UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Facoltà di Ingegneria di Modena

Corso di Laurea in Ingegneria Informatica

MOMIS: progettazione di wrapper per sorgenti di dati in formato RSS

Relatore:
Chiar.mo Prof. Sonia Bergamaschi

Correlatore:
Chiar.mo Prof. Maurizio Vincini

Tesi di Laurea di:
Daniele Grassi

Anno Accademico 2005/2006

Parole chiave:
Integrazione Dati
RSS
XQuery
MOMIS
Wrapper

Indice

Elenco delle Figure

Introduzione

Molto spesso quando si pensa a internet lo si immagina come una fonte immediata, inesauribile e sterminata di informazioni a portata di *click*. In effetti, soprattutto negli ultimi anni, si è avuto un incremento esponenziale dei dati in esso contenuti, giunti ormai a coprire ogni parte della conoscenza umana e reperibili in diversi formati da fonti tanto numerose quanto eterogenee (basti pensare che una stima effettuata nel giugno del 2006 dall'Internet Systems Consortium parla di quasi 440 milioni di computer collegati alla rete Internet).

Sempre più complessi ed evoluti motori di ricerca permettono oggi di rintracciare grandi quantità di dati, rendendo possibile un rapido accesso ad essi. Attualmente, tuttavia, l'enorme disponibilità di dati sta rendendo sempre più difficile la selezione delle informazioni: ciò si traduce in una sempre più complessa fase di elaborazione dei dati trovati, che risultano non solo molto complessi da filtrare, ma anche difficili da presentare in una visione unitaria, provenendo essi spesso da fonti estremamente eterogenee. In pratica si è di fronte ad uno scenario nel quale, accanto alla relativa sicurezza di poter reperire le informazioni cercate, cresce la difficoltà di presentare queste all'utente in maniera da renderle di facile comprensione e, quindi, utili nel prendere determinate decisioni: siamo di fronte ad una situazione di *information overloading* (sovraccarico di informazioni). Si può quindi affermare che, di fronte ad un aumento della quantità delle informazioni a disposizione degli utenti, si assiste ad un progressivo degrado della qualità delle stesse. In un contesto di questo tipo, in cui vanno di pari passo l'eterogeneità delle tipologie delle sorgenti e le strutture dei dati in esse contenuti, per potere scegliere quali siano le informazioni desiderate sarebbe necessario conoscere nello specifico il contenuto, la struttura e il linguaggio di interrogazione delle stesse. L'utente quindi dovrebbe essere in grado di svolgere questo lavoro, scomponendo la propria richiesta in una sequenza di sottointerrogazioni, effettuando poi una rielaborazione ed integrazione dei dati ottenuti, per ottenere una risposta unificata. Essendo tuttavia il numero delle possibili fonti in continua crescita, ed essendo pure l'eterogeneità delle sorgenti sempre più marcata, risulta intuitivo cogliere la necessità di sistemi in grado di automatizzare il processo di acquisizione ed integrazione, rendendo quindi molto più agevole, per l'utente, il reperimento e l'interrogazione delle informazioni contenute nelle varie fonti.

Questa tesi si inserisce in un progetto molto ampio, denominato MOMIS **[1]**, sviluppato dall'Università di Modena con lo scopo di realizzare un sistema semi-automatico in grado di integrare sorgenti eterogenee e distribuite di dati. Esso adotta un'architettura a tre livelli, con un *mediatore* che ne occupa la parte centrale e che ha la funzione di creare una vista integrata a partire dagli schemi locali; tale vista integrata permette di effettuare interrogazioni a prescindere dalla conoscenza della struttura delle singole sorgenti. Le componenti innovative di questo progetto sono indubbiamente l'impiego di un approccio semantico e l'utilizzo di logiche descrittive per la rappresentazione degli schemi delle sorgenti: questi elementi introducono comportamenti intelligenti in grado di sfruttare conoscenze intensionali, estensionali e semantiche, al fine di generare una vista integrata il più espressiva possibile.

Al di sotto del mediatore si trovano i wrapper, meccanismi attraverso i quali avvengono le iterazioni con le diverse sorgenti; i compiti dei wrapper (ciascuno specializzato ad interagire con una tipologia di sorgente) sono essenzialmente due: innanzitutto fornire una

rappresentazione della struttura della sorgente in una logica comune (ODL_{I3}); in secondo luogo permettere l'esecuzione di query locali e fornire i risultati al mediatore.

Obiettivo della presente tesi è stata l'analisi e la realizzazione di un wrapper per feed RSS: è stato realizzato il traduttore, che consente il caricamento automatico in MOMIS di uno schema standard a seconda della versione di RSS della sorgente, mentre si è preferito non addentrarsi nell'implementazione delle query locali per le problematiche che verranno espresse in seguito. Non si è trascurata tuttavia un'approfondita documentazione su quest'ultimo argomento.

La tesi è organizzata nel modo seguente:

Nel **Capitolo 1** vengono presentati accenni sulla teoria dell'Integrazione Intelligente delle Informazioni e sulla struttura e il funzionamento di MOMIS.

Nel **Capitolo 2** viene descritto lo standard RSS nelle sue diverse versioni e varianti; viene inoltre presentata una piccola indagine sulla diffusione di esse nel mondo di internet.

Nel **Capitolo 3** viene esposto il funzionamento del wrapper RSS realizzato nello sviluppo di questa tesi, con riferimento alle problematiche che si sono manifestate durante di esso.

Il **Capitolo 4** presenta i risultati della documentazione e dei test relativi al linguaggio XQuery e al suo possibile utilizzo per effettuare interrogazioni su feed RSS.

Infine, vengono esposte le **conclusioni** di questa tesi.

Capitolo 1

L'integrazione Intelligente delle Informazioni

1.1 Il programma I^3

L'esplosione dell'utilizzo di Internet, e la sua contemporanea espansione, ha portato negli ultimi anni questa tecnologia a diventare una fonte inesauribile e sterminata di informazioni, tanto da sostituire in tantissimi casi, nell'ambito del reperimento di dati, le tradizionali biblioteche. Tuttavia, come noto, avere a disposizione una quantità enorme di informazioni non sempre corrisponde a trovare effettivamente ciò che si cerca; quindi, per descrivere al meglio una buona fonte di dati, accanto all'attributo "quantità" è necessario porre l'attributo "qualità": per qualità di una fonte di informazioni si intende la pertinenza dei risultati che essa produce rispetto alle richieste. Un'altra problematica fondamentale che si incontra nel reperimento delle informazioni è l'eterogeneità delle fonti stesse: spesso ci si trova di fronte a scenari in cui i dati che si cercano risultano distribuiti su più database, costruiti con sistemi e formati diversi tra loro; in questi casi è necessario, dopo aver già compiuto lo sforzo di individuare queste fonti, porre a ciascuna di esse l'interrogazione necessaria a reperire i dati cercati, componendola ogni volta nel formato richiesto dalla fonte; poi, ottenuti i risultati, si deve passare alla fase di integrazione dei dati, spesso molto ostica e lunga da effettuare manualmente.

Nel cercare di risolvere queste problematiche si inseriscono i sistemi di supporto alle decisioni (*DSS, Decision Support System* **[2]**), i *Data Warehouse* **[3]** (raccolte dati di supporto ai processi decisionali), l'integrazione di basi di dati eterogenee ed i database distribuiti. Tuttavia questi sistemi presentano dei limiti: i *datawarehouse* e l'integrazione di basi di dati sono costretti ad utilizzare complessi algoritmi di sincronizzazione per mantenere aggiornate le viste fisiche che realizzano presso gli utenti, mentre in generale i sistemi *decision maker* sono spesso inefficienti nel realizzare lo scopo per il quale sono costruiti.

In questo scenario si inserisce l'*integrazione delle informazioni*, che rappresenta quei sistemi in grado di elaborare dati provenienti da sorgenti eterogenee senza replicare fisicamente i dati in esse contenuti, ma basandosi solamente sulle loro descrizioni (schemi) per ottenere una vista virtuale globale. Se per realizzare questa integrazione si fa uso di tecniche di intelligenza artificiale, possiamo parlare di *Integrazione Intelligente delle Informazioni* (I^3) **[4]**: essa si distingue da altre forme di aggregazione delle informazioni poiché il suo obiettivo non è una mera unione di dati provenienti da fonti eterogenee, bensì l'aumento di valore di questi attraverso l'ottenimento di nuove informazioni da essi.

In questo ambito rientra un'ambiziosa ricerca realizzata dall'ARPA **[5]**, agenzia che fa capo al Dipartimento di Difesa americano, chiamata "*Programma* I^3", mirato ad indicare un'architettura di riferimento per la realizzazione di sistemi in grado di effettuare integrazioni di sorgenti eterogenee in maniera automatica. In questo contesto si è potuto osservare che strumenti necessari per poter dedurre efficacemente informazioni dagli schemi osservati sono le tecniche di intelligenza artificiale, in grado di sviluppare soluzioni automatiche notevolmente flessibili.

Un problema realizzativo relativo a supersistemi in grado di integrare un grande numero di sorgenti eterogenee è la scarsa manutentibilità, abbinata alla scarsa adattabilità a problemi diversi da quelli per i quali esso è stato implementato: a questo proposito il programma I^3 propone l'introduzione di architetture modulari sviluppabili attraverso i dettami di uno standard, che deve dare indicazioni sui servizi da implementare tramite l'integrazione e ridurre i costi di sviluppo e mantenimento. Appare evidente che realizzare nuovi sistemi e funzioni risulta più economico e rapido quando si riesce a riutilizzare tecnologie già esistenti: per ottenere questo è necessaria l'esistenza di interfacce ed architetture standard. Il programma I^3 suggerisce un paradigma per la suddivisione dei servizi e delle risorse nei diversi moduli costruito su due dimensioni:

- orizzontale: livello utente, livello intermedio (qui si inseriscono le tecniche di intelligenza artificiale), livello di sorgenti (dati);
- verticale: domini di appartenenza delle sorgenti.

Il programma I^3 si concentra sul livello intermedio tra quelli sopra descritti: in esso vengono offerti servizi dinamici come la selezione delle sorgenti di dati, la gestione delle query, la ricezione e la combinazione dei dati e, infine, l'analisi e la sintesi degli stessi.

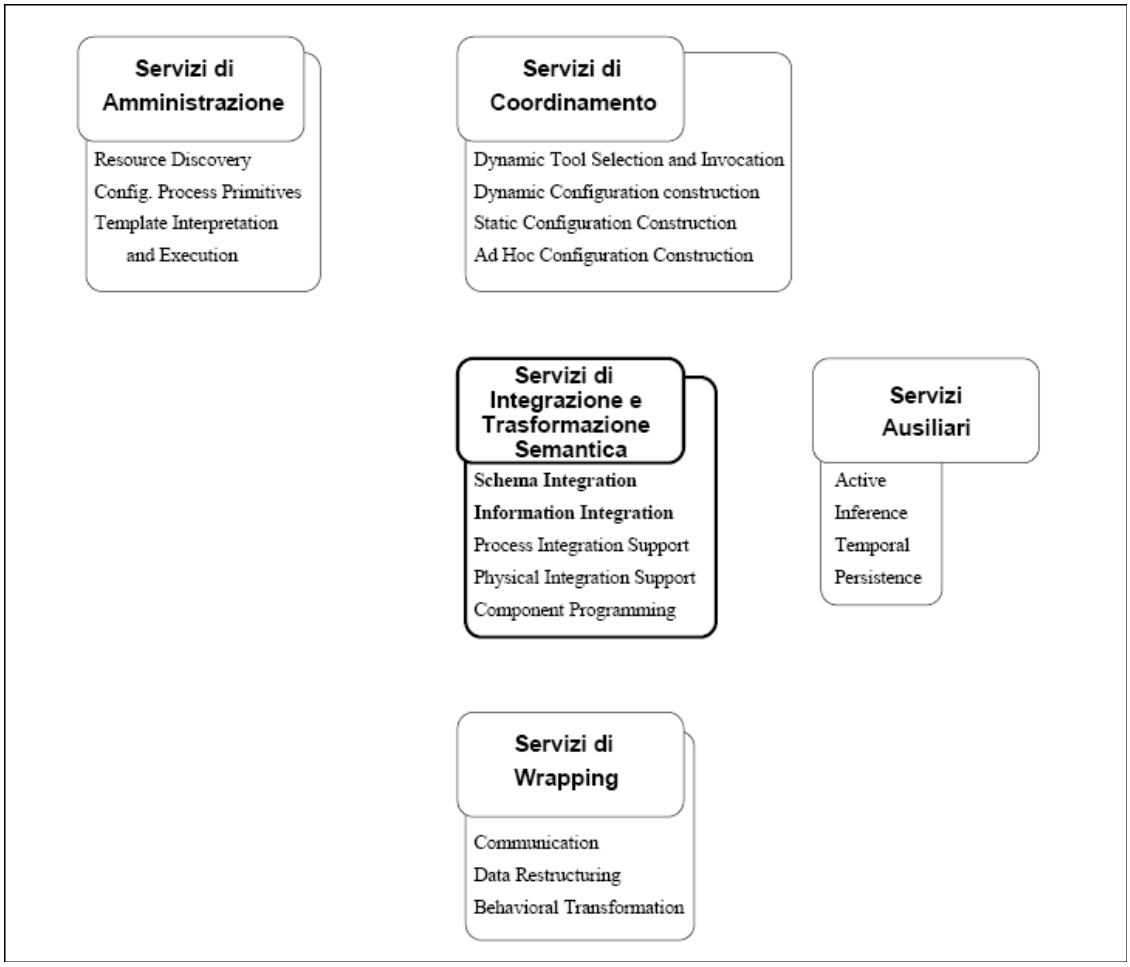

Figura 1: Servizi presenti nel mediatore

L'obiettivo di questo programma è realizzare un modello in grado di ridurre sensibilmente il tempo necessario per la realizzazione di un integratore di informazioni; in particolare, vengono formulate due ipotesi che rappresentano la base del progetto I^3. La prima è relativa alla difficoltà che si incontra nella ricerca di informazioni all'interno di una molteplicità di sorgenti; la seconda è legata al fatto che le fonti di informazioni, pur essendo semanticamente correlate tra loro, non lo sono in una forma semplice o preordinata. Per questi motivi il processo di integrazione può risultare molto complesso, ed è quindi necessario proporre un'architettura di riferimento che rappresenti alcuni dei servizi che un integratore di informazioni deve mettere a disposizione e le interconnessioni tra di esse. Tale descrizione non vuole imporre soluzioni implementative, né è da ritenersi esaustiva delle funzionalità necessarie.

1.2 Il Progetto MOMIS

MOMIS (*Mediator EnvirOnment for Multiple Information Sources*) (www.dbgroup.unimo.it/momis) è un sistema intelligente di integrazione di informazioni da sorgenti di dati strutturati e semistrutturati, la cui realizzazione e` iniziata nel 1998 all'interno del progetto *MURST 40% INTERDATA* attraverso la collaborazione tra i gruppi operativi dell'Università di Modena e Reggio Emilia e di quella di Milano. Come descritto all'interno dei lavori *SIGMOD Record* nella pubblicazione "Semantic Integration of Semistructured and Structured Data Sources", di S.Bergamaschi, S.Castano e M.Vincini **[6]**, il contributo innovativo di questo progetto rispetto ad altri simili risiede nella fase di analisi ed integrazione degli schemi sorgenti, realizzata in maniera semiautomatica; inoltre un lavoro molto approfondito è stato svolto nella fase di query processing, responsabile della traduzione della query generale nelle sottoquery da inviare alle diverse sorgenti di dati.

Come già descritto in precedenza, il continuo aumento delle possibili fonti di informazione accessibili (sia attraverso internet che mediante reti locali come quelle aziendali) ha portato all'esigenza di creare sistemi che, oltre a facilitare il reperimento delle informazioni cercate, siano in grado di proporre all'utente finale una visione integrata di queste, eliminando incongruenze e ridondanze inevitabilmente introdotte a causa della eterogeneità delle sorgenti. A questo proposito è stato introdotto il concetto di *mediatore*, cioè un modulo in grado di reperire e integrare informazioni ottenute da una molteplicità di sorgenti eterogenee. Ovviamente, le inevitabili differenze strutturali esistenti tra le diverse fonti portano alla nascita di varie tipologie di conflitti, quali per esempio differenze nell'utilizzo dei nomi, o incongruenze strutturali, generati dall'utilizzo di modelli differenti per rappresentare le stesse informazioni.

I sistemi che nel tempo sono stati presentati per risolvere questo genere di problemi si possono suddividere a seconda dell'approccio utilizzato, dividendoli quindi in *semantici* e *sintattici*. I sistemi *sintattici* sono caratterizzati dai seguenti punti:

- viene utilizzato un modello *self-describing* per trattare tutti gli elementi presenti nel sistema, ignorando gli schemi concettuali delle diverse sorgenti;
- le informazioni semantiche sono inserite manualmente tramite l'utilizzo di regole;
- l'integrazione di dati semi-strutturati è facilitata dall'utilizzo di un linguaggio *self-describing*;
- è impossibile ottimizzare semanticamente le interrogazioni (soprattutto nei database di grande dimensione) a causa dell'assenza di schemi concettuali;
- sono eseguibili solamente query predefinite.

Invece, i sistemi *semantici*, tra i quali si colloca anche MOMIS, hanno le seguenti caratteristiche:

- ogni sorgente viene codificata nello schema concettuale attraverso l'uso di metadati;
- le informazioni semantiche sono presenti nello schema concettuale, vengono tradotte in una logica descrittiva, e risultano utili sia nella fase di integrazione delle sorgenti, sia nell'ottimizzazione delle interrogazioni;

- è necessario che nel sistema sia presente un modello di dati comune, col quale descrivere in maniera uniforme le informazioni condivise;
- viene realizzata in modo automatico o manuale l'unificazione (parziale o totale) degli schemi concettuali, per giungere alla definizione di uno schema globale.

Come detto, MOMIS adotta un approccio di integrazione delle sorgenti *semantico* e *virtuale*: quest'ultimo termine indica che la vista integrata delle sorgenti (lo schema globale, o GVV) non viene di fatto materializzata, ma creata virtualmente attraverso la decomposizione della query e l'individuazione delle sorgenti da interrogare; inoltre, lo schema globale deve disporre di tutte le informazioni necessarie alla fusione dei risultati ottenuti sulle singole sorgenti, al fine di ottenere una risposta significativa.

Le motivazioni che hanno portato all'adozione di un approccio di questo tipo sono le seguenti:

- avendo a disposizione uno schema globale, all'utente è permesso formulare qualsiasi interrogazione che risulti essere consistente con lo stesso, senza conoscere gli schemi delle sorgenti dati; inoltre le informazioni semantiche in esso incluse possono permettere di effettuare una eventuale ottimizzazione di queste interrogazioni;
- l'utilizzo di una semantica *type as a set* per gli schemi consente di controllarne la consistenza, facendo riferimento alle loro descrizioni;
- la vista virtuale rende il sistema estremamente flessibile riguardo eventuali cambiamenti sia nel numero che nel tipo delle sorgenti.

In MOMIS viene fatto uso di un modello di dati comune (ODM_{I3}, estensione con le primitive I^3 del modello proposto dal gruppo di standardizzazione *ODMG-93* **[7]**); esso è di alto livello e consente di semplificare l'iterazione tra il mediatore e i wrapper (i moduli utilizzati come "interpreti" nelle comunicazioni tra mediatore e le diverse sorgenti di dati); per definire questo modello si è cercato di seguire le raccomandazioni nate in ambito I^3 per la standardizzazione dei linguaggi di mediazione.

Per la descrizione degli schemi si è definito il linguaggio ODL_{I3}, estensione del linguaggio standard *ODL (ODMG-93)*: esso riesce a cogliere le indicazioni emerse in ambito I^3 e di discostarsi il meno possibile da quelle del gruppo sopra citato.

Come linguaggio di interrogazione è stato adottato OQL_{I3}, un sottinsieme del linguaggio *OQL* proposto dallo standard ODMG.

1.2.1 L'architettura di MOMIS

I componenti principali del sistema MOMIS sono il *mediatore* e i *wrapper*.

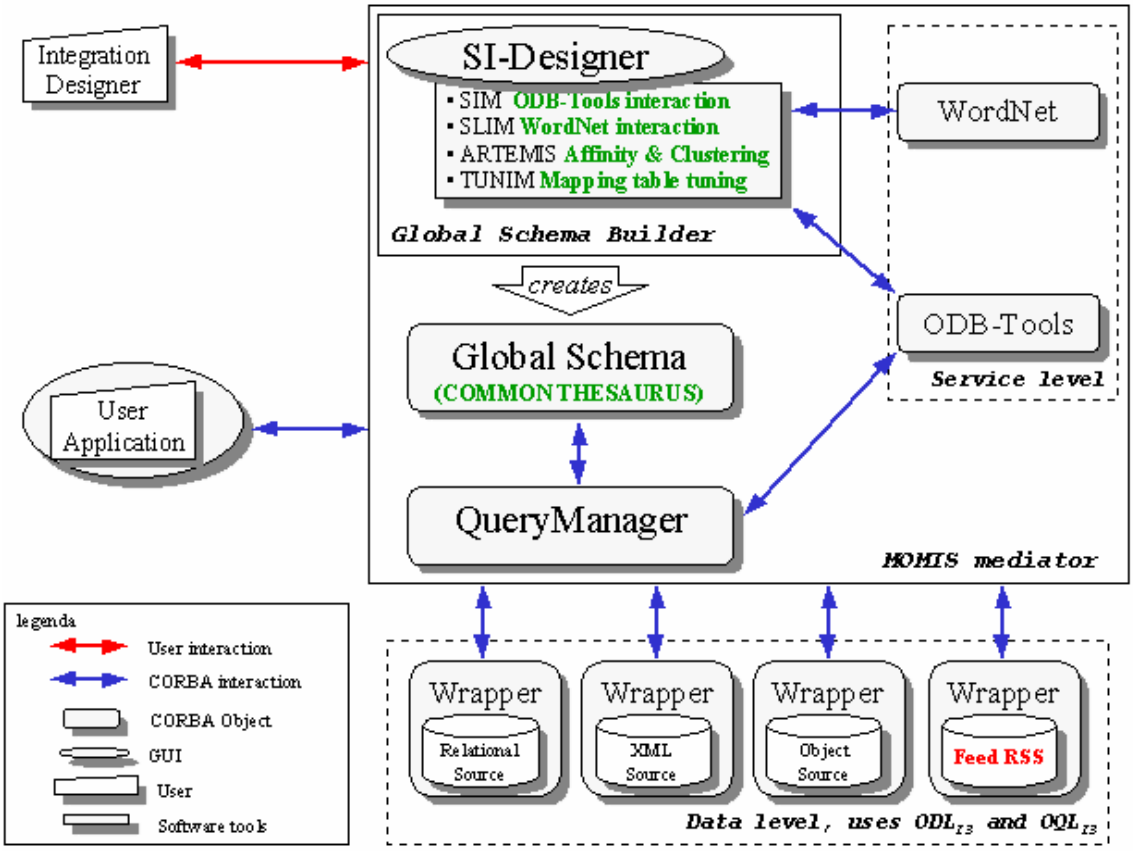

Figura 2: Architettura di MOMIS

Il *mediatore* rappresenta il nucleo centrale del sistema, con il compito fondamentale di costruire la vista globale a partire dalle descrizioni locali delle singole sorgenti eterogenee. Esso è composto da diversi sottomoduli:

- *Global Schema Builder:* il modulo di integrazione degli schemi locali, in grado di generare un unico schema globale partendo dalle descrizioni delle sorgenti; questa fase di integrazione, realizzata in maniera semi-automatica, fa uso degli *ODB-Tools* **[8]** e di tecniche di *clustering (Artemis)*;

- *Query Manager*: modulo di gestione delle interrogazioni, che provvede ad elaborare la query immessa dall'utente, generando un insieme di sottoquery da sottoporre alle fonti locali, e infine ricompone le informazioni da esse ricevute in una unica risposta; inoltre il Query Manager ha il compito di ottimizzare semanticamente le interrogazioni, funzione realizzata utilizzando gli *ODB-Tools*;

I *wrapper* invece sono componenti deputati all'estrazione di informazioni dalla sorgente e alla traduzione del suo schema nel linguaggio comune ODL_{I3}: operando in questo senso, tutte le sorgenti vengono rappresentate nello stesso linguaggio e formalismo, permettendo al mediatore di risultare svincolato dagli specifici formati in cui le sorgenti sono state concepite; i wrapper sono quindi responsabili dell'accesso alle sorgenti, ed è quindi necessario avere, per ogni tipologia di queste, un wrapper specifico in grado di riconoscere ed interpretare il linguaggio utilizzato dalla sorgente ad esso associata. In pratica la loro funzione è duplice: in fase di integrazione forniscono la descrizione della sorgente a loro associata attraverso il

linguaggio ODL_{I3}; in fase di *query processing*, traducono la query ricevuta dal mediatore, espressa nel linguaggio di interrogazione comune OQL_{I3}, in una query comprensibile e realizzabile dalla sorgente stessa, trasferendo infine i risultati al mediatore attraverso il modello di dati comune.

1.2.2 Esempio di processo di acquisizione e integrazione delle sorgenti

- Per prima cosa le sorgenti vengono acquisite e tradotte dai wrapper in schemi ODL_{I3} (fase di *wrapping*);
- Vi è la fase di *annotazione manuale* delle sorgenti, nella quale ogni elemento presente negli schemi locali (nome di tabella, nome di attributo, etc.) viene *annotato* rispetto all'ontologia lessicale *Wordnet* **[9]**: ciò consiste nell'associare un significato al nome di ciascun elemento dello schema locale, permettendo quindi di sfruttare la conoscenza semantica per ottenere un'integrazione fra gli schemi il piu` possibile automatica;
- La terza fase è rappresentata dalla generazione del *Common Thesaurus*, un insieme di relazioni terminologiche che descrivono la conoscenza riguardante le relazioni tra gli attributi e le classi delle sorgenti: essa viene espressa tramite relazioni terminologiche come sinonimie, iponimie e relazioni tra i nomi delle classi e degli attributi; *Wordnet* viene utilizzato anche in questa fase;
- La quarta fase è la generazione dello Schema Globale o Vista Virtuale Globale (*GVV: Global Virtual View*): innanzitutto viene calcolato un valore di affinità tra ogni coppia di classi appartenenti alla stessa sorgente o a due sorgenti differenti sulla base dei loro nomi, della loro struttura e delle relazioni del *Common Thesaurus*; poi si raggruppano le classi più simili, con l'obiettivo di individuare quelle che devono essere integrate; infine, per ogni gruppo di affinità si definisce una Classe Globale rappresentativa dell'insieme, costituita dall'unione dei loro attributi. L'unione delle classi globali e delle loro relazioni diviene la vista globale.

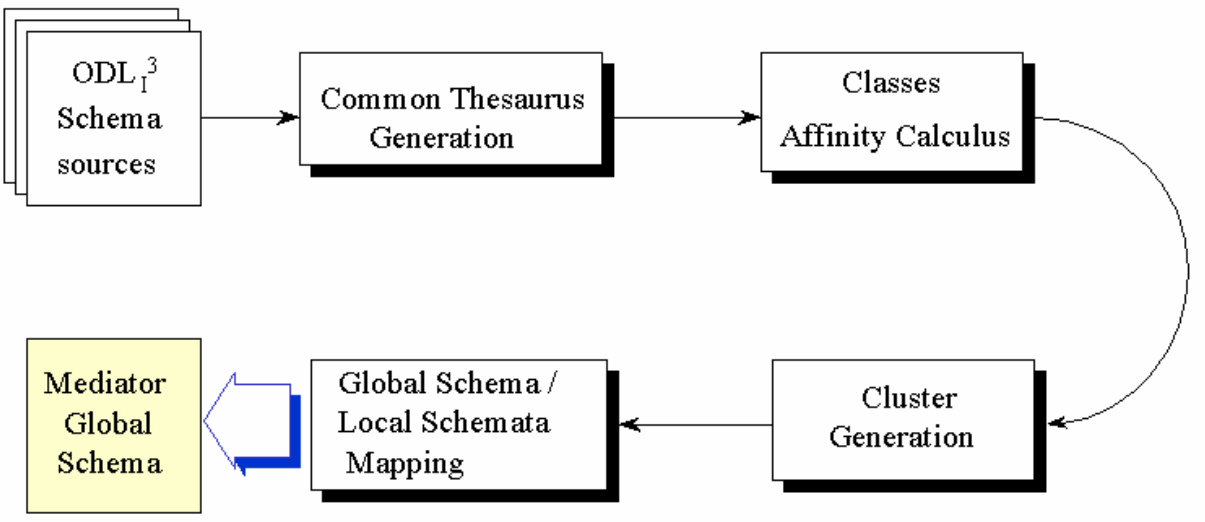

Figura 3: Processo di acquisizione sorgenti in MOMIS

Capitolo 2

Lo Standard RSS

2.1 Cos'è RSS

RSS è un acronimo per "*Really Simple Syndication*", oppure per "*Rich Site Summary*". Esso è un formato basato su XML, destinato alla distribuzione di contenuti. RSS può essere utilizzato dai webmaster per creare dei sommari contenenti titoli e descrizioni di informazioni specifiche; nonostante attualmente la maggior parte dei *feed* RSS contenga notizie o informazioni in tempo reale, le sue possibili applicazioni rimangono numerosissime.

Dal punto di vista tecnico RSS è un'applicazione dell'XML: per questo deve essere conforme alla specifica XML 1.0 e deve essere *ben formato*.

Lo scopo di RSS è rendere possibile la distribuzione su internet di aggiornamenti in tempo reale. Tramite questo standard i webmaster possono fornire *headlines* e contenuti "freschi" in una maniera semplice e concisa; gli utenti, d'altro canto, possono ricevere questi contenuti tramite appositi software ("*RSS Reader*") per catalogare ed esplorare in maniera semplice ed intuitiva i loro feed preferiti da una locazione (o applicazione) centralizzata.

2.2 Definizioni di RSS

"RSS è un formato di distribuzione di contenuti web. Il suo nome è un acronimo per Really Simple Syndication. RSS è un dialetto dell'XML" (Harvard)

"RSS è un formato per distribuire notizie e contenuti di siti di news o simili, sia per grandi portali di news come Wired, sia per community orientate alle news come Slashdot, sia per blog personali" (XML.com)

"Really Simple Syndication (RSS) è un leggero formato XML progettato per condividere headlines e altri contenuti web" (WebReference)

"Really Simple Syndication (RSS) è un formato basato su XML per la distribuzione di contenuti" (CNET)

"RSS è un formato basato su XML per contenuti distribuiti" (IBM)

"RSS è un acronimo per Rich Site Summary, un formato XML per distribuire headlines sul web, conosciuto anche come syndication. Introdotto per la prima volta da Netscape come parte del sito My Netscape, è stato esteso da Dave Winer e Useland. RSS nasce come formato RDF" (Newsmonster)

2.3 Storia di RSS

La storia di RSS comincia nel 1997, in corrispondenza con la creazione del formato RDF **[10]** (*Resource Description Framework*) da parte di Ramanathan V. Guha; questo è un linguaggio "*mark up*" utilizzato per memorizzare metadati (informazioni sulle informazioni stesse: nel caso di un articolo, per esempio, i metadati potrebbero essere l'autore, la lingua, il copyright e tutte le informazioni relative all'articolo stesso).

Nel 1999 Netscape era alla ricerca di un metodo per distribuire news, articoli e informazioni: per ottenere questo sviluppò lo standard RSS, un formato XML simile all'RDF, e lo chiamò RSS 0.90. Successivamente, Dan Libby, un impiegato della Netscape, lo migliorò, sviluppandone la versione 0.91; allo stesso tempo, tuttavia, anche Dave Winer, dipendente della Userland, sviluppò una nuova versione di RSS leggermente differente, chiamandola anch'essa 0.91 e generando così grande confusione: sfortunatamente questo era l'inizio di un trend negativo che non si interruppe.

Netscape abbandonò lo sviluppo di RSS ritenendolo troppo complicato rispetto gli obiettivi che si era prefissata. Contemporaneamente Rael Dornfest (W3C) rilasciò la versione 1.0 di RSS, basata sullo standard RDF e, sopratutto, incompatibile con le release precedenti di RSS. Questo fatto generò notevole confusione sul mercato, poiché la serie 1.0 e la 0.9, nonostante avessero gli stessi obiettivi, erano due specifiche molto differenti.

Cercando di ridurre al minimo la confusione, Userland nominò la successiva versione RSS 2.0, molto simile alla propria serie 0.9 e generalmente compatibile, rimanendo tuttavia molto diversa dalla specifica 1.0.

Lo sviluppatore di RSS 2.0, Dave Winer, resosi conto della difficoltà nel rendere globalmente accettato ed utilizzato il suo standard, lo donò ad un'associazione senza fine di lucro, la *Harvard Law School*, che ora è responsabile per il futuro sviluppo di RSS.

2.4 Le specifiche RSS

Analizziamo ora la storia dello standard RSS nel suo ramo sviluppato da Userland.

2.4.1 RSS 0.91

La versione 0.91 dello standard RSS **[11]**, sviluppata da Userland, viene rilasciata nel giugno del 2000, e nasce come rielaborazione della versione 0.91 targata Netscape: nonostante essa venne dichiarata compatibile con quest'ultima, l'abbandono delle specifiche RDF (che caratterizzavano il lavoro di Netscape) la rese uno standard a sé stante. Ogni documento RSS era definito come un file XML caratterizzato da un elemento root <*rss*> con un attributo obbligatorio <*version*>, in questo caso posto uguale a "0.91", necessario ad identificare la versione dello standard RSS a cui il documento stesso si uniforma. Subordinato all'elemento <*rss*> è posto un unico elemento <*channel*>, che contiene le informazioni che si vogliono distribuire (notizie, annunci, etc…) e metadati sul canale stesso.

Il <*channel*> è composto da elementi obbligatori ed opzionali; quelli richiesti sono:

<*title*>: il nome del canale, utilizzato dagli utenti per riferirsi al canale di informazioni. La lunghezza massima è di 100 caratteri.

<*link*>: contiene l'indirizzo del sito al quale il canale si riferisce; la dimensione massima è di 500 caratteri.

<*description*>: la descrizione del canale; lunghezza massima pari a 500 caratteri.

<*language*>: indica il linguaggio nel quale il canale è stato scritto.

<*image*>: immagine che rappresenta il canale; esso è un elemento XML contenente diversi sottoelementi, composto nel seguente modo: <*url*> è l'indirizzo di un'immagine GIF, JPEG o PNG, di dimensione massima 500 caratteri; <*title*> descrive l'immagine ed è utilizzato come attributo *ALT* quando essa è "renderizzata" in un file HTML (massima lunghezza 100 caratteri); <*link*> è l'indirizzo del sito al quale l'immagine deve puntare una volta renderizzata in HTML (lunghezza massima 500 caratteri); infine ci sono gli elementi opzionali <*width*> (massimo 144, default 88) e <*height*> (massimo 400, default 31) che indicano le dimensioni dell'immagine, e <*description*>, contenente il testo che viene incluso nell'attributo TITLE del link formato con la renderizzazione HTML dell'immagine.

Gli elementi opzionali di <*channel*> invece sono:

<*copyright*>: informazioni di copyright sul contenuto del canale; lunghezza massima pari a 100 caratteri.

<*managingEditor*>: indirizzo e-mail del *Managing Editor* del canale; la lunghezza massima è di 100 caratteri.

<*webMaster*>: indirizzo e-mail del webmaster del canale; lunghezza massima 100 caratteri.

<*rating*>: il rating PICS **[12]** per il canale; lunghezza massima 500 caratteri.

<*pubDate*>: la data di pubblicazione del contenuto del canale; essa (come tutti i valori di *DateTime* in RSS) viene indicata conformemente alle specifiche della RFC 822 **[13]** .

<*lastBuildDate*>: data di ultima modifica del contenuto del canale.

<*docs*>: indirizzo che punta alla documentazione per il formato utilizzato nel file RSS; lunghezza massima 500 caratteri.

<*textInput*>: rappresenta un campo nel quale l'utente può immettere dei dati, ed è costituito da un elemento XML formato da più sottoelementi: <*title*>, l'etichetta del pulsante *Submit* nell'area di immissione del testo, lunghezza massima 100 caratteri; <*description*>, che spiega il significato dell'area di immissione del testo, lunghezza massima 500 caratteri; <*name*>, il nome dell'oggetto di testo, lunghezza massima 20 caratteri; <link>, l'indirizzo dello script CGI che processa le richieste del campo di testo, lunghezza massima 500 caratteri.

<*skipDays*>: elemento XML che contiene fino a 7 sottoelementi <*day*>, i quali valori possono essere Monday, Tuesday, Wednesday, Thursday, Friday, Saturday o Sunday; questo campo serve a permettere agli *aggregators* di non leggere il canale nei giorni elencati in *skipDays*.

<*skipHours*>: elemento XML contenente fino a 24 sottoelementi <*hour*>, I quali valori possono variare tra 1 e 24, rappresentati ore in GMT nelle quali gli *aggregators* non dovrebbero leggere il canale.

Ogni canale contiene inoltre qualsiasi numero di elementi <*item*>, costituiti dai seguenti sottoelementi:

<*title*>: il titolo dell'elemento (notizia); lunghezza massima 100 caratteri.

<*link*>: indirizzo che punta alla versione completa dell'elemento (notizia); la lunghezza massima è di 500 caratteri.

<*description*>: rappresenta la descrizione dell'elemento; lunghezza massima 500 caratteri.

Esempio:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="0.91">
<channel>
	<title>WriteTheWeb</title>
	<link>http://writetheweb.com</link>
	<description>News for web users that write back</description>
	<language>en-us</language>
	<copyright>Copyright 2000, WriteTheWeb team.</copyright>
	<managingEditor>editor@writetheweb.com</managingEditor>
	<webMaster>webmaster@writetheweb.com</webMaster>
	<image>
		<title>WriteTheWeb</title>
		<url>http://writetheweb.com/images/mynetscape88.gif</url>
		<link>http://writetheweb.com</link>
		<width>88</width>
		<height>31</height>
		<description>News for web users that write back</description>
		</image>
	<item>
		<title>Giving the world a pluggable Gnutella</title>
		<link>http://writetheweb.com/read.php?item=24</link>
		<description>WorldOS is a framework.</description>
	</item>
	<item>
		<title>Syndication discussions hot up</title>
		<link>http://writetheweb.com/read.php?item=23</link>
		<description>After a period of dormancy, the Syndication mailing list has
become active again, with contributions from leaders in traditional media and Web
syndication.</description>
	</item>
```

</channel>
</rss>

2.4.2 RSS 0.92

Questa nuova versione **[14]**, rilasciata da Userland nel dicembre del 2000, contiene alcune modifiche ed aggiunte alla precedente. Di seguito sono proposte le modifiche introdotte dalla versione 0.92.

- L'elemento <*language*>, richiesto nella versione 0.91, diviene opzionale;
- Tutti i sottoelementi di <*item*> diventano opzionali;
- Il campo <*description*> permette ora l'inserimento di HTML dentro di esso.

Vengono introdotti i seguenti elementi come sottoelementi di <*item*>:

- <*source*>, che indica il sito dal quale proviene l'articolo in questione; contiene un attributo obbligatorio, *url*, che punta alla versione XML della sorgente stessa;
- <*enclosure*>, che permette di legare una risorsa multimediale ad un articolo RSS; contiene tre attributi obbligatori: *url* indica dove la risorsa si trova, *length* indica la dimensione di essa in byte, *type* indica il suo MIME type;
- <*category*>, che contiene un attributo opzionale *domain* identificante la locazione della tassonomia della categorizzazione; il valore dell'elemento rappresenta una locazione gerarchica nel dominio indicato;

Inoltre, viene introdotto in <*channel*> l'elemento <*cloud*>, che specifica un web service in grado di supportare l'interfaccia rssCloud, che può essere implementata in HTTP-POST, XML-RPC o SOAP 1.1. Questo elemento si compone di cinque attributi: *domain*, *port*, *path*, *registerProcedure*, *protocol*. Per dettagli riguardanti l'elemento <cloud> e la sua utilità si rimanda a **[15]**.

Esempio:

```
<?xml version="1.0"?>
<rss version="0.92">
<channel>
      <title>Dave Winer: Grateful Dead</title>
      <link>http://www.scripting.com/blog/categories/gratefulDead.html</lin
k>
      <description>A high-fidelity Grateful Dead song every day. This is
        where we&amp;apos;re experimenting with enclosures on RSS news items
        that download when you&amp;apos;re not using your computer. If it works
        (it will) it will be the end of the Click-And-Wait multimedia
        experience on the Internet.
```

```
      </description>
      <lastBuildDate>Fri, 13 Apr 2001 19:23:02 GMT</lastBuildDate>
      <docs>http://backend.userland.com/rss092</docs>
      <managingEditor>dave@userland.com (Dave Winer)</managingEditor>
      <webMaster>dave@userland.com (Dave Winer)</webMaster>
      <cloud domain="data.ourfavoritesongs.com" port="80" path="/RPC2"
        registerProcedure="ourFavoriteSongs.rssPleaseNotify" protocol="xml-
        rpc"/>
      <item>
            <description>It&apos;s been a few days since I added a song to
              the Grateful Dead channel. Now that there are all these new
              Radio users, many of whom are tuned into this channel
              (it&apos;s #16 on the hotlist of upstreaming Radio users,
              there&apos;s no way of knowing how many non-upstreaming users
              are subscribing, have to do something about this..). Anyway,
              tonight&apos;s song is a live version of Weather Report Suite
              from Dick&apos;s Picks Volume 7. It&apos;s wistful music. Of
              course a beautiful song, oft-quoted here on Scripting News.
              &lt;i&gt;A little change, the wind and
              rain.&lt;/i&gt;</description>
            <enclosure
              url="http://www.scripting.com/mp3s/weatherReportDicksPicsVol7
              .mp3" length="6182912" type="audio/mpeg"/>
      </item>
      <item>
            <description>Kevin Drennan started a &lt;a
              href=&quot;http://deadend.editthispage.com/&quot;&gt;Grateful
              Dead Weblog&lt;/a&gt;. Hey it&apos;s cool, he even has a
              &lt;a
              href=&quot;http://deadend.editthispage.com/directory/61&quot;
              &gt;directory&lt;/a&gt;. &lt;i&gt;A Frontier 7
              feature.&lt;/i&gt;</description>
            <source
              url="http://scriptingnews.userland.com/xml/scriptingNews2.xml
              ">Scripting News</source>
      </item>
      <item>
            <description>&lt;a
              href=&quot;http://arts.ucsc.edu/GDead/AGDL/other1.html&quot;&
              gt;The Other One&lt;/a&gt;, live instrumental, One From The
              Vault. Very rhythmic very spacy, you can listen to it many
              times, and enjoy something new every time.</description>
            <enclosure url="http://www.scripting.com/mp3s/theOtherOne.mp3"
              length="6666097" type="audio/mpeg"/>
      </item>
</channel>
</rss>
```

2.4.3 RSS 0.93

La versione 0.93 di RSS **[16]**, rilasciata nell'aprile del 2001, oltre a fornire use-case più precisi per *<pubDate>* ed a introdurre la possibilità di specificare più *<enclosure>* per ciascun *<item>*, inserisce *<expirationDate>* come sottoelemento di *<item>*: esso indica la "data di scadenza" di un articolo.

Esempio:

```
<rss version="0.93">
<channel>
      <title>Example Channel</title>
      <link>http://example.com/</link>
      <description>an example feed</description>
      <language>en</language>
      <rating>(PICS-1.1 "http://www.classify.org/safesurf/" l r (SS~~000
        1))</rating>
      <textInput>
            <title>Search this site:</title>
            <description>Find:</description>
            <name>q</name>
            <link>http://example.com/search</link>
      </textInput>
      <skipHours>
            <hour>24</hour>
      </skipHours>
      <item>
            <title>1 < 2</title>
            <link>http://example.com/1_less_than_2.html</link>
            <description>1 &lt; 2, 3 &lt; 4. In HTML, &lt;b&gt; starts a
              bold phrase and you start a link with &lt;a href=
              </description>
            <expirationDate>Sat, 29 Nov 2003 10:17:13 GMT</expirationDate>
      </item>
 </channel>
 </rss>
```

2.4.4 RSS 2.0

Dopo aver prodotto una bozza della versione 0.94 di RSS (mai pubblicata), Userland rilascia nell'agosto del 2002 la versione 2.0 di RSS, seguita dalla 2.0.1 e dalla 2.0.2 (gennaio 2003), che risulta essere ad oggi la versione definitiva. Poiché il numero di versione (attributo *version* di <*rss*>) che identifica queste 3 nuove release è lo stesso (2.0), è di fatto impossibile distinguerle; per questo motivo prenderemo in considerazione solo la 2.0.2 **[17]**.

Quest'ultima versione introduce numerose modifiche rispetto le precedenti.

<*channel*>

Solo i sottoelementi <*title*>, <*link*> e <*description*> rimangono obbligatori, mentre gli altri diventano tutti opzionali; vengono aggiunti i sottoelementi <*generator*>, stringa indicante il programma utilizzato per generare il feed, e <*ttl*>, che indica il numero di minuti per i quali il canale può essere immesso in una cache senza essere aggiornato.

<item>

Vengono aggiunti i seguenti sottoelementi opzionali: <*comments*>, che indica l'URL della pagina in cui è possibile commentare l'articolo; <*author*>, contenente l'e-mail dell'autore dell'articolo; <*pubDate*>, che indica la data di pubblicazione dell'articolo; <*guid*>, stringa in grado di identificare univocamente l'elemento.

Esempio:

```
<?xml version="1.0"?>
<rss version="2.0">
<channel>
      <title>Dictionary.com Word of the Day</title>
      <link>http://dictionary.reference.com/wordoftheday/</link>
      <description>A new word is presented every day with its definition
      and example sentences from actual published works.</description>
      <language>en-us</language>
      <docs>http://blogs.law.harvard.edu/tech/rss</docs>
      <pubDate>Tue, 26 Sep 2006 00:00:03 -0700</pubDate>
      <copyright>Copyright 2006 Lexico Publishing Group, LLC</copyright>
      <ttl>720</ttl>
      <image>
      <title>Dictionary.com</title>
      <url>http://cache.lexico.com/dictionary/graphics/logo88x31.gif</url>
      <link>http://dictionary.reference.com/wordoftheday/</link>
      <width>88</width>
      <height>31</height>
      <description>Free online dictionary, thesaurus and reference guide,
      crossword puzzles and other word games, online translator and Word of
      the Day.</description>
      </image>
      <item>
            <title>ostensible: Dictionary.com Word of the Day</title>
            <link>http://dictionary.reference.com/wordoftheday/archive/2006
            /09/26.html</link>
            <description>ostensible: appearing to be true, but not
            necessarily so.</description>
      </item>
      <textInput>
            <title>Lookup</title>
            <description>Dictionary.com Search</description>
            <name>q</name>
            <link>http://dictionary.reference.com/search</link>
      </textInput>
</channel>
</rss>
```

2.5 La versione 1.0

Le versioni dello standard RSS sopra descritte non sono le uniche esistenti; come già anticipato nel riassunto della storia di RSS, nel 1999 il W3C giunse a conclusione di un

processo di elaborazione di uno standard per la syndication dei contenuti su internet (lo stesso processo dal quale aveva preso le mosse lo sviluppo operato da Netscape). Il prodotto di tale elaborazione fu RSS 1.0 **[18]** (in questo caso acronimo di *RDF Site Summary*), applicazione dell'XML conforme alle specifiche RDF del W3C, estensibile tramite namespace XML e/o attraverso una modularizzazione basata su RDF.

Viene da sé che l'utilizzo dello standard RDF implica una minore leggibilità dei file scritti secondo lo standard 1.0 rispetto a quelli elaborati secondo le altre versioni di RSS; ugualmente maggiore è la complessità, ma allo stesso tempo l'estensibilità risulta notevolmente più ampia, grazie alla possibilità di utilizzare moduli RDF e namespace XML.

Viene qui sotto presentato un esempio di feed RSS 1.0.

```
<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns="http://purl.org/rss/1.0/"
>

  <channel rdf:about="http://www.xml.com/xml/news.rss">
    <title>XML.com</title>
    <link>http://xml.com/pub</link>
    <description>
      XML.com features a rich mix of information and services
      for the XML community.
    </description>

    <image rdf:resource="http://xml.com/universal/images/xml_tiny.gif" />

    <items>
      <rdf:Seq>
        <rdf:li resource="http://xml.com/pub/2000/08/09/xslt/xslt.html" />
        <rdf:li resource="http://xml.com/pub/2000/08/09/rdfdb/index.html"
/>
      </rdf:Seq>
    </items>

    <textinput rdf:resource="http://search.xml.com" />

  </channel>

  <image rdf:about="http://xml.com/universal/images/xml_tiny.gif">
    <title>XML.com</title>
    <link>http://www.xml.com</link>
    <url>http://xml.com/universal/images/xml_tiny.gif</url>
  </image>

  <item rdf:about="http://xml.com/pub/2000/08/09/xslt/xslt.html">
    <title>Processing Inclusions with XSLT</title>
    <link>http://xml.com/pub/2000/08/09/xslt/xslt.html</link>
    <description>
     Processing document inclusions with general XML tools can be
     problematic. This article proposes a way of preserving inclusion
     information through SAX-based processing.
    </description>
  </item>
```

```
  <item rdf:about="http://xml.com/pub/2000/08/09/rdfdb/index.html">
    <title>Putting RDF to Work</title>
    <link>http://xml.com/pub/2000/08/09/rdfdb/index.html</link>
    <description>
     Tool and API support for the Resource Description Framework
     is slowly coming of age. Edd Dumbill takes a look at RDFDB,
     one of the most exciting new RDF toolkits.
    </description>
  </item>

  <textinput rdf:about="http://search.xml.com">
    <title>Search XML.com</title>
    <description>Search XML.com's XML collection</description>
    <name>s</name>
    <link>http://search.xml.com</link>
  </textinput>

</rdf:RDF>
```

Come apparirà chiaro dalla piccola indagine condotta durante lo sviluppo di questa tesi, il corrente utilizzo di RSS 1.0 nel mondo di internet appare alquanto limitato. Per questo motivo, unito alla sostanziale disomogeneità di questa versione rispetto alle altre, si è pensato di non dotare il wrapper RSS del supporto per essa. Infatti, per ottenere questo sarebbe stato necessario uno sforzo progettuale decisamente sproporzionato rispetto ai benefici che si sarebbero ottenuti, mantenuti estremamente limitati proprio dalla scarsa adozione di questa versione dello standard RSS.

2.6 Diffusione dei diversi standard RSS

Come descritto in precedenza, la complicata storia di RSS ha portato alla diffusione di varie versioni di esso, che vengono comunemente utilizzate ancora oggi nel mondo della syndication dei contenuti su internet. E' quindi interessante avere un'idea della reale diffusione delle diverse versioni tramite l'analisi dei siti più visitati che offrono contenuti RSS.

Dalla verifica dei feed provenienti da oltre 60 siti, reperiti tramite la ricerca della parola "rss" su Google, emerge chiara la predominanza della versione 2.0 su tutte le altre. Qui sotto vengono presentati i risultati di questa indagine.

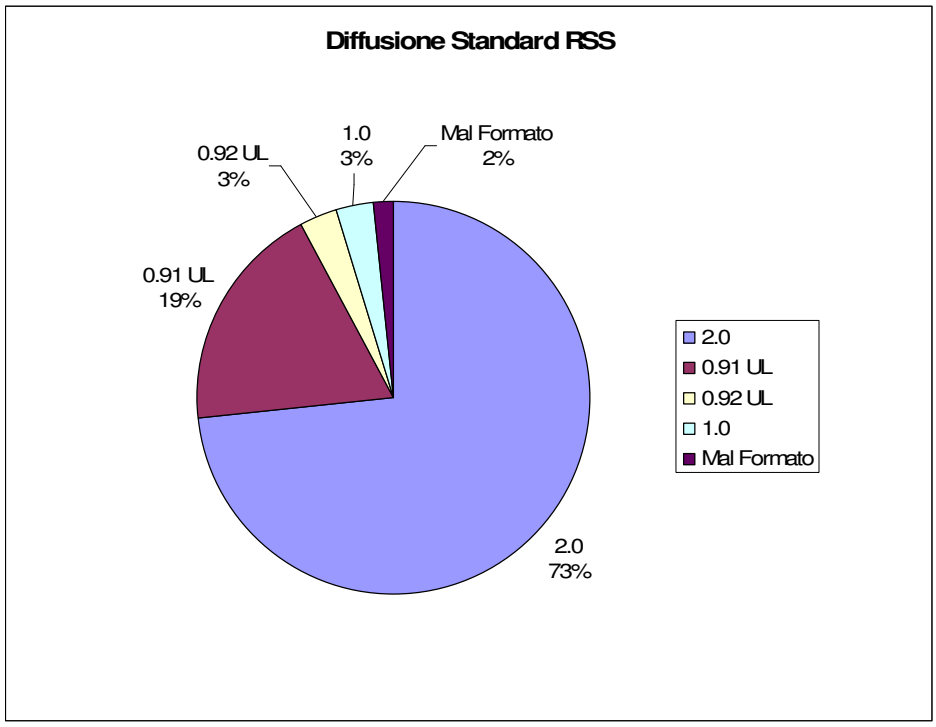

Figura 4: Diffusione degli standard RSS

Versione	Diffusione
2.0	73,44%
0.91 UL	18,75%
0.92 UL	3,13%
1.0	3,13%
Mal Formato	1,56%

Come viene chiaramente evidenziato dal grafico, la versione largamente predominante risulta la 2.0, che da sola rappresenta il 73,44 % della diffusione dello standard RSS. In seconda posizione si attesta la versione 0.91 rilasciata da Userland, rappresentante il 18,75 % del campione. Appare evidente come non ci sia traccia della versione 0.91 di Netscape (la progenitrice dello standard RSS); inoltre, si è verificato come la versione 1.0 rilasciata dalla W3C, che non è stata considerata nell'ambito di questa tesi, sia presente solo nel 3,13 % dei siti analizzati.

In conclusione, la diffusione delle versioni considerate in questa tesi (0.91 UL, 0.92, 0.93, 2.0) rappresenta una fetta molto grande del campione (più del 95 %), per cui tralasciare le restanti versioni non porta ad avere grossi problemi di compatibilità.

Capitolo 3

Il wrapper RSS

Il wrapper RSS, costruito per sovrintendere ai feed RSS, come ogni altro wrapper progettato all'interno del progetto MOMIS deve essenzialmente soddisfare due funzionalità. In primo luogo deve essere in grado di fornire al mediatore una descrizione in linguaggio ODL_{I3} dello schema del feed RSS dato come sorgente; in secondo luogo deve permettere di eseguire query sulla sorgente stessa. Come vedremo di seguito, non è stato possibile implementare quest'ultima funzionalità poiché, nonostante *XQuery* (che presenteremo in seguito) stia emergendo come possibile standard per l'interrogazione di dati semistrutturati (in effetti RSS *è* XML), la messa in opera della componente del wrapper in grado di sfruttare questo query language richiederebbe un processo di elaborazione molto approfondito, motivo per il quale si è preferito concentrarsi sull'implementazione della prima funzionalità richiesta e su uno studio preliminare sui requisiti e sugli strumenti necessari per sviluppare la seconda.

3.1 Gli schemi locali predefiniti

Come abbiamo appena detto, quindi, il primo requisito di funzionalità richiesto al wrapper RSS è quello di fornire a MOMIS la descrizione ODL_{I3} dello schema della sorgente RSS. Come abbiamo visto dall'analisi dello standard RSS, appare evidente che ogni versione di questo possegga specifiche leggermente diverse, che individuano, in pratica, *schemi descrittivi* leggermente diversi. Ciò significa che, a differenza di quanto accade per i wrapper già implementati in MOMIS, in questo caso gli schemi possibili delle sorgenti sono in un numero limitato, pari a quello delle diverse versioni di RSS che si intende supportare. Data una certa versione di RSS, quindi, è possibile individuare uno schema predefinito che ne descriva la struttura, poiché la versione stessa implica quello schema.

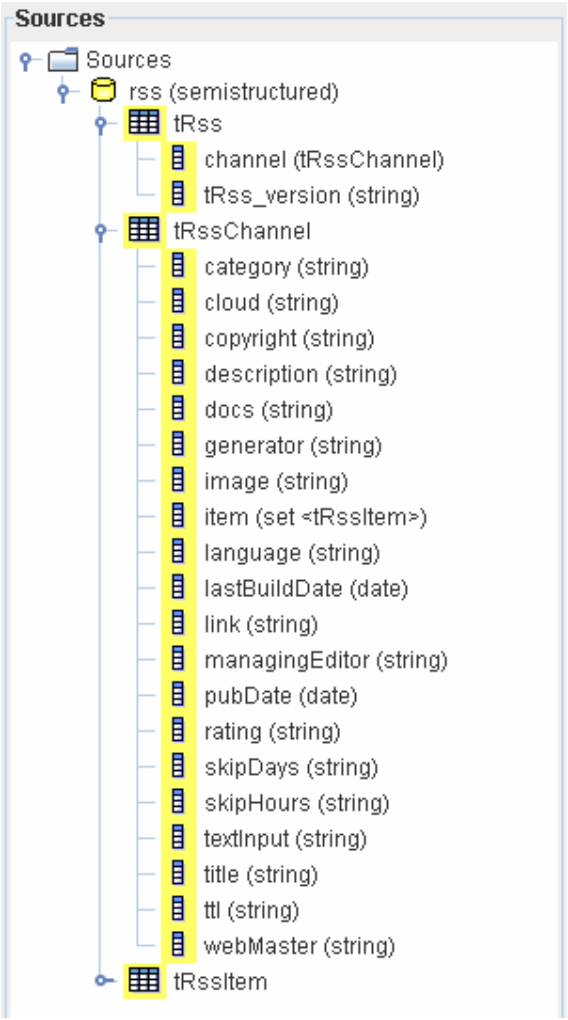

Figura 5: Acquisizione sorgente RSS in MOMIS

Dall'analisi sullo standard RSS, come detto in precedenza, si è deciso di supportare, nella progettazione di questo wrapper, le versioni 0.91 Userland, 0.92, 0.93, 2.0. Quindi, in primo luogo, si è cercato di elaborare quattro diversi schemi che descrivessero la struttura di queste versioni.

Il linguaggio utilizzato per descrivere gli schemi è XML-Schema, adottato poiché di fatto standard per la descrizione di schemi di sorgenti di dati XML (quali, per esempio, i feed RSS).

Il wrapper RSS, quindi, una volta inizializzato, legge dal feed RSS dato come sorgente il numero della versione, a seconda della quale carica lo schema appropriato.

Ovviamente, per ottenere questa informazione, è necessario effettuare il parsing del feed stesso, alla ricerca dell'attributo *version* dell'elemento <rss>; essendo il feed RSS un documento XML, quindi, si è implementato nel progetto un parser XML modificato, dotato di una funzione

public String getStringValue(String RSSFileName, String ElementName)

con il compito di restituire il valore del primo attributo dell'elemento passato come parametro (in questo caso l'attributo *version* di <*rss*>).

Appare evidente che uno schema XML richiede di essere tradotto in ODL_{I3} per poter essere immesso in MOMIS. A questo scopo si è pensato di utilizzare un wrapper già esistente, in grado di tradurre XML-Schema in ODL_{I3}: il wrapper XML-Schema. Per questo motivo, il wrapper RSS viene implementato in ambiente java tramite l'estensione della classe relativa al wrapper XML-Schema.

3.2 Lo schema globale predefinito

Come abbiamo visto il wrapper RSS sfrutta la standardizzazione di questo sistema di syndication per caricare automaticamente schemi locali predefiniti a seconda della versione del feed posto come sorgente. Nel corso dello svolgimento di questa tesi si è pensato di sfruttare la "standardizzazione" di RSS per far effettuare a MOMIS, in automatico, alcune operazioni all'atto dell'immissione in esso di una nuova sorgente RSS: in particolare, è stato automatizzato il processo di creazione delle classi globali e il relativo procedimento di mapping tra queste e le classi locali.

Analizziamo ora il processo di creazione delle classi globali. Innanzitutto, è necessario notare che lo schema di ogni versione di RSS contiene ed estende gli elementi presenti nella versione precedente. Questo consente di affermare che lo schema relativo alla versione 2.0 di RSS contiene tutti gli elementi presenti negli schemi delle versioni 0.91, 0.92 e 0.93. Nel caso di studio, quindi, per ottenere uno schema globale contenente le classi necessarie per mappare lo schema di ogni possibile versione (supportata) di RSS, è stato sufficiente creare una GVV ottenuta tramite l'importazione in MOMIS di una sorgente XML-Schema connessa allo schema relativo alla versione 2.0 di RSS e infine salvarla dopo aver creato le corrispondenti classi globali ed aver eliminato la sorgente. Il file XML così ottenuto quindi rappresenta una sessione MOMIS nella quale la GVV è dotata di 3 classi globali ("rss", "channel", "item") contenenti tutti gli attributi necessari per mappare ogni possibile versione di RSS.

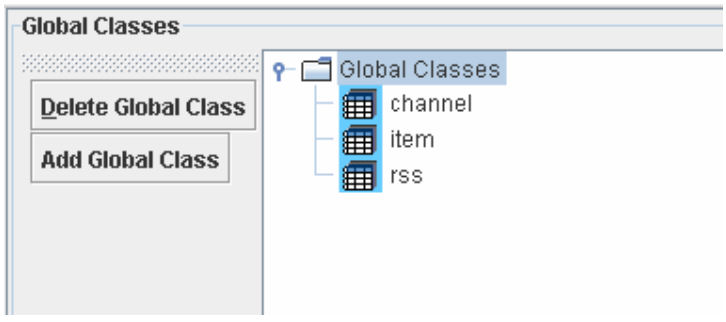

Figura 6: Classi globali per integrazione feed RSS

Nello schema globale sono state inoltre incluse informazioni sull'annotazione delle classi e degli attributi, che quindi vengono caricate automaticamente.

Per ottenere questi risultati, e in particolare sia il caricamento di una GVV predefinita, sia la realizzazione di mapping (che vedremo di seguito), che richiedono di agire al di fuori del wrapper stesso, è stato necessario apportare modifiche al modulo *SAM* di MOMIS, poiché esso è dedito proprio al caricamento dei diversi wrapper.

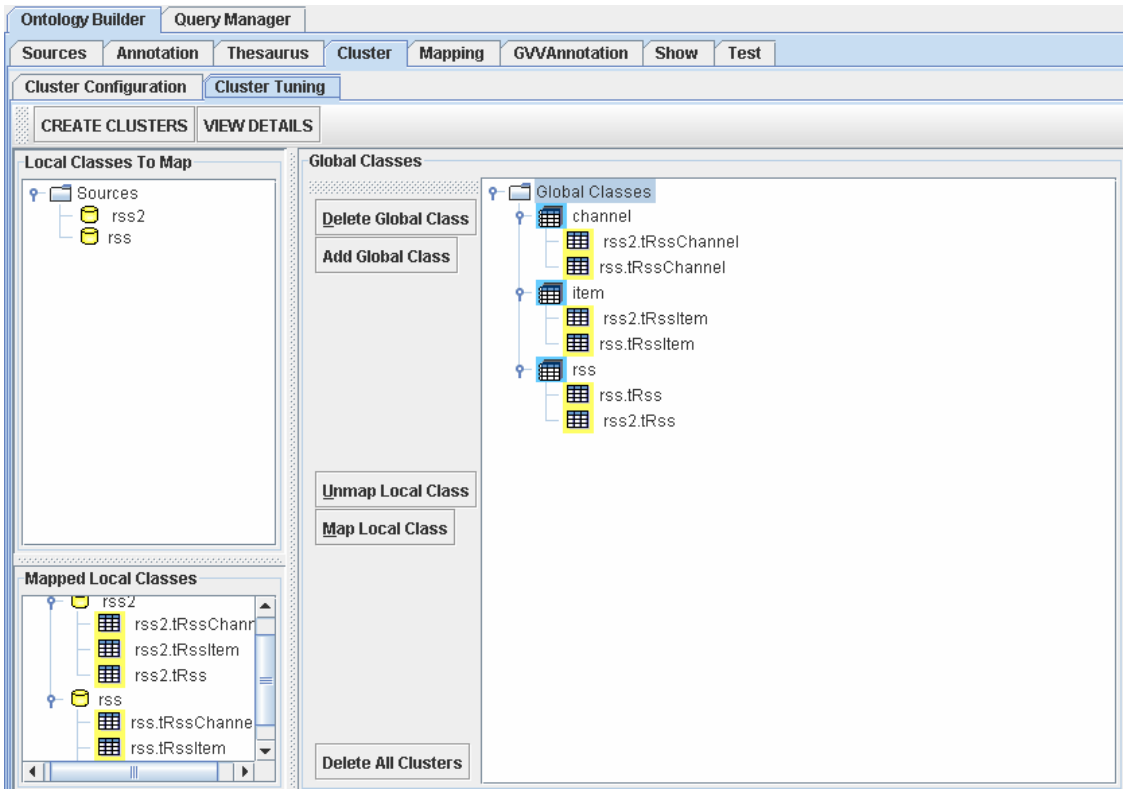

Figura 7: Fase di Clustering in MOMIS

Riassumendo, quando l'utente sceglie di immettere una sorgente RSS, il modulo *SAM*, se la GVV non esiste già, procede al caricamento dello stato di MOMIS contenente la GVV pre-impostata; a questo punto invoca il wrapper RSS, che controlla la versione del feed in esame e a seconda di questa carica il file XML-Schema corrispondente; la descrizione ODL_{I3} viene restituita al modulo *SAM*, che provvede a inserirla nella GVV e a realizzare i mapping necessari.

3.3 La realizzazione dei mapping

Per completare il processo di automatizzazione dei passaggi necessari ad ottenere una GVV completa, rimane quindi la mappatura delle classi e degli attributi delle sorgenti locali rispetto alle corrispondenti globali.

Questo risultato viene ottenuto dal modulo SAM modificato in seguito al caricamento di ciascuna delle sorgenti. Esso procede alla scansione di tutti gli attributi relativi alla sorgente

che viene aggiunta, provvedendo a riconoscere ciascuno di essi, mappandolo poi rispetto all'attributo globale corrispondente. Ciò che si ottiene quindi è una GVV completata con la propria mapping table.

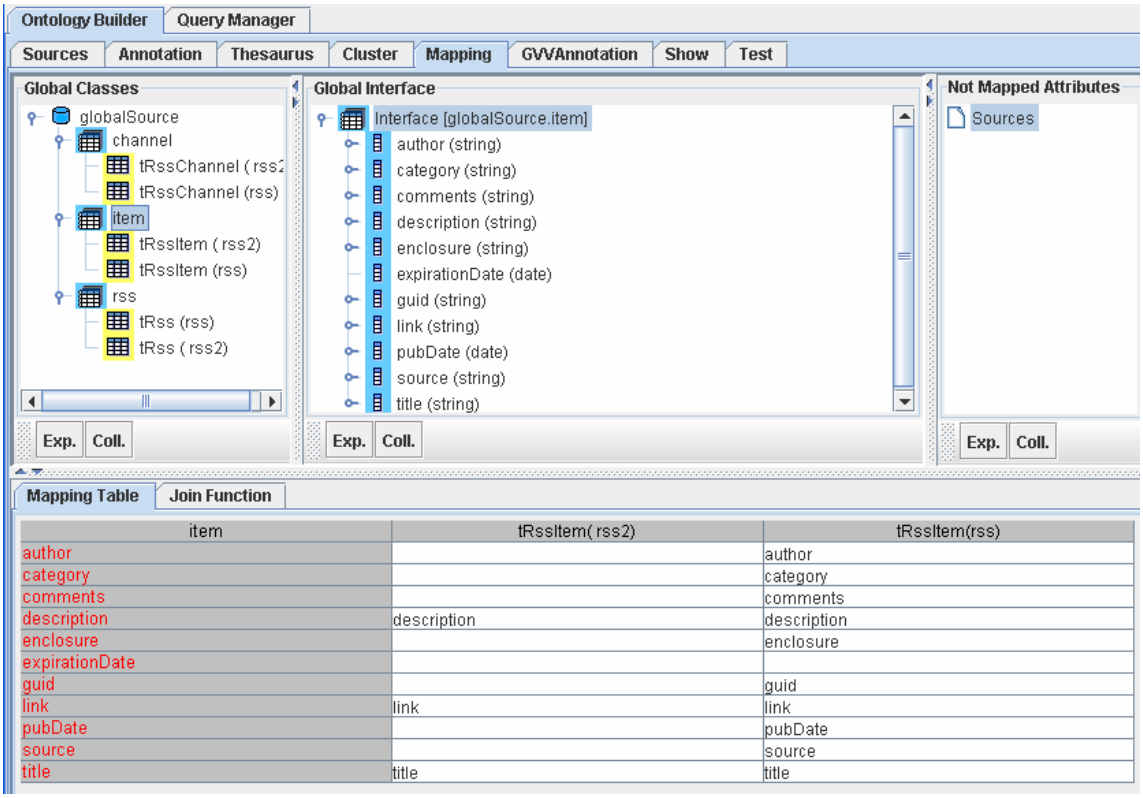

Figura 8: Fase di Mapping in MOMIS

3.4 Problemi Riscontrati

3.4.1 Limiti del Wrapper XSD (XML-Schema)

Abbiamo visto come il wrapper RSS si basi sul modulo per l'acquisizione delle sorgenti XML-Schema, ovvero il wrapper XSD. Durante lo sviluppo di questa tesi sono stati riscontrati alcuni problemi ed alcuni limiti di quest'ultimo, che qui vengono presentati.

Problematiche nell'inserimento del nome del file sorgente

Il wrapper XSD richiede il passaggio come parametro del nome del file contenente lo schema da includere nella GVV come sorgente locale. Ciò è visibile anche nell'interfaccia grafica del wrapper stesso, che richiede di specificare il "*File XSD to wrap*"; come si evidenzia anche

nell'utilizzo del wrapper stesso, è necessario che il nome del file sia preceduto da un doppio slash ("//") per essere accettato dal wrapper: in caso contrario viene sollevata un'eccezione *unknown protocol*. Inoltre, questo non accetta URL (in questo caso viene sollevata un'eccezione che avverte del problema: *a registered resource factory is needed*).

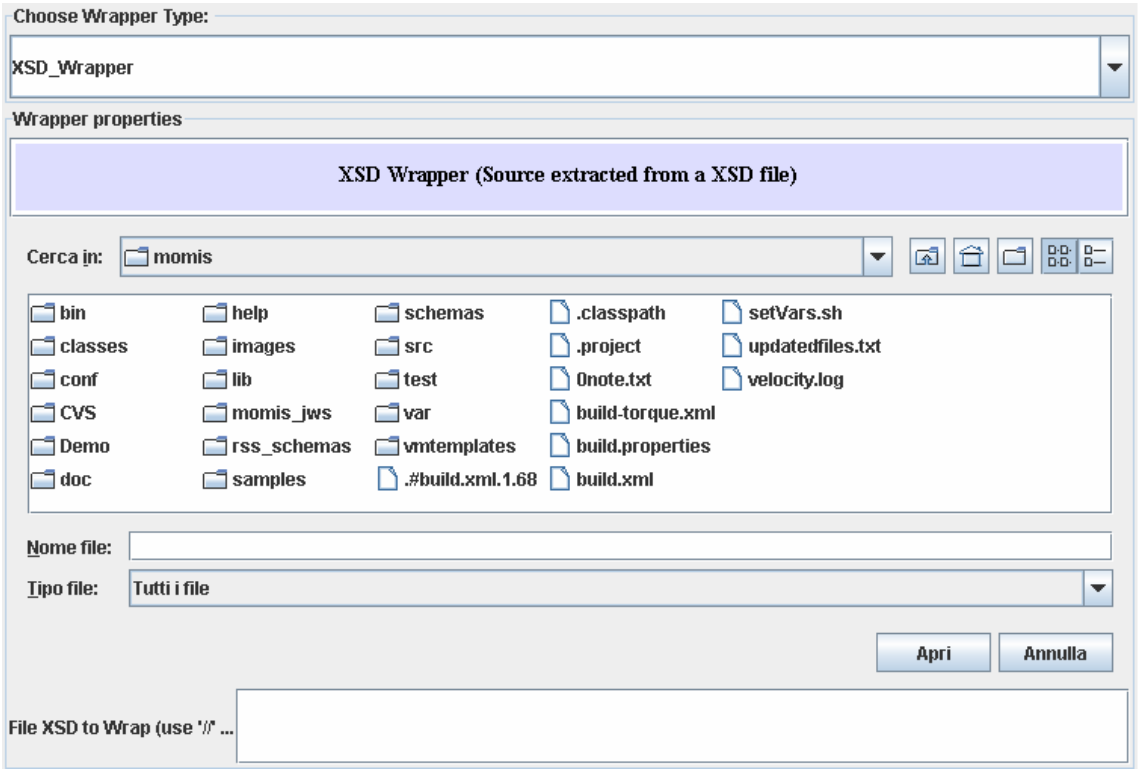

Figura 9: Interfaccia del wrapper XSD

Problematiche nell'acquisizione di schemi gerarchici

Il lavoro nello sviluppo di questa tesi ha permesso di evidenziare una problematica che si manifesta nell'utilizzo dell'interfaccia grafica di MOMIS, durante il passaggio dal "*Cluster Tuning*" al "*Mapping*", e accade quando le classi caricate tra le "*Global Classes*" sono state prese da una sorgente semistrutturata. Nel caso in cui anche solo una delle classi mappate possegga uno o più elementi complessi non mappati a loro volta, passare alla fase di "*Mapping*" produce numerosi eccezioni che creano forti malfunzionamenti sull'interfaccia grafica di MOMIS. E' interessante notare come un salvataggio della GVV tramite il pulsante "*Save All to XML*", e un conseguente ricaricamento dello stesso, provoca la scomparsa del problema sopra descritto.

3.4.2 Il Formato Data RFC-822

Come detto nei paragrafi relativi alle specifiche di RSS, questo standard prevede l'utilizzo di un formato data particolare, conforme al RFC-822, del 1982. Questa specifica, elaborata dall'ente governativo americano ARPA, aveva lo scopo di stabilire lo standard per i messaggi

di testo trasmessi all'interno della nascente internet. In essa troviamo regole relative alla sintassi, all'utilizzo di parole riservate, al formato degli indirizzi e, appunto, al formato delle date. In particolare, il capitolo che ci interessa è il 5:

```
5.  DATE AND TIME SPECIFICATION

     5.1.  SYNTAX

     date-time   =  [ day "," ] date time        ; dd mm yy
                                                 ;  hh:mm:ss zzz

     day         =  "Mon"  / "Tue" /  "Wed"  / "Thu"
                 /  "Fri"  / "Sat" /  "Sun"

     date        =  1*2DIGIT month 2DIGIT        ; day month year
                                                 ;  e.g. 20 Jun 82

     month       =  "Jan"  /  "Feb" /  "Mar"  /  "Apr"
                 /  "May"  /  "Jun" /  "Jul"  /  "Aug"
                 /  "Sep"  /  "Oct" /  "Nov"  /  "Dec"

     time        =  hour zone                    ; ANSI and Military

     hour        =  2DIGIT ":" 2DIGIT [":" 2DIGIT]
                                                 ; 00:00:00 - 23:59:59

     zone        =  "UT"  / "GMT"                ; Universal Time
                                                 ; North American : UT
                 /  "EST" / "EDT"                ;  Eastern:  - 5/ - 4
                 /  "CST" / "CDT"                ;  Central:  - 6/ - 5
                 /  "MST" / "MDT"                ;  Mountain: - 7/ - 6
                 /  "PST" / "PDT"                ;  Pacific:  - 8/ - 7
                 /  1ALPHA                       ; Military: Z = UT;
                                                 ;  A:-1; (J not used)
                                                 ;  M:-12; N:+1; Y:+12
                 / ( ("+" / "-") 4DIGIT )        ; Local differential
                                                 ;  hours+min. (HHMM)
```

Quindi, una data, standardizzata secondo le specifiche sopra riportate, potrebbe apparire così:

Fri, 28 Mar 03 05:18:59 -0800

La limitazione che appare immediatamente evidente nell'utilizzo di questo standard è il formato di scrittura dell'anno, rappresentato solo da due cifre. Per questo motivo la RFC 822 venne sostituita dalla RFC 1123 **[19]** del 1989, e definitivamente resa obsoleta nel 2001 per essere sostituita dalla RFC 2822 **[20]**. Quest'ultima, al paragrafo relativo alle date, recita così:

```
3.3. Date and Time Specification

   Date and time occur in several header fields.  This section specifies
   the syntax for a full date and time specification.  Though folding
   white space is permitted throughout the date-time specification, it
   is RECOMMENDED that a single space be used in each place that FWS
   appears (whether it is required or optional); some older
   implementations may not interpret other occurrences of folding white
   space correctly.
```

```
date-time       =       [ day-of-week "," ] date FWS time [CFWS]

day-of-week     =       ([FWS] day-name) / obs-day-of-week

day-name        =       "Mon" / "Tue" / "Wed" / "Thu" /
                        "Fri" / "Sat" / "Sun"

date            =       day month year

year            =       4*DIGIT / obs-year

month           =       (FWS month-name FWS) / obs-month

month-name      =       "Jan" / "Feb" / "Mar" / "Apr" /
                        "May" / "Jun" / "Jul" / "Aug" /
                        "Sep" / "Oct" / "Nov" / "Dec"

day             =       ([FWS] 1*2DIGIT) / obs-day

time            =       time-of-day FWS zone

time-of-day     =       hour ":" minute [ ":" second ]

hour            =       2DIGIT / obs-hour

minute          =       2DIGIT / obs-minute

second          =       2DIGIT / obs-second

zone            =       (( "+" / "-" ) 4DIGIT) / obs-zone
```

Il formato data specificato nella RFC 822 (utilizzato fra l'altro negli header dei messaggi e-mail) presenta serie limitazioni in quanto esso non rappresenta uno standard utilizzato globalmente: esso non è compatibile con il tipo di dato XML *dateTime*, e ciò lo rende inutilizzabile, senza previa conversione, nelle query effettuate tramite il linguaggio XQuery sui feed RSS.

Essenzialmente, il motivo per cui questo standard è stato adottato nelle specifiche di RSS è da ricercarsi nella *prior art*. Infatti, ricapitolando:

- nel 1982, l'RFC 822 definisce un formato di data;
- nel 1997, Dave Winer rispetta la *prior art* utilizzando quel formato di data per gli elementi del suo formato di syndicate (*Scripting News*); egli avrebbe potuto scegliere un formato differente, ma non lo fece, e ciò, a quel tempo, rappresentava una scelta sensata;
- nel 1999, Netscape rispetta la *prior art* nell'elaborazione di RSS 0.9, prendendo gli elementi dal formato sviluppato da Dave Winer e non modificando il formato delle date;
- nel 2000, Dave Winer, continuando lo sviluppo di RSS, rispetta la propria *prior art*, e quella di Netscape, non modificando il formato delle date.

In conclusione, si segnala la problematica che si porrà nell'implementazione futura di una funzione di interrogazione sulle sorgenti RSS, in quanto sarà necessario adottare un sistema di conversione dal formato data conforme alla RFC 822 al formato utilizzato nelle specifiche XML (*dateTime*).

Capitolo 4

Esecuzione di Query su Feed RSS

4.1 Il linguaggio XQuery

Abbiamo visto come lo standard RSS poggi le proprie fondamenta su XML: essenzialmente esso è infatti un file XML, e per questo porta con se i vantaggi e gli svantaggi dell'*eXtensible Markup Language.*

Tra i principali vantaggi di XML c'è la possibilità di rappresentare dati di qualsiasi genere: con esso rappresentiamo (e quindi memorizziamo) documenti, pagine Web, immagini vettoriali, clienti, ordini, fatture, etc. Viene quindi di conseguenza la necessità di estrarre sottoinsiemi di dati dalla loro rappresentazione in XML. In altre parole necessitiamo di un linguaggio di interrogazione. A questo scopo il W3C ha elaborato il linguaggio di interrogazione XML Query **[21]**, meglio noto nella sua abbreviazione XQuery, tuttora in fase di definizione, che offre un approccio standard all'interrogazione di dati rappresentati mediante XML.

La componente chiave di XQuery, familiare agli utenti avanzati di XML, è XPath **[22]** (anch'essa una specifica della W3C), linguaggio tramite il quale è possibile elaborare delle espressioni per indirizzare parti di un documento XML. XPath è un linguaggio ideato per operare all'interno di altre tecnologie XML, ed è caratterizzato dal fatto di avere una sintassi non XML: in questo modo può essere meglio utilizzato all'interno di URI o come valore di attributi di documenti XML. XPath opera su una rappresentazione logica del documento XML, che viene modellato con una struttura ad albero: esso definisce una sintassi per accedere ai nodi di tale albero. Oltre a questo, XPath mette a disposizione una serie di funzioni per la manipolazione di stringhe, numeri e booleani, da utilizzare per operare sui valori o sugli attributi dei nodi. Le espressioni definite da XPath per accedere ai nodi dell'albero prendono il nome di *Location Path* (percorsi di localizzazione).

La relazione tra XQuery e XPath è evidente: una locazione XPath (per esempio, "*//libro/editore*", che ha il significato di "*trova tutti gli editori di libri nel database corrente*") è un'espressione XQuery corretta.

Dal punto di vista dei dati, la sintassi di XQuery, simile a quella di SQL, dovrebbe essere familiare a coloro che abitualmente utilizzano database relazionali e che conoscono il concetto di variabile di un linguaggio di programmazione.

4.1.1 La sintassi: brevi cenni

Un'interrogazione in linguaggio XQuery, è costituita da un'espressione che legge una sequenza di nodi XML o un singolo valore, restituendo a sua volta, come risultato, una sequenza di nodi o un singolo valore. Le query XQuery sono costituite da espressioni XPath, per selezionare i nodi da analizzare, e da funzionalità aggiuntive specifiche di XQuery, in grado di dare profondità al recupero di informazioni.

Ad esempio, la query

document("libri.xml")//libro[prezzo > 50] SORTBY (autore)

ricerca all'interno del documento "*libri.xml*" tutti nodi <*libro*> che hanno un prezzo maggiore di 50, ordinando poi il risultato in funzione del nome dell'autore.

XQuery introduce un'istruzione avanzata, chiamata *FLWR Expression* (*FOR – LET – WHERE – RETURN*), per formulare interrogazioni complesse. Essenzialmente essa costituisce una generalizzazione del costrutto SQL "*SELECT – FROM – HAVING – WHERE*".

Analizziamo il seguente esempio, che seleziona gli editori che hanno pubblicato almeno cinque libri:

```
FOR $e IN document("libri.xml")//editore
LET $l := document("libri.xml")//libro[editore=$e]
WHERE count($l) > 5
RETURN
  <risultato>
   {$e}
  </risultato>
```

La prima istruzione, introdotta da *FOR*, crea una lista contenente tutti gli editori presenti nel file *libri.xml* (attraverso l'espressione XPath *document("libri.xml")//editore*), associandola alla variabile *$e*. La seconda riga, introdotta da *LET*, associa a ciascun editore presente nella variabile *$e* la lista dei libri di cui è editore (*document("libri.xml")//libro[editore=$e]*), andando a formare una lista ordinata di tuple ($e,$l). Infine, tramite la terza riga, introdotta da *WHERE*, viene applicato un filtro per il quale vengono considerati solamente gli editori che hanno pubblicato più di cinque libri. L'ultima istruzione, introdotta da *RETURN*, crea il risultato, inserendo i nodi <*editore*> che soddisfano i requisiti richiesti all'interno di un elemento chiamato <*risultato*>.

Oltre a *count()*, presente in questo esempio, XQuery mette a disposizione una serie di funzioni per operare sulle liste di elementi; quelle principali sono le seguenti:

- *avg()*, per calcolare il valor medio degli elementi;
- *union(), intersection(), difference()* che realizzano operazioni 'insiemistiche' sugli elementi.

XQuery supporta anche il costrutto *IF – THEN - ELSE* all'interno delle sue espressioni.
Ad esempio:

```
FOR $l IN document("libri.xml")//libro
RETURN
 <risultato>
  {
   IF ($l/editore ='Editore1')
    THEN $l/titolo
    ELSE $l/autore
  }
 </risultato>
```

Il risultato di questa query contiene i titoli dei libri se l'editore corrisponde a "Editore1", altrimenti contiene il nome degli autori dei libri delle altre case editrici.

Altri due costrutti molto utili in XQuery sono *SOME – IN - SATISFIES e EVERY – IN - SATISFIES*, che permettono di verificare determinate proprietà per gli elementi contenuti in una lista. Ad esempio:

```
FOR $l IN document("libri.xml")//libro
WHERE SOME $t IN $l/titolo SATISFIES (contains($t,"XML") AND contains($t,"tutorial"))
RETURN
 <risultato>
  { $l/titolo }
 </risultato>

FOR $l IN document("libri.xml")//libro
WHERE EVERY $t IN $l/titolo SATISFIES contains($t,"XML")
RETURN
 <risultato>
  { $l/titolo }
 </risultato>
```

La prima query restituisce come risultato il titolo di tutti i libri nei cui titoli compare contemporaneamente sia la stringa "XML" che la stringa "tutorial"; mentre la seconda query restituisce i titoli dei libri che contengono la stringa "XML" nel loro titolo.

4.2 Test di utilizzo di XQuery su Feed RSS

Abbiamo visto all'inizio di questa tesi che una delle due funzionalità richieste ad un wrapper, all'interno del progetto MOMIS, è di poter tradurre interrogazioni OQL_{I3} in un linguaggio adatto alla sorgente interessata dalla query, e di restituirne i risultati al mediatore. In questo

caso, quindi, poiché un feed RSS è essenzialmente un file XML, è necessario individuare un linguaggio di interrogazione XML. Nel paragrafo precedente abbiamo evidenziato come il presente e il futuro delle interrogazioni di file XML sia il linguaggio XQuery, del quale esistono diverse implementazioni in ambiente Java. I requisiti che un'implementazione di XQuery deve soddisfare sono: la semplicità d'uso (tramite API di chiaro utilizzo), la velocità di risposta, la precisione e la completezza (nel dare alla possibilità all'utente di utilizzare il numero più grande possibile di funzioni e costrutti descritti nelle specifiche dello standard stesso); inoltre, caratteristica fondamentale è il tipo di licenza, che deve essere libera e gratuita.

Attraverso una ricerca su internet si è cercato di individuare la migliore implementazione di XQuery da utilizzare per effettuare alcuni test sull'interrogazione di feed RSS; ciò ha portato ad individuare, in particolare, la libreria "*XQEngine*" **[23]**, frutto di un progetto *open source* giunto alla versione 0.69.

4.2.1 Caratteristiche di XQEngine 0.69

Le principali caratteristiche di XQEngine, che lo differenziano sensibilmente da altre implementazioni di XQuery reperibili su internet, sono presentate di seguito.

XQEngine gestisce collezioni di documenti multipli, fino ad un limite massimo superiore ai due miliardi; questi documenti possono risiedere sulla macchina stessa, su server remoti oppure possono essere specificati direttamente nel codice java; ogni documento può contenere oltre otto milioni di nodi; non sono richiesti schemi o DTD.

Questa libreria è un motore del tipo *pre-indexing*: ciò significa che ogni documento che viene aggiunto alla collezione viene immediatamente sottoposto ad un processo di indicizzazione. Per questo motivo, XQEngine risponde velocemente alle query. Nella documentazione allegata a questa libreria si parla di una capacità di elaborazione pari a circa 3Mb per secondo.

Un punto di forza della libreria XQEngine è rappresentato dalla semplicità d'uso: nel caso più semplice, è possibile eseguire una query invocando solamente due API principali: *setDocument(docName)* viene chiamata per ogni documento si voglia aggiungere alla collezione; *setQuery(queryString)* interroga la collezione stessa, restituendo i risultati.

Altro elemento a favore di XQEngine è la sua natura *open source*, che rende possibile la modifica dei sorgenti qualora si voglia estendere e semplicemente adottarne il comportamento.

Una funzione built-in molto comoda presente all'interno della libreria XQEngine è la "*contains-word*": essa permette di ricercare tutti gli elementi che contengono una determinata parola all'interno della collezione. Per esempio, la query

contains-word(//Libro/Titolo, "love")

permette di ricercare tutti i titoli che contengono la parola "love"; nel caso volessi cercare tutti i titoli che contengono allo stesso tempo la parola "love" e la parola "hate", la query che dovrei porre sarebbe la seguente:

contains-word(//Libro/Titolo, "love","hate")

oppure

contains-word(//Libro/Titolo, "love hate")

Questa funzionalità è molto importante in quanto una delle ragioni d'essere dello standard XQuery è proprio la possibilità di effettuare ricerche *full-text*, cioè interrogazioni in grado di individuare l'occorrenza di uno o più termini all'interno di un intero testo.

Le query che hanno come radice un singolo slash (/), o due (//), vengono normalmente poste all'intera collezione di documenti indicizzati; per porre una interrogazione ad un singolo documento è possibile utilizzare la funzione *doc()*, propria di XQuery.

4.2.2 Query su Feed RSS

Per testare le funzionalità della libreria XQEngine è stato creato, all'interno dello sviluppo di questa tesi, un piccolo progetto Java il quale compito è stato eseguire interrogazioni su feed RSS tramite il linguaggio XQuery, con lo scopo di rendere possibile una breve analisi della fattibilità dell'applicazione di questa tecnologia all'interno del progetto MOMIS.

La prima parte di questo lavoro si è incentrata sul tentativo, riuscito, di utilizzare le API di XQEngine all'interno del progetto Java: ciò è risultato molto semplice, in quanto, dopo aver fissato un *XMLReader* (necessario per il parsing di XML) tramite la funzione *setXMLReader()* di XQEngine, per porre la query voluta su un singolo documento è stato sufficiente chiamare la funzione *setQuery()*. Per esempio:

```
m_engine.setXMLReader(xr);
ResultList results = m_engine.setQuery("doc(\"c:\\rss\\userland.xml\")//item/title");
```

dove *m_engine* è un'istanza della classe XQEngine, e *doc(\"c:\\rss\\userland.xml\")//** è la query da porre, volta a richiedere tutti i titoli presenti nel file feed RSS *c:\\rss\\userland.xml.*

Il risultato di una query di questo tipo è un oggetto *ResultList* (classe presente all'interno della stessa libreria di XQEngine), serializzabile in un file di testo tramite la funzione *emitXml()*:

```
<title>Giving the world a pluggable Gnutella</title>
<title>Syndication discussions hot up</title>
<title>Personal web server integrates file sharing and messaging</title>
<title>Syndication and Metadata</title>
<title>UK bloggers get organized</title>
<title>Yournamehere.com more important than anything</title>
```

Per valutare la correttezza e la robustezza delle funzioni della libreria sono state eseguite più query, con diverse profondità, a XQEngine, appurando come questa risponda sempre in maniera corretta.

Il problema che si è posto a questo punto è stato quello di individuare una modalità per convertire l'oggetto *ResultList*, istanza di una classe presente nella libreria XQEngine, in *ResultSet* (la classe utilizzata da MOMIS per rappresentare i risultati di una query quando questi vengono passati dal wrapper al mediatore).

Dopo aver notato che l'utilizzo dei dati all'interno di un oggetto *ResultList* appare alquanto problematico (la scarsa documentazione disponibile evidenzia la possibilità di recuperarli solo attraverso l'analisi nodo per nodo dell'oggetto stesso), si è pensato di sfruttare la funzione *emitXml()* citata prima, riversando i risultati in un file (o più semplicemente un testo) XML, rielaborandoli poi attraverso il parsing dello stesso.

Innanzitutto è apparso chiaro che i risultati, riversati in un testo XML, non sempre avrebbero costituito un documento XML well-formed, poiché spesso manca un nodo radice (come nell'esempio precedente; si precisa come questo non sia in alcun modo un problema della libreria XQEngine, poiché alcune tipologie di query effettivamente prevedono come risultato una semplice serie di elementi). Per ovviare a ciò, e quindi evitare errori di parsing, si è pensato di incapsulare i risultati stessi all'interno di un elemento *<results>*. L'esempio precedente quindi diviene:

<results>
<title>Giving the world a pluggable Gnutella</title>
<title>Syndication discussions hot up</title>
<title>Personal web server integrates file sharing and messaging</title>
<title>Syndication and Metadata</title>
<title>UK bloggers get organized</title>
<title>Yournamehere.com more important than anything</title>
</results>

Essendo l'obiettivo finale quello di ottenere un *ResultSet*, utilizzabile da MOMIS, ed essendo *ResultSet* il tipo di oggetto che si ottiene ponendo una query a SQL Server tramite una connessione jdbc, si è pensato di passare attraverso di esso per il conseguimento dello scopo sopra citato.

In primo luogo, quindi, è apparsa necessaria, in seguito all'esecuzione dell'interrogazione XQuery, la creazione su SQL Server di una tabella adatta a contenerne i risultati. Per ottenere ciò, si è estesa la classe *DefaultHandler* presente nel package *org.xml.sax.helpers*, libreria contenente ciò che occorre per effettuare il parsing XML; la nuova classe così creata (denominata *RSSParser*), è stata dotata di un metodo *String getCreateTableQuery(String ResultFilename)*, in grado di restituire, passandogli il nome del file XML contenente i risultati, la query SQL necessaria per creare una tabella (chiamata *XQuery*) contenente le colonne necessarie ad accoglierne i valori.

A questo punto viene stabilita una connessione a SQL Server e, dopo aver effettuato il *drop* della tabella *XQuery* (se questa esiste già), essa viene ricreata utilizzando la stringa ottenuta precedentemente.

Tramite l'implementazione di un ulteriore metodo all'interno della classe *RSSParser*, chiamato *void FillTable(Connection myConn, String ResultFilename)*, si è reso possibile il riempimento automatico della tabella *XQuery*, a partire dal file XML contenente i risultati della query.

L'ultimo passaggio, quindi, è costituito dall'ottenere il *ResultSet* voluto tramite l'imposizione di una semplice query ad SQLServer:

*ResultSet rs = stmt.executeQuery("select * from xquery");*

4.3 Problemi Riscontrati

4.3.1 HTML nel campo <description>

Dalla versione 0.92 di RSS il campo <*description*>, presente all'interno di <*channel*> e <*item*>, può contenere HTML. In termini pratici, quindi, qualora si ponga una query su un feed RSS che produca, tra i risultati, anche il contenuto del campo <*description*>, ci si può trovare di fronte ad una situazione come la seguente:

<results>
*<description>XML.com features a rich mix of information and services
for the XML
community.</description>*
*<description>Processing document inclusions with general XML tools can be
problematic. This article proposes a way of preserving inclusion
information through SAX-based processing.</description>*
*<description>Tool and API support for the Resource Description Framework
is slowly coming of age. Edd Dumbill takes a look at RDFDB,
one of the most exciting new RDF toolkits.</description>*
</results>

Si può notare la presenza del tag <*br*>, proprio di HTML, all'interno del campo <*description*>: tuttavia, contrariamente a quanto richiesto da un documento XML well-formed (requisito necessario per il funzionamento del parser utilizzato), <*br*> non presenta il tag di chiusura <*/br*>.

Per ovviare a questo problema, che avrebbe reso impossibile l'utilizzo del parser XML, si è racchiuso il contenuto del campo <*description*> all'interno del costrutto "*<![CDATA[*", "*]]>*", attraverso il metodo *String ReplaceInvalidEntities(String SourceString)*, implementato e incluso nel progetto di prova.

L'esempio sopra riportato quindi viene trasformato così:

<results>
*<description><![CDATA[XML.com features a rich mix of information and services
for the XML
community.]]></description>*
*<description><![CDATA[Processing document inclusions with general XML tools can be
problematic. This article proposes a way of preserving inclusion
information through SAX-based processing.]]></description>*
*<description><![CDATA[Tool and API support for the Resource Description Framework
is slowly coming of age. Edd Dumbill takes a look at RDFDB,
one of the most exciting new RDF toolkits.]]></description>*
</results>

4.3.2 I tipi di dato

Come si può immaginare, dall'analisi del documento XML contenente i risultati, non è possibile individuare con assoluta precisione il tipo di dato di ciascuna colonna. Per questo motivo, essendo il presente lavoro a scopo documentativo ed esplorativo sulla tematica, si è uniformata la tipologia delle colonne della tabella creata su SQLServer, rendendole tutte di tipo stringa (*varchar*). Ovviamente questo è molto limitante, e la soluzione, resa necessaria anche dalle caratteristiche stesse del *ResultSet* richiesto dal mediatore (esso deve contenere tutti i campi, o colonne, presenti nella descrizione della struttura della sorgente), richiede sicuramente la costruzione della tabella temporanea sulla base della struttura della sorgente stessa (in questo caso, sulla base dello schema locale caricato al momento dell'acquisizione della sorgente), e di conseguenza il dinamico caricamento dei contenuti, effettuato tramite le necessarie conversioni di dato.

A questo proposito si ricorda il problema del formato delle date, in quanto occorrerebbe un algoritmo di conversione per trasformarle dallo standard RFC 822 ad uno compatibile e comprensibile da SQLServer.

4.3.3 Feed malformati o non rispettanti lo standard

Durante l'analisi di innumerevoli feed RSS si è evidenziata la presenza non trascurabile di documenti che non rispettano strettamente lo standard, soprattutto per quanto riguarda i tipi di dato. Per esempio, le date non sempre sono espresse secondo lo standard RFC 822, ma in alcune sporadiche occasioni vengono rappresentate secondo formati differenti. In altri casi si è verificata l'errata indicazione della versione di RSS utilizzata. Per questo motivo si suggerisce l'implementazione di strumenti di controllo che prevengano comportamenti inattesi introdotti da difformità nell'applicazione dello standard RSS.

Conclusioni

La ricerca sull'Integrazione Intelligente delle Informazioni è l'ambito in cui nasce e cresce il progetto MOMIS: esso ha lo scopo di sviluppare un sistema per l'estrazione e l'integrazione di informazioni da sorgenti eterogenee e distribuite.

La caratteristica principale di MOMIS, che lo distingue dagli altri progetti nello stesso campo, è l'introduzione di comportamenti intelligenti nella fase di integrazione delle sorgenti, basati sull'analisi semantica degli schemi delle sorgenti stesse. Da ciò si parte per la costruzione di una vista globale virtuale costruita in un linguaggio comune: essa, che è detta virtuale in quanto non contiene alcun dato, rappresenta la base per rendere possibile l'esecuzione di interrogazioni sulle sorgenti. MOMIS è quindi dotato di strumenti semi-automatici per l'acquisizione e l'integrazione delle sorgenti, che permettono di velocizzare l'acquisizione stessa.

Nell'ambito di questa tesi è stato analizzato lo standard RSS nelle sue varie versioni, delineando poi il comportamento richiesto ad un wrapper per sorgenti di dati RSS; è stata realizzata una piccola indagine sulla diffusione delle diverse specifiche di RSS, ed è stata quindi realizzata la componente del wrapper stesso in grado di acquisire lo schema del feed RSS connesso; si è data la possibilità al progettista di integrare in MOMIS lo schema di feed RSS sia memorizzati localmente che risiedenti su server remoto. Non è stata implementata la funzionalità di interrogazione sulle sorgenti stesse, in quanto la struttura dei feed RSS, che si basa su XML, introduce innumerevoli difficoltà in questo senso, comuni in generale a tutti i procedimenti di interrogazione su dati semi-strutturati; nonostante ciò, è stato svolto un lavoro di documentazione su questa problematica, analizzando il linguaggio di interrogazione per documenti XML (XQuery), ed arrivando ad individuare una libreria per Java (XQEngine) in grado di sfruttare questa specifica; inoltre è stato realizzato un progetto Java di prova nel quale sono state sperimentate le funzionalità di questa libreria, delineando un procedimento che potrà essere seguito in futuro nello sviluppo del modulo di interrogazione per feed RSS.

L'estrema diffusione dello standard RSS nel mondo di internet evidenzia l'importanza che può avere la possibilità di integrare informazioni provenienti da esso in un progetto come MOMIS. Per ottenere questo scopo finale, sarà necessario provvedere alla realizzazione del modulo deputato all'interrogazione dei feed RSS stessi; in questa successiva estensione del wrapper RSS potrà rendersi necessario effettuare dei cambiamenti sul modulo di acquisizione degli schemi, oppure agli schemi locali predefiniti, in questa prima fase semplificati per ovviare alle problematiche sorte durante lo sviluppo.

Bibliografia

1: “MOMIS Home Page”, http://www.dbgroup.unimo.it/Momis/
2: “A brief history of Decision Support System”, http://dssresources.com/history/dsshistory.html
3: “Data warehouse”, http://it.wikipedia.org/wiki/Data_warehouse
4: “I3 Initiative Home Page”, http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-6/web-agent/www/i3.html
5: “DARPA Home”, http://www.darpa.mil/
6: “Sigmod Record, Vol. 28 No. 1“, http://www.sigmod.org/sigmod/record/issues/9903/
7: “ODMG Home Page”, http://www.odmg.org/
8: “ODB-Tools”, http://www.dbgroup.unimo.it/Momis/prototipoPub/modules/ODBTools/
9: “WordNet”, http://wordnet.princeton.edu/
10: “Resource Description Framework Model and Syntax”, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
11: “RSS 0.91”, http://backend.userland.com/rss091
12: “PICS Technical Specifications”, http://www.w3.org/PICS/#Specs
13: “RFC 822, Standard for the format of ARPA internet text messages”, http://asg.web.cmu.edu/rfc/rfc822.html
14: “RSS 0.92 Userland Specifications”, http://backend.userland.com/rss092
15: “SOAP meets RSS”, http://www.thetwowayweb.com/soapmeetsrss
16: “RSS 0.93 Userland Specifications”, http://backend.userland.com/rss093
17: “RSS 2.0 Specifications”, http://blogs.law.harvard.edu/tech/rss
18: “RDF Site Summary”, http://web.resource.org/rss/1.0/
19: “RFC 1123, Requirements for Internet Hosts”, http://www.ietf.org/rfc/rfc1123.txt
20: “RFC 2822, Internet Message Format”, http://www.faqs.org/rfcs/rfc2822.html
21: “W3C XML Query”, http://www.w3.org/XML/Query/
22: “XML Path Language”, http://www.w3.org/TR/xpath
23: “XQEngine”, http://xqengine.sourceforge.net/