi è, dunque, quello di studiare alcuni dei principali progetti di ricerca nell'ambito della gestione dei database nei sistemi P2P. Come vedremo, i dati devono essere gestiti attraverso appositi sistemi (definiti Peer Data Management Systems - PDMS) che permettono la condivisione di dati eterogenei in modo distribuito e scalabile. Normalmente, per il numero di

sorgenti trattate, tale condivisione si basa sulla definizione di mapping semantici tra gli elementi delle sorgenti coinvolte, piuttosto che sulla realizzazione di uno Schema Globale sintesi dei dati delle sorgenti coinvolte.

Verranno inoltre studiati la caratterizzazione logica di tali sistemi e varrà analizzato il loro funzionamento.

Il testo è composto dai seguenti 4 capitoli:

1 Panoramica sui Peer Data Management Systems (PDMS)
Questo capitolo è un' introduzione ai PDMS: si vuole puntare l'attenzione sui loro punti di forza e sulle difficoltà intrinseche nella loro realizzazione.

In conclusione al capitolo viene data un'ampia introduzione sui principali sistemi studiati in questa tesi.

2 Modello logico per la caratterizzazione dei P2P database system
Questo capitolo introduce una formalizzazione dei linguaggi di mapping e delle semantiche in alcuni dei sistemi descritti nel primo capitolo. Viene inoltre proposto un esempio di mapping semantici per chiarirne il significato.

3 Analisi del sistema Bibster
Studio delle caratteristiche e del funzionamento dei sistemi P2P tramite Bibster, un sistema per la condivisione e lo scambio di metadati bibliografici. In conclusione al capitolo verrà svolto un test del sistema, consistente nel sottoporgli un insieme di query.

4 Conclusioni

Quest'ultimo capitolo rappresenta una riflessione sul lavoro svolto; vengono inoltre proposti alcuni temi di ricerca sui quali concentrare il lavoro futuro.

Capitolo 1

PANORAMICA SUI **P**EER **D**ATA **M**ANAGEMENT **S**YSTEMS (**PDMS**)

In questo primo capitolo vengono fornite le conoscenze di base sui PDMS e sulle loro caratteristiche, necessarie per la comprensione del resto del testo. Il capitolo si conclude con una panoramica su alcuni dei principali progetti di ricerca attualmente attivi nell'ambito della gestione dei dati nei sistemi P2P.

1.1 Che cosa sono i PDMS?

I *Peer Data Management System* (PDMS) sono sistemi distribuiti di gestione delle informazioni appartenenti alle reti Peer-to-Peer (P2P).

Un ambiente P2P è costituito da più nodi (individui, società, organizzazioni, dipartimenti, e altro ancora), o peer, che mettono in comune i propri dati e le proprie risorse rendendole disponibili all'intera comunità. Non esistono componenti che hanno una conoscenza globale delle informazioni del sistema o che possono interegire globalmente con esso; al contrario, ogni peer agisce localmente e autonomamente, sulla base delle informazioni che possiede sullo stato della rete: in questo senso si dice che il controllo della rete è distribuito sui nodi che la compongono (Gnutella [1] ne è un esempio). Infatti, ogni nodo interagisce con i propri vicini al fine di ottenere le informazioni desiderate.

I PDMS nascono dunque dalla necessità di coordinazione e interazione fra i database della rete P2P; essi consistono in un insieme finito di sorgenti informative e devono permettere la condivisione di dati eterogenei in modo distribuito e scalabile, condivisione che non può basarsi sulla realizzazione di uno Schema Globale sintesi dei dati delle sorgenti coinvolte [11, 20] come proposto dai sistemi di integrazione dei dati noti in letterature, ma che si basa sulla definizione, come vedremo in seguito, di mapping semantici tra gli elementi delle sorgenti.

L'obbiettivo dei PDMS è proprio quello di soddisfare questo bisogno: l'utilizzo di una struttura decentralizzata, facilmente estendibile nella quale ogni utente può apportare nuove informazioni e risorse (come, ad esempio, creare nuovi schemi che gli altri utenti possono usare, aggiungere informazioni sugli schemi esistenti e regole di mapping con altri schemi), con l'intento di sostituire al singolo schema una collezione di mapping semantici tra gli schemi dei singoli database dei peer appartenenti alla rete.

1.2 Caratteristiche principali dei PDMS

I PDMS possono avere caratteristiche diverse dovute alle scelte progettuali compiute. In generale un buon PDMS dovrebbe presentare le seguenti caratteristiche:

- ***Decentralizzazione:*** caratteristica per la quale non esiste uno Schema Globale per tutti i database della rete P2P. Al contrario ogni

peer deve essere in grado di instaurare i propri "acquaintance" (vicini). Tale proprietà porta interessanti benefici a tutto il sistema P2P: grazie ad essa infatti la robustezza del sistema, le sue prestazioni e la disponibilità di risorse crescono con il numero di peer che partecipano al sistema stesso. Inoltre la decentralizzazione elimina gli interessi proprietari sulle infrastrutture del sistema, in quanto non si sente il bisogno di amministrazione e non esistono infrastrutture dedicate da gestire.

- ***Autonomia:*** legata alla precedente; indica la capacità del sistema di operare senza interventi dall'esterno.

- ***Replica e consistenza dei dati:*** le stesse informazioni possono essere presenti in più database, il ché può migliorare le prestazioni generali del sistema, in quanto aumenta la disponibilità dei dati e riduce la possibilità che si verifichino eventuali colli di bottiglia. Si rendono necessari però dei processi di aggiornamento dei dati che rappresentano un grosso problema per gli attuali PDMS (§ 1.3).

- ***Uguaglianza dei nodi:*** non esiste un nodo "centrale" sul quale fare affidamento per qualche servizio.

- ***Scalabilità:*** è la proprietà che rappresenta l'abilità della rete e di ogni singolo nodo di continuare a funzionare in modo efficiente anche quando le sue dimensioni aumentano notevolmente. Questa caratteristica, come si può capire, è fondamentale per i PDMS che si

trovano a gestire dati in reti con migliaia o milioni di utenti. Infatti, nei sistemi P2P decentralizzati, ogni nodo gestisce le comunicazioni individualmente: questo può causare dei tempi molto lunghi per le operazioni di recupero dei dati e, quindi, provocare problemi di traffico nella rete.

- ***Supporto alla dinamicità dei partecipanti:*** ogni utente del sistema ha la possibilità di lasciare il sistema quando vuole. Per questo motivo i peer connessi in un dato istante al sistema possono non essere sempre gli stessi e nemmeno nello stesso numero. Anche se durante il processo di risposta ad una query la topologia della rete cambia (perché nuovi peer si sono collegati o perché sono scomparse o sono state create nuovi mapping semantici), un PDMS deve essere in grado di fornire comunque una risposta all'interrogazione posta.

- ***Supporto all'eterogeneità delle sorgenti:*** i partecipanti ad una rete P2P possono avere database con schemi differenti gli uni dagli altri. Caratteristica principale di ogni PDMS è quella di supportare questa condizione definendo delle regole di mapping che collegano tra loro i database della rete qualsiasi schema presentino (rifiuto di un singolo Schema per tutti i partecipanti).

Non tutte queste caratteristiche appartengono a tutti i PDMS nello stesso modo; come vedremo in seguito alcuni sistemi possono prevedere l'utilizzo di

qualche nodo particolare, cioè di peer con funzioni che lo distinguono dagli altri peer (§1.4, [20]).

1.3 Problematiche da affrontare nella realizzazione dei PDMS

Le caratteristiche menzionate nel paragrafo precedente sono tutte desiderabili in un PDMS, ma la loro realizzazione porta a dei problemi che non sono tutti risolti dai sistemi commerciali attualmente disponibili o risolti nello stesso modo.

I principali problemi che affliggono gli attuali sistemi P2P commerciali nascono dal fatto che essi sono carenti nelle aree riguardanti la semantica e le relazioni tra i dati. Oggigiorno, tali aree sono soggetto di studio da parte di numerosi ricercatori che cercano di risolvere il problema dell'allocazione fisica dei dati (data placement problem) attraverso le tecniche di gestione dei dati.

Di seguito sono riportati i principali problemi riguardo la localizzazione dei dati e il loro recupero, oltre i problemi di sicurezza e di qualità del servizio.

1.3.1 Problemi causati dalla dinamicità dei partecipanti

Come detto nel paragrafo precedente, una delle caratteristiche principali dei PDMS è quella di permettere la dinamicità dei peer. In realtà questa proprietà è fonte di diversi problemi riguardo la consistenza, in quanto, in certi momenti, risulta impossibile accedere a certi nodi. Inoltre, questa caratteristica comporta un meccanismo di recupero dei dati più complesso.

1.3.2 Problemi causati dall'eterogeneità delle sorgenti

In un sistema P2P possono partecipare nodi che presentano schemi diversi gli uni dagli altri: in questo contesto risulta estremamente complesso fornire un'integrazione semantica globale delle sorgenti. Infatti, anche se tali sorgenti condividono un certo insieme di concetti comuni, non è detto che essi siano espressi con le stesse relazioni o con gli stessi vocaboli. Le cause di queste diversità semantiche sono da ricercare nella diversa concettualizzazione della realtà che progettisti diversi possono fornire e dall'utilizzo di diversi modelli per la rappresentazione della realtà in questione (ad esempio, determinate relazioni presentano strutture diverse a seconda che si utilizzi un modello relazionale o uno relazionale/ad oggetti).

1.3.3 Problemi legati all'ottimizzazione del recupero dei dati

Questo problema consiste nel sapere, al fine di ottimizzare i processi di risposta alle query, quali nodi contengono viste che possono velocizzare il processo in corso e quindi migliorare le prestazioni del sistema. Esistono diverse alternative alla soluzione di questo problema, come viene illustrato in [20]: l'utilizzo di un catalogo centralizzato che contenga i riferimenti ai nodi cercati, l'utilizzo di un'organizzazione gerarchica e distribuita (come ad esempio il DNS), oppure l'utilizzo di tecniche derivate dai protocolli di routine (in realtà le prime due soluzioni sono ancora poco adatte alla dinamicità e alla scalabilità; la più convincente è la terza alternativa).

1.3.4 Problemi causati dall'estensione dei sistemi P2P

Aumentare le dimensioni del sistema, e quindi la mole delle informazioni che esso gestisce, ha dei benefici, come ad esempio l'aumento della qualità del servizio (Quality of Service - QoS), ma contemporaneamente provoca maggiori complessità nei meccanismi di aggiornamento e recupero dei dati. Gli stessi problemi si manifestano anche nel caso di un alto grado di replica delle informazioni su diversi peer.

1.3.5 Relazioni tra autonomia, efficienza e robustezza dei meccanismi di ricerca

Questo problema, discusso in [15], riguarda il meccanismo di ricerca dei dati: in pratica si afferma che esistono dei legami tra autonomia, efficienza e robustezza del sistema, dove per efficienza si intendono i risultati ottenuti in base alle risorse consumate (come banda e memoria), mentre per robustezza si intende la stabilità del sistema in presenza di errori,che possono compromettere le prestazioni per quanto riguarda la ricerca dei dati nel sistema. In particolare in [15] si afferma che l'autonomia del meccanismo di ricerca è correlata sia con l'efficienza che con la robustezza dello stesso. Per quanto riguarda il binomio autonomia/efficienza, si sostiene che una maggiore autonomia implica una minore efficienza e viceversa: riuscire a disaccoppiare queste due proprietà significherebbe avere meccanismi di ricerca che siano contemporaneamente autonomi, efficienti e robusti. Il secondo binomio, quello tra autonomia e robustezza, risulta infatti essere

meno problematico da disgiungere (Gnutella [1], ad esempio, presenta sia un alta autonomia che un'alta robustezza [15]).

1.3.6 Problemi di sicurezza

I sistemi P2P hanno una natura aperta ed autonoma, in quanto ogni peer può apportare nuovi dati al sistema: si viene così a formare un nuovo problema da risolvere riguardante la sicurezza delle informazioni che appartengono alla rete [9]. Gli aspetti principali di questo problema riguardano:

- *Disponibilità dei dati e delle risorse:* ogni nodo della rete deve essere in grado di comunicare con gli altri nodi per poter accedere alle risorse che essi mettono a disposizione; questo servizio può essere danneggiato da attacchi *Denial-of-Service* (DoS), nei quali un nodo riempie la banda disponibile di un altro nodo con informazioni inutili. In questo caso, tutte le risorse offerte dal nodo attaccato sono indisponibili per tutta la rete P2P. Infatti, un nodo può attaccare direttamente qualsiasi particolare risorsa offerta dagli altri peer; ad esempio, la disponibilità di CPU in un nodo può essere attaccata spedendogli un numero anche modesto di query complesse, oppure la disponibilità di memoria può essere attaccata mandando al nodo documenti falsi o inutili da memorizzare.

- *Autenticità dei dati:* il problema è sapere se i dati forniti in risposta ad una query sono effettivamente autentici o se sono stati alterati da altri

utenti malintenzionati, ovvero: data una query e un insieme di documenti in risposta, quali di essi sono risposte "autentiche" per la query? L'autenticità dei documenti può essere valutata secondo diversi meccanismi: la copia di un documento considerato autentico può essere quella più vecchia (*oldest document*); oppure quella considerata tale da un nodo "esperto" o autoritario (meccanismo *expet-based*); un terzo meccanismo prevede che il documento autentico è quello votato da un certo numero di nodi considerati "esperti" (meccanismo *voting-based*); infine, esiste un meccanismo simile al precedente, dove però i voti dei nodi "esperti" vendono pesati in base alla reputazione del nodo stesso (meccanismo *reputation-based*).

- *Anonimia:* riguarda il preservare l'anonimia dell'autore di un documento o di un suo usufruttuario, allo scopo di evitare eventuali persecuzioni o censure. Il problema consiste nel garantire l'anonimia preservando però altre caratteristiche importanti del sistema, come l'efficienza e la decentralizzazione di un meccanismo di ricerca. Ad esempio, consideriamo il vincolo tra anonimia dei peer e efficienza della ricerca dei dati: se vogliamo garantire l'anonimia dei peer, diventa difficile determinare quali nodi memorizzano un documento, mentre se vogliamo avere un efficiente sistema di ricerca delle informazioni, dobbiamo sapere esattamente quali nodi contengono i dati cercati. Esistono diverse forme di anonimia: *autore*, un autore di un documento non può essere identificato, *editore*, l'editore di una

pubblicazione non può essere identificati, *lettore*, chi usufruisce delle informazioni non può essere identificato, *server*, non è possibile identificare i server dove i dati sono memorizzati basandosi sui dati stessi, *documento*, i server non conoscono i documenti che gestiscono, *query*, un server non può riferire informazioni riguardo i documenti che utilizza per rispondre alle query.

- *Controllo di accesso:* alcuni documenti possono avere una natura privata, e quindi non devono essere disponibili per gli altri peer; altre informazioni potrebbero essere accessibili solo da un gruppo ristretto di peer.

1.4 Stato dell'arte nell'ambito della ricerca dei PDMS

Nell'area della gestione dei dati nei sistemi P2P, sono stati sviluppati diversi progetti di ricerca, alcuni dei quali sono: coDB [17, 19], Piazza [23], LRM [13, 24], SON [14] e SWAP [16].

1.4.1 Il sistema *coDB*

Il sistema **coDB**, sviluppato presso l'Università di Bolzano [17, 19], considera i nodi della rete come un insieme di database relazionali, possibilmente con schemi diversi, e si propone di interconnetterli applicando regole di mapping semantico a livello di schema dette *coordination rules*: ogni nodo può scambiare dati con i propri vicini se fra di essi si è definita una o più coordination rule. Il

tipo di mapping tra due schemi è *Global-Local-As-View*, anche detto *GLAV*; ciò significa che la testa di una coordination rule è una query congiuntiva che si riferisce a qualche relazione locale in un dato nodo, e il corpo è un'altra query congiuntiva la quale si riferisce a qualche relazione in un acquaintance. coDB è un sistema robusto nel senso che supporta la dinamicità della rete: se durante il processo di risposta ad una query dei nodi o delle coordination rule scompaiono o appaiono, l'algoritmo termina in ogni caso con una risposta all'interrogazione posta. Inoltre tale sistema differisce da taluni altri PDMS anche per consentire delle regole di mapping cicliche che comportano una topologia ciclica della rete P2P, e per avere un processo di aggiornamento dei singoli database dei peer (detto *global update*) che usa tutte la definizioni di coordination rule che essi mantengono: tale processo inizia quando un nodo pone una richiesta di aggiornamento ai propri vicini, contenente le definizioni delle appropriate coordination rule; questi peer rispondono alla query e inviano il *global update* ai rispettivi vicini, e così via. La propagazione della richiesta di aggiornamento termina in un nodo quando esso non ha vicini o quando ha già ricevuto questo messaggio. Per distinguere le diverse richieste di aggiornamento, il nodo nel quale la richiesta viene generata ha il compito di aggiungere al messaggio un identificatore univoco.

Per quanto riguarda l'architettura (che si rifà a quella dell'LRM – [LRM]), mostrata in figura 1, un nodo è costituito da un P2P Layer, da un Local Database (LDB) e da un Database Schema (DBS) che ha il compito di descrivere quelle parti di LDB condivise per gli altri nodi. Il primo consiste di

una User Interface (UI), con la quale l'utente ha accesso a tutte le funzioni del sistema, di un Database Manager (DBM), di un JXTA Layer [5] e di un Wrapper.

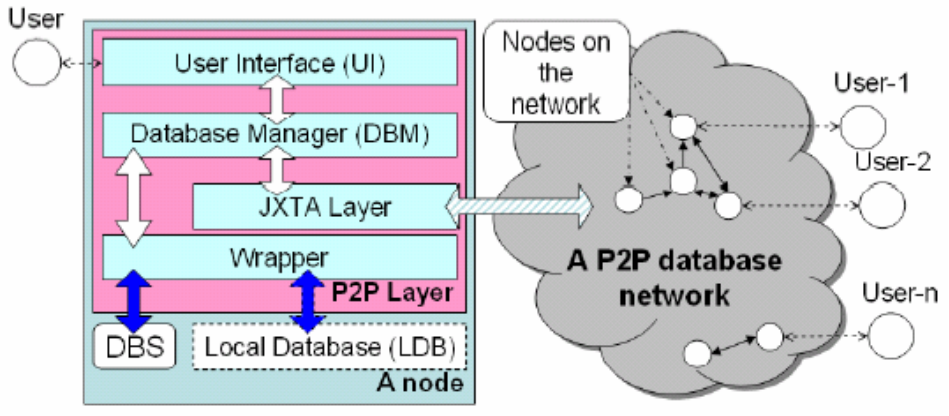

Figura 1. Architettura di un nodo coDB

La UI permette all'utente di formulare le query e le richieste di aggiornamento, di visualizzare i dati in risposta ad una query, di avviare le procedure di esplorazione della topologia della rete, e altro ancora. DBM è responsabile per la propagazione delle query, delle richieste di aggiornamento e dei risultati delle query e degli aggiornamenti nella rete; inoltre esso elabora le query (e le loro risposte) e le richieste di aggiornamento (e le loro risposte) poste dall'utente o in arrivo dagli altri nodi. JXTA Layer è responsabile di tutte le attività del nodo nella rete: creazione di coordination rule con altri peer, esplorazione della topologia della rete, invio e ricezione di messaggi, ecc. Il Wrapper ha il compito di gestire le connessioni con LDB e di eseguire le operazioni di manipolazione dei database. LDB rappresenta l'RDBMS del node: in coDB viene usato MySQL. Il modulo DBS è essenziale per

consentire ad un nodo di partecipare attivamente alla rete. Infine, come si nota dalla figura, le frecce nel nodo sono di colore diverso, in quanto esse denotano funzionalità differenti: le frecce bianche indicano procedure invocate tra diversi moduli, quelle blu indicano che la comunicazione dipende dal tipo di LDB; le frecce che collegano il JXTA Layer con la rete indicano che la comunicazione è supportata da JXTA.

1.4.2 Il sistema *Piazza*

Il sistema **Piazza**, sviluppato all'interno dell'Università di Washington [23], è un PDMS per dati XML [9] che si focalizza sulla definizione di integrazione fra gli schemi e tecniche di mapping fra i sistemi P2P. La sua architettura è gerarchica, e i suoi nodi sono autonomi e possono contribuire con nuovi dati con schemi; è presente anche un nodo centrale che ospita un indice necessario per il routing delle query e per la riformulazione delle stesse. La riformulazione delle query è eseguita dal nodo che riceve le query. Ogni nodo ha uno schema XML, detto *peer schema*, che descrive il modo in cui il peer vede i dati che il sistema gli mette a disposizione: il peer schema è indipendente dallo schema dei dati che il nodo contiene, quindi si rende necessario un ulteriore livello di mapping. I nodi che apportano nuove informazioni al sistema hanno anche un secondo schema, detto *storasge schema*, il quale descrive la struttura dei dati. In questo contesto, ogni query formulata da un peer è posta in base al peer schema del nodo, e riformulata in base allo storage schema dei peer della rete rilevanti per la query. Per questo motivo

Piazza prevede due tipi di mapping: *peer description*, che relaziona due o più peer schema, e *storage description*, la quale mappa i dati contenuti in un peer all'interno del peer schema. Come in molti altri PDMS, anche in Piazza non esiste uno schema centralizzato, ma le riformulazioni delle query sono eseguite usando solo peer description e storage description. Peer description e storage description vengono espresse tramite relazioni di inclusione con delle viste, in modo da poter utilizzare tecniche GAV e LAV per la riformulizzazione delle query sulle viste, e sono contenute in un nodo centrale.

Il linguaggio utilizzato per le query e per i mapping si basa sul linguaggio XQuery: l'intento è quello di definire un linguaggio di mapping che adotti elementi di XQuery, ma più semplice da usare. I mapping nel linguaggio sono definiti come uno o più mapping definition fra una sorgente e una destinazione; essi consistono nel prelevare un frammento dello schema della destinazione e annotarlo con le espressioni di XQuery che definiscono quali dati della sorgente devono essere mappati nel frammento. I risultati dei vari mapping definition sono combinati per formare un mapping completo dal documento sorgente a quello di destinazione.

1.4.3 Local Relational Model (*LRM*)

Il modello **LRM** (Local Relational Model), sviluppato presso l'Università di Trento [13, 24], assume che l'insieme dei dati nella rete consiste in database locali relazionali, ognuno con un set di acquaintance, i quali definiscono la

topologia della rete P2P. In questo sistema, le *domain relation* definiscono le regole di transizione tra dati, mentre le *coordination formula* definiscono le dipendenze semantiche tra due database (cioè dicono come i dati in un peer sono in relazione con i dati in un acquaintance). Tali formule possono anche agire come vincoli o come strumenti per la propagazione degli aggiornamenti. La semantica in LRM è definita in termini di *relational space* ognuno dei quali modella lo stato dei database nella rete p2p. Un relational space è una coppia <db, r>, dove db è un insieme di database relazionali su I (dove I è un insieme di indici distinti) e r è una funzione che associa ad ogni $i,j \in I$ un domain relation r_{ij} da i a j. Per quanto riguarda l'architettura (figura 2), deve essere uguale per tutti i nodi. I due elementi principali sono LRM Layer e il Local Information Source (LIS). Il primo è composto da: User Interface (UI), la quale permette agli utenti di impostare le query, ricevere risultati e messaggi dagli altri peer e controllare gli altri moduli dell'LRM Layer; Query Manager (QM) e Update Manager (UM), responsabili della propagazione delle query e degli aggiornamenti; Wrapper, il quale fornisce uno strato di traduzione tra QM, UM e LIS. Le comunicazioni tra peer e tra QM e UM avvengono usando messaggi XML, mentre la comunicazione tra wrapper e LIS dipende dal LIS (SQL, HTTP,…). Le strategie per la propagazione delle query e degli update sono raggruppate in un insieme di *coordination rule* (o ECA rule). Queste regole descrivono quando, dove e come una query o un update devono essere propagati.

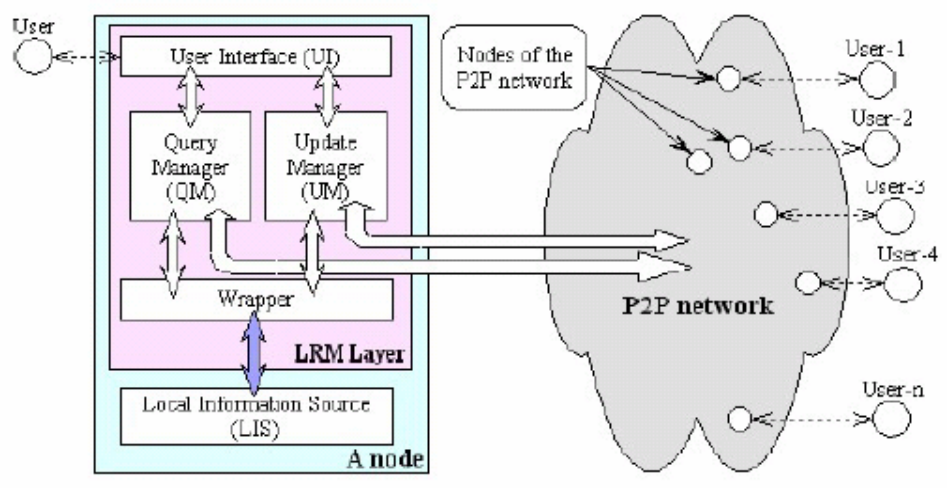

Figura 2. Architettura di un nodo

1.4.4 Semantic Overlay Networks (*SON*)

Le **Semantic Overlay Networks** (SONs), progetto sviluppato all'Università di Stanford [14], sono reti che raggruppano nodi con contenuto semanticamente simile; ogni nodo può appartenere ad una o più SON. In un sistema di questo tipo le query sono eseguite identificando quali SON contengo i dati richiesti; successivamente la query viene spedita ad un nodo di questi SON e mandata poi a soli altri componenti del SON stesso. Più SON sono organizzate in una gerarchia: l'obbiettivo è quello di definire un insieme di SON tali per cui, data una richiesta, si possa selezionare un numero ristretto di essi i cui nodi presentano un alto numero di hit, in modo da ridurre il tempo di risposta ad una richiesta (in quanto i nodi selezionati hanno molti hit) e di consentire ai nodi con poche risposte di poter servire più velocemente altre richieste (in quanto la query in questione non gli viene inviata).

Il processo di formazione e di utilizzo dei SON è il seguente:

- si trova una buona gerarchia in base all'attuale distribuzione dei dati;
- i nodi lanciano un *document classifier* per classificare i loro documenti;
- il *node classifier* assegna i nodi ai SON; questo processo è realizzato sulla base della classificazione dei documenti del nodo;
- quando viene posta una query, essa viene indirizzata ai SON appropriati; i nodi all'interno dei SON trovano i risultati usando un qualsiasi meccanismo di propagazione.

Per quanto riguarda la classificazione dei documenti e delle query, i progettisti hanno scelto un processo ibrido (in parte manuale e in parte automatico) per la classificazione dei primi e quello manuale per la classificazione delle query. Infine, la classificazione dei nodi usata è tale che un nodo appartiene ad un determinato SON se un numero significativo dei suoi documenti rientra in quel SON; questa scelta è prevalsa sull'approccio che prevede che un nodo appartiene ad un determinato SON se ha almeno un file classificato in quel SON.

1.4.5 Il progetto *SWAP*

SWAP (Semantic Web and Peer-to-Peer) è un progetto europeo sviluppato presso l'Università di Karlsruhe [16], che si prefigge di superare i problemi derivati dalla combinazione del paradigma P2P con le tecnologie del Semantic Web per creare un supporto agli ambienti decentralizzati. Dal punto di vista

architettonico, ogni nodo, chiamato **SWAP Node**, consiste nei seguenti componenti, mostrati in figura 3:

Sorgenti di Conoscenza (Knowledge Sources): i peer possono avere sorgenti di informazioni locali, come il file system locale, le e-mail directory o le liste di bookmark. Tali sorgenti rappresentano il corpo delle conoscenze di un peer; in esse vengono memorizzate tutte la informazioni che il nodo condivide con la rete.

Integratore della Conoscenza della Sorgente (Knowledge Source Integrator): sono i responsabili per l'estrazione e l'integrazione delle sorgenti di conoscenza interne ed esterne, nel Deposito Locale del Nodo (*Local Node Repository*). Questo processo, che si basa sul modello di metadati di SWAP (*SWAP Metadata Model*), inizia con l'accesso alle sorgenti di conoscenza locali per estrarre una rappresentazione RDF [6] della conoscenza memorizzata (*extraction*); successivamente vengono selezionati gli oggetti RDF da integrare nel deposito locale del nodo (*selection*); nel terzo passaggio (*annotation*) tali oggetti vengono annotati con i metadati (vengono aggiunti gli oggetti "Swabbi"); a questo punto, gli oggetti che si trovano nel deposito locale che si riferiscono alle stesse risorse, ma che provengono da peer diversi, possono avere nomi diversi: lo scopo dell'ultimo processo (*merging*) è quello di incorporare gli oggetti nell'ontologia dell'utente, cioè si vuole associare a questi oggetti lo stesso nome.

Deposito Locale del Nodo (Local Node Repository): il suo compito è quello di gestire un modello integrato della conoscenza che esiste localmente

per il peer stesso e remotamente sugli altri peer della rete; questa conoscenza può essere rappresentata come file system, e-mail, database, ontologie e altre ancora. Per poter includere le diverse sorgenti di informazione, viene estratta una rappresentazione RDF delle stesse.

Interfaccia Utente (User Interface): permette di avere accesso alla conoscenza disponibile nelle sorgenti locali come nel resto della rete.

Informatore (Informer): ha il compito di scoprire quali sorgenti di conoscenza sono momentaneamente disponibili nella rete.

Controllore (Controller): è quel componente che controlla il processo di distribuzione delle query: esso riceve le query dall'interfaccia utente e le distribuisce ai peer in base al loro contenuto.

Adattatore della Comunicazione (Communication Adapter): è il componente responsabile per la comunicazione fra i peer; agisce come uno strato di trasporto per le altre parti del sistema.

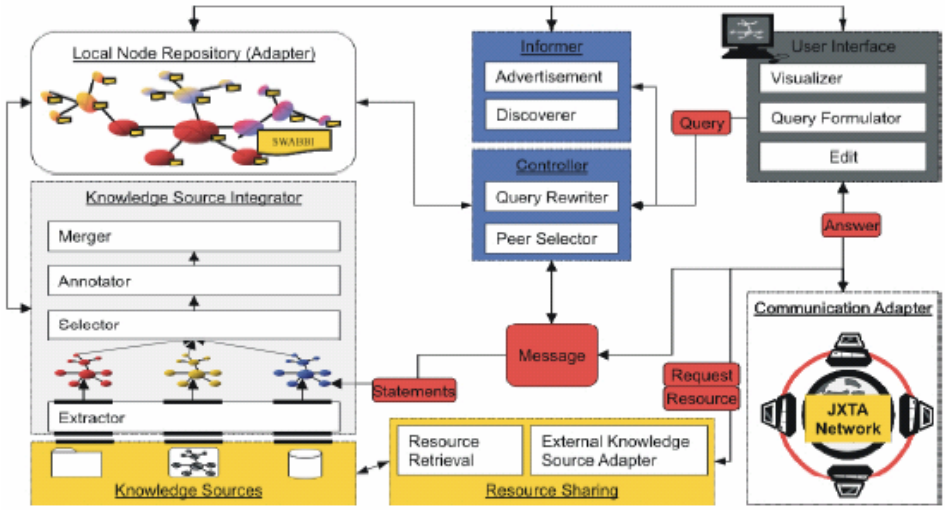

Figura 3. Architettura di un nodo SWAP

Il processo di invio di query e ricezione della risposta è il seguente : la query, dall'interfaccia utente, viene passata al controllore il quale, esaminato il suo contenuto, la distribuisce ai nodi che conosce. Quando un peer riceve la query da un altro peer, prova a rispondere o, in caso non abbia informazione che soddisfa la query, la rispedisce ad altri nodi che vengono decisi dal *Peer Selector* in base ai peer che il nodo conosce. Le risposte ricevute sono trasferite all'interfaccia e all'integratore della conoscenza della sorgente, il quale integra le risposte nel deposito locale.

Capitolo 2

MODELLI LOGICI PER LA CARATTERIZZAZIONE DEI P2P DATABASE SYSTEM

2.1 Introduzione

Scopo di questo capitolo è quello di fornire una comprensione sulle caratteristiche e proprietà dei modelli logici che caratterizzano i database management system nelle architetture P2P.

Come già affermato più volte, i sistemi P2P sono composti da nodi indipendenti che possono avere database eterogenei. Diventa, quindi, necessaria una coordinazione fra di essi, cioè quella capacità dei peer di gestire le dipendenze semantiche tra i loro database in un ambiente decentralizzato, distribuito e dinamico, necessaria per impostare lo scambio dei dati fra coppie di database. Chiameremo le formule che legano in modo semantico gli schemi di due peer *coordination rule* (regole di coordinazione). Il compito delle coordination rule è quello di creare un mapping diretto tra i nodi in modo da stabilire come trasferire i dati da un nodo sorgente a un nodo di destinazione.

Introduciamo un semplice esempio. Si considerino una rete P2P e due nodi, il primo con la sorgente relazionale **University**, e il secondo con la sorgente ad oggetti **Computer_Science**. La prima (UNI) descrive la base di dati di una Università, mentre la seconda (CS) la base di dati di un dipartimento di Computer Science, e si suppone che i database delle due sorgenti siano

composti esclusivamente da una relazione ciascuno (nel § 2.3 è proposto l'esempio completo):

```
UNI: Article(year, subject, title, journal, article_code)

CS: Publication(year, title, authors)
```

Supponiamo inoltre che i due database siano interconnessi dalle seguenti coordination rule:

(1) University.Article.year SYN Computer_Science.Publication.year

```
UNI.Article.year → CS.Publication.year
```

che esprime la sinonimia (SYN) dell'istanze `year` nelle due relazioni `Article` e `Publication`.

(2) University.Article.title SYN Computer_Science.Publication.title

```
UNI.Article.title → CS.Publication.title
```

che esprime la sinonimia dell'istanze `title` nelle due relazioni.

Essendo la relazione di sinonimia bilaterale, sono valide anche le coordination rule opposte:

```
CS.Publication.year → UNI.Article.year
```

per la relazione (1), e

```
CS.Publication.title → UNI.Article.title
```

per la relazione (2).

In questo esempio, le coordination rule mostrano che `year` in `Publication` e `year` in `Article` (lo stesso vale per `title`) indicano lo stesso concetto e quindi possono essere interscambiate nei due database.

Nel prossimo paragrafo verranno introdotte le principali caratteristiche dei modelli logici e verranno analizzati i modelli di alcuni dei sistemi presentati nel Capitolo 1, dando tutte le definizioni necessarie per la loro comprensione. Nell'ultimo paragrafo viene presentato un esempio completo di coordination rule.

2.2 Modello logico di un PDMS

I modelli logici possono essere definiti sulla base di diverse caratteristiche che li contraddistinguono. Essi si basano, nella maggior parte, sulla definizione di regole di mapping che, come detto, forniscono un legame tra gli schemi dei database. Le differenze consistono in come questi mapping sono implementati: ad esempio nel *Local Relational Model* (LRM), le *coordination formula* oltre ad esprimere query, hanno anche il compito di guide per gli aggiornamenti, o di esprimere vincoli tra i nodi in uno "spazio relazionale". Gli autori del sistema *coDB*, al contrario, hanno studiato un sistema di mapping che, pur simile a quello di LRM, consente solo di soddisfare il trasferimento dei dati da una sorgente ad un destinatario al momento della richiesta. Inoltre, alcuni sistemi prevedono regole di mapping cicliche (è il caso di coDB): in essi, la propagazione delle query è più difficoltosa che nei

sistemi con mapping aciclici (Piazza); in ogni caso, in quasi tutti i sistemi studiati, le regole di mapping sono sempre instaurate dai nodi in modo autonomo, senza l'intervento di amministratori dall'esterno.

Un'altra importante caratteristica per un PDMS è quella di non propagare l'inconsistenza locale di un database. Infatti, in un sistema P2P con migliaia di nodi, è molto probabile che alcuni di essi contengano dati inconsistenti. Se un modello logico non prevede questa possibilità, allora l'intero database della rete sarebbe inconsistente.

Infine, tali modelli logici sono sostanzialmente differenti da quelli dei sistemi classi di integrazione dei dati. Questi sistemi [11, 21], infatti, forniscono un'interfaccia uniforme alle varie sorgenti informative attraverso uno "schema di mediazione" (*mediation schema*) dove le regole di mapping sono definite tra tale schema e le relazioni nelle sorgenti. In tali sistemi vengono proposti inoltre due tipi di mapping: **Global-as-View** (**GAV**), dove le relazioni nello schema di mediazione sono definite come viste sulle relazioni nelle sorgenti, e **Local-as-View** (**LAV**), nelle quali le relazioni nelle sorgenti sono definite come viste su quelle nello schema.

Nei moderni sistemi P2P, invece, il mapping è locale, cioè fra nodi (non esiste uno schema di mediazione), e si tende ad introdurre un nuovo formalismo, detto **Global-Local-as-View** (**GLAV**), il quale concilia i due precedenti (GAV e LAV). Con questo approccio, sia il capo che il corpo di una regola di mapping possono contenere query congiuntive. Se tra due nodi sono presenti

una o più regole di questo tipo, allora un nodo può essere interrogato sul suo schema per delle informazioni che può prelevare dai suoi vicini.

Sistemi come coDB [18] e LRM [24] seguono questo tipo di approccio, mentre il sistema Piazza [23] presenta qualche eccezione.

Studieremo infine il modello dei metadati di SWAP [16], il quale si basa sulla definizione di ontologie, realizzate utilizzando il linguaggio RDF, utilizzate per attribuire un significato preciso ai dati gestiti. Tale modello si distingue dai precedenti per essere basato su RDF Schema [7] e per essere, quindi, un modello informale.

2.2.1 Modello logico alla base del sistema coDB

Gli autori di questo sistema hanno formalizzato un modello logico di PDMS che si basa sulle seguenti definizioni.

Database Locale (**Local Database**). *Sia I un insieme non vuoto e finito di indici {1, 2,…, n} e C un insieme di costanti. Per ogni coppia di distinti i,j ∈ I, sia* L_i *una logica di primo ordine senza simboli di funzione, con firma disgiunta da* L_j *ma per le costanti condivise C. Un* database locale DB_i *è definito come una teoria sul linguaggio di primo ordine* L_i.

Ogni database è interconnesso tramite "regole di coordinazione" che consentono di trasferire dati tra un nodo e i suoi vicini. Tali regole sono definite come:

Regole di Coordinazione (**Coordination Rules**). *Una* regola di coordinazione *è un'espressione della forma*

$$j_1 : b_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \ldots \wedge j_k : b_k(\mathbf{x}_k, \mathbf{y}_k) \rightarrow i : h(\mathbf{x})$$

dove $j_1, \ldots, j_k$ *e* i *sono sono indici distinti (che si riferiscono a distinti database), ogni* $b_l(\mathbf{x}_l, \mathbf{y}_l)$ *sono formule di* L_{jl} *,* $h(\mathbf{x})$ *è una formula di* L_i *, e* $\mathbf{x} = \mathbf{x}_1 \cup \ldots \cup \mathbf{x}_k$.

Le coordination rule possono contenere query congiuntive sia nel capo che nel corpo: in questo senso il tipo di mapping è GLAV. Inoltre esse possono essere cicliche.

In questo contesto un **sistema P2P** può essere definito come una *tupla MDB* $= \langle LDB, CR \rangle$ *, dove* $LDB = \{ DB_1, \ldots , DB_n \}$ *è un insieme di database locali, e CR è l'insieme di coordination rule.*

In questo sistema, come nella maggior parte dei sistemi P2P, le coordination rule vengono interpretate solo come relazioni necessarie per lo scambio di dati fra un peer e i suoi vicini, e non per computazione logica. In quest' ultimo caso, infatti, si potrebbero verificare delle situazioni di errore [18].

Per quanto riguarda la semantica delle coordination rule, vengono formalizzati due approcci: *semantica globale* e *semantica locale.*

La **semantica globale** (**global semantics**) introduce la nozione di un modello globale il quale lavora sopra i singoli linguaggi dei nodi. Essa viene definita come:

sia Δ *un insieme non vuoto di oggetti, incluso C, e sia MDB un sistema P2P. Un'interpretazione di MDB su* Δ *è un n-tupla* $m \equiv \blacksquare m_1, m_2, \ldots, m_n \blacksquare$ *dove ogni* m_i *è un classica interpretazione FOL di* L_i *sul dominio* Δ *che interpreta costanti come se stesse. Se m è un'interpretazione, allora* m_i *denota l'i-esimo elemento di m.*

Un modello globale *M per MDB,* $M \models_{global} MDB$*, è un insieme non vuoto di interpretazioni tale che:*

1. *il modello soddisfa localmente le condizioni di ogni database, cioè*

$$\forall m \in M.(m_i \models DB_i)$$

2. *e il modello soddisfa le coordination rule al meglio, cioè, per ogni coordination rule*

$$j_1 : b_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \ldots \wedge j_k : b_k(\mathbf{x}_k, \mathbf{y}_k) \rightarrow i: h(\mathbf{x})$$

e per ogni assegnamento a, cioè l'assegnamento delle variabili **x**

agli elementi di Δ*, si ha la seguente relazione:*

$$(\forall m \in M.(m_{j1} \models \exists \mathbf{y}.b_1(\mathbf{x}_1, \mathbf{y})) \wedge \ldots \wedge (m_{jk} \models \exists \mathbf{y}.b_k(\mathbf{x}_k, \mathbf{y}))) \rightarrow$$

$$(\forall m \in M.(m_i \models h(\mathbf{x})))$$

La semantica P2P per le coordination rule afferma quindi che se il corpo di una coordination rule è vero in qualsiasi modello possibile del nodo sorgente, allora il capo della regola è vero in qualsiasi modello possibile del nodo di destinazione.

In questa semantica la "risposta ad una query" (**query answer**) è definita nel seguente modo:

*sia Q_i(**x**) una query locale con variabili* **x**; *l'*insieme delle risposte *di Q_i è l'insieme delle sostituzioni di* **x** *con costanti* **c**, *tali che ogni modello M do MDB soddisfa la query, cioè*

$$\{\mathbf{c} \in C \times \ldots \times C \mid \forall M.(M \models_{global} MDB) \rightarrow \forall m \in M.(m_i \models Q_i(\mathbf{c}))\}$$

La risposta ad una query è dunque l'insieme di tuple di quei valori che, sostituiti alle variabili delle query, rendono la query vera per ogni modello globale ristretto al nodo stesso.

Nella **semantica locale** (**local semantics**), invece, i modelli dei sistemi P2P hanno una natura centrata sul nodo. Prima di definire la semantica locale, diamo la definizione di "modello locale derivato".

Modello locale derivato (**Derived local model**). *Il* modello locale derivato $\boldsymbol{M_i}$ *è l'unione degli i_esimi componenti di tutti i modelli di MDB:*

$$\boldsymbol{M_i} = \bigcup_{\substack{m \in M, \\ M \models_{global} MDB}} m_i$$

La tupla ⟨ $\mathbf{M}_1, \ldots, \mathbf{M}_n$ ⟩ rappresenta il modello del sistema P2P.

La **semantica locale** viene definita come:

un modello locale *M per MDB, $M \models MDB$, è una sequenza $<M_1, \ldots, M_n>$ tale che:*

1. *ogni M_i è un insieme non vuoto di interpretazioni di L_i su Δ*
2. $\forall m_i \in M_i.(m_i \models DB_i)$

3. *per ogni coordination rule*

$$j_1 : b_1(\mathbf{x}_1, \mathbf{y}_1) \wedge \ldots \wedge j_k : b_k(\mathbf{x}_k, \mathbf{y}_k) \rightarrow i : h(\mathbf{x})$$

e per ogni assegnamento a alla variabile **x**, *si ha la seguente relazione:*

$$(\forall m_{j1} \in M_{j1}.(m_{j1} \models \exists \mathbf{y}.b_1(\mathbf{x}_1, \mathbf{y}))) \wedge \ldots \wedge$$

$$(\forall m_{jk} \in M_{jk}.(m_{jk} \models \exists \mathbf{y}.b_k(\mathbf{x}_k, \mathbf{y})))) \rightarrow$$

$$(\forall m_i \in M_i.(m_i \models h(\mathbf{x}))$$

In questo caso, la **risposta ad una query** è definita come:

*sia Q_i una query locale; l'*insieme delle risposte *di Q_i è l'insieme delle sostituzioni di* **x** *con costanti* **c**, *tali che ogni modello M do MDB soddisfa la query, cioè*

$$\{\mathbf{c} \in C \times \ldots \times C \mid \forall M.(M \models_{global} MDB) \rightarrow$$

$$\forall m_i \in M_i.(m_i \models Q_i(\mathbf{c}))\}$$

In realtà, si nota che l'insieme delle risposte ad una query locale nella semantica locale coincide con l'insieme delle risposte ad una query nella semantica globale.

Gli autori di coDB introducono in realtà anche un terzo tipo di semantica, detta "semantica autoepistemica" (**autoepistemic semantics**) che non approfondiamo.

I modelli semantici precedentemente introdotti non definiscono però l'inconsistenza locale di un database; ciò significa che in tali modelli, definiti in

questa maniera, l'inconsistenza locale implica l'inconsistenza globale del sistema. Per eliminare questa improprietà, si estende la semantica locale (**semantica locale estesa** - **extended local semantics**) affermando che M_i può essere l'insieme vuoto. In questo modo si riesce a definire l'inconsistenza di un database locale: un database locale DB_i è inconsistente se M_i è vuoto per ogni modello del sistema P2P. Quindi un database che è collegato tramite coordination rule ad un database inconsistente preserva la sua consistenza; viene così garantita la consistenza globale del sistema anche in presenza di un'inconsistenza locale.

La rete P2P viene invece descritta tramite le definizioni di *dependency edge*, *dependency path* e *maximal dependency path*. Su queste definizioni si basa anche la descrizione del comportamento dinamico della rete.

Dependency edge. *E' presente un* dependency edge *da un nodo i ad un nodo j se esiste almeno una regola di mapping con capo al nodo i e il corpo al nodo j.*

Dependency path. *Un* dependency path *per un nodo i è un percorso* $\langle i_1,i_2,\ldots,i_n\rangle$ *di dependency edge, tale che:*

1. $i_1 = i$;
2. $\blacksquare\ i_1,\ldots,i_{n-1}\ \blacksquare$ *è un percorso semplice, cioè nessun nodo appare due volte.*

Maximal dependency path. *Un* maximal dependency path *per un nodo i è un dependency path tale che se si aggiunge un qualsiasi nodo al percorso, il risultato non è più un dependency path.*

Dalla definizione di dependency edge si capisce che la direzione di questi edge è opposta alla direzione delle formule di coordinazione; mentre infatti il verso di tali formule è il verso nel quale i dati vengono trasferiti, i dependency edge presentano la direzione inversa.

Come detto, in coDB è permesso ad ogni nodo di abbandonare e rientrare nella rete in ogni momento: questo comporta che la topologia della rete cambi, cioè comporta la rimozione o la creazione di formule di coordinazione tra il nodo scomparso (o ricollegato) e i suoi vicini. Il cambiamento della topologia della rete è implementato tramite due operazioni:

- *addLink(i,j,rule,id)*: aggiunge la formula di coordinazione *rule* di nome (unico) *id* dal nodo *j* (corpo della formula) al nodo *i* (capo della formula).
- *deleteLink(i.j.id)*: rimuove la formula di nome *id* tra i nodi *i* e *j*.

Quindi un cambiamento μ della rete altro non è che una sequenza di operazioni addLink e deleteLink su MDB, mentre un sottocambiamento μ_A di μ rispetto ad $A \subset LDB$ è un insieme di operazioni atomiche di μ, relativo ad A, e ordinate con lo stesso ordine in μ.

Si possono dare ora le seguenti definizioni:

Risposta certa (**Sound answer**). *Una* sound answer *di una query Q in una rete soggetta a cambiamenti in runtime con riguardo al cambiamento* μ *della rete, è una risposta alla query che è inclusa nel risultato che otterremmo se eseguissimo tutte le operazioni addLink prima di propagare Q, e non eseguissimo nessuna operazione deleteLink.*

Risposta completa (**Complete answer**). *Una* complete answer *di una query Q in una rete soggetta a cambiamenti in runtime, è una risposta alla query che contiene il risultato che ci aspettiamo di ottenere se eseguissimo tutte le operazioni di deleteLink prima di lanciare Q, e non eseguissimo nessuna operazione addLink.*

2.2.2 Modello logico di LRM

In questo modello, dal quale hanno preso spunto gli autori di coDB, ogni database è supposto essere relazionale. Il modello logico di questo sistema si basa sulla definizione degli "spazi relazionali" (*relational space*) i quali modellano lo stato dei database della rete P2P. Prima di poter definire correttamente uno spazio relazionale, è necessario dare le seguenti definizioni.

Linguaggio relazionale (**Relational language**). *Un* linguaggio relazionale *è un linguaggio di primo ordine L con un insieme finito di costanti dom, senza simboli di funzione e con un insieme finito* **R** *di predicati.*

A esempio, il linguaggio della sorgente **University**, descritta nel paragrafo precedente, contiene il simbolo relazionale `Article` e i predicati `year`, `subject`, `title`, `journal` e `article_code`.

Database relazionale (**Relational database**). *Un* database relazionale *è un'interpretazione m di primo ordine di un linguaggio relazionale L sull'insieme delle costanti dom, tale che m(d) = d per tutte le costanti d di L, dove d è un generico elemento della tupla* $\mathbf{d} = \langle d_1, ..., d_n \rangle$ *appartenete a dom.*

Grazie a questa definizione è possibile ora capire se un database si trova nello stato completo, incompleto o inconsistente; in particolare db_i è:

- completo se $|db_i|=1$, cioè se la sua cardinalità è uguale ad 1; ad esempio $db_1 = \{m_1\}$;
- incompleto se $|db_i|>1$; ad esempio $db_2 = \{m_2, m_3\}$;
- inconsistente se $db_i = \varnothing$.

Questi concetti di consistenza/inconsistenza valgono solo a livello locale, in quanto in LRM non esiste una nozione di consistenza globale per un insieme di database locali (ovvero quei database membri di un insieme di database coordinati).

Viene presentata ora la definizione delle "relazioni di dominio" le quali definiscono le regole di trasferimento dei dati.

Relazioni di dominio (**Domain relations**). *Siano L_i e L_j due linguaggi relazionali con dominio dom_i e dom_j rispettivamente. Una* relazione di dominio *r_{ij} da i a j è un qualsiasi sottoinsieme di $dom_i \times dom_j$, cioè del prodotto cartesiano fra i due domini.*

Ciò significa che, una domain relation r_{ij} rappresenta l'abilità del database j di trasferire, e rappresentare nel proprio dominio, dati dal database i; inoltre, nella maggior parte dei casi, esse non sono simmetriche.

Queste relazioni permettono di rappresentare la sovrapposizione (*overlap*) tra più database appartenenti alla rete. Tale sovrapposizione consiste nel fatto che le stesse informazioni possono essere rappresentate in due o più database:

questo significa che entità denotate da simboli diversi in diversi database possono essere relazionate.

Spazio relazionale (**Relational space**). *Uno* spazio relazionale *è una coppia <db, r>, dove db è un insieme di database locali su I (con I={1,2,...,n}) e r è una funzione che associa ad ogni i,j ∈ I, una domain relation* r_{ij} *da i a j.*

Da questa definizione si comprende che uno spazio relazionale è una collezione di database interconnessi da un insieme di mapping: tale definizione coincide con quella di *sistema P2P* nel sistema coDB.

Formule di coordinazione (**Coordination formulas**). *L'insieme delle* formule di coordinazione *CF nella famiglia dei linguaggi relazionali* $\{L_i\}$*, con* $i \in I$*, è definita nel seguente modo:*

$$CF ::= i:\varphi \mid CF \rightarrow CF \mid CF \wedge CF \mid CF \vee CF \mid \forall i:x.CF \mid \exists i:x.CF$$

dove φ *è una formula su* L_i*, e* x *è un generico elemento della tupla di variabili* $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$.

Le formule di coordinazione esprimono le dipendenze semantiche tra database locali, ovvero spiegano come i dati in un peer devono essere relazionati ai dati in un suo vicino; esse adottano un linguaggio indipendente dai linguaggi supportati dai singoli database. Inoltre, in LRM le coordination formula possono essere usate sia per esprimere vincoli che devono essere soddisfatti da uno spazio relazionale, sia per esprimere query ; in quest'ultimo caso una coordination formula viene interpretata come una regola deduttiva

che deriva da nuove informazioni basate su quelle già esistenti in altri database.

i-query. *Una* i-query *definita sulla famiglia di linguaggi relazionali* $\{L_i\}$*, con* $i \in I$*, è una coordination formula del tipo* $A(\mathbf{x}) \rightarrow i{:}q(\mathbf{x})$*, dove* $A(\mathbf{x})$ *è una coordination formula, q è un nuovo predicato n-ario di* L_i *e* $\mathbf{x}$ *contiene n variabili.*

Risposta globale ad una i-query (**Global answer to an i-query**). *Sia* $<db, r>$ *uno spazio relazionale su* $\{L_i\}$*, con* $i \in I$*; la* risposta globale ad una i-query *del tipo* $A(\mathbf{x}) \rightarrow i{:}q(\mathbf{x})$ *in* $<db, r>$ *è l'insieme:*

$$\{\mathbf{d} \in (dom)^n \mid <db, r> \models \exists i : \mathbf{x}.(A(\mathbf{x}) \wedge i : \mathbf{x} = \mathbf{d})\}$$

dove $\mathbf{x} = \mathbf{d}$ *sta per* $x_1 = d_1 \wedge \ldots \wedge x_n = d_n$ *, e* $\exists i : \mathbf{x}$ *sta per* $\exists i : x_1 \ldots \exists i : x_n$.

Una risposta globale ad una i-query è quindi processata valutando localmente in db_j tutte le coordination formula con indice j contenute in A, e componendo e mappando ricorsivamente, tramite le domain relation, questi risultati, in accordo con le congiunzioni che compongono la coordination formula A.

2.2.3 Modello logico alla base del sistema Piazza

Piazza, al contrario degli altri PDMS analizzati, non è definito per i soli database relazionali, ma, al contrario, utilizza XML come modello dei dati, in quanto esso è sicuramente la rappresentazione più utilizzata per condividere dati in modo semantico nel Web. Comunque, anche il sistema Piazza prevede,

come vedremo, una definizione del modello logico relazionale, utile per definire le proprietà del PDMS.

Nel modello di Piazza [17] ogni peer è provvisto di uno schema XML, detto *peer schema*, che definisce la terminologia e i vincoli strutturali del nodo; le relazioni che appartengono a questo schema sono le *peer relation*. Esistono anche un secondo tipo di relazioni, dette *stored relation*, le quali sono analoghe alle sorgenti informative nei sistemi di integrazione dei dati. Questo PDMS è progettato in modo che ogni nodo possa, oltre che apportare nuovi dati al sistema, anche relazionare i dati con degli schemi esistenti e creare nuovi schemi che gli altri peer della rete possono utilizzare per rispondere alle query.

Per riuscire in questo intento, nel sistema Piazza sono previsti i seguenti tipi di mapping: *storage description* e *peer mapping*.

Le **storage description**, contenute nei peer, descrivono i dati memorizzati in un nodo relazionando tra loro le stored relation con le peer relation; ovvero una storage description si presenta nella forma A : R ⊆ Q, dove A è un nodo, Q è la query formulata sullo schema del nodo A, e R è una stored relation del peer.

Ad esempio, se la sorgente UNI, vista nel paragrafo 2.1, fosse composta dalle *peer relation*

```
School_Member(name,faculty,year)

Department(dept_name,dept_code,budget)
```

allora, la *storage description* che si viene a creare relaziona la *storage relation* `students` con le due *peer relation* nel seguente modo:

```
UNI: students(name,faculty,department,year)⊆
UNI: School_Member(name,faculty,year),
UNI: Department(dept_name,…)
```

Questa *storage description* afferma che la *storage relation* `students` è un sottoinsieme del join di `School_Member` e `Department`.

I **peer mapping** consentono di relazionare schemi appartenenti a peer diversi; ovvero, il loro compito è quello di descrivere come la terminologia e la struttura di un nodo corrispondono a quelle in un secondo nodo. Esistono due tipi di peer mapping: inclusione/uguaglianza, e definitional mapping. Il primo è della forma $Q_1(\bar{A}_1)=Q_2(\bar{A}_2)$ o $Q_1(\bar{A}_1) \subseteq Q_2(\bar{A}_2)$ per l'inclusione, dove Q_1 e Q_2 sono delle query congiuntive, e $\bar{A}_1$ e $\bar{A}_2$ sono insiemi di peer; nell'inclusione è possibile solamente inferire istanze della destinazione da quelle della sorgente, mentre nell'uguaglianza è possibile anche inferire istanze della sorgente da quelle della destinazione. Il secondo tipo di peer mapping sono mapping direzionali (da una sorgente ad una destinazione) le cui relazioni sono peer relation. Se il primo caso di mapping peer è di tipo GLAV, in quanto sono presenti query congiuntive sia nel capo che nel corpo della relazione, le definitional mapping sono di tipo GAV in quanto non possono contenere query nel corpo.

Il linguaggio utilizzato per le query e per i mapping si basa sul linguaggio XQuery: l'intento è quello di definire un linguaggio di mapping che adotta elementi di XQuery, ma più semplice da usare.

La semantica presentata, si basa sull'estensione del significato delle "risposte certe" (*certain answers*), introdotte dalla letteratura dei sistemi classici di integrazione dei dati.

Per definire le risposte certe, bisogna prima definire le "istanze di dati consistenti" (*consistent data instance*) in un PDMS. Quindi, sia N un generico PDMS e D un'istanza per le stored relation, come, ad esempio, un insieme di tuple D(R) per ogni stored relation $R \in (R_1 \cup \ldots \cup R_n)$. Una **istanza di dati** per N è un assegnamento di un insieme di tuple con ogni relazione in ogni peer. Si denota con I(R) l'insieme di tuple assegnate alle relazione R da I , e con Q(I) il risultato di della valutazione della query Q sui dati estensionali in I. La definizione di **istanze di dati consistenti** è la seguente:

un'istanza di dati I viene detta consistente *con un PDMS N e un'istanza D per per le stored relation di N se:*

1. *Per ogni storage description in D_N, se è nella forma $A : R = Q_1$ (o $A : R \subseteq Q_1$), allora $D(R) = Q_1(I)$ (oppure $D(R) \subseteq Q_1(I)$).*
2. *Per ogni peer description di N:*
    - *se è nella forma $Q_1(A_1) = Q_2(A_2)$, allora $Q_1(I) = Q_2(I)$;*
    - *se è nella forma $Q_1(A_1) \subseteq Q_2(A_2)$, allora $Q_1(I) \subseteq Q_2(I)$;*

- *se è una definitional description il cui capo contiene il predicato p, allora siano $r_1,\ldots,r_m$ tutte le definitional mapping con p nel capo e sia $I(r_i)$ il risultato della valutazione del corpo di r_i sull'istanza I; allora, $I(p) = I(r_1) \cup \ldots \cup I(r_m)$.*

Da questa definizione si arriva quindi a quella di **risposte certe**:

sia Q una query sullo schema di un peer A in un PDMS N, e sia D un'istanza delle stored relation di N. Una tupla ā è una risposta certa *di Q se ā è m Q(I) per ogni istanza di dati che è consistente con N e D.*

Riassumendo, una istanza di dati I è consistente con N e con D se descrive un possibile stato del mondo ammissibile in base ai dati e ai peer mapping e in base a D; le risposte certe sono quelle che appartengono in tutte le possibili istanze consistenti di dati.

Come menzionato all'inizio del paragrafo, il modello dei dati di Piazza è basato su XML, quindi le istanze dati sono tutte istanze XML. I questo contesto la semantica dei mapping è definita come: *data un'istanza XML I_s del nodo sorgente S e il mapping al nodo di destinazione T, il mapping definisce un sottoinsieme di istanze I_t per il nodo di destinazione.* Il fatto che I_t sia un sottoinsieme delle istanze di T significa che elementi di T possono non esistere in S. I_t è una proiezione di alcune istanze complete I_t' di T su un sottoinsieme dei suoi elementi e attributi. Infatti, I_t definisce un insieme di istanze complete di T la cui proiezione è I_t'.Il processo di risposta delle query poste sul nodo di

destinazione T produce come risultato solo le risposte che sono consistenti con tutte le istanze I_t'; tali risposte sono appunto le risposte certe.

Infine, per quanto riguarda le storage description, come nel caso dei peer mapping, esse possono essere di due tipi, semanticamente opposti: *open-world assumption* o *closed-world assumption*. Nel primo caso, il nodo può non memorizzare tutti i tipi di dati che sono modellati dal suo schema, mentre nel secondo caso, nel nodo sono presenti tutti i dati rilevanti che caratterizzano il suo schema.

2.2.4 Modello logico alla base del sistema SWAP

SWAP è provvisto di un modello di metadati (**SWAP Metadata Model**) progettato per fornire semantica ai dati sia interni che esterni al nodo; tale modello, basato su RDFS [7], combina le strutture ontologiche con l'informazione necessaria per allineare, evolvere ed usare queste strutture per la formulazione e valutazione delle query. Le informazioni provenienti da diversi sorgenti o da diversi nodi possono essere integrati con questo modello per rendere possibile un eventuale futuro prelievo di questi dati. Nel modello dei metadati, ogni risorsa possiede metadati che indicano da dove essa è stata prelevata, cioè informazioni riguardo l'origine della risorsa stessa. Inoltre il modello gestisce l'inconsistenza delle risorse nel deposito locale. Questo problema nasce dal fatto che le risorse vengono prelevate e importate da più peer diversi causando inconsistenza nel deposito locale. La soluzione adottata è duplice: il modello assegna, tramite opportuni parametri, sia alle singole

risorse che ai peer della rete un valore di fiducia sulla loro attendibilità, che per le risorse prende il nome di *confidence rating* e per i peer *trust level*. Un alto valore di content rating indica che una determinata risorsa, ottenuta in risposta ad una query, può essere considerata attendibile, cioè pertinente alla semantica della query; allo stesso modo il valore di trust level aiuta un peer a identificare quei nodi che memorizzano risorse utili per rispondere alle query che pone. Grazie a questi valori il sistema è in grado di fornire risorse utili.

Il modello presenta infine ulteriori caratteristiche come il controllo di accesso per determinate risorse e la possibilità di utilizzare un servizio di cache il quale migliora le prestazioni del sistema.

Questo modello, mostrato in figura 4, è composto da due classi RDFS: la classe *Swabbi*, classe alla quale appartengono tutte le informazione disponibili in un nodo (ad esempio le risorse ad esso associate), e la classe *Peer*, alla quale appartengono le informazioni sui peer conosciuti da un nodo.

La classe Swabbi presenta le seguenti proprietà:

- *hasPeer:* serve per tenere traccia dei peer al quale l'oggetto "Swabbi" è associato.

- *uri:* serve per tenere traccia del peer che ha originato la risorsa.

- *location:* è un identificatore utilizzato per accedere alle risorse (è, ad esempio, il percorso nel file system).

- *label:* memorizza come l'informazione è chiamata nel peer dal quale è

stata prelevata.

- *confidence:* esprime, in una scala da 0 a 1, quanto un'informazione è affidabile; 1 esprime che l'informazione è sicuramente vera, 0 che l'informazione è falsa.
- *security:* fornisce un controllo di accesso alle informazioni.
- *visibility:* permette di nascondere le risorse, invece di cancellarle.
- *additionDate:* tiene traccia della data nella quale la risorsa è stata inserita o aggiornata nel deposito locale del nodo; viene usata per determinare il parametro *confidence*.
- *cache:* è la proprietà necessaria per poter tenere in cache le informazioni.

La classe *Peer* presenta, invece, le seguenti proprietà:

- *peerID:* rappresenta l'identificatore del peer.
- *peerLabel:* questo attributo memorizza la label del peer, ovvero una descrizione del peer in linguaggio naturale, leggibile e comprensibile anche per gli utenti.
- *peerTrust:* è l'attributo usato per misurare l'affidabilità dei peer; la scala di misurazione va da 1 (massima affidabilità) a 0 (peer non affidabile).

Vediamo ora come viene applicato il modello dei metadati in SWAP. Esistono due processi: creazione e integrazione del contenuto del deposito creato automaticamente da una sorgente informativa locale e remota; valutazione dell'informazione memorizzata da un nodo basandosi sulla fiducia che abbiamo sulla sua attendibilità.

Il primo processo, già brevemente introdotto, prende il nome di "integrazione della conoscenza della sorgente" (*knowledge source integration*) e consiste di 4 sottoprocessi:

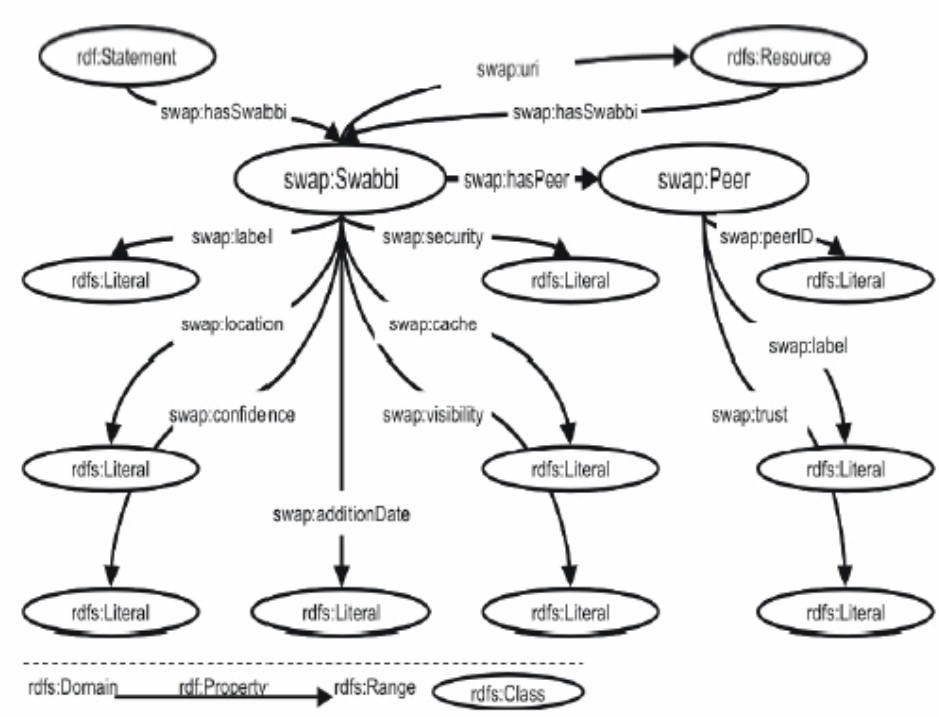

Figura 4. Modello dei Metadati di SWAP

Estrazione: questo processo consiste nell'estrarre una rappresentazione RDF(S) dalle differenti sorgenti informative, con lo scopo di includere le stesse sorgenti. Infatti, lo scopo di SWAP è quello di fornire un modello integrato di

diverse strutture (file system, database, e-mail, ontologie, ecc) che esistono localmente sul peer e remotamente sugli altri nodi della rete. Successivamente, l'informazione estratta può essere arricchita semanticamente aggiungendo prima la semantica implicita contenuta nelle strutture.

Selezione: in questo processo si decide quale informazione è utile, cioè quale informazione deve essere inclusa nel deposito locale del nodo.

Annotazione: le informazioni scelte nel processo di selezione vengono ora annotate con i metadati, cioè vengono aggiunti gli oggetti "Swabbi". Per quanto riguarda le strutture estratte dal nodo, viene creato un oggetto "Swabbi" che viene collegato all'informazione usando un costruttore RDF di reificazione; inoltre, tramite una relazione *hasSwabbi*, si esprime il collegamento tra la risorsa e l'oggetto "Swabbi". Per quanto riguarda invece l'informazione selezionata dagli altri peer della rete vale quello gia detto per le informazioni estratte dal peer stesso, con l'aggiunta che viene inclusa una relazione *hasPeer* all'oggetto "Swabbi" per mantenere l'informazione sul peer sorgente.

Fusione: nel processo di fusione, le informazioni nel deposito locale vengono unite al modello di conoscenza dell'utente, allo scopo di creare un unico modello. Infatti, nel deposito locale lo stesso oggetto può essere presente più volte, ma con nomi diversi, in quanto le risorse e le informazioni vengono estratte da diversi nodi della rete. Esistono diverse strategie per identificare la similitudine degli oggetti tra cui: *word matching*, che consiste nella verifica delle label degli oggetti; *feature matching*, che consiste nella verifica dei predicati degli

oggetti; *semantic-neighborhood matching*, il quale controlla la vicinanza contestuale di due entità; *instance matching*, nel quale si controllano le istanze e dove esse sono classificate nella struttura. Dopo tale passo avviene la fusione nel deposito locale.

Nel secondo processo, che prende il nome di "modello di valutazione del contenuto" (*content rating model*), viene fatta una stima di attendibilità delle informazioni memorizzate nel deposito locale del nodo attribuendo ad ognuna di esse un certo livello di fiducia. Tale processo è reso necessario dal fatto che le informazioni nei peer possono essere incomplete, vaghe o anche completamente false. Ciò che si vuole realizzare è un modello nel quale ogni nodo non deve considerare un'informazione sempre vera, ma deve valutare la sua attendibilità basandosi sul livello di fiducia che attribuisce al nodo sorgente, valutato in base alle precedenti esperienze avute con quel nodo. Tale modello consiste dei seguenti aspetti:

Assegnamento della stima di fiducia all'informazione in un peer: nel caso dell'informazione estratta dal nodo stesso, si assegna un alto valore di fiducia; se l'informazione è quella selezionata dagli altri peer, allora esistono due casi: se il nodo sorgente non è conosciuto da quello ricevente, quest'ultimo attribuirà un basso valore di fiducia all'informazione ricevuta; se il nodo ricevente ha già ricevuto altre informazioni da quello sorgente, il tasso di fiducia è stimato sulla base di una media pesata dei precedenti valori di fiducia, dove i pesi sono determinati dalla distanza semantica tra le precedenti informazioni ricevute e quelle ricevute attualmente. Ad esempio, se un nodo

A chiede ad un nodo B delle informazioni riguardo gli ostelli della gioventù e B conosce che a è un esperto di hotel, allora B ne deriva che i concetti di hotel e ostelli della gioventù hanno un significato simile (cioè semanticamente distano poco).

Aggiornamento delle stime di fiducia: il valore della fiducia aumenta quando altri peer, diversi dal nodo sorgente originale, confermano l'informazione ripetendola.

Determinazione degli esperti da essere interrogati: questo sottoprocesso consiste nel determinare dei nodi considerati "esperti" ai quali indirizzare le query ricevute alle quali il nodo non riesce a rispondere. Il sistema cerca i nodi "esperti" sulla base dell'argomento della query.

Meccanismo di svalutazione della stima di fiducia: questo meccanismo prevede che le informazione nel deposito locale che sono memorizzate da molto tempo e che hanno un basso valore di fiducia vengano eliminate.

Nel prossimo capitolo vedremo come questi principi sono applicati in Bibster, un sistema P2P per la condivisione di metadati bibliografici.

2.3 Un esempio completo

Viene proposto di seguito un esempio di coordination rule, ripreso da quello già parzialmente introdotto.

In tale esempio sono presenti due sorgenti **University** e **Computer_Science**, le quali descrivono uno scenario relativo al contesto universitario:

- la sorgente **University** (*UNI*) è una base di dati relazionale la quale contiene dati sul personale e sugli studenti di una specifica università. La sorgente si compone delle seguenti relazioni: **Research_Staff**, **School_Member**, **Department**, **Section**, **Room** e **Article**. Per un determinato professore (in **Research_Staff**) è memorizzato il suo codice (**rs_code**), il suo dipartimento (**dept_code**), la sezione (**section_code**) e l'indirizzo e-mail (**e_mail**). Nella relazione **School_Member** sono memorizzate le informazioni sugli studenti e sulla facoltà alla quale appartengono attraverso gli attributi **name**, **year** e **faculty** e **matr** che indica il numero di matricola. Il **Department** ha un codice (**dept_code**), un nome (**dept_name**) e un **budget**. Anche la sezione **Section** ha un codice (**section_code**), un nome (**section_name**), e una **length** ed è collegata ad una stanza con un codice (**room_code**). Facendo riferimento alla stanza **Room**, si dispone del codice (**room_code**), del numero di sedie (**seats_number**) ed è possibile introdurre alcune note (**notes**). Infine vengono memorizzati gli articoli (**Article**) che sono scritti da un **author**, sono caratterizzati da un titolo (**title**), e da un codice (**article_code**), un anno di pubblicazione (**year**) e un soggetto (**subject**).

- La sorgente **Computer_Science** (*CS*) rappresenta le informazioni sul personale collegato a un dipartimento di Computer Science. La sorgente è implementata attraverso una base di dati ad oggetti e si

compone di nove classi: **CS_Person**, **Professor**, **Student**, **Office**, **Location**, **ClassRoom**, **Essay**, **Publication** e **Course**. I dati contenuti in questa sorgente sono simili a quelli contenuti nella precedente: sono memorizzati infatti dati relativi a professori (**Professor**), di cui viene memorizzato anche l'ufficio (**Office**) e a studenti (**Student**). La classe **Location** contiene la posizione all'interno del dipartimento delle stanze. Relativamente agli studenti, vengono memorizzati i corsi che tengono (**takes**), gli anni (**year**), il reddito (**rank**) e il numero di matricola (**matr**). Inoltre sono memorizzate informazioni sulle pubblicazioni (**Publication**) o sui saggi (**Essay**) scritti da membri del dipartimento e sulle classi

(**ClassRoom**) dove sono tenuti i corsi (**Course**).

Il modello schematico per la sorgente **University** (*UNI*) è il seguente:

```
Research_Staff(rs_code,relation,name,dept_code,section_code,
e_mail)
   AK: name
   FK: dept_code REFERENCES Department
   FK: section_code REFERENCES Section

School_Member(name,faculty,year,matr)

Department(dept_name,dept_code,budget)

Section(section_name,section_code,length,room_code)
   FK: room_code REFERENCES Room

Room(room_code,seats_number,notes)

Article(year,subject,author,title,journal,article_code)
   FK: author REFERENCES Research_Staff(name)
```

Lo schema della sorgente **Computer_Science** (*CS*) è composto dalle seguenti relazioni:

```
CS_Person(first_name,last_name)

Professor:CS_Person(belongs_to,rank,title,code)

Student:CS_Person(year,takes,rank,matr)
   FK: takes REFERENCES Course

Office(description,address,code)

Location(floor,number,building name)

Course(course code,course_name,taught_by,taught_in)
   FK: taught_by REFERENCES Professor
   FK: taught_in REFERENCES ClassRoom

Essay:Publication(subject,journal)

Publication(year,authors,title,pub_code)
   FK: authors REFERENCES Professor

ClassRoom(seats,code,address,notes)
```

Di seguito verranno elencate le coordination rule tra le due sorgenti tramite relazioni create in [15] allo scopo di fornire un esempio di integrazione dei dati utilizzando MOMIS. Queste relazioni possono essere dei seguenti tipi:

- **SYN** (synonym-of): questa relazione è definita tra due termini distinti t_i e t_j che sono considerati sinonimi, cioè che possono essere

interscambiati nelle sorgenti, identificando lo stesso concetto del mondo reale.

- **BT** (broader-term): questa relazione è definita tra due termini distinti t_i e t_j , tali che t_i ha un significato più generale di t_j .
- **NT** (narrow-term): questa relazione è concettualmente l'inversa della precedente, cioè t_i ha un significato più stretto di t_j .
- **RT** (related-term): questa relazione è definita tra due termini distinti t_i e t_j che possono essere usati nello stesso contesto, tra i quali esiste un legame generico.

Relazione *SYN*

1. University.Article.year *SYN* Computer_Science.Publication.year

```
UNI.Article.year → CS.Publication.year
CS.Publication.year → UNI.Article.year
```

2. University.Article.year *SYN* Computer_Science.Student.year

```
UNI.Article.year → CS.Student.year
CS.Student.year → UNI.Article.year
```

3. University.School_Member.year *SYN* Computer_Science.Publication.year

```
UNI.School_Member.year → CS.Publication.year
CS.Publication.year → UNI.School_Member.year
```

4. University.School_Member.year *SYN* Computer_Science.Student.year

```
UNI.School_Member.year → CS.Student.year
CS.Student.year → UNI.School_Member.year
```

5. University.Article.author *SYN* Computer_Science.Publication.authors

```
UNI.Article.author → CS.Publication.authors
CS.Publication.authors → UNI.Article.author
```

6. University.Research_Staff *SYN* Computer_Science.Professor

```
UNI.Research_Staff → CS.Professor ∧
CS.Professor.code = UNI.Research_Staff.rs_code ∧
CS.Professor.belongs_to =UNI.Research_Staff.dept_code ∧
UNI.Research_Staff.name =
concat(CS.Professor.first_name, CS.Professor.last_name)

CS.Professor → UNI.Research_Staff ∧
UNI.Research_Staff.rs_code = CS.Professor.code ∧
UNI.Research_Staff.dept_code = CS.Professor.belongs_to ∧
UNI.Research_Staff.name =
concat(CS.Professor.first_name, CS.Professor.last_name)
```

7. University.Section *SYN* Computer_Science.Course

```
UNI.Section → CS.Course ∧
CS.Course.course_name = UNI.Section.section_name ∧
CS.Course.course_code = UNI.Section.section_code ∧
CS.Course.taught_in = UNI.Section.room_code

CS.Course → UNI.Section ∧
UNI.Section.section_name = CS.Course.course_name ∧
UNI.Section.section_code = CS.Course.course_code ∧
UNI.Section.room_code = CS.Course.taught_in
```

11. University.Article.title *SYN* Compuer_Science.Publication.title

```
UNI.Article.title → CS.Publication.title
CS.Publication.title → UNI.Article.title
```

12. University.School_Member *SYN* Computer_Science.Student

```
UNI.School_Member → CS.Student ∧
```

CS.Student.matr = UNI.School_Member.matr ∧

CS.Student.year = UNI.School_Member.year ∧

UNI.School_Member.name =

concat(CS.Student.first_name,CS.Student.last_name)

CS.Student → UNI.School_Member ∧

UNI.School_Member.matr = CS.Student.matr ∧

UNI.School_Member.year = CS.Student.year ∧

UNI.School_Member.name =

concat(CS.Student.first_name,CS.Student.last_name)

13. University.Room.notes *SYN* Computer_Science.ClassRoom.notes

UNI.Room.notes → CS.ClassRoom.notes

CS.ClassRoom.notes → UNI.Room.notes

14. University.Room.seats_number *SYN* Computer_Science.ClassRoom.seats

UNI.Room.seats_number → CS.ClassRoom.seats

CS.ClassRoom.seats → UNI.Room.seats_number

15. University.Article.journal *SYN* Computer_Science.Essay.journal

UNI.Article.journal → CS.Essay.journal

CS.Essay.journal → UNI.Article.journal

Relazione *BT*

1. Computer_Science.CS_Person *BT* University.Article.author

∀x ∈ UNI.Article ∃author → ∃y ∈ CS.CS_Person ∧

x.author=concat(y.first_name,y.last_name)

2. Computer_Science.Publication *BT* University.Article.journal

∀x ∈ UNI.Article ∃journal → ∃y ∈ CS.Publication ∧

x.year= y.year ∧ x.title=y.title ∧ x.author=y.authors

Relazione *NT*

1. Computer_Science.Professor.title *NT* University.Research_Staff.relation

∀x ∈ CS.Professor ∃title → ∀y ∈ UNI.Research_Staff ∃relation ∧ y.name=cocncat(x.first_name,y.last_name) ∧ y.dept_code = x.belongs_to

2. University.Research_Staff *NT* Computer_Science.CS_Person

∀x ∈ UNI.Research_Staff → ∃y ∈ CS.CS_Person ∧ x.nome=concat(y.first_name, y.last_name)

3. University.School_Member *NT* Computer_Science.CS_Person

∀x ∈ UNI.School_Member → ∃y ∈ CS.CS_Person ∧ x.name=concat(y.first_name, y.last_name)

4. Computer_Science.Office.address *NT* University.Room.room_code

∀x ∈ CS.Office ∃address → ∀y ∈ UNI.Room ∃room_code ∧ y.room_code = x.code

5. Computer_Science.ClassRoom.address *NT* University.Room.room_code

∀x ∈ CS.ClassRoom ∃address → ∀y ∈ UNI.Room ∃room_code ∧ y.room_code = x.code ∧ y.seats_nember = x.seats ∧ y.notes = x.notes

6. Computer_Science.ClassRoom *NT* University.Room

∀x ∈ CS.ClassRoom → ∃y ∈ UNI.Room ∧ y.room_code = x.code ∧ y.seats_nember = x.seats ∧ y.notes = x.notes

7. Computer_Science.CS_Person.first_name *NT* University.Research_Staff.name

∀x ∈ CS.CS_Person ∃first_name → ∀y ∈ UNI.Research_Staff

∃name ∧ y.name = concat (x.first_name, x.last_name)

8. Computer_Science.CS_Person.last_name *NT* University.Research_Staff.name

∀x ∈ CS.CS_Person ∃last_name → ∀y ∈ UNI.Research_Staff

∃name ∧ y.name = concat (x.first_name, x.last_name)

9. Computer_Science.CS_Person.first_name *NT* University.School_Member.name

∀x ∈ CS.CS_Person ∃first_name → ∀y ∈ UNI.School_Member

∃name ∧ y.name = concat (x.first_name, x.last_name)

10. Computer_Science.CS_Person.last_name *NT* University.School_Member.name

∀x ∈ CS.CS_Person ∃last_name → ∀y ∈ UNI.School_Member

∃name ∧ y.name = concat (x.first_name, x.last_name)

11. Compuer_Science.Publication.title *NT* University.Research_Staff.name

∀x ∈ CS.Publication ∃title → ∀y ∈ UNI.Research_Staff

∃name ∧ y.name = x.authors

12. Compuer_Science.Publication.title *NT* University. School_Member.name

∀x ∈ CS.Publication ∃title → ∀y ∈ UNI.School_Member

∃name ∧ y.name = x.authors

13. University.Article *NT* Computer_Science.Publication

∀x ∈ UNI.Article → ∃y ∈ CS.Publication ∧

y.year = x.year ∧ y.author = x.authors ∧

y.title = x.title

Relazione *RT*

1. University.Section *RT* Computer_Science.Professor

∀x ∈ UNI.Section → ∃y ∈ CS.Professor ∧ ∃z ∈ CS.Course ∧

z.taught_by = concat(y.first_name,y.last_name) ∧

z.course_code = c. section_code ∧

z.course_name = x.section_name ∧

z.taught_in = x.room_code

2. University.Article *RT* Computer_Science.Professor

∀x ∈ UNI.Article → ∃y ∈ CS.Professor ∧

x.author = concat(y.first_name, y.last_name)

3. University.Department *RT* Computer_Science.Professor

∀x ∈ CS.Professor → ∃y ∈ UNI.Department ∧ ∃z ∈

UNI.Research_Staff ∧ y.dept_code = z.dept_code ∧

z.dept_code = x.belongs_to

4. Computer_Science.Office *RT* University.Research_Staff

∀x ∈ UNI.Research_Staff → ∃y ∈ CS.Office ∧ ∃z ∈

CS.Professor ∧ z.code = x.rs_code ∧

z.belongs_to = x.dept_code ∧ x.name =

concat(z.first_name, z.last_name)

5. Computer_Science.Course *RT* University.Research_Staff

∀x ∈ CS.Course → ∃y ∈ UNI.Research_Staff ∧ ∃z ∈

UNI.Section ∧ z.section_code = y.section_code

z.section_code = x.course_code ∧ y.rs_code = x.taught_by

∧ z.section_name = x.course_name ∧ z.room_code =

x.taught_in

6. Computer_Science.Publication *RT* University.Research_Staff

∀x ∈ CS.Publication → ∃y ∈ UNI.Research_Staff ∧ ∃z ∈

UNI.Article ∧ y.name=z.author ∧ z.author = x.authors ∧

z.year = x.year ∧ z.title = x.title ∧

z.article_code = x.pub_code

7. University.Room *RT* Computer_Science.Course

$$\forall x \in CS.Course \rightarrow \exists y \in UNI.Room \wedge \exists z \in UNI.Section \wedge z.room\_code = y.room\_code \wedge z.section\_code = x.course\_code \wedge z.section\_name = x.course\_name \wedge z.room\_code = x.taught\_in$$

8. Computer_Science.Course *RT* University. School_Member

$$\forall x \in CS.Course \wedge \forall y \in CS.Student \wedge x.course\_code = y.takes \rightarrow \exists z \in UNI.School\_Member \wedge z.name = concat(y.first\_name, y.last\_name) \wedge y.year = z.year \wedge y.matr = z.matr$$

9. Computer_Science.ClassRoom *RT* University.Section

$$\forall x \in CS.ClassRoom \rightarrow \exists y \in UNI.Section \wedge \exists z \in UNI.Room\ y.room\_code = z.room\_code \wedge z.room\_code = x.code \wedge z.seats\_number = x.seats \wedge z.notes = x.notes \wedge y.room\_code = x.code$$

10. University.Room *RT* Computer_Science.Location.building_name

$$\forall x \in UNI.Room \rightarrow \exists y \in CS.Location.building\_name$$

2.4 Considerazioni finali

In questo capitolo sono stati analizzati i concetti principali sui quali sono basati i PDMS presentati. Abbiamo visto che PDMS come coDB, LRM e Piazza gestiscono dati appartenenti ad un insieme di database distribuiti interconnessi fra di loro tramite mapping semantici locali, cioè senza uno Schema Globale; tali mapping possono essere utilizzati per relazionare i dati

nei differenti nodi (come le formule di coordinazione in LRM o le regole di coordinazione in coDB) e per definire le regole di trasferimento di dati (come le relazioni di dominio in LRM e i peer mapping in Piazza). Abbiamo inoltre visto come tali mapping abbiano un approccio diverso da quello dei sistemi di integrazioni dei dati: mentre questi sistemi usano approcci di tipo GAV o LAV, coDB, LRM e Piazza adottano una nuova tipologia di mapping, detta GLAV, la quale unisce le caratteristiche positive delle due precedenti.

E' stato presentato inoltre come i sistemi gestiscano l'inconsistenza: il modello di coDB permette ad un database locale di essere inconsistente senza che questo comprometta la consistenza globale del sistema, mentre SWAP assegna dei valori di fiducia alle risorse e ai peer: in questo modo il sistema riesce a rintracciare le risorse e i nodi che contengono informazioni che soddisfano le query poste dagli utenti.

Tali sistemi presentano comunque differenze sostanziali. Sia coDB che LRM definiscono la rete P2P come un insieme di database relazionali, dove i dati da gestire costituiscono uno "spazio relazionale". L'approccio utilizzato dai progettisti di Piazza è diverso: anch'essi danno una definizione relazionale del PDMS ma solo per delinearne le proprietà; in realtà, il sistema Piazza, al contrario dei due precedenti, utilizza XML come modello dei dati, cioè i dati della rete sono rappresentati tramite XML e gli schemi dei nodi sono definiti utilizzando XML Schema [10].

Analogamente a Piazza, anche SWAP non basa il proprio modello sulla definizione di uno spazio relazionale, ma sulla definizione di ontologie le quali

facilitano l'accesso ai dati e permettono la creazioni di query semantiche; tali ontologie vengono realizzate tramite l'utilizzo di RDF e RDF Schema.

E' interessante notare come sia Piazza che SWAP possono, grazie ai loro modelli dei dati, fornire delle basi per la realizzazione di applicazioni per il Semantic Web come, ad esempio il meccanismo di valutazione del contenuto utilizzato in SWAP per assegnare i valori di fiducia alle risorse. Infatti molti sono gli obbiettivi comuni: sia i PDMS che il Semantic Web sono progettati per fornire una condivisione di dati, agevolando la formulazione di query più sofisticate e permettendo la creazione di processi complessi che coinvolgono i siti Web, ed entrambi intendono studiare a fondo il significato dei dati e le relazioni fra di essi. A questo proposito, le tecnologie utilizzate dai PDMS in questione sono le stesse utilizzate nella realizzazione del Semantic Web, ovvero XML, XML Schema, RDF(S) e le ontologie.

Capitolo 3

ANALISI DEL SISTEMA BIBSTER

Scopo di questo capitolo è la verifica dei concetti introdotti tramite l'utilizzo di Bibster, un sistema Peer-to-Peer utilizzato dai ricercatori per lo scambio di informazioni su pubblicazioni scientifiche. Il primo paragrafo introduce gli aspetti fondamentali del sistema e la sua architettura, mentre nel secondo viene analizzato il suo funzionamento e nel terzo paragrafo viene effettuato un test del sistema.

3.1 Il sistema Bibster

Bibster [16] è un sistema P2P sviluppato da un consorzio che comprende l'Istituto AIFB dell'Università di Karlsruhe, l'Università Vrije di Amsterdam e Empolis Polonia, all'interno del progetto SWAP, cioè si pone come dimostratore delle funzionalità di SWAP.

Lo scenario preso in considerazione è quello della condivisione, da parte di più ricercatori, di metadati bibliografici, per rendere più efficiente lo scambio di pubblicazioni scientifiche fra di essi. Tramite questo sistema gli utenti (ricercatori, ma non solo) possono ricercare le informazioni desiderate interrogando un singolo peer (anche se stessi), o un insieme di peer selezionati manualmente, oppure tutti i nodi della rete.

Da un punto di vista esterno (utente), il funzionamento di Bibster consiste di tre passaggi: il ricercatore, innanzitutto, deve scegliere quali nodi interrogare: se deve ricercare delle informazioni localmente può selezionare solo se stesso, oppure, in caso contrario, può selezionare un' insieme di peer connessi alla rete o tutti quelli disponibili; successivamente deve inserire i parametri di ricerca (autore, anno di pubblicazione, ecc.): in questo caso può limitarsi ad alcune parole chiave o effettuare query più complesse, ovvero delle ricerche semantiche (come pubblicazioni con attributi con valori specifici); infine, i dati ottenuti come risposta alla query possono essere integrati nella conoscenza locale e usati per scopi futuri: ad esempio, possono essere utilizzati per rispondere ad alcune query poste da altri utenti del sistema o per aggiornare metadati già memorizzati dal ricercatore.

Da un punto di vista interno (sistema), Bibster si basa sull'utilizzo di ontologie e di tutte le tecnologie sviluppate nell'ambito del Semantic Web. Infatti, tali tecnologie, forniscono un valido supporto all'eliminazione di alcuni dei principali problemi introdotti dall'alto grado di distribuzione dell'architettura P2P come, ad esempio, problemi di routine delle query nella rete e di dinamicità della topologia della rete, la mancanza di un singolo Schema Globale e altri ancora. L'utilizzo delle ontologie è dunque molteplice: ogni volta che nuovi dati sono disponibili per essere utilizzati, essi vengono strutturati e classificati in base a due diverse ontologie (SWRC e la Gerarchia degli Argomenti ACM) per aiutare gli utenti nella formulazione della query, anche complesse; vengono usate per migliorare il routing delle query nella

rete; sono infine utilizzate per rilevare la presenza, nelle risposte ad una interrogazione, di duplicati.

Il linguaggio utilizzato per formulare le query semantiche è SeRQL (**Se**same **R**DF **Q**uery **L**anguage) un linguaggio di query RDF/RDFS sviluppato anch'esso all'interno del progetto SWAP; infine, la comunicazione fra i peer nella rete è implementata tramite la piattaforma JXTA [5].

3.1.1 Architettura

Bibster è implementato sulla base del sistema SWAP, per cui essi presentano la stessa architettura. Come già spiegato nel Capitolo 1, ogni nodo di SWAP è costituito: da un *Adattatore della Comunicazione*, responsabile della comunicazione fra i peer della rete, che nel caso di Bibster è implementato tramite la piattaforma JXTA; da una *Sorgente di Conoscenza*, che nel sistema in esame è rappresentata da sorgenti di metadati bibliografici nel formato BibTeX memorizzati nel file system dell'utente; dall'*Integratore di Conoscenza della Sorgente*, il quale è responsabile del processo di estrazione semantica dai file BibTeX e di integrazione di tali dati nel deposito locale; dal *Deposito Locale del Nodo*, responsabile per la gestione e il mantenimento dei dati, delle viste e delle informazioni sugli schemi acquisiti dalla rete o appartenenti al nodo locale, che in Bibster è basato sul RDF-S Repository Sesame, il deposito di dati RDF di Sesame; dall'*Informatore*, il cui compito è quello di esplorare la rete per cercare peer che contengano informazioni utili per rispondere alle query poste dall'utente e per rilevare le conoscenze disponibili di un peer inviando

informazioni riguardo l'abilità (*expertise*) di un peer; dal *Replicante di Query*, il quale controlla il processo di distribuzione delle query; dall'*Interfaccia Utente*.

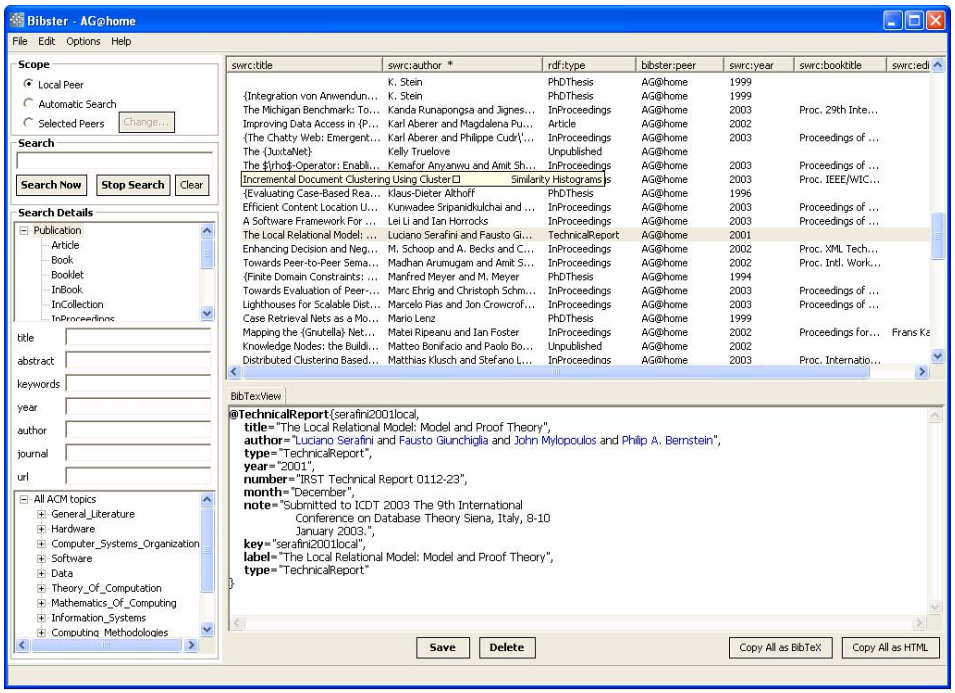

Figura 5. Bibster: Interfaccia Utente

Vediamo, in particolare, la composizione della UI (figura 5). La finestra *Scope* permette di selezionare i nodi ai quali indirizzare le query (è possibile scegliere solo il proprio nodo, un'insieme di nodi o tutti i nodi connessi al sistema); le finestre *Search* e *Search Details* consentono di impostare i parametri per la definizione delle query: è possibile definire query basate su semplici parole chiave o su semantiche più complesse (è possibile indicare se la pubblicazione che si cerca è un articolo, un libro, ecc. e anche di che argomento tratta, come, ad esempio, software, hardware, information systems e altri ancora); la finestra *Result Table* visualizza le informazioni (titolo, autore, peer sorgente,

ecc.) appartenenti ai documenti ottenuti in risposta ad una query; la finestra *BibTexView* visualizza in formato BibTeX i dati ricevuti.

Tutti i risultati di una query sono visualizzati in un lista che raggruppa i documenti replicati. Tramite le opportune funzioni nel menu, essi possono essere integrati nel deposito locale del nodo o esportati in formati come BibTeX, HTML e RDF.

3.1.2 Ontologie

Come anticipato nel paragrafo 3.1, Bibster estrae dai dati due tipi di ontologie: la prima, *SWRC* (Semantic Web Research Comunity Ontology), descrive diversi aspetti generici dei metadati in ingresso, cioè quegli aspetti che sono validi in diversi campi di ricerca (author, publication type, title, ecc.), mentre la seconda, la *Gerarchia degli Argomenti ACM*, descrive caratteristiche che appartengono alla letteratura della ricerca in Computer Science (Software, Data, Theory of Computation, ecc.).

Ogni query posta dall'utente viene formulata in termini delle due ontologie, quindi ogni query può consistere di campi generali inerenti l'ontologia SWRC, oppure concetti specifici di Computer Science.

Le ontologie vengono anche utilizzate nei processi di routing delle query, in quanto l'informazione semantica che contengono fornisce un supporto per stabilire l'abilità (*expertise*) di un peer di rispondere ad una query. Infatti, ogni

query viene instradata nella rete in basa al modello di abilità dei peer, il quale descrive a quali concetti dell'ontologia ACM un peer è in grado di rispondere.

Le ontologie risolvono inoltre un ultimo problema, quello della presenza di informazioni duplicate nelle risposte ad una query. Più precisamente, nell'insieme delle risposte possono essere presenti dati che, a causa dalla natura semi-strutturata dei metadati bibliografici, sono simili, ma non perfettamente uguali: le ontologie aiutano ad identificare queste similarità semantiche tra più metadati e rimuovere quelli duplicati, per poter visualizzare e memorizzare un unico risultato.

3.2 Funzionamento di Bibster

3.2.1 Installazione e configurazione

Il sistema Bibster è interamente open source ed è disponibile, sia per piattaforme Linux che per piattaforme Windows, all'indirizzo web http://bibster.semanticweb.org/download/download.htm.

Per poter utilizzare il sistema Bibster è necessario dotarsi di una piattaforma Java sul proprio computer; la versione raccomandata è J2SDK 1.4.2 o successive.

Per l' installazione in ambiente Windows:

- Scaricare il file `bibster-1.1.3-win32-install.jar` dal sito indicato.

- Eseguire il comando

```
java -jar bibster-1.1.3-win32-install.jar
```

- Seguire le istruzioni indicate nell' interfaccia grafica del programma di installazione.

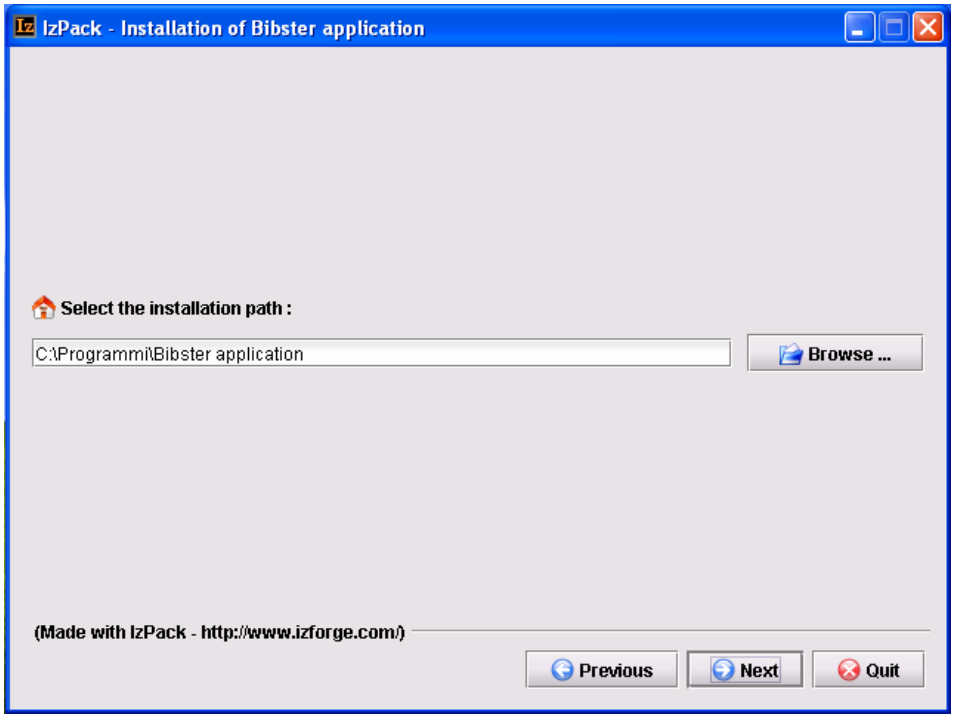

Figura 6. Installazione Bibster: scelta del direttorio di destinazione

Per prima cosa, l' applicazione richiederà di selezionare il direttorio di destinazione (figura 6).

In seguito, il programma inizia a copiare tutti i file necessari per il funzionamento del sistema nel direttorio di destinazione (figura 7). Al termine di questa operazione Bibster è installato sul computer.

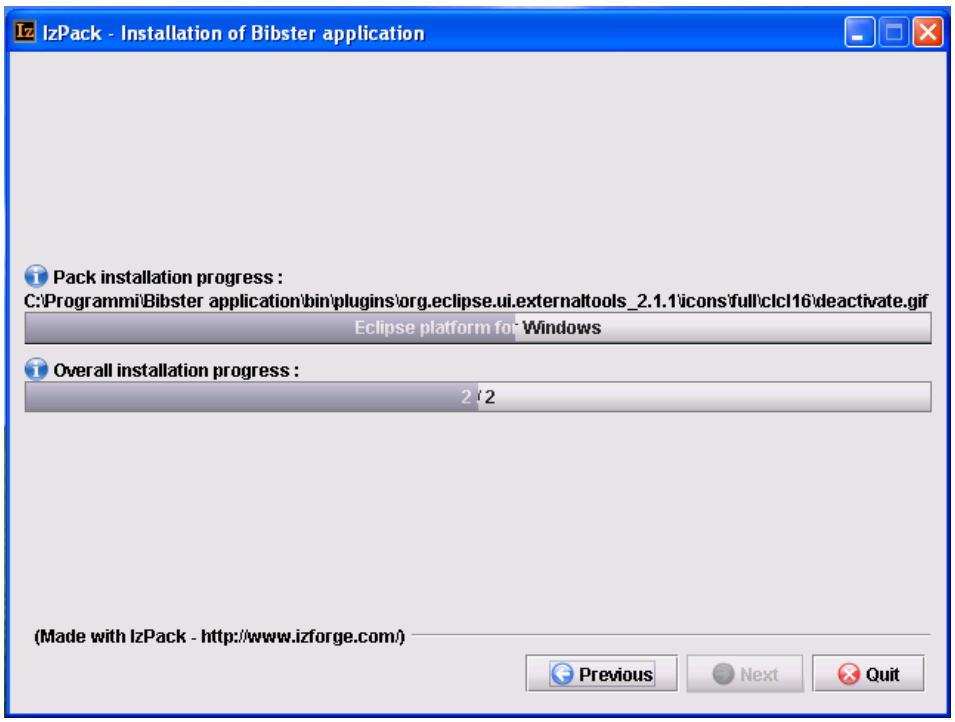

Figura 7. Installazione Bibster: avanzamento dell'installazione

Terminata la procedura di installazione è possibile lanciare il programma eseguendo il file `bibster.bat` nel direttorio di installazione, ad esempio

C:\Programmi\Bibster application\bibster.bat

Successivamente, solo la prima volta che si lancia l' applicazione, Bibster richiede di configurare la rete (JXTA Configuration): in questo caso è sufficiente indicare l' identificatore per il peer locale (figura 8).

In ambente Linux il processo di installazione consiste dei seguenti passaggi:

- Scaricare il file bibster-1.1.3-linux.tar.gz dal sito.

- Scompattare il file scaricato nel direttorio scelto.

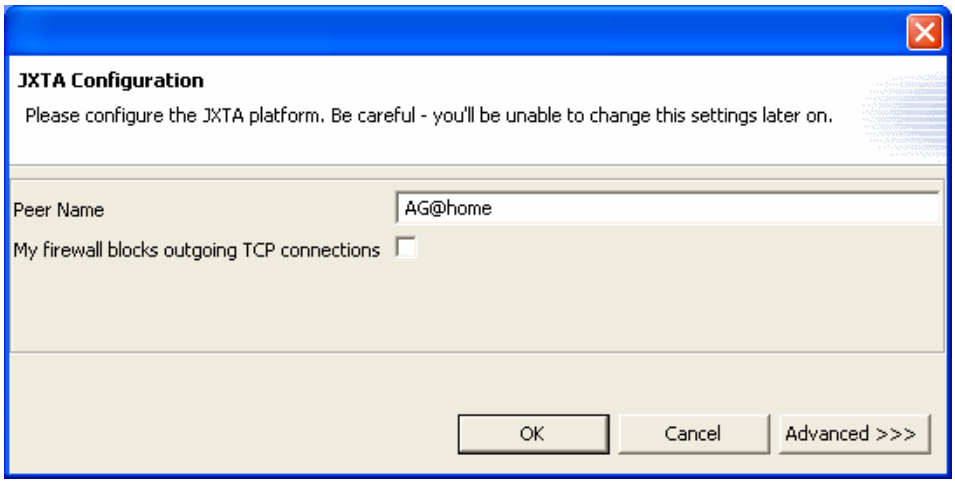

Figura 8. Bibster: configurazione della rete

A questo punto è possibile avviare Bibster eseguendo il comando `start.sh` nel direttorio di installazione.

Come nel caso di Windows, la prima volta che si usa il programma è nacessario configurare JXTA inserendo l' identificatore del peer.

3.2.2 Creazione, eliminazione e aggiornamento di metadati bibliografici

Queste tre operazioni sono rese possibili dall'apposito comando nel menu del programma. La creazione dei metadati tramite inserimento manuale richiede la selezione del tipo di documento che si vuole inserire e la compilazione dell'apposito form (figura XX) nel quale si inserisce il valore delle proprietà appartenenti all'ontologia SWRC; bibster permette anche di definire nuovi tipi di documento specificando le opportune proprietà definite dall'ontologia SWRC, o di modificare quelli già esistenti. Un secondo metodo di creazioni di metadati consiste nell'importarli dai file BibTeX: in questo caso, un

componente di Bibster, *BibToOnto*, trasforma tali file in una rappresentazione di conoscenza basata sull'ontologia SWRC estraendo la conoscenza esplicita del file; BibToOnto, inoltre, classifica automaticamente i dati in ingresso secondo l'ontologia ACM. Esiste infine un terzo metodo per la creazione di metadati che consiste semplicemente nel memorizzare le informazioni ricevuti come risposta ad una query posta. Le informazioni estratte o inserite manualmente o ricevute dagli altri nodi vengono successivamente memorizzate nel deposito locale del nodo e, quindi, rese disponibili per rispondere alle query poste dagli utenti della rete.

Altre operazioni sui metadati rese disponibili da Bibster sono la modifica e l'eliminazione ed anche la rilevazioni di possibili duplicati fra i metadati memorizzati nel deposito locale.

Infine, è anche possibile esportare il contenuto del deposito locale, ovvero, contrariamente all'importazione dei metadati, essi vengono esportati nei formati RDF, BibTeX o HTML.

3.2.3 Formulazione delle query

Abbiamo già detto che il deposito locale del nodo è un deposito RDF e che il linguaggio di query utilizzato per interfacciarsi ad esso è SeRQL (**Se**same **R**DF **Q**uery **L**anguage), sviluppato dai progettisti di SWAP per superare alcuni problemi e limiti degli altri linguaggi attualmente disponibili.

Le query possono essere formulate dall'utente direttamente dalla UI, selezionando e compilando gli opportuni campi per definire i parametri di ricerca dei documenti e si distinguono in query locali, nelle quali viene interrogato solo il deposito locale del nodo, e query globali, le quali sono inoltrate anche agli altri peer della rete. Ad esempio, immaginiamo che un ricercatore stia lavorando ad una pubblicazione e debba citare un altro libro; per risparmiare tempo, invece di riscrivere il dato a mano può interrogare il proprio deposito locale per ottenere il file BibTeX: questa rappresenta una query locale.

Un esempio di query globale è rappresentato da uno studente che voglia acquisire materiale sui sistemi P2P; la query posta sarà del tipo: ricerca tutti i documenti riguardanti la tecnologia Peer-to-Peer.

Ogni volta che viene posta una query da parte dell'utente, l' UI la invia al replicante di query il quale prova a rispondere ponendo la query, formulata nel linguaggio SeRQL, al deposito locale del nodo o la distribuisce agli altri peer, in base all'abilità dei peer di rispondere all'interrogazione; questo meccanismo è descritto nel prossimo paragrafo.

3.2.4 Instradamento delle query

Nel precedente capitolo abbiamo visto che molti modelli logici prevedono la definizione di mapping semantici, necessari per lo scambio di dati tra due nodi; questi mapping sono vere e proprie regole che coordinano i database dei nodi coinvolti preparandoli al trasferimento dei dati. La necessità di

coordinare i database nasce dal fatto che non esiste uno Schema Globale valido per tutti i peer della rete; al contrario, ogni singolo dominio può essere descritto in modi differenti, in base alle diverse ontologie utilizzate: i mapping devono quindi fornire delle relazioni semantiche tra gli elementi in schemi diversi.

Bibster, contrariamente a tali sistemi, prevede che tutti i peer utilizzino solo le due ontologie, SWRC e Gerarchia degli Agomenti ACM, che fungono anche da supporto per il trasferimento dei metadati, oltre che per la formulazione e l'instradamento delle query. Ciò comporta che, in Bibster, non è presente un mapping tra schemi diversi, ma lo scambio di informazione è gestito esclusivamente dalle due ontologie comuni.

Ad ogni modo, ciò non significa che i peer della rete non siano collegati semanticamente tra loro: come già anticipato, in Bibster è implementato un meccanismo di instradamento delle query secondo il quale i nodi vengono selezionati in base alla similarità tra il soggetto della query, espresso secondo l'ontologia ACM, e la semantica dei metadati memorizzati dal nodo, sempre espressa tramite ACM, venendo così a formare una topologia semantica della rete. In particolare, questo meccanismo di instradamento delle query si basa sulla definizione di "abilità" (*expertise*) di un peer: una *descrizione di abilità* è una descrizione semantica del deposito locale del nodo basata sull'ontologia condivisa ACM e, quindi, rappresenta l'insieme degli argomenti ACM per i quali il nodo fornisce dei metadati. Ogni nodo promuove le proprie abilità nella rete inviando agli altri partecipanti degli "avvisi" (*advertisements*) i quali

associano al peer un insieme di abilità. I peer che ricevono questi avvisi possono decidere, in modo autonomo, se accettarli (cioè memorizzarli nei propri registri) o meno, in base alla similarità semantica delle descrizioni di abilità. Grazie all'utilizzo di questi avvisi i peer possono conoscere le abilità degli altri peer: in questo senso si crea fra i nodi della rete una topologia semantica che è completamente indipendente da quella fisica. Tramite la "funzione di similarità" (*similarity function*), il sistema determina se il soggetto di una query è correlato alla descrizione semantica di un peer: in questo modo, un nodo che ponga una query riesce a individuare quei peer che possono rispondergli. Tale funzione si basa sull'idea che la similarità degli argomenti è funzione della loro distanza nella gerarchia ACM, cioè gli argomenti sono simili se la loro distanza è limitata. Il risultato espresso può assumere un range di valori compresi tra 0 e 1: una bassa similarità semantica è indicata con un valore prossimo allo 0 (un valore nullo indica che soggetto della query e abilità del nodo sono completamente diversi), mentre valori prossimi ad 1 indicano similarità (in particolare 1 esprime l'identicità delle semantiche). L'algoritmo di selezione dei peer utilizza dunque questi strumenti; il suo compito è quello di ritornare al nodo un insieme classificato di peer, dove il rank è stabilito in base ai risultati della funzione di similarità (i primi nodi in classifica sono quelli con valore più alto, cioè sono quelli la cui abilità è più simile al soggetto della query). In base a questa classifica, il nodo che ha posto la query seleziona i peer ai quali indirizzare la query: in Bibster il nodo seleziona i primi n nodi della lista, dove n può essere specificato.

3.2.5 Ricezione e memorizzazione dei risultati di una query

I risultati ricevuti vengono passati dal replicante di query all'interfaccia grafica; l'utente può così decidere se memorizzare i dati nel deposito locale: in questo caso le informazioni vengono passate all'integratore di conoscenza il quale integra i dati selezionati nel deposito.

Prima di memorizzare il dato nel deposito, il sistema controlla che tale informazione non sia già mantenuta in memoria locale, ovvero il sistema effettua una ricerca semantica dei duplicati, ovvero di quelle informazioni che si riferiscono alla stessa pubblicazione, persona o organizzazione nel mondo reale, ma che sono modellate come risorse diverse. Tale rilevazione dei duplicati avviene anche al momento della visualizzazione dei risultati ottenuti nella interfaccia utente. Essa avviene, come nel processo di selezione dei peer, tramite una funzione di similarità per le risorse RDF basata sulle ontologie SWRC e ACM, la quale presenta le stesse proprietà della precedente. Tramite questa funzione, le similarità di ogni tipo di entità (pubblicazione, persone, organizzazione) può essere valutata in base alla similarità di alcune specifiche istanze (ad esempio, per le pubblicazioni le istanze sono l'autore, il titolo, il tipo di pubblicazione, ecc.) tramite funzioni di similarità individuali, cioè specifiche per quel tipo di istanza. Queste funzioni individuali sono raggruppate in tre livelli diversi: al livello di *valore dei dati* la funzione agisce sul paragone dei dati; a livello di *struttura del grafo* la funzione studia le relazioni tra le risorse coinvolte; il livello di ontologia è un estensione del precedente e si applica in base alla Gerarchia degli Argomenti ACM. Dalle funzioni di

similarità individuali si ottiene un certo valore: due risorse sono considerate simili se il valore ottenuto supera un certo valore soglia.

I metadati considerati simili, vengono visualizzati e memorizzati come un'unica risorsa (vengono uniti): tali risorse sono composte dall'unione delle proprietà dei singoli metadati duplicati.

3.3 Test del sistema

In questo paragrafo ci proponiamo di testare il sistema Bibster ponendo delle query ai nodi della rete e valutando il numero delle risposte ottenute. Ogni query verrà posta tre volte: la prima a tutti i nodi della rete connessi al momento della prova, la seconda ad un insieme casuale di nodi e la terza ai nodi scelti in base all'algoritmo di selezione descritto nella sezione precedente. Il test del sistema è stato effettuato su un numero di 12 nodi e il nodo locale, al momento della prova, memorizzava 105 metadati bibliografici.

3.3.1 Risultati

Nel periodo di tempo del test sono state poste 30 query, ottenendo, in totale, 5217 risposte per le query poste a tutti i peer connessi, 4978 per le query indirizzate ai peer selezionati in base alla descrizione di abilità e 3159 per le query poste a nodi casuali. Tra le query poste al sistema, 9 utilizzano solo l'ontologia SWRC, 13 l'ontologia degli argomenti ACM, 4 utilizzano entrambe

le ontologie e solo 4 contengono attributi standard (come anno, autore, titolo, ecc.).

In tabella vengono presentati i risultati specifici per ogni query.

	Tipo di query			Risultati ottenuti in base ai nodi selezionati		
Query	**Standard**	**SWRC**	**ACM**	**Tutti**	**Casuali**	**Abilità**
1	X			123	67	123
2	X			110	80	110
3		X		158	65	158
4		X		110	85	110
5			X	137	92	137
6			X	129	78	129
7			X	291	132	251
8		X		148	107	148
9		X	X	149	125	149
10			X	151	122	151
11		X		236	128	236
12		X		144	128	144
13			X	178	146	178
14		X		161	123	161
15		X		95	36	95
16			X	159	121	159
17		X	X	143	85	143
18			X	425	88	295
19			X	270	186	270
20			X	121	74	121
21		X	X	95	71	95
22		X		145	103	145
23	X	X		119	24	119
24			X	294	231	251
25	X			242	135	242
26			X	140	113	140
27		X	X	94	60	94
28		X		168	61	168
29			X	262	138	236
30			X	220	155	220

3.3.2 Considerazioni

I risultati del test, organizzati nel grafico di figura 9, mostrano che il numero di risposte ottenute attraverso l'algoritmo di selezione dei peer è, nella

maggior parte dei casi, uguale a quello ottenuto ponendo le query a tutti i nodi della rete; si notano solo quattro eccezioni nelle quali il numero di risposte ottenute da tutti i nodi della rete supera quelle ottenute dai nodi "abili": ciò probabilmente significa che i risultati ottenuti nel secondo caso sono più specifici e inerenti all'argomento ACM della query posta rispetto alle risposte ottenute nel primo caso. Comunque, nel caso di sottomissine della query ai peer "abili", si osserva un miglioramento nelle prestazioni, in quanto l'intervallo di tempo che va dalla sottomissione delle query al sistema alla fine della ricerca dei risultati, e quindi alla fine della ricezione degli stessi, è lievemente minore. Il numero di risultati ottenuti utilizzando l'algoritmo di selezione casuale dei nodi è invece inferiore, spesso anche in modo consistente, ai due precedenti, a testimonianza del fatto che questo procedimento è poco efficiente.

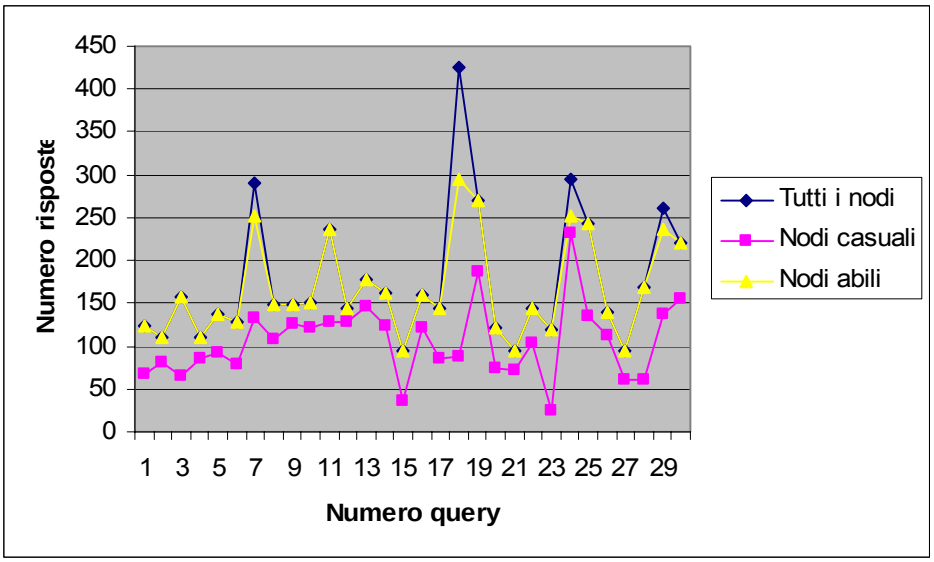

Figura 9. Andamento dei risultati del test

I risultati ottenuti non dicono che la differenza fra i tre algoritmi adottati, o almeno nei primi due analizzati (selezione di tutti i nodi e selezione dei soli peer "abili") è ridotta; piuttosto la similarità dei risultati è dovuta al basso numero di nodi che costituiscono la rete al momento del test: questo limite ha compromesso l'intero esito di questa semplice verifica. Infatti se la rete fosse stata molto popolata, probabilmente avremmo notato più significativamente che l'algoritmo di selezione casuale dei peer non dà garanzie di ottenere delle risposte e che selezionare tutti i peer disponibili significa avere tempi di risposta molto lunghi, rischiando di incrementare notevolmente il traffico nella rete..

3.4 Conclusioni

Bibster presenta interessanti soluzioni ad alcuni dei principali problemi dei sistemi P2P, in particolare per quanto riguarda la ricerca dei dati nella rete. Come visto, la ricerca si può basare su semplici parole chiave, come l'anno di pubblicazione o l'autore di un documento, oppure su query complesse basate sulle due ontologie. Infatti, grazie all'efficacia espressiva del linguaggio di interrogazione SeRQL, è possibile tradurre ed integrare le ontologie nelle query, riuscendo così a descrivere in modo completo i metadati cercati.

Altri vantaggi si hanno dal punto di vista dell'efficienza e dell'autonomia del sistema di ricerca dei dati: Bibster consente di selezionare i nodi da interrogare sia manualmente, sia in base all'algoritmo di selezione dei peer. Nel primo

caso si riesce ad aumentare l'autonomia della ricerca, anche se a scapito dell'efficienza, mentre nel secondo caso si riesce a trovare un buon compromesso fra le due proprietà: si aumenta notevolmente l'efficienza della ricerca, in quanto i peer selezionati hanno un'alta probabilità di contenere informazioni utili, mantenendo contemporaneamente buoni livelli di autonomia, in quanto i peer possono comunque scegliere il numero dei primi nodi in classifica ai quali inviare le query. Tale algoritmo permette inoltre al sistema di essere scalabile, cioè di funzionare in modo corretto ed efficiente anche in presenza di un elevato numero di peer.

L'utilizzo di un efficace linguaggio di interrogazione e l'adozione di un algoritmo di selezione dei peer che incrementa l'efficienza mantenendo un sufficiente livello di autonomia comportano alti valori di qualità del servizio (Quality of Service, QoS) che sono percepiti dall'utente come alta velocità di ricerca dei metadati (l'intervallo di tempo che va dalla sottomissione della query al sistema alla ricezione delle risposte è breve) e come un numero elevato di risposte ottenute.

Infine, Bibster permette agli utenti di mantenere la proprio anonimia, in quanto è impossibile rintracciare l'indirizzo IP dei nodi connessi, mentre non fornisce nessun meccanismo di controllo di accesso ai dati (del resto l'applicazione non gestisce informazioni riservate o che devono essere protette).

Capitolo 4

CONCLUSIONI E LAVORO FUTURO

In questa tesi è stato affrontato lo studio del problema della gestione dei dati nei sistemi Peer-to-Peer. In particolare, dopo un'introduzione sulle caratteristiche dei PDMS e sulle problematiche inerenti la loro progettazione (Capitolo 1), ci siamo soffermati sull'analisi della caratterizzazione logica di alcuni dei principali sistemi di gestione dei dati nelle reti P2P sviluppati all'interno di progetti di ricerca universitari (Capitolo 2). In tale sezione abbiamo visto come tutti i progetti presentati enfatizzino l'importanza di coordinare tra loro i database della rete, piuttosto che integrarli (scelta, quest'ultima, che caratterizza i sistemi di integrazione dei dati). Tale coordinazione è definita fornendo un insieme di mapping semantici, stabiliti fra due o più nodi (peer), usati per propagare le query e trasferire i dati tra i peer e i suoi "vicini" (acquaintance). In questo modo ogni peer può richiedere dati ad un altro nodo partecipante il quale, ricevuta la query, può interrogare il proprio database e inviare la risposta al peer vicino, oppure richiedere tali informazioni ad un terzo nodo, e così ancora. L'intento di questi mapping è quello di creare una rete che relazioni tra loro, in modo semantico, i peer del sistema, evitando l'utilizzo di uno Schema Globale dei dati.

Abbiamo anche visto come tali sistemi superino il problema dell'inconsistenza dei database: il modello logico di coDB [12], in particolare, permette ad uno o

più database di essere inconsistenti localmente, senza che questo comprometta la consistenza globale del sistema.

Inoltre, abbiamo notato che, specialmente con Piazza [17] e SWAP [10], un PDMS è inteso come supporto per costruire applicazioni per il Semantic Web: infatti, tali sistemi utilizzano molte tecnologie che sono alla base del web semantico, come RDF(S), XML, XML Schema e le ontologie, per rappresentare e trasferire le informazioni.

Infine è stato studiato e analizzato il funzionamento di Bibster (Capitolo 3), un sistema P2P per la condivisione di metadati bibliografici [16] e abbiamo notato che tale sistema presenta interessanti soluzioni riguardo il problema della ricerca dei dati nella rete, il che garantisce un elevato livello di efficienza globale del sistema.

In conclusione, alcune tematiche sulle quali dovrà focalizzarsi la ricerca in futuro riguardano: l'approfondimento dello studio sulle "relazioni di dominio" (domain relations), introdotte dal modello LRM in [7], ovvero quelle funzioni che mappono tra loro domini diversi nella rete; lo studio sulle complessità della computazione delle query, di nuovi algoritmi di ricerca dei dati più efficienti e anche di algoritmi di aggiornamento dei dati nelle reti; lo studio riguardo la sicurezza dei dati nei sistemi P2P, in particolare i problemi di autenticità e di controllo di accesso ai dati; infine, lo studio e la progettazione di applicazioni che utilizzano le tecnologie derivate dal Semantic Web come, ad esempio, dei meccanismi di selezione dei peer o dei meccanismi che consentano gli aggiornamenti delle ontologie.

GLOSSARIO

ACM: Association for Computing Machinery, è un'organizzazione scientifica internazionale il cui scopo è quello di divulgare e approfondire lo studio sull'arte, le scienze e le applicazioni riguardanti l'information technology.

Acquaintance: un nodo appartenente ad una rete P2P è acquaintance ("vicino" o "conoscente") di un altro peer se fra essi è possibile lo scambio diretto di dati.

BibTeX: è un programma e un formato dei file creato nel 1985 per il sistema di preparazione dei documenti di LaTeX. E' un formato interamente basato sui caratteri ed è il più utilizzato in Internet per la rappresentazione di dati bibliografici.

DNS: Domain Name Server, è il sistema dei nomi di dominio, il cui compito è quello di fornire un servizio di directory per Internet. Esso è composto da un database distribuito implementato in una gerarchia di server dei nomi e da un protocollo che permette agli host di comunicare con i server dei nomi in modo da fornire il servizio di traduzione.

DoS: Denial of Service, sono attacchi di negazione di servizio. Tali attacchi rendono la rete, un terminale, o altri pezzi dell'infrastruttura di rete indisponibili agli utenti autorizzati. Tipicamente, un attacco DoS opera creando talmente tanto lavoro per l'infrastruttura attaccata che non si può effettuare il lavoro desiderato.

GAV: Global-As-View o Schema Globale Centrico, è un approccio per la definizione dei mapping semantici nei sistemi di integrazione dei dati. Tale approccio definisce le relazioni dello schema globale in termini di relazioni delle sorgenti coinvolte, cioè, dato un database come sorgente dati, il mapping GAV fornisce una informazione diretta su quali dati soddisfano gli elementi dello schema globale.

GLAV: Global-Local-As-View, è un approccio per la definizione dei mapping semantici utilizzato in particolare nei sistemi P2P; esso combina gli aspetti positivi degli approcci GAV e LAV.

HTML: Hiper Text Markup Language, è un formato, non proprietario, basato su SGML (Standard Generalized Markup Language), nato per pubblicare informazioni ipertestuali sul World Wide Web.

JXTA: è una piattaforma di programmazione, progettata da Sun Microsystems, il cui compito è quello di risolvere numerosi problemi inerenti la computazione distribuita, in particolare per la architetture P2P. La piattaforma JXTA è costituita da un insieme di protocolli i quali consentono ad ogni dispositivo collegato di comunicare, scambiare informazioni e condividere risorse.

LAV: Local-As-View o Schema Sorgente Centrico, è, come GAV, un secondo approccio per la definizione dei mapping semantici nei sistemi di integrazione dei dati. In questo caso le relazioni delle sorgenti che devono essere integrate sono definite come espressioni delle relazioni nello schema globale.

Metadati: sono informazioni sui dati; vengono memorizzati insieme ai dati e servono per fornire un insieme di servizi alle applicazioni che li utilizzano.

Ontologie: rappresentano un'esplicita specificazione di una concettualizzazione. Possono essere considerate come dei vocabolari che esprimono l'insieme degli oggetti e delle relazioni che intercorrono fra di essi in un dominio, cioè come la conoscenza stessa del dominio.

P2P: Peer-to-Peer, architettura distribuita composta da soli "pari" (peer), cioè nella quale ogni nodo è potenzialmente sia cliente che servitore di un servizio; in tali sistemi ogni partecipante può mettere a disposizione della rete nuovi dati e risorse e può usufruire di quelli apportati dagli altri nodi.

PDMS: Peer Data Mangement System, sistema distribuito di gestione dei dati nei sistemi P2P.

Peer: "pari", sono i nodi di una rete P2P.

QoS: Qualità of Service, esprime la qualità di un sistema percepita dall'utente; può essere misurata in base a diverse metriche come il numero dei risultati, il tempo di risposta, ecc.

RDF: Resource Description Framework, è un linguaggio per la descrizione delle informazioni delle risorse del Web; è stato creato con il particolare intento di rappresentare i metadati delle risorse Web (come autore, titolo, data dell'ultima modifica di una pagina Web).

RDFS: RDF Schema, è il linguaggio utilizzato per la descrizione delle proprietà di RDF e delle relazioni che intercorrono fra esse. Esso definisce

classi e proprietà che possono essere usate per descrivere classi proprietà ed altre risorse

Sesame: è un database RDF open source che permette di memorizzare e interrogare dati RDF Schema; utilizza SeRQL come linguaggio di interrogazione.

Semantic Web: è un'estensione del Web attuale in cui alle informazioni sono date un senso, un significato ben definito, migliorando in questo modo la cooperazione tra i computer e le persone.

SeRQL: Sesame RDF Query Language, linguaggio di interrogazione di dati RDF/RDFS, sviluppato all'interno del progetto SWAP.

Wrapper: componenti utilizzati per far sì che le fonti di informazioni abbiano una modellazione standard, che può essere interna o proveniente dal mondo esterno con cui il sistema vuole interfacciarsi.

XML: Extendible Markup Language, è uno standard W3C per la formattazione di documenti e dati strutturati sul Web.

XQuery: linguaggio di interrogazione dei database; è stato creato per rispondere alla necessità crescente di compiere interrogazioni su dati non più basati sul modello relazionale, ma di tipo semi-strutturato, i quali sono sempre più utilizzati grazie anche alla diffusione di XML a scapito di HTML.

BIBLIOGRAFIA

[1] Gnutella. World Wide Web: http://www.gnutella.com, 2004.

[2] Groove Networks. World Wide Web: http://www.groove.net, 2004.

[3] HTML. World Wide Web: http://www.w3.org/MarkUp, 2004.

[4] Napster. World Wide Web: http://www.napster.com, 2004.

[5] Progetto JXTA. World Wide Web: http://www.jxta.org, 2004.

[6] RDF. World Wide Web: http://www.w3.org/RDF, 2004.

[7] RDF Schema. World Wide Web: http://www.w3.org/TR/rdf-schema, 2004.

[8] Seti@Home. World Wide Web: http://setiathome.ssl.berkely.edu, 2004.

[9] XML. World Wide Web: http://www.w3.org/XML, 2004.

[10] XML Schema. World Wide Web: http://www.w3.org/XML/Schema, 2004.

[11] D. Beneventano, S. Bergamaschi. The MOMIS Methodology for Integrating Heterogeneous Data Sources. IFIP World Computer Congress. Toulouse, France, 22-27 August 2004.

[12] S. Bergamaschi. "SEWASIE: a Semantic Search Engine", in "Laboratoire des Sciences de l'Information et des Systèmes", Giovedì 27 novembre, 2003, Marsiglia.

[13] Philip A. Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision. WebDB2002.

[14] Arturo Crespo and Hector Garcia-Molina. Semantic Overlay Networks. Stanford University, 2003.

[15] Neil Daswani, Hector Garcia-Molina and Beverly Yang. Open Problems in Data-Sharing Peer-to-Peer Systems, In ICDT, 2003. 1-15.

[16] Marc Ehrig, Peter Haase, Ronny Siebes, Steffen Staab, Heiner Stuckenschmidt, Rudi Studer, Christoph Tempich. The SWAP Data and Metadata Model for Semantic-Based Peer-to-Peer Systems. In Michael Schillo and Matthias Klusch and Jorg P. Muller and Huaglory Tianfield, Proceedings of MATES-2003. First German Conference on Multiagent Technologies, volume 2831 of LNAI, pp. 144-155. Springer, Erfurt, Germany, September 2003.

[17] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko (2004). Efficient query processing in P2P dynamic networks of heterogeneous databases. Submitted, 2004.

[18] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko and Luciano Serafini. A Robust Logical and Computational Characterisation of Peer-to-Peer Database Systems. Proceedings of the VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03), Berlin, Germany, September 2003.

[19] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko, Ilya Zaihrayeu (2004). The coDB Robust Peer-to-Peer Database Systems. Proc. of the 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid'04), 2004.

[20] Steven Gribble, Alon Halevy, Zachary Ives, Maya Rodrig, Dan Suciu. What Can Databases do for Peer-to-Peer? WebDB Workshop on Databases and the Web, June 2001.

[21] F. Guerra. Dai Dati all'Informazione: il sistema MOMIS. Tesi di Dottorato di Ricerca, Università di Modena e Reggio Emilia, 2003.

[22] P. Haase, J. Broekstra, M.Ehrig, M. Menken, P.Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, C. Tempich. Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. To be published in: Proceedings of the International Semantic Web Conference (ISWC2004), November 9-11, 2004, Hiroshima, Japan

[23] Alon Y. Halevy, Zachary G. Ives, Jayant Madhavan, Peter Mork, Dan Suciu, Igor Tatarinov. The Piazza Peer Data Management System. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 7, July 2004.

[24] L. Serafini, F. Giunchiglia, J. Mylopoulos, P. Bernstein. Local Relational Model: a logical formalization of database coordination. DIT University of Trento. Technical Report DIT-03-002.