UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA - "ENZO FERRARI"
Corso di Laurea Magistrale in Ingegneria Informatica

Progettazione e sviluppo di tecniche per l'integrazione di dati tradizionali e multimediali nel sistema MOMIS

Candidato:
Marco Maria Santese
Matricola 76215

Relatore:
Prof. Domenico Beneventano

Correlatori:
Dott. Alberto Corni
Ing. Mirko Orsini (DataRiver S.R.L.)

Anno Accademico 2013–2014

Alla mia Famiglia, al DBGroup e agli amici che mi hanno aiutato a raggiungere quest'obiettivo.

"Un anello per domarli, un anello per trovarli,
Un anello per ghermirli e nel buio incatenarli."
J. R. R. Tolkien

Indice

Parole chiave:
Integrazione Dati
Interrogazioni Dati Tradizionali e Multimediali
MOMIS
WINDSURF
Elaborazione Analisi Mediche

Abstract

Il lavoro di tesi tratta l'Integrazione Dati provenienti da sorgenti eterogenee ed ha l'obiettivo dell'elaborazione di interrogazioni tramite fusione di dati "tradizionali" (database, file testuali, ...) e "multimediali" (immagini, video, ...).

La tesi si colloca nell'ambito del sistema di Integrazione Dati MOMIS (Mediator envirOnment for Multiple Information Sources), sviluppato dal DBGroup dell'Università di Modena e Reggio Emilia e distribuito in versione open source da DataRiver, spin-off universitario presso il quale ho svolto il tirocinio.

Il Sistema MOMIS "tradizionale" è in grado di integrare dati provenienti da sorgenti tradizionali e di generare uno Schema Globale integrato rappresentativo di tali sorgenti. L'utente effettua quindi un'interrogazione sullo Schema Globale integrato, specificando criteri di ricerca, e riceve una risposta unificata ottenuta tramite fusione dei dati che provengono dalle varie sorgenti locali, fusione basata sostanzialmente su operazioni standard del linguaggio SQL.

Consideriamo ora una sorgente di dati "multimediali", quali ad esempio immagini, che consente la specifica delle cosiddette "Top-K query", in grado di restituire le immagini più simili (in base ad opportuni criteri di similarità) ad un'immagine di riferimento.

Un Sistema di Integrazione tra dati tradizionali e multimediali deve essere in grado di costruire uno Schema Globale integrato rappresentativo sia delle sorgenti locali tradizionali che di quelle multimediali, e deve consentire all'utente di specificare interrogazioni che contengano congiuntamente sia criteri di ricerca tradizionali che criteri di ricerca per similarità sui dati multimediali. Il sistema deve essere quindi in grado di effettuare la fusione dei dati che provengono dalle sorgenti locali, sia tradizionali che multimediali, per restituire all'utente una risposta unificata.

Nella presente tesi, dopo un'analisi delle soluzioni già presentate in letteratura, in particolare di quella precedentemente sviluppata nel contesto del Sistema MOMIS, viene sviluppata ed implementata una nuova soluzione per la fusione di dati tradizionali e multimediali, la cui caratteristica fondamentale è quella di utilizzare ancora solo operazioni standard del linguaggio SQL. In altri termini, nella soluzione proposta ed implementata i criteri di ricerca per similarità sono tutti gestiti dalla sorgente

dati multimediali e la fusione con dati tradizionali avviene con un “SQL-engine” standard.

Capitolo 1

Introduzione

Nel corso degli anni è diventata sempre più rilevante la necessità di accedere ad informazioni distribuite e nello stesso modo è incrementata anche l'esigenza di riuscire ad integrare fra loro le informazioni provenienti da sorgenti eterogenee [21]. Le imprese attualmente hanno a disposizione moltissime fonti di informazione, che possono contenere dati correlati tra loro ma spesso ridondanti, eterogenei e non sempre consistenti. La necessità è quella di poter accedere in modo semplice a tutte le informazioni aziendali distribuite ed anche poter costruire applicazioni che utilizzino in tempo reale tali informazioni. Il problema dell'integrazione dell'informazione (data integration o information integration) consiste nell'integrare i dati correlati, residenti nelle diverse sorgenti e fornire all'utente una vista integrata di tali dati. I problemi che devono essere affrontati in questo ambito sono principalmente dovuti ad eterogeneità strutturali e applicative (basti pensare alle differenti piattaforme hardware, DBMS, modelli dei dati ...), nonché alla mancanza di una ontologia comune, che porta a differenze semantiche tra le fonti di informazione. A loro volta le differenze semantiche possono dare origine a diversi tipi di conflitti, che vanno dalle semplici incongruenze nell'uso dei nomi (sinonimi: quando nomi differenti sono utilizzati in sorgenti diverse per identificare gli stessi concetti) a conflitti strutturali (quando modelli diversi sono utilizzati per rappresentare le stesse informazioni). Questo problema di Integrazione Dati si amplifica in contesti dove le informazioni in esame sono di natura differente dal classico dato testuale contenuto in un possibile RDBMS, come per esempio nel caso di dati multimediali (immagini, video, ecc...).

La tesi si colloca nell'ambito del sistema di Integrazione Dati MOMIS (Mediator envirOnment for Multiple Information Sources)[9], sviluppato dal DBGroup dell'Università di Modena e Reggio Emilia e distribuito in versione open source da DataRiver S.R.L.[1], spin-off universitario presso il quale ho svolto il tirocinio. L'approccio seguito dal sistema MOMIS per l'integrazione di varie sorgenti di informazioni è

[1] http://www.datariver.it/

quello che si basa sulla creazione di uno "schema globale integrato", detto "Global Schema", che permette di mantenerle unite. In questo modo gestendo tutti i dati in un modo comune, il Global Schema permette all'utente di effettuare una query sulla base della visione d'insieme delle informazioni trattate: la query effettuata su uno schema mediato verrà scomposta in varie sotto-query che saranno eseguite sulle singole sorgenti di dati coinvolte; ciò è realizzato per mezzo delle operazioni di *unfolding* e *rewriting* che sono effettuate in base alla struttura dello schema mediato e delle sorgenti trattate. Infine il risultato di queste sotto-query verrà unificato utilizzando tecniche di fusione dati in cui vengono risolti eventuali conflitti.

Considerando anche una sorgente di dati "multimediali", quali ad esempio immagini, che consente la specifica delle cosiddette "Top-K query", in grado di restituire le immagini più simili (in base ad opportuni criteri di similarità) ad una data come riferimento. L'obiettivo della tesi è quello di creare un Sistema di Integrazione tra dati tradizionali e multimediali in grado di costruire un Global Schema integrato rappresentativo di entrambe le possibili tipologie di sorgenti, e quindi di consentire all'utente di specificare interrogazioni che contengano congiuntamente sia criteri di ricerca tradizionali che criteri di ricerca per similarità sui dati multimediali. Il sistema deve essere in grado di effettuare la fusione dei dati che provengono dalle sorgenti locali, sia tradizionali che multimediali, per restituire all'utente una risposta unificata.

La tesi è stata svolta nell'ambito del progetto "DataRiver Data Integrator"[2], all'interno del quale il sistema MOMIS è stato esteso per integrare sorgenti multimediali gestite dal framework **Windsurf**[3] (Wavelet-based INDexing of ImageS Using Region Fragmentation) [19], sviluppato dall'Università di Bologna. Un quesito fondamentale alla base di tale progetto era il seguente: "Come integrare una sorgente dati multimediale (gestita da Windsurf) che supporta Top-K query nel sistema a mediatore MOMIS che non ha tale capacità?" Più precisamente, "Come effettuare la fusione dei risultati delle Top-K query provenienti da Windsurf nel sistema MOMIS la cui elaborazione delle interrogazioni è basata su un *SQL*-engine tradizionale?".

In altri termini l'obiettivo è quello di individuare delle soluzioni per eseguire "Top-K Global Query" nel sistema MOMIS, lasciando sostanzialmente inalterato il suo metodo di elaborazione delle interrogazioni. Questo è un aspetto centrale del progetto che lo differenzia rispetto a soluzioni precedentemente proposte. In particolare, in [5] il sistema MOMIS era stato integrato con **MILOS** [1] (Multimedia dIgital Library for On-line Search), un sistema di gestione dei dati multimediali utilizzato per l'implementazione efficace di biblioteche digitali. Il framework ottenuto era in

[2]Finanziamento ottenuto da DataRiver dal MISE, "Fondo per l'Innovazione Tecnologica - FIT Start-Up".

[3]http://www-db.deis.unibo.it/Windsurf/

grado di elaborare "Top-K Global Query" però aveva richiesto, a livello del sistema a mediatore di MOMIS, un metodo di elaborazione delle interrogazioni (basato sull'algoritmo *MEDRANK* [14]) completamente differente rispetto a quello per dati tradizionali (basato su operazioni di *Full Outer Join* e quindi effettuabile tramite un SQL-engine).

1.1 Cos'è la Data Integration

L'Integrazione dei Dati (Data Integration - DI) permette di combinare i dati provenienti da diverse sorgenti eterogenee per averne una visione unificata. La gestione dei dati (data management) disciplina nota come anche quella dell'integrazione dei dati (DI), ha subito un'impressionante espansione nell'ultimo decennio. Oggigiorno esistono molteplici tecniche utilizzate in diverse applicazioni e contesti aziendali che permettono di realizzare ciò. L'integrazione dei dati è un insieme di tecniche fra le quali ricordiamo l'ETL (Extract Transform Load), la più usata, la Data Federation, Database Replication, sincronizzazione dei dati e sorting. Tutte queste tecniche sono sviluppate attraverso tools che permettono di accedere a database, ad applicazioni o a file per caricare ed estrarre le informazioni. Le soluzioni basate su queste tecniche richiedono l'implementazione di codice manuale, automatico o un mix tra le due. L'integrazione dei dati è uno strumento estremamente importante, basti pensare al caso due società che necessitano di fondersi e che per questo hanno la necessità di riunire i loro sistemi informativi in un solo sistema informativo aziendale. Di solito il sistema da popolare con i dati provenienti dai due sistemi è un Data Warehouse (DW), perché permette alle aziende di eseguire anche analisi statistiche migliori, sulla base dei dati gestionali memorizzati.

1.2 Integrazione di Dati Tradizionali e Multimediali

I dati multimediali sono di natura differente rispetto ai dati tradizionali e logicamente risulterà esser diverso anche il modo di interrogazione. Per esempio il metodo per effettuare query su dati tradizionali si basa sulla verifica che alcuni record (che dovranno esser selezionati) soddisfino determinati vincoli imposti dall'utente, come per esempio l'uguaglianza, relazione di "maggiore, minore", ecc... . Possiamo subito affermare che una ricerca effettuata basandoci sulle stesse modalità di esecuzione utilizzate su dati tradizionali non è utilizzabile su quelli multimediali: la **ricerca esatta** di feature in immagine non ha nessun significato in quanto le feature usualmente vengono paragonate fra loro usando funzioni di distanza. Quindi in questo contesto introdurremo il concetto di **similarità**, di **funzione di distanza** basata sulle nozioni matematiche di "spazio metrico". La ricerca di similarità è diventata

un punto fondamentale in molte aree di applicazione, per esempio nell'estrazione di informazioni da un contesto multimediale, data mining, pattern recognition, machine learning. Questo problema era stato originariamente analizzato dalla comunità della "geometria computazionale", dove era conosciuto come "closest point", "nearest neighbor" e "post office problem". Attualmente la similarità ha catturato l'attenzione anche della comunità delle "basi di dati", in quanto risolvere questo problema porterebbe ad un miglioramento dell'efficienza che è un elemento di interesse e necessità attuale, amplificata dall'aumento dei volumi di dati da analizzare.

L'obiettivo è quello di trattare dati multimediali e tradizionali in modo omogeneo, cioè offrendo all'utente un'interfaccia di query unica, in modo efficiente, cioè con performance comparabili a quelle dei sistemi di database tradizionali. Questa è una delle giustificazioni della scelta del framework **Windsurf** [19], che permette di effettuare le ricerche su contenuti multimediali in modo efficace ed efficiente grazie all'utilizzo di indici.

1.3 Integrazione di Dati Tradizionali e Multimediali: scenari applicativi

La ricerca di elementi multimediali e l'unificazione dei risultati di natura eterogenea ha molti campi di applicazione. Riportiamo ad esempio quello considerato in [5], dove un rappresentante di marmi che si trova da un cliente, deve dare risposta sulla disponibilità di un blocco di marmo simile a quello già posseduto. Il rappresentante vorrebbe essere aiutato da un sistema che gli consigli qual'è il marmo più simile a quello di cui è già in possesso il cliente. Il rappresentante per esempio scatterà una foto del marmo tramite smartphone, aggiungerà alcuni possibili vincoli extra come tipo, dimensioni, disponibilità, costo, somiglianza rispetto a quello fotografato ed infine aspetta di ricevere come risposta solo i blocchi simili ad esso, con tutti i dettagli a lui necessari. Quindi in questo semplice scenario si può intravedere la necessità di supporto per ricerche combinate fra elementi multimediali ed elementi testuali tradizionali.

Lo scenario applicativo che è stato considerato durante il tirocinio e che verrà discusso nella tesi si colloca nell'ambito medico. In questo caso, lo scenario è caratterizzato da dati "tradizionali" contenuti nelle basi di dati, mentre i dati "multimediali" (fondamentalmente saranno le analisi tipo TAC, Radiografie, Risonanze Magnetiche, PET, ecc...) sono contenuti in directory condivise fra gli ospedali. L'obiettivo è fornire al medico strumenti di supporto su alcune decisioni, per permettergli, per esempio di effettuare un rapida comparazione di determinati risultati di analisi, per analizzarne i comportamenti intrapresi in condizioni analoghe da altri specialisti.

In particolare, verrà considerato il caso degli esami PET[4] (Positron Emission Tomography) con la possibilità di fornire in input un'analisi e dei dati (che potranno essere l'età, il sesso, e altre caratteristiche del paziente) ed di ottenere in output lo storico dei comportamenti intrapresi in casi analoghi in passato, per basare la nuova decisione avendo presenti questi dati utili organizzati secondo alcuni criteri scelti appunto dal medico.

1.4 Organizzazione della Tesi

La tesi si colloca nell'ambito del sistema di Integrazione Dati MOMIS (Mediator envirOnment for Multiple Information Sources), sviluppato dal DBGroup dell'Università di Modena e Reggio Emilia e distribuito in versione open source da DataRiver, spin-off universitario presso il quale ho svolto il tirocinio.

Il tirocinio e quindi il lavoro di tesi è stato svolto nell'ambito del progetto "DataRiver Data Integrator", finanziato dal Ministero dello Sviluppo Economico italiano, "Fondo per l'Innovazione Tecnologica - FIT Start-Up". In questo contesto le tecniche di integrazione di dati del sistema MOMIS sono state estese verso due direzioni principali:

- La creazione di tecnologie intelligenti semantiche per l'annotazione e la scoperta di mapping, con il fine di aumentare il grado di automazione dei processi di integrazione.

- La gestione dei dati multimediali, per riuscire ad integrare assieme sorgenti di dati "multimediali" e "tradizionali".

Il mio lavoro di tesi si colloca nell'ambito di questo secondo aspetto.

Nel capitolo 2 verrà descritto il sistema MOMIS, e ci concentreremo sugli aspetti che maggiormente interessano il lavoro di tesi. In particolare, la prima parte del capitolo riguarda la descrizione dell'architettura del sistema MOMIS, utile soprattutto per gli aspetti implementativi. In questa sezione riporteremo il minimo indispensabile per comprendere il resto della tesi, ovvero quei componenti di MOMIS, in particolare il Query Manager, che sono stati maggiormente utilizzati (e modificati) durante lo sviluppo della tesi. Una sezione di questo capitolo sarà dedicata anche alle definizioni formali, indispensabili per comprendere il funzionamento del Query Manager.

Nel capitolo 3 si analizzeranno i due approcci per gestire la multimedialità in un sistema a mediatore quale MOMIS:

[4] `https://it.wikipedia.org/wiki/Tomografia_a_emissione_di_positroni`

- Il primo approccio è quello usato nel sistema **MOMIS-MILOS**, nel quale le risposte locali sono delle "ranked list" e come metodo di fusione si utilizza MEDRANK e si presuppone di poter accedere alla lista elemento per elemento; per realizzare ciò si necessita di modificare la logica alla base del Query Manager di MOMIS.

- Il secondo è un nuovo metodo proposto, che sarà oggetto della presente tesi, indicato come **MOMIS-Windsurf**, nel quale le risposte locali sono delle "scored list" e come metodo di fusione si utilizza sempre il Full Outer Join, quindi non è necessario accedere alla lista elemento per elemento; in altre parole, come verrà discusso ampiamente nel seguito, il Query Manager resta concettualmente e sostanzialmente invariato.

Nel capitolo 4 si analizza la progettazione ed implementazione del framework MOMIS-Windsurf che permette l'elaborazione delle interrogazioni di dati tradizionali e multimediali. Si forniranno le motivazioni delle scelte, si analizzeranno le caratteristiche tecniche di Windsurf e si analizzerà la sua architettura. Questo capitolo si conclude con la presentazione del wrapper necessario per mettere in comunicazione il mediatore (MOMIS) con la sorgente locale multimediale (Windsurf).

Nel capitolo 5 verrà analizzato un importante campo applicativo del framework MOMIS-Windsurf, che si colloca nell'ambito delle Analisi Mediche e riguarda gli esami PET (Positron Emission Tomography).

Nel capitolo 6 viene presentato un esempio pratico del framework MOMIS-Windsurf realizzato che mostra il funzionamento del sistema complessivo e fornisce anche una guida all'utilizzo.

Nel capitolo 7 verranno esposti i dettagli implementativi per la realizzazione del Framework MOMIS-Windsurf, svolti durante la tesi. L'implementazione realizzata può essere schematizzata nella seguente struttura:

- Modifiche a Windsurf
- Modifiche a MOMIS
- Creazione wrapper Windsurf per MOMIS

Ognuna di queste classi di modifiche verrà analizzata nel dettaglio in questo capitolo.

Per concludere, nei capitoli finali si analizzerà concretamente ciò che è stato realizzato, gli obiettivi raggiunti e i possibili miglioramenti sviluppabili in un prossimo futuro.

Capitolo 2

MOMIS

MOMIS è un sistema di Integrazione di Dati Open Source che sfruttando la semantica presente nelle sorgenti informative è in grado di integrare sorgenti dati eterogenee (strutturate e semi-strutturate) e distribuite. Il software è giunto alla release 1.2 ed è scaricabile previa registrazione gratuita.

In questa sezione verrà analizzato MOMIS, oggetto di numerose tesi presso il DBGroup. Per chiarezza riporterò nel seguito una breve descrizione del sistema MOMIS tratta sia da documenti scientifici che da precedenti lavori di tesi presso il DBGroup [21, 13].

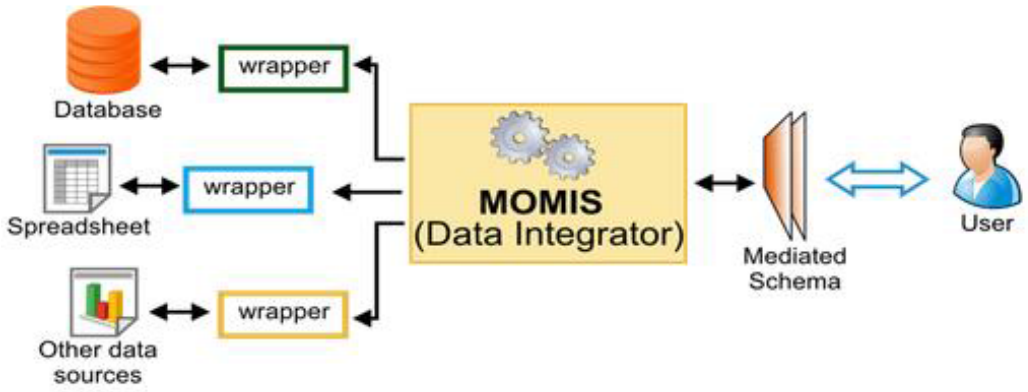
2.1 Introduzione a MOMIS

MOMIS (Mediator envirOnment for Multiple Information Sources) è un sistema Open Source per l'estrazione e l'integrazione dei dati da sorgenti strutturate e semi-strutturate, distribuito dall'Aprile 2010, da DataRiver (http://wwww.datariver.it), spin-off universitaria istituita nel giugno 2009. Lo sviluppo di MOMIS è iniziato nell'ambito del progetto nazionale INTERDATA, e nell'ambito del progetto D2I, sotto la direzione della Professoressa S. Bergamaschi. L'attività di ricerca è continuata all'interno del progetto di ricerca europeo IST SEWASIE: Semantic Webs and Agents in Integrated Economies (2002/2005). È stato inoltre esteso nell'ambito del progetto MUR NeP4B: Networked Peers for Business (2006/2009) e del progetto IST-EU RDT STASIS (SofTware for Ambient Semantic Interoperable Services) (2006-2009). Il sistema MOMIS è basato sull'architettura *wrapper/mediator* (vedere Figura 2.1): il wrapper rende disponibile la sorgente dati locale da integrare, il mediatore genera uno schema mediato, detto anche Global Virtual View (GVV) o Global Schema (GS), degli schemi delle sorgenti locali. Una volta ottenuto lo schema globale, l'utente interroga lo schema ed ottiene una vista completa ed unificata dei dati contenuti nelle sorgenti locali, in modo totalmente trasparente per l'utente.
Nei seguenti step viene spiegato il processo di esecuzione di una query:

1. Un utente interroga il sistema mediatore;
2. La query viene riscritta generando un insieme di subquery che verranno eseguite sulle sorgenti locali collegate tramite wrapper;
3. I risultati parziali vengono fusi dal mediatore, che presenta all'utente una risposta finale unificata di tali dati.

I dati non vengono replicati e la risposta ad ogni interrogazione riporta i dati aggiornati Il GS è una concettualizzazione che descrive un insieme di sorgenti dati

Figura 2.1: MOMIS: struttura wrapper/mediator



distribuite ed eterogenee, l'utente può formulare delle query sul GS e ricevere un'unica risposta ottenuta tramite la fusione dei risultati provenienti dalle sorgenti locali, in modo trasparente rispetto alle sorgenti coinvolte.

2.2 MOMIS: Architettura del Sistema

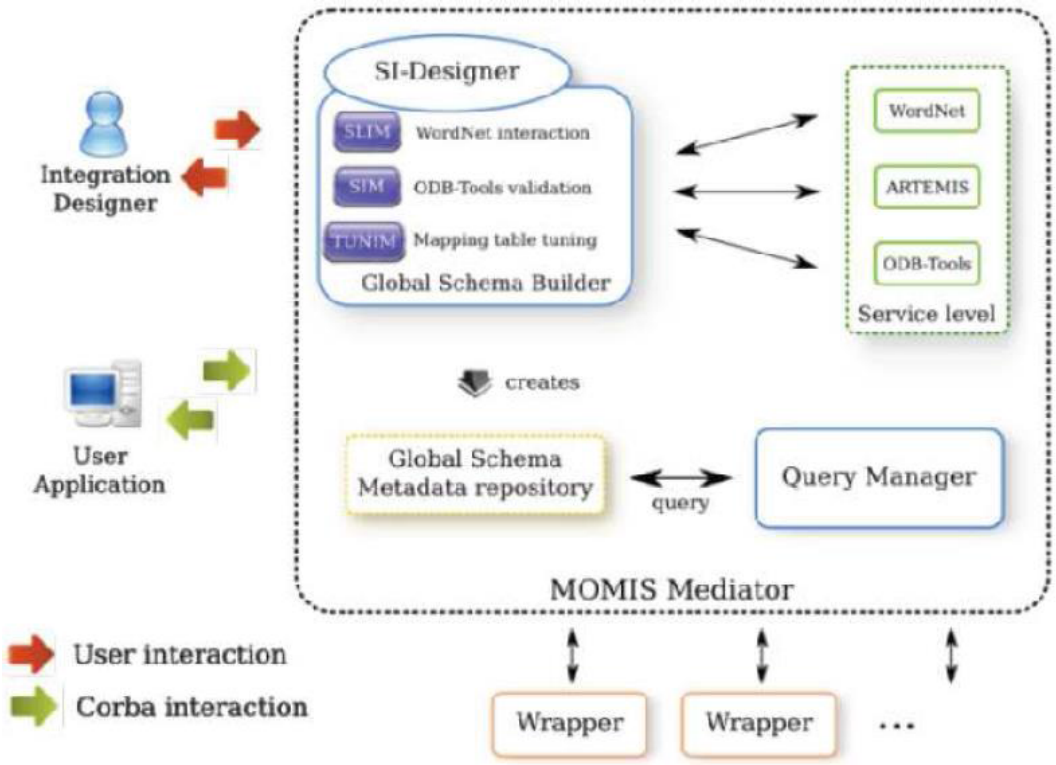
MOMIS è stato progettato per fornire un accesso integrato ad informazioni eterogenee memorizzate in sorgenti dati strutturate (es. database relazionali, database object oriented) e semi-strutturate (es. file XML, EXCEL, CSV). In [21] viene descritta in maniera più che dettagliata l'architettura di MOMIS, per comprendere meglio lo strumento ne riportiamo una parte. I componenti principali di MOMIS, illustrati in Figura 2.2), sono dettagliati nel seguito.

2.2.1 Wrapper

Wrapper: Posti al di sopra di ciascuna sorgente, sono i moduli che rappresentano l'interfaccia fra il Mediatore e le sorgenti dati locali. La loro funzione è duplice:

- In fase d'integrazione forniscono la descrizione dell'informazione in essa contenute. Questa descrizione è fornita attraverso il linguaggio ODL_{I^3}.

Figura 2.2: MOMIS: Componenti della struttura wrapper/mediator



- In fase di query processing traducono la query ricevuta dal Mediatore in un'interrogazione comprensibile dalla sorgente stessa. Devono inoltre esportare i dati ricevuti come risposta all'interrogazione, presentandoli al mediatore, attraverso il modello comune di dati utilizzato dal sistema. I wrapper sono quindi equivalenti ai connettori già visti per gli strumenti ETL analizzati per la prima delle la funzioni offerte, quella di trasformazione dati.

2.2.2 Mediatore

Il mediatore rappresenta il cuore del sistema ed è composto da due sotto-moduli:

- Global Schema Builder (GSB): è il modulo che integra gli schemi locali, il quale partendo dalle descrizioni delle sorgenti, espresse attraverso il linguaggio ODL_{I^3}, genera un unico schema globale da presentare all'utente. Il progettista usa l'interfaccia grafica SI-Designer per integrare le sorgenti e poi, tutte le informazioni create con SI-Designer vengono salvate in un oggetto CORBA: "Global Schema Object". Tale oggetto permetterà al modulo Query Manager di effettuare interrogazioni sullo schema integrato.

- Query Manager (QM): è il modulo di gestione delle interrogazioni. In particolare, utilizzando tecniche di unfolding genera le query da inviare ai wrapper,

partendo dalla singola query formulata dall'utente sullo schema globale. Il QM genera automaticamente la traduzione della query nelle corrispondenti sotto-query da spedire alle singole sorgenti, effettua la fusione dei risultati parziali e restituisce all'utente una risposta unificata. Per la fusione dei risultati parziali viene utilizzato un DBMS (DataBase Management System) di supporto; essendo MOMIS un progetto open source nasce la necessità di utilizzare come DBMS di appoggio non una soluzione proprietaria, ma una soluzione Open Source. La scelta è caduta su un RDBMS open source, completamente scritto in Java, HSQLDB (Hyper Structured Query Language DataBase), che verrà descritto nei prossimi paragrafi.

2.2.3 HSQLDB

HSQLDB (Hyper Structured Query Language DataBase) nasce nel 2001 dal Hypersonic SQL Project, è un DBMS relazionale Open Source (viene distribuito con licenza BSD license1), completamente scritto in Java (quindi completamente portabile). HSQLDB è conforme allo standard SQL92, inoltre supporta alcuni statement SQL previsti nello standard SQL2008. Nella prima implementazione del Query Manager è stato scelto come DBMS di supporto Microsoft SQL Server, un RDBMS prodotto da Microsoft ad alte prestazioni, progettato per gestire altissimi volumi di operazioni transazionali in ambiente multiutente. Però dall'altra parte presenta degli svantaggi: per poterlo utilizzare all'interno di una applicazione è necessaria una licenza commerciale ed inoltre questo RDBMS è compatibile solo con la piattaforma Windows. Dopo avere installato MOMIS, per eseguire delle query su un GS, generato tramite il Global Schema Builder, è necessario installare Microsoft SQL Server e creare un database definito nel file di configurazione di MOMIS, sul quale verranno create le tabelle temporanee necessarie per l'elaborazione delle query. Per questi motivi è stato scelto come nuovo DBMS di supporto al query processing, HSQLDB, un RDBMS Open Source e completamente portabile perché scritto in Java. Il database engine HSQLDB viene integrato nell'applicazione MOMIS quindi non è necessario installarlo a parte. Inoltre, diversamente dagli altri inner database engine (es. Derby, H2) Open Source implementati in Java, HSQLDB supporta l'operatore Full Outer Join, operatore che viene utilizzato nella Mapping Query.

2.3 MOMIS: Definizioni

Un Data Integration System di MOMIS $DIS = \langle GS, \mathcal{L}, \mathcal{M} \rangle$ è costituito da[1]:

[1]Sia lo schema locale che quello globale devono essere espressi nel linguaggio ODL_{I^3} [10]. Comunque, per l'obiettivo di questa tesi noi consideriamo sia il GS che LS_i come schemi relazionali (ricordandoci che in questi capitoli ci riferiremo ai loro elementi rispettivamente come global e local

- GS è il *Global Schema*, un set di *classi globali*, denotate da G;
  una classe globale G è una relazione con schema $S(G) = (GA_1, GA_2, \ldots, GA_m)$, dove GA_i è un *attributo globale* di G.

- $\mathcal{L}$ è un set di *schemi locali*, $\mathcal{L} = \{LS_1, \ldots, LS_k\}$; uno schema LS è un set di *classi locali*, denotate da L;
  una classe locale L è una relazione con schema $S(L) = (LA_1, LA_2, \ldots, LA_n)$, dove LA_i sono gli *attributi locali* di L;

- $\mathcal{M}$ è un set *Global-As-View* (GAV) di dichiarazioni di mapping [24] fra GS e $\mathcal{L}$ espresse come segue:

  per ogni classe globale $G \in GS$ definiamo:

  1. un set non vuoto di classi locali denotate da $\mathcal{L}(G)$, che appartengono alle sorgenti locali in $\mathcal{L}$.
  2. una query Q_G su $\mathcal{L}(G)$, detta Mapping Query.

Intuitivamente, il GS è l' *Intensional Representation*[2] dell'informazione offerta dal *DIS*, mentre il mapping $\mathcal{M}$ specifica come GS è in relazione con le sorgenti locali integrate. Il GS e $\mathcal{M}$ devono essere definiti a tempo di progetto dall' Integration Designer. Uno degli elementi di innovazione del Framework MOMIS è che questo processo di data integration è effettuato in modo semi-automatico come ampiamente descritto in [3, 7]; qui consideriamo il risultato di tale processo di data integration in una forma opportunamente semplificata per focalizzarci sulla data fusion. Per ogni classe globale G, una *Mapping Table* (MT) è generata, le cui colonne rappresentano il set $\mathcal{L}(G) = \{L_1, \ldots, L_n\}$ delle *classi locali appartenenti* a G, e le cui righe rappresentano $S(G)$, in altre parole gli attributi globali GA di G. Un elemento $MT[GA][L]$ è un set di attributi locali L mappati su un attributo globale GA; l'*Integration Designer* può definire le *Data Transformation Functions* per specificare come i valori degli attributi locali devono essere trasformati nei corrispettivi valori degli attributi globali. In questa sezione, per semplicità consideriamo una versione semplificata del framework generico di MOMIS [3, 7], dove un elemento $MT[GA][L]$ rappresenta un singolo attributo locale di L che è mappato su un global attribute GA, oppure $MT[GA][L]$ risulta essere vuoto (non ci sono attributi locali L mappati sull'attributo globale GA).

Inoltre consideriamo $S(G) = \cup_{L \in \mathcal{L}(G)} S(L)$, quindi i nomi degli attributi locali e globali sono gli stessi.

class per rispettare la terminologia di MOMIS).

[2] `http://en.wikipedia.org/wiki/Intensional_definition`

Esempio: Settore del Tessile

Consideriamo l'esempio che era stato introdotto in [2], dove è stato mostrato il nostro metodo applicato all'integrazione di cinque siti Web, tre Italiani e due Americani, che descrivono aziende e prodotti nel settore tessile.

In questo paragrafo sono state riportate tre delle 5 sorgenti dati, due Americane (*U=Usawear* e *F=Fibre2fashion*) ed una Italiana (*I=Ingromarket*); lo schema locale (con istanze) è mostrato in figura 2.3.

Il processo di integrazione è stato omesso ed in questo capitolo noi consideriamo le tre classi globali del risultante GS:

1. `Enterprise`, con classi locali $\mathcal{L}(\texttt{Enterprise}) = \{\texttt{UE}, \texttt{FE}\}$ e schema $S(\texttt{Enterprise}) = (\texttt{ID}, \texttt{A}, \texttt{B}, \texttt{C}, \texttt{D})$;
2. `Business`, con classi locali $\mathcal{L}(\texttt{Business}) = \{\texttt{IB}\}$ e schema $S(\texttt{Business}) = (\texttt{ID}, \texttt{A}, \texttt{B}, \texttt{E})$;
3. `Category`, con classi locali $\mathcal{L}(\texttt{Category}) = \{\texttt{UC}, \texttt{IC}\}$ e schema $S(\texttt{Category}) = (\texttt{E}, \texttt{F})$.

Local Schema *U=Usawear*

UE: U.Enterprise

ID	A	B	C
a	20	MZ	YES
c	18	MR	NO
d	22	NULL	YES
e	20	MZ	YES
f	16	NULL	YES

UC : U.Category

B	F
MZ	NULL
MR	s

Local Schema *F=Fibre2fashion*

FE : F.Enterprise

ID	A	B	D
a	18	ZZ	123
b	22	MK	234
c	18	MR	345
d	22	MK	567
e	20	ZZ	567

Local Schema *I=Ingromarket*

IB : I.Business

ID	A	B	E
b	19	MW	345
q	22	MZ	678

IC : I.Category

B	F
MZ	s
MW	s

Figura 2.3: Istanze delle Classi Locali. (A = NumOfEmployee, B = Category, C = ContactPerson, D = Fax, E = Phone, F = SubCateg)

Data Fusion

Il primo passo per il processo di Data Fusion process è l'*Object Identification*, l'identificazione di uno stesso oggetto in differenti sorgenti dati; questo è un compito impegnativo affrontato in molti lavori come per esempio [18, 30], ma questi studi sono posti al di fuori del campo di applicazione del framework di MOMIS.

Si assuma che l'*Object Identification* sia stata già effettuata ed introdurremo il concetto di *Join Conditions* come un modo conveniente per effettuare l'identificazione degli oggetti quando si può supporre che esistano identificatori di oggetti condivisi (senza errori) fra le diverse sorgenti.

Nell'esempio, per la classe globale `Enterprise` l'attributo `ID` è l' object identifier e la Join Condition è `JC(UE, FE) : UE.ID = FE.ID`. Sulla base di queste Join Conditions, record multipli con lo stesso identificatore (i.e., companies a,c,d,e) sono unificate in un singolo record dal tramite l'operatore di Merge *Full Outer Join* proposto in [28] e adattato nel framework di MOMIS in [7]. Intuitivamente, ciò corrisponde alle seguenti due operazioni: (1) Calcolo del *Full Outer Join* e (2) Applicazione delle *Funzioni di Risoluzione*.

Resolution Functions

In una classe globale G, notiamo come potrebbero sorgere conflitti per gli attributi globali mappati su più di una classe locale; la *Data Reconciliation* può essere effettuata dalle *Funzioni di Risoluzione* come è stato proposto in [29, 11].

Un Global Attribute *mappato* in una sola classe locale, ovvero tale per cui c'è un solo elemento $MT[GA][L]$ non vuoto, viene chiamato *Omogeneo*. Per ogni Global Attribute GA tale che vi sia più di un elemento $MT[GA][L]$ non vuoto, possiamo definire una funzione di risoluzione per ottenere il valore di GA. MOMIS offre alcune funzioni di risoluzione standard:

- *Funzioni di Aggregazione* per attributi numerici: SUM, AVG, MIN, MAX, ecc;
- *Funzioni di Precedenza*: alcuni conflitti possono essere risolti preferendo un dato proveniente da una determinata classe locale;
- *All values*: tutti i valori sono mantenuti e la risoluzione dei conflitti è demandata all'utente o all'applicazione stessa (cioè, verrà intrapresa l'azione di default di risoluzione del conflitto).

Le definizioni formali non sono riportare, ma intuitivamente, l'operatore per il *Full Outer Join Merge* può essere riscritto in standard SQL, ad eccezione di (alcune) Resolution Functions, come è mostrato in Figura 2.4 per la classe globale `Enterprise`:

- `A=AVG(UE.A, FE.A)`: `AVG` è una funzione (non standard SQL) che calcola il valore *medio*;

- `B = COALESCE(UE.B,FE.B)`: `COALESCE` è una funzione standard SQL che restituisce il primo valore non nullo, in questo caso si vede che `UE.B` ha *precedenza* rispetto al valore `FE.B`.

Mapping Table

	UE	FE
ID	ID	ID
A	A	A
B	B	B
C	C	
D		D

Mapping Query

```
SELECT ID ,
     A = AVG(UE.A, FE.A),
     B = COALESCE(UE.B,FE.B),
     C = UE.C,
     D = FE.D
FROM UE FULL OUTER JOIN FE
          USING (ID)
```

Instance

ID	A	B	C	D
a	19	MZ	YES	123
b	22	MK	NULL	234
c	18	MR	NO	345
d	22	MK	YES	567
e	20	MZ	YES	567
f	16	NULL	YES	NULL

Figura 2.4: Mapping Table, Mapping Query e l' Istanza della global class `Enterprise`

Completiamo l'esempio fornendo l'istanza di due altre classi globali:

- `Business` ha una sola classe locale `IB` e assumiamo che non c'è nessuna trasformazione di dati. Quindi, l'istanza di `Business` coincide con `IB`.
- `Category` (la sua mapping table è banale e per questo è stata omessa) è definito come il Full Outer Join Merge di `UC` e `IC`, utilizzando l'attributo `B` come identificatore e la funzione di risoluzione `COALESCE` per l'attributo comune `F`. Infine, l'istanza di `Category` sono le tre tuple $(MZ, s), (MR, s)$ and (MW, s).

2.4 Elaborazione Query Globali

In questa sezione viene illustrato il funzionamento del Query Manager attualmente utilizzato nel sistema MOMIS; ci riferiremo a tale sistema con il termine Query Manager Tradizionale (QMT) per evidenziare il fatto che gestisce solo dati tradizionali. Siccome nei capitoli successivi vogliamo descrivere le modifiche effettuate al QMT allo scopo di effettuare anche l'elaborazione delle query multimediali, usiamo come riferimento l'esempio di Mapping Table mostrato in figura 2.5, relativa alla classe globale MarbleSlab ottenuta come integrazione di tre classi locali e che contiene anche attributi multimediali (ovviamente in questa sezione tali attributi verranno trascurati).

Figura 2.5: Mapping Table della classe globale MarbleSlab

MarbleSlab	Marble	Slab	MySlab
Id (join)	id	s-id	slb-id
Code	code	code	code
Block-id	block-id	block-id	blockid
Material	material	class	type
Thickness	thickness	–	thickness
Price	euro	price	cost
Area	–	area_in_meters	–
Sold	sold	out	sold
Width	width_cm	width_in_centimeters	width
Height	height_cm	height_in_centimeters	height
Photo	photo	img	image
Description	description	notes	annotations

Una query globale senza multimedialità é del tipo:

```
SELECT Id, Material, Price
FROM MarbleSlab
WHERE (Thickness = 3) AND (Area = 2) AND (Price < 100)
```

L'Architettura Funzionale del QMT è mostrata in Figura 2.6.

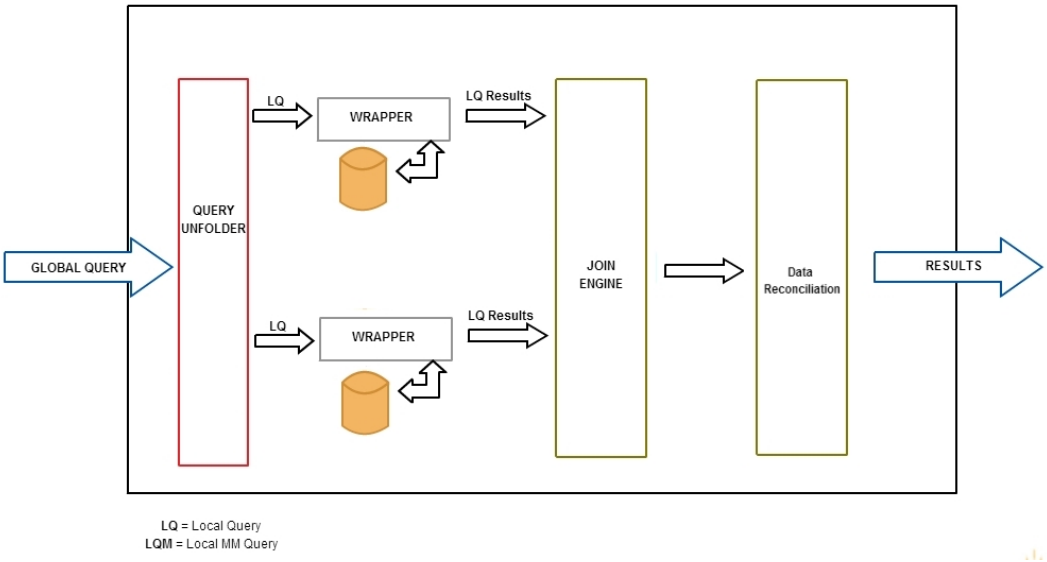


Figura 2.6: Architettura Funzionale del **Query Manager Tradizionale (QMT)**

L'elaborazione di una query globale avviene attraverso i seguenti passi.

1. **Query Unfolding** (modulo **Query Unfolder**):
   Per risolvere la query globale, essa deve essere riscritta in un set di query locali (equivalenti ad essa). Queste query sono risolte tenendo in considerazione il mapping fra la classe globale e quelle locali, quindi nell'approccio GAV il "query unfolding" significa dire che si sta effettuando l' "Espansione della global query"

in accordo con quanto espresso nel mapping contenente le relazioni fra la classe globale e le sue classi locali. Più precisamente, nel caso specifico del sistema MOMIS, tale mapping è espresso dalla Mapping Query Q_G su $\mathcal{L}(G)$ introdotta nella sezione precedente.

2. **Fusione delle query locali** (modulo **Join Engine**):
   Dopo che ogni query locale è stata calcolata sulla risorsa locale corretta, i risultati ottenuti devono essere fusi a livello globale. Se la query globale non contiene predicati legati alla Multimedialità la fusione avviene utilizzando la Mapping Query Q_G e, quindi in definitiva attraverso un Full Outer Join (eseguito dal motore SQL integrato in MOMIS). Invece, nel caso di query globali che contengono predicati di similarità, discuteremo nel capitolo 3 le opportune soluzioni proposte.

3. **Applicazione della funzione di risoluzione e dei predicati residui** (modulo **Data Reconciliation**):
   Ogni attributo non omogeneo della global query è calcolato dalla relativa funzione di risoluzione, infine i predicati residui (non ancora applicati) saranno applicati su questo attributo.

Nel seguito ogni step precedentemente indicato viene discuto nel dettaglio.

2.4.1 Query Unfolding

Il processo di Query unfolding su una query globale restituisce varie local query, in particolare una su ogni classe locale in gioco, ed anche i predicati residui da applicare successivamente.

Local Queries

La query locale su una classe locale è ottenuta dalla riscrittura di ogni singolo predicato atomico sugli attributi omogenei. I Predicati Atomici su attributi omogenei sono riscritti in base alla funzione di mapping (definita nella Mapping Table). In questo esempio il predicato $Area = 2$ è tradotto in $area\_in\_meters = 2$ per la classe locale *Slab*, ed invece non è tradotta in nessun predicato per la classe locale $Marble$ poiché non contiene questo attributo.

In questo esempio le query generate sono:

```
LQ_Marble = SELECT id, material, euro,thickness
            FROM Marble

LQ_Slab =  SELECT s-id, class, price
           FROM Slab
```

```
                WHERE area_in_meters = 2

LQ_MySlab  = SELECT slb-id, thickness,cost
               FROM MySlab
```

Predicati Residui

I predicati "atomici" su attributi non omogenei non possono essere tradotti in predicati locali e vengono detti residui, inoltre essi dovranno essere risolti a livello globale. In questo esempio *Price* è un attributo non omogeneo calcolato come ***AVG*** (funzione di risoluzione). Quindi il predicato $Price < 100$ non può essere "inoltrato" alla risorsa locale, in quanto la funzione di media (AVG) deve essere calcolata a livello globale: questa condizione potrebbe risultare globalmente vera ma localmente falsa. In questo esempio anche l'attributo *Thickness* risulta essere non omogeneo.

$$(Thickness = 3)AND(Price < 100)$$

Gli attributi locali in corrispondenza di attributi globali presenti nei predicati residui sono aggiunti alla lista di selezione delle query locali, inoltre i predicati di similarità non hanno effetto sulla condizione costituita da predicati residui.

2.4.2 Fusione dei risultati delle query locali

Calcolo del Full outer join e semplificazione

Come detto in precedenza, in questo caso in cui non ci sono predicati legati alla multimedialità, la fusione avverrà utilizzando la Mapping Query Q_G e, quindi in definitiva attraverso un Full Outer Join, eseguito dal motore SQL integrato in MOMIS [8].

Ricordandoci quanto abbiamo detto del dover inoltrare il predicato alla risorsa locale, questa azione può portare ad avere più risultati di quelli attesi, in particolare alcuni con NO-MAPPING. Ricordiamo come i record locali con lo stesso valore per gli attributi di join locale sono lo stesso oggetto, per questo motivo essi vengono fusi in un unico record globale. Per semplicità, *ID* denota l'attributo di join globale ed anche quello locale per ogni classe locale.

L'input per eseguire la fusione sono N query locali LQ con attributi:

$$LQ.ID, LQ.A_1, LQ.A_{\mathrm{m}}$$

con $m \geq 0$ ed il risultato in output è una relazione:

$$R\_F(ID, LQ_1.A_1, ..., LQ_1.A_{\mathrm{m_1}}, ..., LQ_{\mathrm{n}}.A_1, ...LQ_{\mathrm{n}}.A_{\mathrm{m_n}})$$

Nell'esempio:

```
R_F(ID, LQ_Marble.material, LQ_Marble.euro,
    LQ_Marble.thickness,LQ_Slab.class,
    LQ_Slab.price,LQ_MySlab.thickness,LQ_MySlab.cost)
```

2.4.3 Applicazione della funzione di risoluzione e dei predicati residui

Per ogni attributo GA non omogeneo della global query, la funzione di risoluzione ad esso relativa è applicata ad ogni record di R_F. Questo restituisce la relazione R_S i cui attributi sono specificati nella lista SELECT della global query, nel nostro esempio: R_S(ID, material, price).

I predicati residui sono applicati ad ogni record di R_S, e ciò ci permette di ottenere la risposta alla query globale.

2.4.4 Dettagli sul No-Mapping elements

Come avevamo precedentemente introdotto, il Full Outer Join in determinati contesti potrebbe restituirci più risultati di quelli attesi. Per spiegare questo problema conviene partire da un semplice esempio:

Si hanno due classi locali L1 ed L2 ed una classe globale G, la Mapping table è la seguente:

Mapping Table

G	**L1**	**L2**
ID	ID	ID
A (AVG)	A	A
B	B	-

È presente una funzione di risoluzione su A che è il calcolo della MEDIA (AVG), quindi essa dovrà esser calcolata solo sulla global query.

La query globale su G è la seguente:

```
SELECT   *
FROM     G
WHERE    A = 10
AND      B = 10
```

per quanto detto in precedenza, la fase di unfolding produce le seguenti query locali:

```
Local Query LQ1:
       SELECT   *
```

```
        FROM     L1
        WHERE    B = 10
```

ed in:

```
Local Query LQ2:
        SELECT   *
        FROM     L2
```

Possiamo notare come non viene impostato nessun predicato su A, in quanto c'è una funzione da applicare a livello globale.

Applicazione della funzione di risoluzione

Si procede effettuando il full outer join delle sotto-query locali X e Y e si applicano le eventuali funzioni di risoluzione.

Si ha quindi che la query globale usata per creare la vista z è:

```
VIEW Z:
        SELECT COALESCE(LQ1.ID=LQ2.ID) as ID , AVG(L1.A,L2.A) as A, B
        FROM     LQ1 FULL OUTER JOIN LQ1  ON (LQ1.ID=LQ2.ID)
```

che si può scrivere in modo equivalente tramite USING:

```
VIEW Z:
        SELECT  ID , AVG(L1.A,L2.A) as A, B
        FROM     LQ1 FULL OUTER JOIN LQ1  USING(ID)
```

Applicazione dei predicati residui

Infine vengono applicati i predicati residui, in quanto essi devono esser validi a livello globale.

```
QUERY Z:
        SELECT *
        FROM Z
        WHERE A = 10
```

Possiamo però notare che in questo modo, a causa dell'uso del Full Outer Join, saranno inclusi nel risultato anche quei record provenienti solo dalla L2 che non soddisfano il predicato B=10. Siccome a livello implementativo l'applicazione delle funzioni di risoluzione non avviene tramite SQL ma in modo equivalente agendo direttamente sul Result Set, è possibile individuare tali record ed evidenziarli, riportando nel risultato finale mostrato all'utente l'indicazione NO-MAPPING (invece del

valore NULL) proprio per evidenziare che sono record presenti solo in una sorgente senza "mapping" nell'altra.

Come è stato analizzato nell'articolo [4], un modo per risolvere questo problema dovrebbe essere quello di utilizzare il Left Outer Join invece che quello Full Outer.

```
VIEW Z':
       SELECT ID , AVG(L1.A,L2.A), B
       FROM     LQ1 LEFT OUTER JOIN LQ1  USING(ID)
```

quindi in questo modo si ottiene il risultato corretto:

```
QUERY Z':
       SELECT *
       FROM Z'
       WHERE A = 10
```

Questa parte non è stata ancora implementata nel sistema MOMIS, dove per risolvere il problema si "riapplica" a livello globale una seconda volta il predicato già applicato sulla sorgente L1, continuando ad utilizzare il Full Outer Join.

```
Query Z'':
       SELECT *
       FROM     Z
       WHERE    A = 10
       AND      B = 10
```

Capitolo 3

MOMIS e Multimedialità

Come abbiamo introdotto nel capitolo iniziale, l'obiettivo di questa tesi è rendere MOMIS uno strumento capace di gestire sorgenti eterogenee, in particolare si vuole renderlo in grado di lavorare su dati multimediali. In questo capitolo mostriamo quali sono i framework per la multimedialità utilizzati.

Per gestire la multimedialità dal punto di vista degli Schemi Locali e dello Schema Globale, si suppone di avere attributi di un opportuno tipo; a tale scopo ODL_{I^3} è stato esteso con tipi di dato per la gestione di dati multimediali; in particolare per le immagini è stato introdotto il tipo `Image`. Pertanto supponiamo di avere nella Mapping Table di Figura 2.5 un attributo di tipo `Image` in ciascuna delle tre sorgenti locali che corrisponde all'attributo `Photo` di tipo `Image` nella classe globale. Nel seguito uno schema globale derivante dall'integrazione di sorgenti tradizionali e multimediali verrà chiamato anche Data and Multimedia Global Schema (DMGS).

In modo intuitivo, una query globale con multimedialità sulla `MarbleSlab` è del tipo

```
SELECT Id, Material, Price,  Photo
FROM MarbleSlab
WHERE (Thickness = 3) AND (Area = 2) AND (Price < 100)
<MULTIMEDIA_PREDICATE>
```

ovvero seleziona, e quindi riporta in output, anche l'attributo multimediale `Photo` (oltre ad altri attributi tradizionali) e contiene un `<MULTIMEDIA_PREDICATE>` su tale attributo `Photo`, ad esempio per esprimere la similarità rispetto ad una immagine data come riferimento.

La scelta del sistema MOMIS è quella di demandare il `<MULTIMEDIA_PREDICATE>` completamente alla sorgente locale, ovvero il sistema MOMIS (a livello di mediatore) non deve elaborare predicati multimediali ma durante l'elaborazione delle interro-

gazioni deve solo effettuare la Fusione dei risultati delle query locali (vedi sezione 2.4.2).

Verranno presentati due metodi:

- Il primo approccio è quello usato in **MOMIS-MILOS** (sezione 3.1) nel quale le risposte locali sono delle "ranked list" e come metodo di fusione si utilizza MEDRANK e si presuppone di poter accedere alla lista elemento per elemento;

- Il secondo è un nuovo metodo proposto (sezione 3.2.1), indicato come **MOMIS-Windsurf**, nel quale le risposte locali sono delle "scored list" e come metodo di fusione si utilizza sempre il Full Outer Join, quindi non è necessario accedere alla lista elemento per elemento; in altre parole, come verrà discusso ampiamente nel seguito, il Query Manager resta concettualmente e sostanzialmente invariato.

3.1 Elaborazione Query con dati multimediali: Sistema MOMIS-MILOS

Il primo tentativo per aggiungere in MOMIS la possibilità lavorare con elementi multimediali è stato sfruttando il framework di nome MILOS.

3.1.1 MILOS

MILOS (Multimedia dIgital Library for On-line Search) [1] è un sistema di gestione dei dati multimediali utile per la progettazione e l'implementazione efficace di applicazioni di biblioteche digitali. MILOS supporta la memorizzazione ed il recupero content-based di documenti multimediali le cui descrizioni sono fornite utilizzando modelli di metadati rappresentati usando XML. MILOS adotta un database XML nativo, che supporta le query standard del linguaggio XML come XQuery e XPath, ed offre una ricerca avanzata e l'indicizzazione sui documenti XML. Gli standard di nuova generazione dei metadata, come per esempio MPEG-7[1], includono nella loro descrizione anche alcune caratteristiche estratte automaticamente da "documenti multimediali", come per esempio gli istogrammi colore, le texture, spigoli e forme. Quindi oltre a fornire altre prestazioni di ricerca e recupero di documenti XML strutturati grazie alla struttura a indice, MILOS fornisce anche una ricerca testuale, la classificazione automatica e funzioni di ricerca per similarità anche per elementi multimediali.

[1] `http://it.wikipedia.org/wiki/MPEG-7`

3.1.2 Data and Multimedia Source

Una sorgente locale gestita tramite il sistema MILOS viene indicata come *DMS* (Data and Multimedia Source); una classe locale M di *DMS* ha m attributi multimediali denotati da S_1,...,S_m e h attributi standard A_1,...,A_h; una *DMS* supporta **query locali multimediali** che in sintassi pseudo-SQL viene espressa come:

```
SELECT S1, . . . A1, . . .
FROM M
     WHERE A1 op1 val1
     AND A2 op2 val2
     ...
     ORDER BY S1(Q1), S2(Q2),  ....
     LIMIT k
```

dove la clausola $WHERE$ consiste in una congiunzione di predicati atomici su attributi standard con OPi operatori relazionali, mentre la clausola ORDER BY esprime una congiunzione di predicati di similarità nella forma Si(Qi) , dove Si è un attributo multimediale e Qi è una costante multimediale. Ad esempio, per la classe `LQ_Marble` con l'attributo multimediale `photo` di tipo Image, data l'immagine vera e propria di confronto (in questo caso "slab12345.jpg"), il predicato di similarità sarà `photo("slab12345.jpg")`. In tal caso i risultati si intendono ordinati rispetto alla similarità con tale immagine. Il predicato "LIMIT K" sta ad indicare un classico predicato di selezione Top-K elementi. In definitiva, una DMS è in grado di classificare i risultati di una query multimediale (query aventi predicati di similarità). Si consideri il caso generale dove non si conosce il modo in cui una *DMS* effettua questa classifica (rank) dei risultati.

3.1.3 Ranked Query Fusion

Una query globale con Multimedialità e quindi con calcolo di similarità sfruttando il framework MILOS potrà essere del tipo:

```
SELECT Id, Material, Price
FROM MarbleSlab
WHERE (Thickness = 3) AND (Area = 2) AND (Price < 100)
ORDER BY Photo("slab12345.jpg")
LIMIT 100
```

In questa sezione analizzeremo solo i gli elementi differenti nell'esecuzione rispetto al processo di elaborazione di Query Globali descritto in sezione 2.4 riferito solo a dati tradizionali.

Query Unfolding

Molto simile al caso precedente.

Local Queries La query locale su una classe locale è ottenuta dalla riscrittura di ogni predicato atomico sugli attributi omogenei e dei predicati di similarità della global query nei corrispettivi predicati supportati dalla classe locale. Per i predicati atomici su attributi omogenei, la riscrittura rimane ovviamente invariata, mentre la clausola ORDER BY per un attributo multimediale globale è tradotto in un ORDER BY sulla classe locale che possiede l'attributo multimediale e che supporta il predicato. Nell'esempio, le query generate sono:

```
LQ_Marble = SELECT id, material, euro,thickness
            FROM Marble
            ORDER BY photo(‘‘slab12345.jpg’’)

LQ_Slab =  SELECT s-id, class, price
           FROM Slab
           WHERE area_in_meters = 2
           ORDER BY img(‘‘slab12345.jpg’’)

LQ_MySlab  = SELECT slb-id, thickness,cost
             FROM MySlab
             ORDER BY image(‘‘slab12345.jpg’’)
```

Predicati Residui I predicati di similarità non influenzano la condizione dei predicati residui.

Fusione dei risultati delle query locali In questo step della soluzione delle query ci sono molti elementi che differenziano l'esecuzione del solo MOMIS rispetto a MOMIS integrato con MILOS.

Per ogni query locale LQ_i, la sua risposta è una ranked list $LQ_i = \langle r_{1,i}, ..., r_{n,i} \rangle$ formata da n_i elementi ordinati in ordine decrescente della rilevanza da 1 a n_i (notiamo come utilizziamo lo stesso simbolo della local query per la sua risposta). La fusione produce una lista n_G di record legati alla global class $R_G = \langle r_{1,g}, ..., r_{k,G} \rangle$ ordinati in modo decrescente di rilevanza da 1 a n_G.

Per semplicità consideriamo il caso dove tutte le query locali hanno un predicato di similarità, successivamente consideriamo un caso più generico dove la similarità è presente solo in alcune query locali.

Esistono differenti algoritmi per il calcolo della distanza e quindi per la valutazione dell'ordine globale delle risposte, inoltre, come possiamo immaginare, è difficile sceglierne uno universalmente valido.

Per la realizzazione del progetto MOMIS-MILOS si è deciso di utilizzare **MEDRANK** [5], un metodo ottimo per il calcolo dell'aggregazione del rank basato su distanza. Il ranking globale è quindi quello che ha la minima distanza calcolata dalle differenti sorgenti (vedi Appendice A).

MEDRANK permette di calcolare la distanza fra liste di elementi, e restituisce un singolo elemento per volta e permette così anche il calcolo di query di tipo Top-K. Esso lavora a livello di oggetti. In MOMIS-MILOS i record il cui valore dell'attributo (di join) ID risulta identico a quello di un altro record, saranno considerati uno stesso oggetto. MEDRANK scansiona le ranked list che sono restituite dalle query locali: il primo valore di ID che compare in più di una lista ranked è considerato dall'algoritmo un valore "TOP ID value", Il successivo valore di ID che è presente in più di una lista sarà considerato un altro valore Top, e così via fino a che esso non riesce a calcolare un numero di record k necessari alla Top-K query.

Applicazione della funzione di risoluzione e dei predicati residui Per il calcolo della funzione di risoluzione dobbiamo introdurre la funzione MOST_SIMILAR, che permette di risolvere i conflitti fra gli attributi multimediali scegliendo l'oggetto multimediale che è più simile a quello espresso nella query (se presente). Questa funzione si basa sul metodo di fusione che basato su MEDRANK.

Un altro possibile approccio per l'implementazione di questa funzione di risoluzione MOST_SIMILAR è quella che si basa sulla soluzione "Mediator Based": un mediatore implementa una funzione di similarità (uno per ogni attributo multimediale) che quindi è in grado di decidere quale attributo multimediale della query è più "simile" a quello della query.

Invece per quanto riguarda i predicati residui non è presente alcun comportamento diverso da quello analizzato nel caso senza predicato multimediale.

3.2 Elaborazione Query con dati multimediali: Sistema MOMIS-Windsurf

L'aspetto fondamentale del metodo esposto nella sezione precedente è la Ranked query fusion (3.1.3): la fusione di query locali tradizionali e multimediali richiede di cambiare completamente il metodo di fusione dei risultati delle query locali (sezione 2.4.2) valido per dati tradizionali basato sul Calcolo del Full outer join ed implementato nel Query Manager Tradizionale basato su un SQL-engine.

L'obiettivo del progetto in esame era quello di individuare in quali casi e sotto quali ipotesi era possibile effettuare una fusione di query locali tradizionali e multime-

diali basata ancora sul calcolo del Full Outer Join e che quindi fosse implementabile nel Query Manager Tradizionale con modifiche minime.

Innanzitutto, rispetto al framework generale precedentemente discusso, è stata introdotta l'ipotesi restrittiva di una sola sorgente multimediale (contenente immagini), che, come mostrato nel seguito, è fondamentale per effettuare la fusione utilizzando ancora il Full Outer Join.

Inoltre siccome il sistema MILOS non era più tenuto aggiornato e le sue caratteristiche non consentivano una semplice integrazione nel Query Manager Tradizionale, si è deciso di cambiare il framework per la gestione della multimedialità e dopo un periodo di analisi delle varie possibilità presenti sul mercato (vedere sezione 4.1), si è scelto di utilizzare il framework Windsurf.

In definitiva, il lavoro svolto nella tesi si è concentrato sulla progettazione ed implementazione del nuovo framework MOMIS-Windsurf, per l'interrogazione di dati tradizionali e multimediali. In sintesi, l'idea è sempre quella iniziale di demandare tutti i criteri di ricerca per similarità alla sorgente dati multimediali gestita da Windsurf ma il nuovo aspetto caratteristico è quello di effettuare la fusione tra dati tradizionali e multimediali utilizzando, ancora una volta, il Query Manager Tradizionale. A tale scopo è stato considerato inizialmente lo scenario semplificato con:

- N Sorgenti Tradizionali
- 1 Sorgente Multimediale (gestita tramite Windsurf)

Più precisamente, una classe globale G potrà avere al massimo una classe locale multimediale corrispondente e classi locali tradizionali; quindi nel caso più semplice si ha solo una classe locale tradizionale ed una multimediale. Una Query Globale su una tale classe globale G conterrà, oltre ai predicati su attributi tradizionali, anche i predicati multimediali supportati dal sistema Windsurf.

Nel capitolo conclusivo verranno discusse le problematiche introdotte nel gestire N Sorgenti Multimediali e le possibili soluzioni per questo caso generale.

3.2.1 Windsurf

Windsurf (**W**avelet-based **IND**exing of Image**S** **U**sing **R**egion **F**ragmentation) è una libreria sviluppata dall'Università di Bologna, che permette di avere uno strumento "redy-to-use" utile per la comparazione di performance legate a efficacia ed efficienza degli algoritmi di query presenti in altri programmi.

Windsurf è un framework per la risoluzione efficiente di query "content-based" su dati multimediali. Questo framework pone particolare attenzione al paradigma Region-Based Image Retrieval (RBIR), ma ciò non toglie la possibilità di implementare e quindi utilizzare altri paradigmi definiti dall'utente. In questo modo Windsurf

offre uno strumento per comparare in modo equo differenti implementazioni, dando la possibilità di analizzarne la differenza di efficienza e efficacia fra le varie soluzioni. Nel nostro caso questa libreria verrà utilizzata come strumento da integrare in MOMIS per la gestione delle query multimediali.

Windsurf offre varie possibilità di estensibilità e personalizzazione del framework, fra le quali ricordiamo:

- Possibilità di scegliere differenti rappresentazioni a basso livello di immagini, differenti metodi per la loro segmentazione ed estrazione.
- Differenti costrutti per la comparazione di regioni di immagini (comparazione 1-1 , N-M , ecc...).
- Differenti metodi di esecuzione di query, basate su Rank oppure su preferenze (es. k-nearest neighbour, range, query qualitative di tipo skyline).

I dati multimediali che il sistema può trattare possono essere: Foto, Video, Musiche, GIF animate, Documenti medici contenenti vari sotto-elementi multimediali

Inoltre le possibili query multimediali che Windsurf mette a disposizione sono di tre tipologie:

- Query di Range
- Top-K query
- Next Nearest Neighbour

3.2.2 Windsurf Local Source

Come abbiamo detto, una sorgente locale multimediale gestita tramite Windsurf (Windsurf Local Source - WLS) ha un contenuto informativo solo di tipo multimediale, ed in particolare è solo l'immagine stessa, ovvero la sorgente multimediale non gestisce altre informazioni, né di carattere tradizionale né di carattere multimediale. Tale ipotesi non è restrittiva in quanto in presenza di una sorgente con immagini e attributi tradizionali, essa può essere sempre rappresentata come integrazione di due sorgenti.

Di conseguenza, una Windsurf Local Source può essere rappresentata da uno schema locale "costante", ovvero con la stessa struttura per ogni sorgente multimediale, costituito dai seguenti attributi:

- **ID** : utilizzato solo internamente alla sorgente WLS Windsurf.
- **NAME** : Identificatore con il quale l'oggetto multimediale (l'immagine nel nostro caso) può essere riferito anche all'esterno della sorgente WLS Windsurf;

In un contesto Object Fusion esso ha il ruolo di Object Identifier (in termini più semplici, questo elemento sarà quello che verrà utilizzato da MOMIS per effettuare le condizioni di join); il termine "NAME" è stato scelto in quanto si tratta generalmente di una stringa, il "NAME" del file immagine.

- **IMAGE** : l'immagine vera e propria (convertita in stringa).
- **DISTANCE** : un attributo derivato in fase di interrogazioni multimediale (per interrogazioni non MM assume un valore non significativo).

In definitiva, una WLS ha come local schema la seguente relazione

```
WLS(ID,NAME,IMAGE,DISTANCE)
```

Abbiamo detto che le possibili query multimediali che WINDSURF mette a disposizione sono di tre tipologie: Query di Range, Top-K Query e Next Nearest Neighbour. Le prime due tipologie sono *set-oriented*, nel senso che restituiscono un insieme di tuple (le immagini con il relativo ID e SCORE), mentre l'ultima tipologia è *tuple-oriented* nel senso che restituisce una singola tupla ad ogni chiamata. Siccome il QMT di MOMIS è *set-oriented* in quanto basato sul Full Outer Join delle query locali, si considerano solo le prime due tipologie di query multimediali; di conseguenza una query locale multimediale può essere espressa in una sintassi Pseudo-SQL:

```
SELECT        *
FROM          WLS
(DISTANCE BETWEEN X AND Y | TOP K).IMAGE(<imageSTRING>)
```

dove il predicato multimediale è costituito da due parti: nella prima viene indicata la tipologia del predicato multimediale, ovvero una range query espressa tramite `DISTANCE BETWEEN X AND Y` oppure una Top-K query espressa tramite `TOP K`, mentre `IMAGE(<imageSTRING>)` rappresenta l'elemento multimediale, più precisamente una immagine convertita in stringa (si vedrà successivamente il perché) con cui il predicato deve essere valutato.

Quindi la risorsa multimediale Windsurf è in grado di restituire i risultati che soddisfano il predicato espresso.

3.2.3 Fusione tra dati tradizionali e multimediali tramite FOJ

Nello scenario considerato, ovvero N Sorgenti Tradizionali e una sola Sorgente Multimediale (gestita tramite Windsurf, cioè WLS), una classe globale G potrà avere al massimo una classe locale multimediale corrispondente rappresentata dalla relazione `WLS(ID,NAME,IMAGE,DISTANCE)`. Una Query Globale su una tale classe globale G sarà del tipo:

```
SELECT      *
FROM        G
WHERE       <TRADITIONAL PREDICATES>
 (DISTANCE BETWEEN X AND Y | TOP K).IMAGE(<imageSTRING>)
```

Come già detto in precedenza, nella presente tesi verrà discusso come elaborare questa interrogazione con predicati di similarità utilizzando ancora il Query Manager Tradizionale e demandando la risoluzione del criterio di ricerca (per similarità) alla WLS, sorgente dati multimediali gestita da Windsurf. Pertanto, dal punto di vista dell'architettura funzionale del Query Manager, la soluzione è mostrata in Figura 3.1: rispetto al Query Manager Tradizionale di Figura 2.6 è stato aggiunto il *Multimedia Wrapper*, per gestire la WLS, ed è stato esteso il Query Unfolder per "estrarre" la parte multimediale della global query.

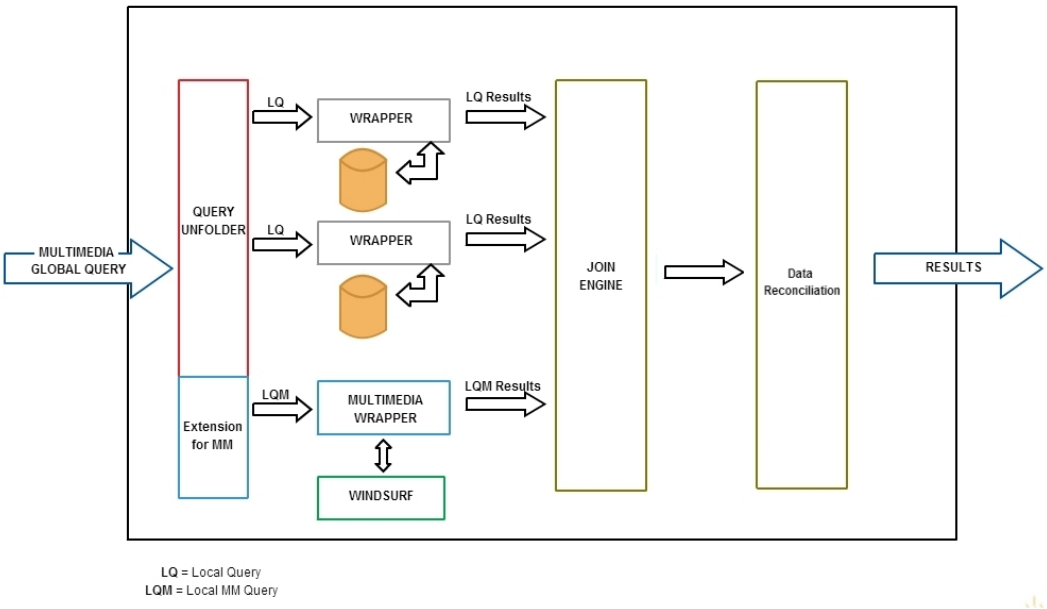


Figura 3.1: Architettura Funzionale del QMT esteso ai dati multimediali

Sintetizziamo l'elaborazione di una query globale multimediale, evidenziando per ogni passo le differenze rispetto al caso di soli dati tradizionali.

Query unfolding Concettualmente questo step rimane inalterato rispetto a quello effettuato in MOMIS senza la presenza di predicati multimediali. Vengono sottoposti ad unfolding classico i predicati tradizionali, mentre il predicato multimediale viene "estratto ed inoltrato" soltanto alla sorgente multimediale Windsurf interessata.

Fusione dei risultati delle query locali e semplificazione Questo è il punto fondamentale del metodo proposto: la fusione di query locali tradizionali e

multimediali è basata ancora sul calcolo del Full Outer Join (in sostanza viene utilizzato il modulo Join Engine praticamente invariato rispetto al caso di soli dati tradizionali). Come vedremo questa soluzione consentirà di gestire le interrogazioni di Range in maniera completa ed esatta, mentre per le interrogazioni Top-K in alcuni casi la soluzione completa ed esatta non è possibile e verrà gestita una soluzione approssimata. Le problematiche sono enunciate in questa sezione ed analizzate in dettaglio nella sezione 3.3.2.

Applicazione della funzione di risoluzione e dei predicati residui In questo step non sono presenti comportamenti differenti rispetto a quelli analizzati nei casi precedenti.

Consideriamo il problema del NO-MAPPING discusso in sezione 2.4.4 nel caso di query multimediali.

Data una query che ha sia una parte tradizionale che una parte multimediale, una operazione di join dovrebbe essere utilizzata tra la query locale tradizionale e quella multimediale. Attualmente ciò non è ancora implementato, quindi si usa la soluzione introdotta in sezione 2.4.4; ma mentre per la parte tradizionale il predicato viene aggiunto automaticamente dal MOMIS, per la parte multimediale deve essere inserito manualmente dall'utente mentre formula la query. Nel seguito vogliamo illustrare quali sono le problematiche nell'elaborare le interrogazioni multimediali usando il QMT ovvero query unfolding e Fusione dei risultati delle query locali basato sul full outer join come spiegato in sezione 2.4.2. Intuitivamente, il problema dell'incompletezza dei risultati si presenta per le Top-K, quindi se la MM query possiede solo il predicato Top-K , il risultato della fusione risulta essere esatto; se la MM query possiede oltre al predicato Top-K altri predicati tradizionali, il risultato potrà essere appunto "incompleto", ovvero la query restituirà K' $\leq$ K risultati. Per ovviare a tale incompletezza posso specificare Top-K" , con K $''$ $\gg$ K , in particolare imponendo K" pari alla cardinalità della sorgente ho sicuramente tutti i risultati attesi ma chiaramente questa soluzione non è applicabile.

Per chiarezza espositiva consideriamo dapprima un caso simbolico costituito da due sorgenti:

```
WLS(ID,NAME,IMAGE,DISTANCE)
```

```
TRAD(ID,ATTR1,ATTR2)
```

Dalla quali, impostando come attributo di join il campo ID si ottiene la Classe Globale:

Tabella 3.1: Classe Globale

G	**WLS**	**TRAD**
ID	ID	ID
NAME	NAME	
ATTR1		ATTR1
ATTR2		ATTR2
IMAGE	IMAGE	
DISTANCE	DISTANCE	

Quindi, sappiamo che in una query globale possono essere presenti sia predicati tradizionali che predicati multimediali, in particolare essi possono presentarsi in varie combinazioni.

Caso 1 È il caso in cui il predicato tradizionale è presente solo sulla risorsa tradizionale (TRAD) e non è presente alcun predicato multimediale. Il risultato di questa query è **corretto**.

Caso 2 Il predicato tradizionale è sia su TRAD che su WLS, ma non è presente nessun predicato multimediale. Anche in questo caso il risultato calcolato risulta essere **corretto**.

Caso 3 Il predicato tradizionale è sia su TRAD che su WLS, ed è anche presente un predicato multimediale di RANGE su WLS. Il risultato restituito è **corretto**.

Caso 4 Il predicato tradizionale è sia su TRAD che su WLS, ed è presente un predicato multimediale di tipo Top-K sulla risorsa Multimediale. il risultato **può essere non completo**. Quindi il sistema MOMIS-Windsurf potrà restituire K'<K risultati richiesti (nel caso in cui effettivamente erano presenti più di k elementi che soddisfavano il predicato).

Entrando nel dettaglio dell'esempio, una query di questo tipo potrebbe essere:

```
QUERY Globale:
       SELECT *
       FROM G
       WHERE ATTR1 = X
                  AND NAME = Y
        (TOP K).IMAGE(<IMG_STR>)
```

Dove essa viene scomposta e sulla risorsa tradizionale viene inoltrata la query:

```
QUERY su TRAD: QT
```

```
SELECT *
FROM TRAD
WHERE ATTR1 = X
```

Invece su quella multimediale viene eseguita:

```
QUERY su WLS: QMM
        SELECT *
        FROM WLS
        WHERE NAME = Y
        (TOP K).IMAGE(<IMG_STR>)
```

Infine si uniscono i risultati tramite:

```
QUERY di Unificazione:
        SELECT *
        FROM QT foj QMM USING (ID)
```

Concludendo, non è detto che questi K risultati estratti da WLS siano presenti anche nella risorsa tradizionale e che oltretutto essi soddisfino anche l'altra clausola tradizionale imposta.

Caso 5 Il predicato tradizionale è solo sulla risorsa TRAD ed è presente anche una parte di query multimediale inoltrata a WLS: il risultato in caso di Query di Range ricade in **Caso 3**, invece se è Top-K si ricade nel **Caso 4**.

È stato quindi mostrato che elaborando interrogazioni Top-K con il Query Manager Tradizionale e demandando quindi la query Top-K alla sorgente dati multimediale (gestita da Windsurf) potrebbe sorgere un problema di incompletezza dei risultati. Come accennato in precedenza, per ovviare a tale incompletezza posso specificare un K" opportunamente maggiore di K. Ovviamente la questione ora è quella di decidere il valore di K". La soluzione banale di imporre K" pari alla cardinalità della sorgente WLS fa si che in questo caso tutti i risultati attesi vengano restituiti, ma chiaramente ciò non è applicabile (costo di esecuzione).

Un'altra soluzione è quella di calcolare K" utilizzando una stima sulla **selettività dei predicati tradizionali**; questo aspetto verrà discusso nella sezione 3.3.2.

Un'ulteriore soluzione che verrà discussa nella sezione 3.3.3 è basata sulla modifica del piano di esecuzione della query globale.

Anticipiamo che la scelta effettuata a livello implementativo nel progetto di tesi è stata quella di non realizzare nessuna delle due soluzioni "teoriche" discusse nella sezione 3.3.2 ma quella di effettuare un prototipo che ammette la presenza di risposte

incomplete. La motivazione di questa scelta è duplice. Da una parte ha consentito di realizzare un primo prototipo completamente funzionante dell'elaborazione delle interrogazioni su dati tradizionali e multimediali nei tempi stabiliti da DataRiver per il progetto in questione. Inoltre, la soluzione si basa sull'utilizzo del QMT che è un componente oramai collaudato del sistema MOMIS e quindi questo ha consentito di avere una soluzione "robusta" anche se in forma prototipale, un altro requisito imposto da DataRiver per il progetto, soprattutto in vista del collaudo del sistema in casi reali, quale quello medico, come discuteremo in sezione 5.

3.3 Esempio e Analisi delle Problematiche

Analizzeremo l'esecuzione di una query locale ed una query globale che coinvolgono una risorsa multimediale Windsurf con lo scopo di analizzarne il funzionamento e mostrare i possibili casi problematici e alcune soluzioni individuate.

3.3.1 Query Locali

Innanzitutto analizzeremo il caso di Query locale per mostrare con chiarezza la questione della possibile approssimazione del risultato della Top-K Query. La risorsa multimediale ha la struttura mostrata in tabella 3.2.

Tabella 3.2: Risorsa Multimediale : WLS

Nome	**(Tipo)**
ID	(INT)
NAME	(String)
IMAGE	(MM)
DISTANCE	(INT,DERIVATO)

Nel caso di Query di Range e Query Top-K non accompagnate da altri predicati, la loro esecuzione restituisce un risultato corretto e quindi non sono presenti problemi.

Anche se ci trovassimo nel caso di Query solo *non multimediali* rivolte alla sorgente WLS, non risulta esserci nessun problema, il risultato calcolato è corretto.

```
SELECT *
FROM WLS
WHERE ID < 5
```

I casi di query con combinazione di *predicati tradizionali* e *predicati multimediali* sono da analizzare:

Caso Range e predicati tradizionali Questo non è un caso problematico, in quanto posso applicare prima uno o l'altro e non ci sono problemi. In realtà noi applicheremo sempre prima il predicato MM ed otteniamo un set di risultati, per poi filtrare ulteriormente, ma il concetto non cambia e il risultato è corretto in tutti e due i casi.

Caso di Top-K e predicati tradizionali I possibili problemi della non correttezza (completezza) dei risultati restituiti sono dovuti a come si risolve e con quale dei due strumenti che l'Università di Bologna ha messo a disposizione in Windsurf:

Il **primo strumento** ci consente di chiedere i primi k risultati relativi alla parte multimediale per poi *solo* su questi applicare i predicati tradizionali. Questo metodo non è corretto in quanto non è detto che mi restituisca k risultati se presenti, anzi siamo sicuri gli elementi restituiti sono K' < K, in quanto fra i primi K presi, alcuni non soddisferanno il predicato tradizionale. Questa modalità di esecuzione di query miste **non** è stata utilizzata in questo progetto.

Il **secondo strumento** e quello da noi utilizzato, sfrutta un cursore messo a disposizione da Windsurf per scorrere i risultati: in questo modo Windsurf restituisce un risultato per volta che soddisfa il solo predicato multimediale, e per ogni risultato restituito il wrapper verifica se esso soddisfa anche il predicato tradizionale. Si procede iterativamente fino a che si hanno K elementi che soddisfano entrambe le clausole imposte sulla query inoltrata alla risorsa multimediale locale (oppure se non finiscono i risultati) . In questo modo siamo sicuri che, se sono presenti K risultati che soddisfano i predicati (entrambi, combinati), effettivamente verranno restituiti K risultati, quindi il risultato calcolato è corretto e completo.

Il discorso cambia radicalmente nel caso delle Query Globali.

3.3.2 Query Globali

Lo scenario di riferimento è quello di base, in cui vi è una sorgente tradizionale, R(id,A,B,...), e una sorgente multimediale (di immagini, per esempio), WLS(img_id,img,score), in cui l'attributo score è calcolato in base alla similarità con un'immagine input di riferimento. La connessione (qui 1-1) tra R e WLS è data dalla table RWLS(id,img_id), in cui per ogni oggetto di R si stabilisce qual è (se esiste) la corrispondente immagine in WLS, e viceversa. La forma generale di query che si considera è pertanto (query Q1):

```
SELECT * FROM
R FULL JOIN RWLS ON (R.id = RWLS.id)
                FULL JOIN WLS ON (RWLS.img_id = WLS.img_ID)
WHERE <pred. locali su R>
```

```
ORDER BY DISCANCE DESC
LIMIT K
```

Il problema con Q1 è che formalmente non è possibile commutare Top-K e (outer) join, ovvero non si possono semplicemente prendere le migliori K immagini e poi farne il join con R e RWLS.

Quindi, nel caso in cui fossero presenti in contemporanea predicati multimediali di tipo Top-K e predicati non multimediali, si otterrebbe un risultato non completo.

I motivi di ciò sono essenzialmente due:

- **a)** Per alcune immagini potrebbe non esserci un match in RWLS, ovvero non esserci nessuna oggetto di R corrispondente;
- **b)** In presenza di predicati locali su R, alcuni match potrebbero essere eliminati.

È poi evidente che il caso più complesso **c)**, è dato dalla contemporanea presenza di **a)** e **b)**.

Caso A - Assenza di match fra risorsa tradizionale e risorsa multimediale

Il caso A è, in un certo senso, il meno problematico e potrebbe essere risolto, in maniera approssimata, come segue:

1. Si stabilisce la frazione **P**, di immagini con match nella risorsa tradizionale essendo:
$$P = \frac{numero\_elementi\_mm}{numero\_elementi\_non\_mm};$$
2. Si reperiscono da Windsurf le migliori $K' = \mathrm{K/P}$ immagini.

È da notare che, anche se l'intuizione porterebbe a dire che il numero di risultati medio in uscita sarà pari a K, ciò è garantito solo sotto specifiche condizioni. A puro titolo di esempio, si consideri la seguente tabella con il match fra risorsa tradizionale e risorsa multimediale.

img_id	Img1		Img2	Img3	Img4	Img5		Img6	
id	O1	O2		O3	O4		O5		O6

Posto K = 2, ed essendo P = 3/6, si pone K' = 4. Affinché si abbiano K = 2 match, in media, è necessario che tutti gli insiemi formati da 4 immagini siano equiprobabili, oppure che, per ogni set di 4 immagini risultato, il fatto di avere o meno dei match si possa considerare un evento indipendente in termini probabilistici.

Questo problema è correntemente in discussione con i ricercatori responsabili del sistema Windsurf ed è stato oggetto di un articolo scientifico sottomesso alla

Conferenza Nazionale delle Basi di Dati **SEBD**[2] (Italian Symposium on Advanced Database Systems).

Questa parte teorica non è inclusa nell'attuale implementazione; come vedremo nelle sezioni successive, nell'implementazione attuale si opera in un contesto di assenza di stime e quindi viene semplicemente data la possibilità all'utente di eseguire la query nuovamente con un valore di k più alto.

3.3.3 Una Proposta Alternativa

Dati i problemi sopra descritti, in particolare per quello relativo al calcolo della selettività dei predicati su RT, si propone un approccio alternativo, che tuttavia richiede una parziale modifica alla logica usata da MOMIS per risolvere le query. Si consideri sempre uno schema globale, con le risorse locali RT e WLS, dove la tabella che prima indicavamo con RWLS ora è integrata all'interno di WLS.

```
WLS(ID,IMAGE,DISTACE)

RT(ID,A)
```

Tabella 3.3: Classe Globale

Nome	**(Tipo)**
coalesce(WLS.ID,RT.ID) as ID	(INT)
RT.A as A	(INT)
WLS.IMAGE as IMAGE	(MM)
WLS.DISTANCE as DISTANCE	(INT,DERIVATO)

Quindi una query globale del tipo:

```
SELECT *
FROM G
WHERE IMAGE(<PRED_MM>)
          AND A > 10
TOP 2
```

Viene scomposta in:
Query su RT:

```
SELECT ID,A
FROM RT
WHERE A > 10
```

[2] http://sebd2015.dia.uniroma3.it/Home.html

Query su WLS:

```
SELECT ID,IMAGE,DISTANCE
FROM WLS
WHERE IMAGE(<PRED_MM>)
TOP 2
```

Ma come abbiamo detto prima, ciò da origine ad un possibile errore del numero di risultati selezionati che potrebbe essere ovviato se ragionassimo in modo differente. Se eseguissimo **prima** la query locale su RT (non multimediale), quindi in questo modo selezionassimo gli ID che soddisfano quella clausola e **dopo** li inoltrassimo a WLS (ricordiamo che la tabella di mapping fra RT e WLS è già inserita in WLS), in modo tale che i K risultati selezionati siano fra quelli presenti negli ID ricevuti (probabilmente saranno presenti anche altri ID non presenti nella risorsa multimediale), che a loro volta soddisfano anche il predicato tradizionale, avremmo il numero corretto di risultati.

Quindi si avrebbe la vista T formato da elementi ottenuti tramite:

```
VIEW T:
       SELECT ID,A
       FROM RT
       WHERE A > 10
```

Quindi, si inoltra una query Top-K a Windsurf del tipo:

```
SELECT ID,DISTANCE,IMAGE
FROM WLS
WHERE IMAGE(<PRED_MM>)
          AND ID IN  (    SELECT ID
                          FROM T )
TOP K
```

Che fornisce in input anche l'insieme T (elementi che già soddisfano le clausole su risorsa tradizionale). Poiché Windsurf supporta query "Next Nearest Neighbour", quindi la query "get-next", è possibile sfruttare questa interfaccia per trovare le K migliori immagini in RWLS che sono all'interno di T.

A livello implementativo viene dapprima calcolato il predicato multimediale e si sfrutta la funzione "*get-next*" dell'algoritmo che restituisce i risultati in ordine di DISTANCE rispetto al predicato multimediale. Ogni volta che ottengo in questo modo un elemento, vedo se esso soddisfa anche la clausola non multimediale tramite la sua presenza nell'insieme T. Questo processo è ripetuto fino a che non ottengo i **K** elementi che erano stati richiesti.

In questo modo, sono sicuro che se sono presenti almeno K risultati (che soddisfano i due predicati, uno tradizionale e l'altro multimediale), essi saranno restituiti in output.

Si sottolinea ancora una volta che tutte queste problematiche e le varie soluzioni proposte sono state analizzate in collaborazione con l'Università di Bologna, e sono contenute in un articolo scientifico sottomesso alla Conferenza Nazionale delle Basi di Dati **SEBD**[3] (Italian Symposium on Advanced Database Systems).

[3] `http://sebd2015.dia.uniroma3.it/Home.html`

Capitolo 4

Il Framework MOMIS-Windsurf

In questo capitolo si discute in dettaglio la progettazione ed implementazione del framework MOMIS-Windsurf per l'elaborazione delle interrogazioni di dati tradizionali e multimediali.

Le motivazioni della scelta del sistema Windsurf per la gestione delle sorgenti locali multimediali è stata discussa nella sezione 4.2.1.

Da un punto di vista implementativo, la principale motivazione di tale scelta risiede nelle alte possibilità di configurazione del sistema Windsurf, nella possibilità di variare il codice sorgente e quindi di modificarlo per adattarlo al meglio per il contesto di lavoro, non dimenticando inoltre le funzionalità di ricerca efficiente in contesti multimediali che esso mette a disposizione.

Il capitolo è strutturato come segue.
In sezione 4.1 viene sintetizzata la fase di analisi dei sistemi per la gestione di dati multimediali e quindi viene ulteriormente motivata la scelta ricaduta sul sistema Windsurf.
Nella sezione 4.2 si riportano le principali caratteristiche tecniche di Windsurf, utili per lo sviluppo del progetto.
Nella sezione 4.3 invece è presente una descrizione dell'architettura funzionale di MOMIS e di come essa viene estesa per includere sorgenti multimediali gestite tramite WindSurf; infine nella sezione successiva (sezione 4.4) si descrive l'implementazione per la creazione di un wrapper per Windsurf.

4.1 Alcuni Sistemi di Gestione dati Multimediali

Il contenuto informativo dei dati multimediali è un elemento di grande interesse per il mondo attuale, nonostante risulti una sfida per il recupero, estrazione, analisi ed utilizzo di queste informazioni. La natura complessa di questi dati multimediali (come video, immagini, pagine web, ecc...) rende difficile ed anche poco efficiente l'utilizzo di soluzioni generiche già presenti sul mercato (come per esempio le soluzio-

ni per ricerca in contesto puramente testuale). In molti casi, per effettuare l'analisi dei dati multimediali si è visto che i modelli classici di information retrieval (IR) non possono essere utilizzati, a meno di rinunciare all'efficienza, all'efficacia o accettando una pesante semplificazione della query originaria. Gli elementi multimediali potrebbero anche avere una struttura gerarchica, dove per ottenere la risposta ad una query si necessita il calcolo degli elementi di interesse per ogni sotto-elemento, per poi comporre la risposta al livello "più esterno". Prima di realizzare ciò, ci sono vari problemi da analizzare e chiarire:

- Come può essere effettuata una comparazione fra un elemento (multimediale nel nostro caso) ed un altro?
- Come si classifica l'importanza di un elemento a livello globale?
- Si può indicizzare il "documento"? Se si in che modo?

Purtroppo la risposta non può esser data in modo univoco, ma dipende dal caso in cui ci si trova.

I principali sistemi analizzati sono stati: Google Image Search , LIRe (Lucene Image Retrieval), Oracle Multimedia, IMARS (IBM Multimedia Analysis and Retrieval System).

Il progetto *Google Image Search*[1][2] è stato abbandonato ufficialmente il 26 maggio del 2011 (ma queste API sono state integrate in Google Search). Il problema principale è dato dal fatto che la ricerca avverrebbe con elementi multimediali di Google, quindi non con un Database definito dall'utente, ed il secondo problema (oltre ad essere quello più grave nel campo medico) è la questione della riservatezza dei dati e quindi la non possibilità di caricarli in rete.

LIRe[3] (Lucene Image Retrieval) è una libreria che offre la possibilità del recupero di immagini e foto tramite ricerca sullo spazio colore e caratteristiche delle texture. LIRe crea un indice Lucene[4] sulle feature delle immagini per un recupero content-based degli elementi (CBIR - Content Based Image Retrieval). Il sistema risulta essere altamente scalabile ed è offerto con licenza GNU GPL. Il problema principale di questa libreria è dovuto al fatto che si basa solamente sul "CBIR" senza possibilità di adattare il sistema per altri paradigmi.

Oracle Multimedia[5] (InterMedia) è una feature che permette al DBMS Oracle di immagazzinare, gestire e recuperare dati multimediali tramite un metodo integrato con le informazioni contenute nella base di dati. Questa estensione è inclusa nei pacchetti "Oracle Database Standard Edition One" e "Standard Edition and Enterprise

[1] `https://developers.google.com/image-search/`
[2] `http://en.wikipedia.org/wiki/Google_Images`
[3] `http://www.lire-project.net/`
[4] `http://lucene.apache.org/core/`
[5] `http://docs.oracle.com/cd/B19306_01/appdev.102/b14302/ch_intr.htm`

Edition". Essa fornisce supporto per ricerche di immagini, immagini mediche, audio e video. Il suo problema principale è che si tratta di un software closed source, oltre che un sistema strettamente legato al solo mondo dei DB Oracle.

IMARS[6] (IBM Multimedia Analysis and Retrieval System) è un sistema per l'analisi e il recupero di elementi multimediali creato da IBM ed è uno strumento estremamente potente che può esser usato per creare indici, classificare ed effettuare ricerche su grandi collezioni di immagini e video. Purtroppo, oltre che essere solamente un progetto di ricerca, è disponibile unicamente per i sistemi Microsoft Windows XP, 2003 e Vista.

4.2 Windsurf

In questa sezione vengono analizzati i dettagli e le specifiche tecniche di WindSurf, tratte da [32, 19].

4.2.1 Informazioni generiche su Windsurf

Windsurf non nasce come un programma "stand alone" per la risoluzione di query multimediali, ma risulta essere un ottimo componente di un sistema complesso, che utilizza questa libreria per la risoluzione di determinate query. Questa libreria è stata anche utilizzata per il tagging automatico di segmenti di video, utile per effettuare categorizzazione e ricerca di determinati elementi nei video e anche per la ricerca automatica di segmenti multimediali in altri video (CCD - content-based copy detection).

Windsurf offre varie possibilità di estensibilità e personalizzazione del framework, fra le quali ricordiamo:

- Possibilità di scegliere differenti rappresentazioni a basso livello di immagini, differenti metodi per la loro segmentazione ed estrazione.

- Differenti costrutti per la comparazione di regioni di immagini (comparazione 1-1 , N-M , ecc...).

- Differenti metodi di esecuzione di query, basate su Rank oppure su preferenze (es. K-nearest neighbour, range, query qualitative di tipo skyline).

Questo prodotto è stato rilasciato sotto licenza QPL[7] (Q Public License) ed è gratuitamente scaricabile all'indirizzo `http://www-db.deis.unibo.it/Windsurf/`.

[6] `http://researcher.watson.ibm.com/researcher/view_group.php?id=877`

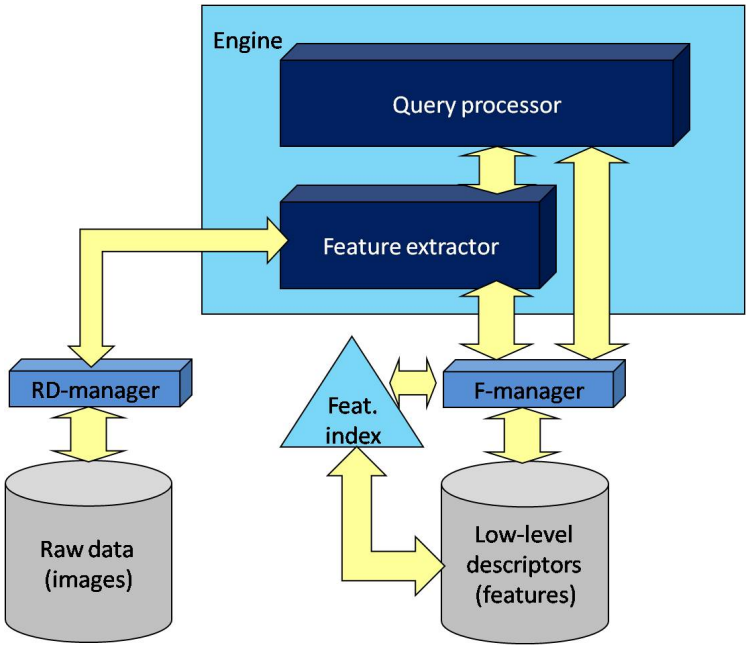
[7] `http://en.wikipedia.org/wiki/Q_Public_License`

4.2.2 Architettura di Windsurf

Windsurf è una libreria scritta in JAVA che sfrutta funzioni C++ di basso livello invocate tramite JNI (Java Native Interface). Le classi java sono state strutturate in modo tale che possano essere facilmente estese per realizzare il modello RBIR desiderato. L'architettura di Windsurf è principalmente formata da quattro elementi e la loro interazione è schematizzata in figura 4.1:

- **Query processor** : elemento incaricato della risoluzione efficiente delle query.
- **Feature extractor** : elemento che effettua l'estrazione a basso livello delle feature e la segmentazione dell'immagine.
- **RD-manager** e **F-manager** : la persistenza dei dati grezzi e delle feature è garantita da questi due elementi; i manager inoltre sono incaricati di gestire i dati su file di testo e su RDBMS.
- **Feature index** : per accedere efficientemente alle feature di basso livello, un Indice di feature è creato (su richiesta) per risolvere le query, Windsurf supporta l'utilizzo di varie tipologie di indici, includendo anche un implementazione dell'indice M-Tree[8] [35, 12].

Figura 4.1: Architettura di Windsurf

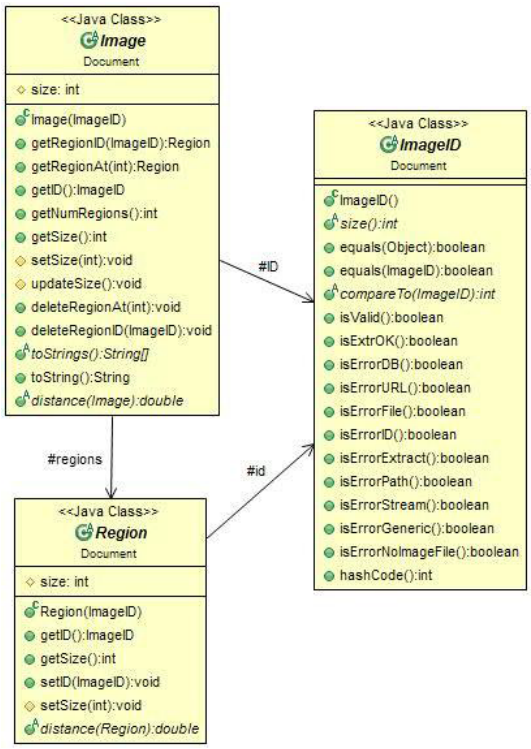


[8] http://www-db.deis.unibo.it/Mtree/

4.2.3 Funzionamento di Windsurf

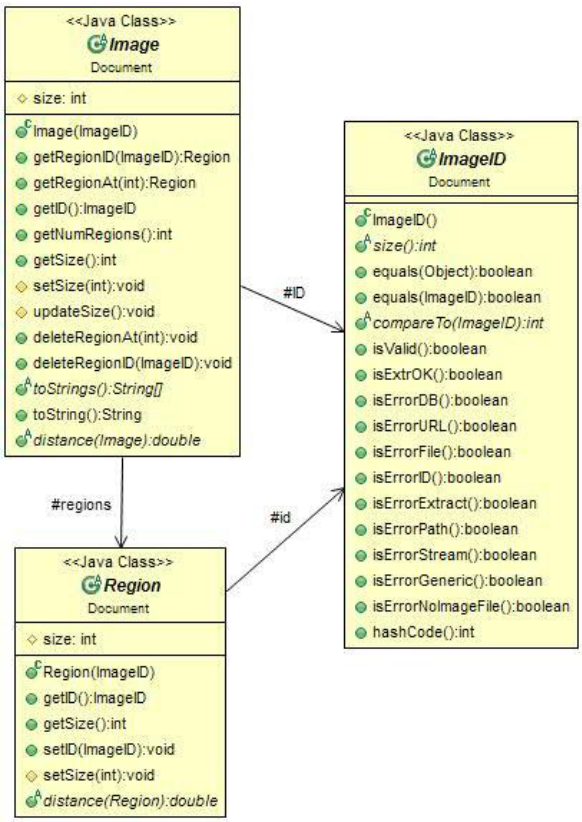
Il framework Windsurf è stato realizzato per la gestione di query su immagini e loro feature. Fondamentalmente Windsurf utilizza i concetti di Image e di Region come elementi costitutivi delle immagini gestite: un'immagine fisica viene introdotta nel framework e le viene associata un'istanza di classe Image specifica; questa viene processata per estrarne le features che la caratterizzano (rappresentate da istanze specifiche di classe Region) e i dati così raccolti vengono inseriti nel DataBase dedicato. Image e Region sono quindi classi Java astratte che devono essere estese da classi concrete che ne implementano i metodi astratti in modo specifico, ma nelle quali sono già definiti attributi e metodi di tipo generico: ogni istanza di Image contiene un'ImageID che ne permette il collegamento con i dati nel DataBase e una collezione di Region che la caratterizzano. Allo stesso modo anche la classe astratta Region contiene un'ImageID per l'associazione con i dati nelle tabelle (particolare attenzione va posta sul metodo astratto distance presente sia in Image che in Region il quale sarà fondamentale nelle query di estrazione). I dati sono salvati su un DBMS, nel nostro caso MySQL, ed in tale DataBase devono essere definite due tabelle per l'utilizzo standard di Winsurf: una contenente i dati relativi ad ogni singola immagine e una contenente i dati di tutte le features estratte dal framework (si possono intendere queste due tabelle come collezioni di dati relativi a istanze di Image la prima e relativi a istanze di Region la seconda).

Figura 4.2: Schema relazione Image e Region



Windsurf effettua quindi query di inserimento ed estrazione mettendo in relazione i dati tra implementazioni di Image/Region e quelli contenuti nelle tabelle appena descritte. Questo processo avviene tramite le implementazioni di due manager, classi astratte che definiscono in modo generico i metodi per la comunicazione da oggetti a DataBase e viceversa: la classe astratta RawDataManager e la classe astratta FeatureManager. Le implementazioni specifiche di queste due classi possono essere adattate a qualsiasi DBMS o file di testo, in quanto sono definiti astratti i metodi createConnection, closeConnection e tutti i metodi di accesso ai dati. Queste classi espongono tutti i metodi per l'inserimento e l'estrazione di dati relativi a oggetti di classe Image o Region, in modo che si possano definire estensioni specifiche per casi particolari. Le estensioni della classe RawDataManager si occuperanno nello specifico della gestione delle query relative a oggetti di classe Image intesi in questo caso come "raw data" ovvero dati generici che non è necessario preprocessare. Le estensioni della classe FeatureManager si occuperanno invece dell'estrazione e inserimento nella tabella delle features di tutti i dati relativi alle regioni delle immagini elaborate dal RawDataManager. È importante sottolineare che questa classe si occupa solamente di fornire un'interfaccia tra oggetti di classe Region e la loro rappresentazione del DataBase, e non si occupa quindi della manipolazione delle immagini al fine di estrarne features.

Figura 4.3: Schema FeatureManager e RawDataManager

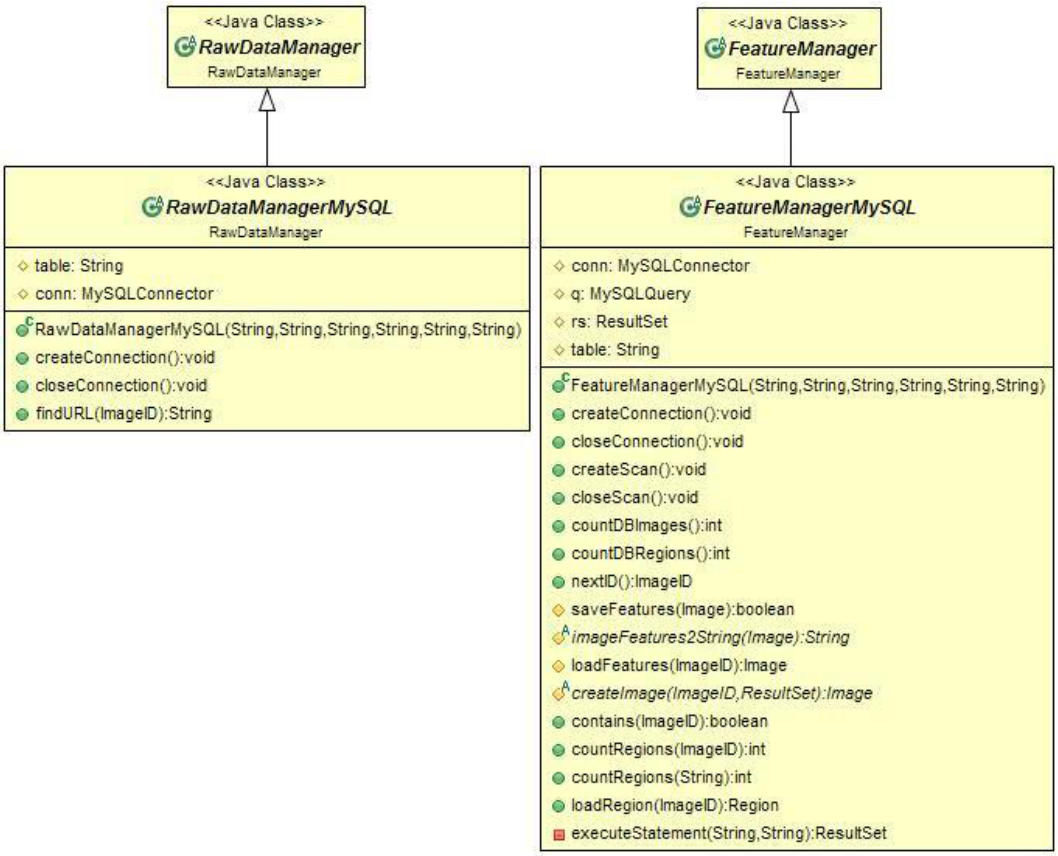


In Windsurf esiste infatti un modulo creato appositamente per l'estrazione delle features partendo dalle immagini inserite (FeatureExtractor)

Windsurf nella sua versione standard estende le classi Image e Region con le classi specifiche WindurfImage e WindsurfRegion.Le classi manager sono invece estese per poter utilizzare DataBase MySQL tramite la classe astratta RawDataManagerMySQL (nel framework standard ulteriormente estesa da RawDataManagerWindsurfMySQL) e la classe astratta FeatureManagerMySQL (ulteriormente estesa da WindsurfFeatureManagerMySQL nel framework standard).

Queste classi realizzano i metodi di connessione e disconnessione dal DataBase MySQL, la quale configurazione è inserita all'interno di un file config.txt ausiliario. I metodi per la realizzazione effettiva delle queries rimangono però astratti, in modo da poterle adattare al tipo di dati considerati.

Figura 4.4: Schema Classi Manager

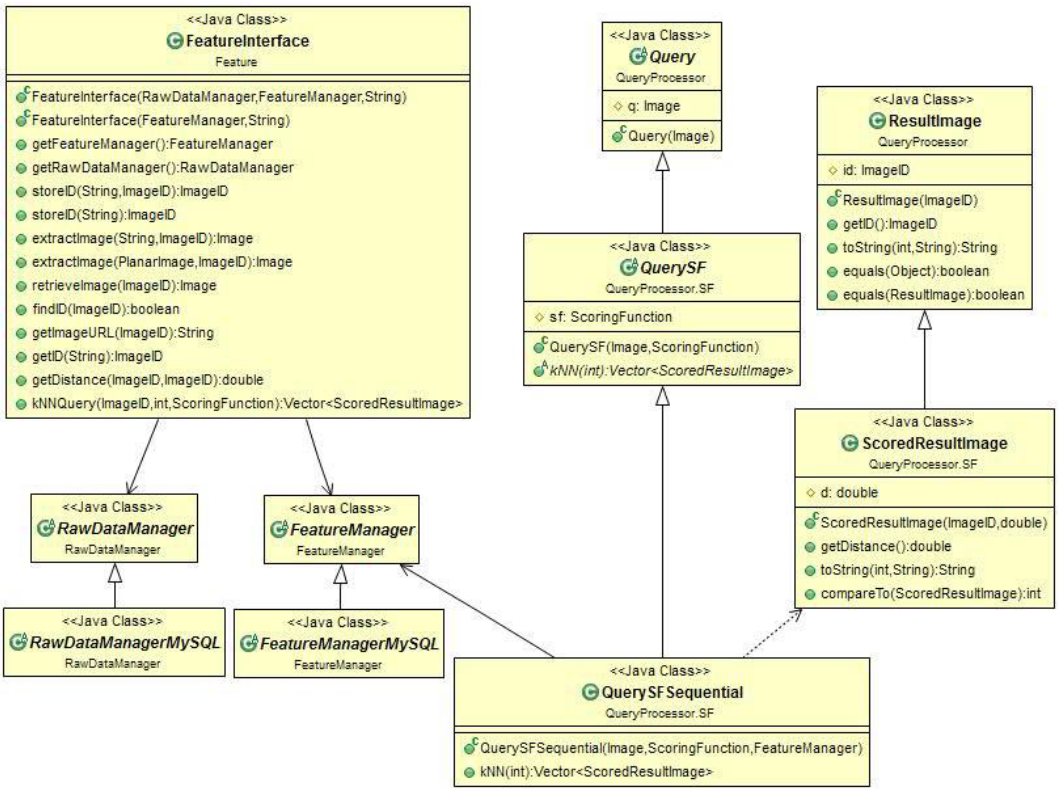


Le classi manager sono racchiuse nella classe FeatureInterface (la quale contiene riferiementi a RawDataManager e FeatureManager) che è il vero e proprio core del framework: questa classe espone tutti i metodi per l'inserimento e recupero di immagini e loro feature (storeID, extractImage e retrieveImage) e quelli per l'estrazione di dati simili (kNNQuery e getDistance). Questa classe fondamentale è anche alla base dei wrapper realizzabili per semplificare l'utilizzo del framework da parte dell'utente

(WrapperFeatureWindsurfMySQL, WrapperFeatureWindsurfTxt, ecc...).

Per le interrogazioni si utilizza la classe astratta Query che viene implementata dalla classe QuerySF in Windsurf (poiché per la ricerca di similarità tra immagini è necessaria una Scoring Function) la quale estrae immagini ordinate sotto forma di implementazioni della classe ResultImage (o ScoredResultImage nello specifico). Le estensioni della classe Query si occupano quindi di definire i metodi per recuperare dati dal DataBase a partire dal confronto con un elemento di classe Image passato come parametro.

Figura 4.5: Schema Classi Query



4.2.4 Classi ed Elementi direttamente utilizzati

Dopo aver analizzato nel dettaglio l'architettura di Windsurf, spieghiamo quali sono le classi direttamente sfruttate per risolvere query multimediali.

Come discusso nella sezione 3.2.2 su Windsurf Local Source, le tipologie di query che interessano e verranno usate nel nostro progetto sono:

- Query di Range
- Top-K Query

Tutti gli elementi che si occupano di risolvere tali interrogazioni di similarità in Windsurf si trovano all'interno del package ***QueryProcessor***, le cui classi principali sono **ResultImage** e **Query**.

La prima classe (ResultImage) modella il risultato e contiene sempre almeno l'id dell'immagine risultato, in particolare è utilizzata per restituire un'istanza di **scoredResultImage**, che include anche il valore di distanza rispetto alla query.

La seconda classe (Query) rappresenta un immagine query, per ottenerla ci sono due modi:

- Creare una query su un'immagine presente nel db:

  ```
  Image query= wf.retrieveImage(query_id)
  ```

- Creare una query a partire da un file:

  ```
  Image query = f_extractor.extract(query_imageFileName, query_id)
  ```

  dove *f_extractor* è un'istanza di FeatureExtractor e query_id un identificativo numerico (irrilevante per i nostri scopi, quindi possiamo supporlo = -1)

Nello specifico, nell'utilizzo in questo progetto, interessa sfruttare le feature di Windsurf, quindi utilizzeremo la sottoclasse **QuerySF** derivata da Query. Questa richiede di specificare, all'atto della creazione dell'istanza, anche la scoring function da utilizzare per combinare le distanze a livello di regione nella distanza a livello di immagine: un esempio è la classe **ScoringFunctionEMD** che realizza la **Earth mover's distance**[9]. L'unico metodo della classe QuerySF è **kNN** che ci permette di risolvere le query **k-nearest neighbor**, ovvero di ottenere le k immagini più simili alla query. QuerySF, però, è ancora una classe astratta, in quanto ora dobbiamo scegliere come risolvere la nostra interrogazione: se usando l'indice M-Tree (classe **WindsurfQuerySFMtree**) o un algoritmo sequenziale (classe **QuerySFSequential**); nel nostro caso si userà l'indice, che fornisce anche altri metodi, come il metodo range (che permette di restituire tutte le immagini abbastanza vicine alla query) o l'accesso tramite cursore per restituire (eventualmente) tutte le immagini in ordine non-decrescente di distanza rispetto alla query.

4.3 Architettura Funzionale di MOMIS-Windsurf

Per poter "integrare" Windsurf nell'architettura di MOMIS si deve realizzare un opportuno wrapper per gestire gli elementi multimediali. In particolare gli elementi multimediali dovranno essere completamente gestiti da Windsurf, implicando che

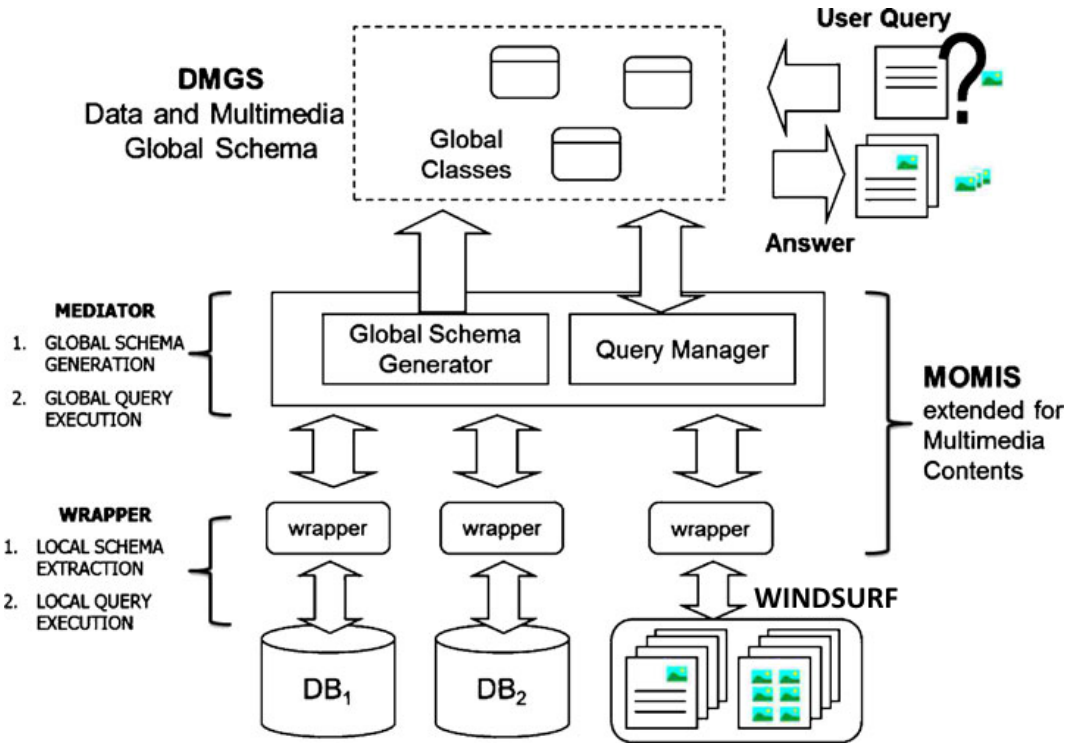
[9] `http://en.wikipedia.org/wiki/Earth_mover%27s_distance`

MOMIS vedrà questa libreria come una sorgente di dati multimediali, tipo black box, capace di rispondere a determinate query. Il nostro obiettivo è creare un wrapper per MOMIS, che nasconda la reale implementazione della risorsa Windsurf e che permetta la corretta comunicazione fra questi due applicativi.

Si può subito notare la necessità di avere una risorsa di tipo "Windsurf": questa sorgente di dati ingloba in se la libreria Windsurf che permette di calcolare le interrogazioni multimediali, ricordandoci che la mediazione avverrà grazie al wrapper creato ad-hoc.

Premettendo che nell'attuale implementazione di Windsurf vengono gestite solo immagini in formato JPEG[10], quando si parlerà di "elementi multimediali" nell'attuale realizzazione ci si riferisce alla gestione di sole immagini.

Figura 4.6: Architettura MOMIS-Windsurf



Per utilizzare la risorsa Windsurf, l'Integration Designer dovrà "creare" una risorsa locale Windsurf partendo da una directory contenente le immagini sulle quali si vuole fare uno studio o una comparazione. È stata creata una semplice shell che permette all'utente di scegliere quali operazioni di Windsurf (stand alone) utilizzare, mostrando a video i risultati.

Per utilizzare Windsurf sono necessari i seguenti elementi:

- Una directory contenente file multimediali

[10] `http://en.wikipedia.org/wiki/JPEG`

- La libreria Windsurf (attualmente un file JAR[11])
- La DLL con implementazione dell'M-Tree Index, compilata per la macchina sulla quale si sta lavorando
- Le librerie necessarie a Windsurf per essere eseguito (JAI Codec,JAI Core, JSON,MYSQL Server e MySQL Connector)
- Un file di testo contente le configurazioni della risorsa
- Una directory per i file temporanei
- Una directory per gli indici
- [*OPZIONALE*] Un file CSV contenente le informazioni extra legate ad ogni elemento multimediale

Per creare correttamente la risorsa multimediale di Windsurf bisogna aver completato il file di configurazione con tutti parametri ad esso necessari, quindi successivamente bisogna lanciare il software Windsurf da linea di comando; è stato messo a disposizione dell'utente un file di aiuto che spiega come utilizzare il tutto, sottolineando come sia molto più semplice per l'utente andare a modificare i file di script disponibili per macchine Microsoft Windows (file ".bat") che per macchine GNU-Linux (file ".sh"). Ricordiamo inoltre che la shell di Windsurf offre all'utilizzatore il comando "help" che permette di avere informazioni su quali funzioni esso mette a disposizione e con quali parametri invocarle. La "creazione" della risorsa, in realtà non è altro che la popolazione di alcune tabelle di un database (nel nostro caso il DBMS utilizzato è MySQL) e la creazione di un file indice. Nel caso in cui venisse fornito anche il file con informazioni extra (tipicamente un file CSV), verrà popolata anche una tabella aggiuntiva contenente le informazioni extra che sono state appunto fornite dall'utente. Si noti che la parte extra fornita come parametro opzionale è un aspetto importante del wrapper, in quanto consente di passare da una **WLS** con *schema fisso* (cioè con i quattro attributi fissi, quindi ID, NAME, IMAGE, DISTANCE) ad una **WLS_EXTRA** che ha anche eventuali altri attributi non più fissati ma definibili dall'utente all'atto della creazione della risorsa.

Una volta popolate queste tabelle, la risorsa è pronta per essere utilizzata all'interno di MOMIS o direttamente dall'utente tramite la libreria Windsurf: è stata realizzata anche una parte del software che permette di utilizzare funzioni di ricerca direttamente da linea di comando.

[11] `http://en.wikipedia.org/wiki/JAR_(file_format)`

4.4 Creazione Wrapper per Windsurf

Grazie al funzionamento tramite wrapper di MOMIS, il Query Manager (elemento core del sistema) può trattare tutte le risorse in modo omogeneo, in quanto ogni wrapper nasconde e astrae la reale implementazione della medesima e fa si che essa restituisca un ResultSet comprensibile e navigabile da MOMIS. Si premette che in questo capitolo si utilizzerà spesso il termine "wrapper" dandogli il significato di "risorsa".

Si è deciso di creare il wrapper di Windsurf, elemento che permette a MOMIS di lavorare con dati multimediali. Per vincoli di progetto legati alla modularità del sistema, MOMIS deve rimanere completamente slegato da Windsurf, e per riuscire in tale proposito si è data la possibilità di importare il wrapper di Windsurf tramite l'import di una risorsa "Generica"; in questo caso "Generica" sta a significare che è possibile importare una qualunque tipologia di risorsa (quindi un wrapper) a condizione che si conoscano tutti i parametri ad esso necessari per la sua configurazione. Il wrapper di Windsurf è stato costruito ad-hoc per MOMIS, in modo tale che fosse possibile invocare i comandi su esso e strutturare i risultati da restituire.

Il Query Manager all'atto dell'import della risorsa chiederà di descriversi (primo comando inviato al wrapper di una risorsa), per poi all'atto di interrogazione inoltrargli la query locale calcolata basandosi sulla Mapping Table (dopo il Query Unfolding). Il wrapper quindi calcolerà la risposta ed inoltrerà al QM un ResultSet, che altro non è un cursore per navigare i risultati che soddisfano le clausole della query ad esso inoltrata.

4.4.1 Logica del Wrapper Windsurf

Continuando l'analisi della logica del sistema MOMIS e del principio di scomposizione della query globale su sorgenti locali analizzata nel paragrafo 2.4, analizziamo il comportamento del Query Manager in dettaglio.

Prima che venga iniziato il processo di query unfolding della Global Query inserita dall'utente, il sistema controlla se è presente un predicato multimediale (ricordandoci che l'oggetto multimediale è stato già trasformato in stringa ed è caratterizzato dalla keyword __JSON__FOR_MULTIMEDIA); per non alterare il principio di funzionamento del Query Manager, si è deciso che il modo migliore per estendere il processo di Query Unfolding alle query multimediali è quello di effettuare un preparsing della query inserita dall'utente, rimuovendo la porzione di query contenente l'elemento multimediale ed i possibili predicati legati alla similarità. Dopo di ciò, il QM esegue la scomposizione della Query Globale in Query Locali (come già spiegato nel paragrafo 3.2) e all'atto dell'inoltro di essa verso la giusta sorgente (wrapper), viene effettuato il controllo se il wrapper risulti essere di tipo multimediale. In questo

caso dovrà essere aggiunto in coda alla query locale la parte multimediale precedentemente estratta, dopo aver effettuato la trasformazione degli attributi globali in attributi locali in accordo con la Mapping Table.

Il wrapper che riceve la query locale per prima cosa controlla se sono presenti dati multimediali, in caso positivo li analizza e li ritrasforma in elementi multimediali. Quindi viene effettuato il parsing[12] e viene eseguito quanto espresso nella query. La sua esecuzione si conclude con la restituzione di un ResultSet in formato comprensibile e navigabile da MOMIS.

4.4.2 Gestione degli Elementi Multimediali in MOMIS-Windsurf

Ogni elemento multimediale all'interno del sistema integrato MOMIS-Windsurf è trasformato in stringa utilizzando una particolare codifica, in particolare ogni immagine indirizzata al wrapper o restituita da esso è trasferita come stringa utilizzando lo schema *Data URI*[13], che è uno standard sviluppato da IETF (Internet Engineering Task Force) e rilasciato con il **RFC 23975**. Questo standard permette di trasferire qualunque documento utilizzando il MIME code[14] (**RFC 2045**) in quanto ogni wrapper di MOMIS può essere eseguito su differenti server di rete e può essere raggiunto tramite i protocolli "Web Service".

Dobbiamo sottolineare inoltre che questa operazione permette di lasciare a livello di logica inalterato il QM e tutti gli altri elementi di MOMIS, ma ha lo svantaggio di rendere *triplicata* la dimensione degli elementi multimediali rispetto a quella del file originale.

[12] Non ancora implementato nel wrapper.

[13] `http://en.wikipedia.org/wiki/Data_URI_scheme`

[14] `http://en.wikipedia.org/wiki/MIME`

Capitolo 5

MOMIS-Windsurf ed Analisi Mediche

Questo capitolo illustra il Case Study scelto per utilizzare e verificare il framework MOMIS-Windsurf: la gestione degli esami PET[1].

5.1 Campo di Applicazione

Gli esami PET sono degli esami di Tomografia a Emissione di Positroni (Positron Emission Tomography), che si basa sulla tecnica di medicina nucleare utilizzata per la produzione di bioimmagini (immagini del corpo). La PET fornisce informazioni di tipo fisiologico, a differenza di TC[2] e RM[3] che invece forniscono informazioni di tipo morfologico. Con questo esame si ottengono informazioni sui processi funzionali all'interno del corpo.

5.1.1 Descrizione procedura PET

La procedura inizia con l'iniezione di un radiofarmaco formato da un radioisotopo tracciante con emivita breve, legato chimicamente a una molecola attiva a livello metabolico, detta vettore. Dopo un tempo di attesa, nel quale questa molecola attiva raggiunge una determinata concentrazione all'interno dei tessuti organici da analizzare, il soggetto viene posizionato nello scanner. L'isotopo di breve vita media decade, emettendo un positrone. Dopo un percorso che può raggiungere al massimo pochi millimetri, il positrone si annichila con un elettrone producendo una coppia di fotoni gamma entrambi di energia 511 KeV emessi in direzioni opposte tra loro (fotoni back to back).

[1] https://it.wikipedia.org/wiki/Tomografia_a_emissione_di_positroni
[2] https://it.wikipedia.org/wiki/Tomografia_computerizzata
[3] https://it.wikipedia.org/wiki/Risonanza_magnetica_nucleare

Questi fotoni sono rilevati quando raggiungono uno scintillatore, nel dispositivo di scansione, dove creano un lampo luminoso, rilevato attraverso dei tubi fotomoltiplicatori. Punto cruciale della tecnica è la rilevazione simultanea di coppie di fotoni: i fotoni che non raggiungono il rilevatore in coppia, cioè entro un intervallo di tempo di pochi nanosecondi, non sono presi in considerazione. Dalla misurazione della posizione in cui i fotoni colpiscono il rilevatore, si può ricostruire l'ipotetica posizione del corpo da cui sono stati emessi, permettendo la determinazione dell'attività o dell'utilizzo chimico all'interno delle parti del corpo investigate. Lo scanner utilizza la rilevazione delle coppie di fotoni per mappare la densità dell'isotopo nel corpo, sotto forma di immagini di sezioni (generalmente trasverse) separate fra loro di 5 mm circa. La mappa risultante rappresenta i tessuti in cui la molecola campione si è maggiormente concentrata e viene letta e interpretata da uno specialista in medicina nucleare al fine di determinare una diagnosi ed il conseguente trattamento.

5.1.2 Applicazioni

La PET è usata estensivamente in oncologia clinica (per avere rappresentazioni dei tumori e per la ricerca di metastasi) e nelle ricerche cardiologiche e neurologiche.

Ad ogni modo, mentre gli altri metodi di scansione, come la TAC e la RMN permettono di identificare alterazioni organiche e anatomiche nel corpo umano, le scansioni PET sono in grado di rilevare alterazioni a livello biologico molecolare che spesso precedono l'alterazione anatomica, attraverso l'uso di marcatori molecolari che presentano un diverso ritmo di assorbimento a seconda del tessuto interessato. Con una scansione PET è possibile visualizzare e quantificare con discreta precisione il cambio di afflusso sanguigno nelle varie strutture anatomiche (attraverso la misurazione della concentrazione dell'emettitore di positroni iniettato).

La PET gioca un ruolo sempre maggiore nella verifica della risposta alla terapia, specialmente in particolari terapie anti-cancro.

5.1.3 Utilizzo PET e loro valutazione

La valutazione delle aree ipercaptanti sulle immagini di norma viene fatta in maniera qualitativa da medici esperti; tuttavia esistono dei casi dubbi in cui un'analisi semi-quantitativa può essere utile. Il principale parametro utilizzato in tal senso è il **SUV**[4] (Standardized Uptake Value) che si può calcolare su ogni area dubbia mediante la seguente formula:

$$SUV = \frac{Concentrazione\_della\_radioattivitá\_dei\_tessuti(t)}{Attivitá\_iniettata(t)/Massa\_corporea}$$

[4] `http://en.wikipedia.org/wiki/Standardized_uptake_value`

A parole, il SUV rappresenta il rapporto fra la concentrazione della radioattività introdotta trovata in una parte del corpo al tempo t, e la concentrazione della radioattività di un ipotetico caso in cui la radioattività fosse stata iniettata nell'intero corpo, le due misure dovranno essere misurate ad uno stesso punto, quindi al tempo dell'iniezione o al tempo del frame in esame. In un possibile caso, l'attività iniettata è da correggere per il decadimento fisico fra il tempo di iniezione (t=0) e il tempo a cui è stato preso il frame (t=τ).

Tale rapporto mostra quante volte capta di più (o di meno) l'area interessata rispetto a quanto capterebbe un'area di uguale massa. Il calcolo del SUV poi può essere ulteriormente corretto da altri parametri, come la superficie corporea o la massa magra. Altri modelli più accurati si basano sull'analisi della cinetica di captazione, rilevata tramite acquisizioni e misurazioni del pool circolante seriate.

Quindi in questo progetto sono state effettuate varie modifiche al framework di Windsurf per permettere la gestione corretta delle analisi PET e i loro valori SUV.

5.2 Gestione delle PET in MOMIS-Windsurf

Questa sezione descrive come gestire le PET nel sistema MOMIS-Windsurf; la modularità del framework realizzato fa si che nessuna modifica sarà richiesta al sistema MOMIS (ed in particolare al Query Manager) e le uniche presenti riguarderanno esclusivamente la sorgente locale WindSurf.

Il calcolo del SUV è un esempio significativo della necessità di integrare dati tradizionali e dati multimediali; infatti il suo numeratore, *Concentrazione della radioattività dei tessuti(t)*, è un valore che deve essere calcolato sulla base delle immagini allegate alle PET, mentre il denominatore contiene valori (quali *Attività iniettata(t)* e *Massa corporea*) che sono legati alla cartella clinica del paziente e quindi come ogni altro dato tradizionale di questo tipo sarà memorizzato in opportuni database.

Dall'esempio è anche evidente che per la gestione delle PET in MOMIS-Windsurf ci sono alcuni elementi, quali il calcolo della *Concentrazione della radioattività dei tessuti(t)* che deve essere fatto internamente alla libreria Windsurf (e quindi di competenza dei relativi ricercatori responsabili) mentre la parte più propriamente di integrazione deve essere fatta all'interno di MOMIS. In maggior dettaglio, la modifica realizzata sul framework Windsurf consiste nell'introduzione di esami PET come unità di lavoro, ognuno contenente una serie di valori SUV. Una volta inseriti tutti gli esami noti, un nuovo esame effettuato può essere confrontato con i valori SUV degli esami già presenti nel DataBase in modo da estrarne altri con valori analoghi. Questo permetterà un confronto preliminare in modo da analizzare se casi simili hanno richiesto esami ulteriori (magari più invasivi). Nel caso specifico un esame

PET sarà una cartella contenente un insieme di immagini relative allo stesso esame, dalle quali un esperto sia in grado di estrarre valori SUV.

5.2.1 Descrizione delle modifiche in PET-Windsurf

Per sfruttare l'infrastruttura di Windsurf già presente ed adattare questo framework al calcolo dei valori SUV delle PET, si è deciso di abbinare i concetti di IMAGE e REGION a quelli di PET e SUV. Possiamo subito notare come questa mappatura non è una vera e propria estensione delle classe analizzate nel capitolo 4.2.3 di Windsurf, in quanto una PET in realtà risulterà essere una directory di immagini e non una singola immagine, e il valore SUV un singolo numero decimale e non una regione di immagini.

Sfruttando le classi già esistenti sarà possibile utilizzare le relazioni che intercorrono fra le classi Image e Region, per consentire l'inserimento dei valori di SUV nel database senza modificare gli elementi già esistenti, viene creata la classe di supporto SUVDetails che incapsula il valore numerico (SUV) ed il path dell'immagine su cui il valore SUV è stato calcolato.

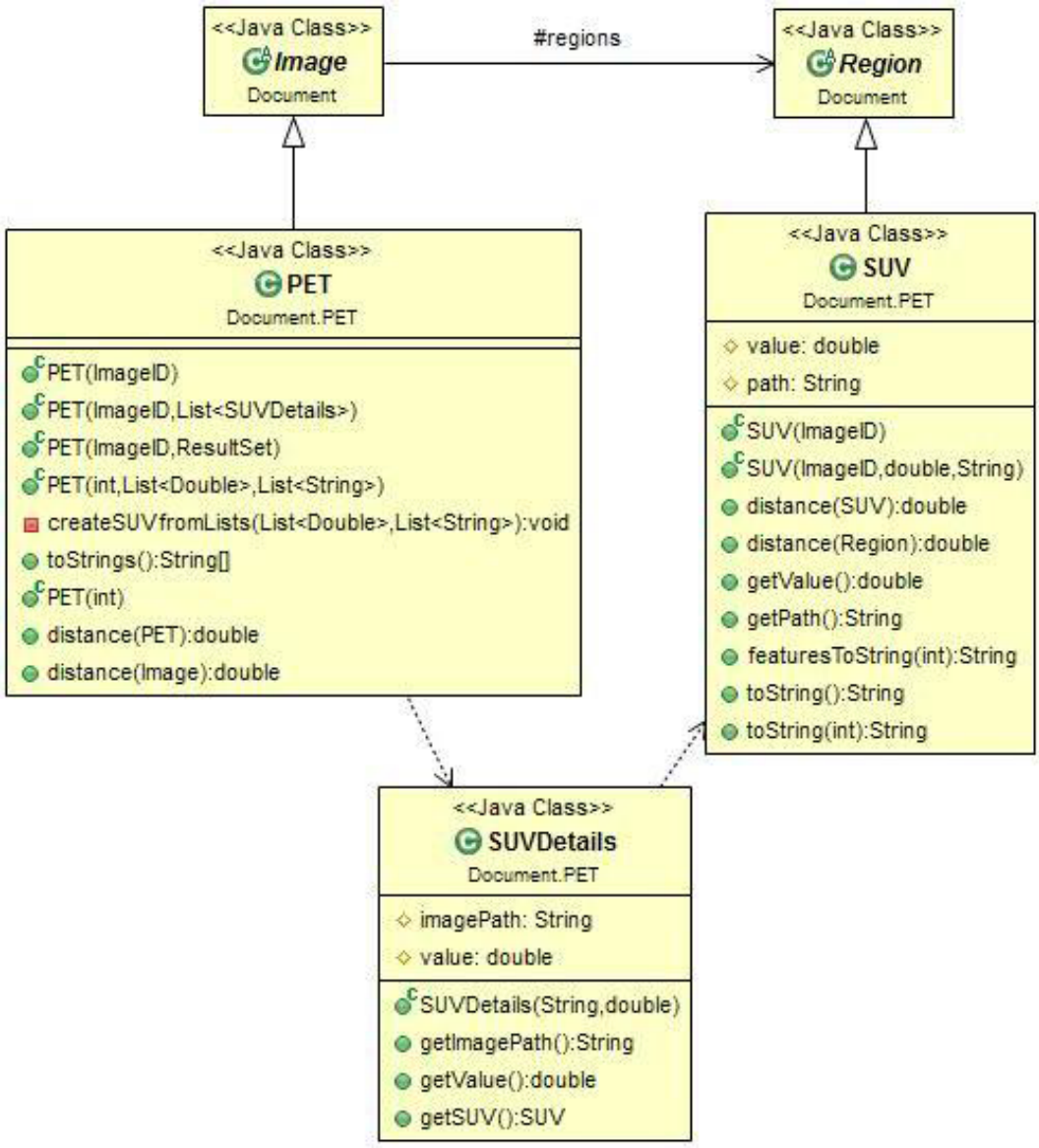
Si è resa inoltre necessaria la modifica delle tabelle del database per il salvataggio dei nuovi dati. La tabella che descrive le immagini rimane invariata in quanto contiene semplicemente un path, che originariamente era un path dell'immagine (singola), mentre ora si riferirà ad una cartella (PET) che conterrà le immagini, ma la "sostanza" non cambia. Mentre per la tabella features, non serviranno più tutti quei campi che descrivevano la feature, prima necessari per il calcolo della distanza fra un'immagine ed un altra, ma solo di un valore numerico double in relazione alla cartella PET. Ricordando quanto detto in precedenza (capitolo introduttivo su Windsurf 4.2.3), questi elementi sono gestiti dalle classi DataManager: per la PET non ci sarà alcun cambiamento (al RawDataManager), mentre si necessiterà di una nuova classe per la gestione dei dati relativi ai SUV, in particolare, partendo da FeatureManager, la classe PETFeatureManagerMySQL (che esprimerà i metodi per la gestione degli attributi della classe SUV) avrà:

- STRING *imageFeatures2String(IMAGE image)*: che realizza la stringa SQL con i valori dei SUV partendo dagli attributi della PET in esame.

- IMAGE *loadFeaturesNext()*: che carica la PET successiva prelevando i dati dei valori SUV che la compongono.

- INT *contains(IMAGE i)*: che verifica se l'immagine è già contenuta nel DataBase (utilizzato per sapere se effettuare un nuovo inserimento o un semplice aggiornamento di una PET già presente).

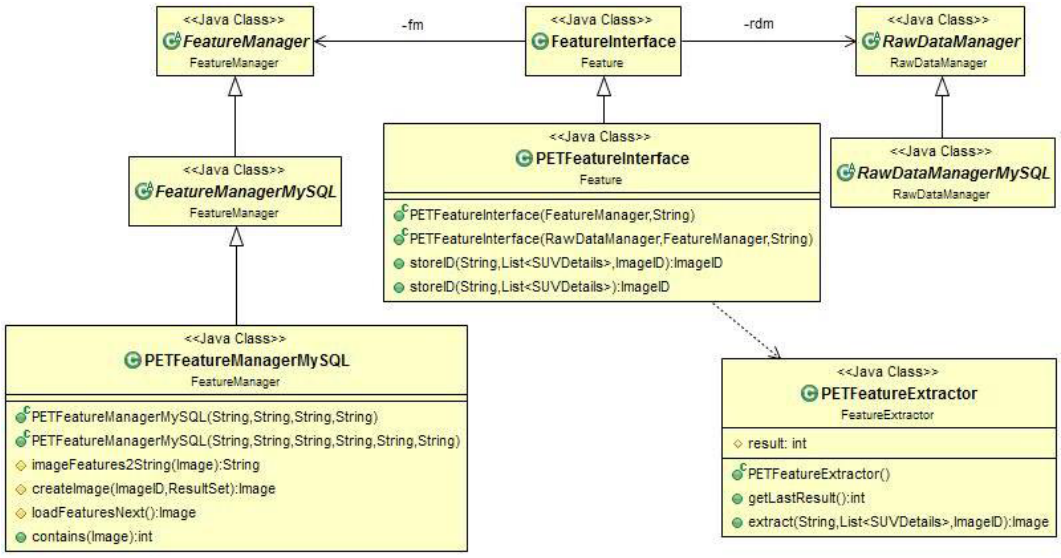
Figura 5.1: Schema Classi Relazioni



La classe PETFeatureInterface utilizza questi due manager per effettuare l'accesso ai dati nelle tabelle introdotte. Questa classe estende FeatureInterface di cui usa molti metodi, ridefinendo il metodo di salvataggio storeID, il quale cerca una PET passata come parametro nel DataBase e la inserisce se non la trova. Il metodo storeID utilizza anche un'altra classe implementata, PETFeatureExtractor, che trasforma i dati estratti dalla PETFeatureInterface in oggetti di tipo PET e SUV.

In aggiunta alle funzionalità già evidenziate per l'inserimento e l'aggiornamento dei dati relativi alle PET e ai loro valori SUV, sono poi state implementate nuove classi per il recupero dei valori SUV simili. Queste classi devono fornire servizi in grado di recuperare un elenco di SUV features simili ad uno passato come parametro: sia i Top-K Nearest Neighbors SUV, sia i SUV che distano meno di un valore soglia.

Figura 5.2: Schema Classi realizzazione PET e SUV



Per fare questo non è più possibile utilizzare le classi già disponibili Query e ResultImage poiché il framework Windsurf è stato creato per il recupero di immagini simili, ma nel nostro caso la classe che estende Image è PET mentre a noi serve recuperare SUV, quindi Region simili.

Le due classi Query e ResultImage sono quindi sostituite da QueryRegion e ResultRegion che permettono l'estrazione di un valore SUV (estensione di Region) al posto di un'immagine.

I valori SUV restituiti saranno delle istanze di ResultRegion, le quali potranno essere utilizzate per ottenere i SUV corrispondenti utilizzando l'ImageID estratto.

È offerto anche un metodo retrieveRegion che restituisce un'istanza di SUV in base agli ID di PET e SUV passati; questo metodo semplifica l'estrazione dei SUV simili poiché le istanze di ResultRegion contengono i riferimenti all'ImageID della PET e del SUV che descrivono.

Capitolo 6

Framework MOMIS-Windsurf: Demo

Questo capitolo costituisce una sorta di manuale d'uso del framework realizzato, riportandone una sua demo completa.

6.1 Configurazione del Progetto

L'analisi della creazione di un progetto MOMIS-Windsurf con varie sorgenti è stata suddivisa in due macro-blocchi:

1. Creazione della Risorsa Multimediale Windsurf
2. Creazione del Progetto MOMIS per l'integrazione
    - Importare da 1 a N sorgenti tradizionali e 1 risorsa multimediale Windsurf
    - Creazione del nuovo Global Schema
    - Interrogazione della risorsa globale integrata

Nei prossimi capitoli analizzeremo nel dettaglio ognuna di queste fasi.

6.1.1 Elementi Necessari al Funzionamento

Per far funzionare il sistema MOMIS-Windsurf, necessiteranno i seguenti elementi installati sul calcolatore.

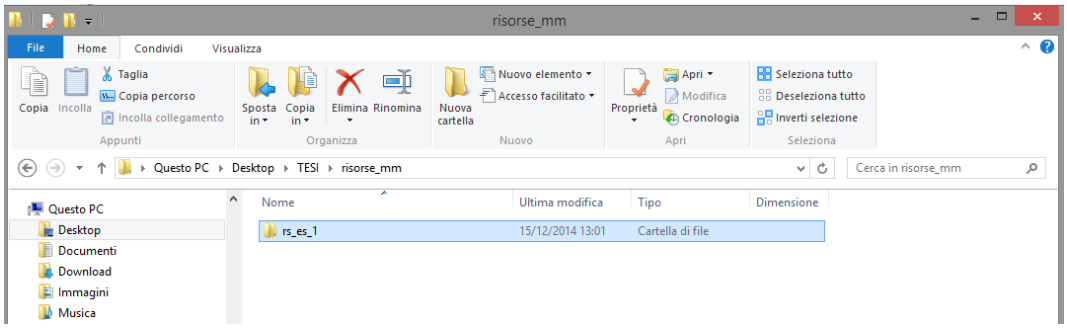
- JRE (Java Runtime Environment) 1.8 32bit: l'utilizzo della versione solo a 32 bit è dovuta al limite di JAI (java advanced image) che è implementata in hardware solo per macchine a 32bit, in particolare la parte C sulla quale si basa è stata compilata solo per macchine a 32bit.

- IA-32: l'architettura "Intel 32bit" è necessaria in quanto l'implementazione dell'indice M-Tree utilizzata da Windsurf non è mai stata ne testata ne compilata per macchine a 64bit.
- MOMIS 1.2 : ultima versione disponibile.
- Libreria Windsurf.
- MySQL Server 5.6.
- Editor di testo (consigliato Notepad++).

6.2 Creazione della risorsa multimediale

Premettendo che sul computer su cui si sta eseguendo quest'operazione devono essere presenti tutti gli elementi elencati precedentemente, il primo passo per la creazione della risorsa multimediale è quello di creare una cartella in un path scelto dall'utente dove ha i privilegi di scrittura e modifica file. Questa directory risulterà essere il contenitore della risorsa Windsurf, nel nostro esempio il nome scelto è "rs_es_1"

Figura 6.1: Creazione Risorsa MM - 1



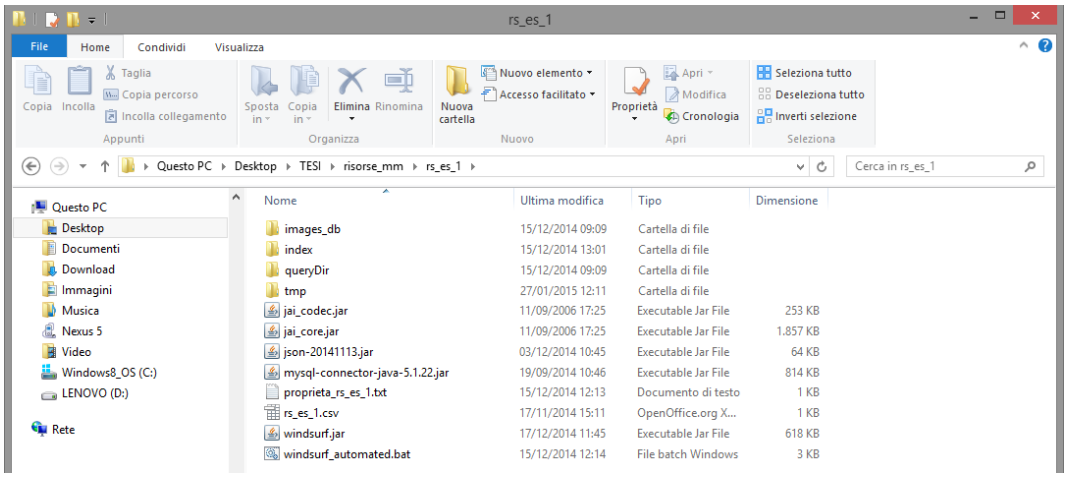
All'interno di questa cartella andremo ad inserire le librerie necessarie a Windsurf, quindi:

- jai_codec.jar
- jai_core.jar
- json-20141113.jar
- mysql-connector-java-5.1.22.jar

Inoltre dovremo anche inserire la libreria di Windsurf stesso, che nel nostro caso risulta essere contenuta nel file "*windsurf.jar*". Creiamo inoltre le directory che dovranno contenere le immagini, gli indici ed i file temporanei necessari al funzionamento di Windsurf. Dovremo creare un file di configurazione che conterrà tutte

le informazioni necessarie a Windsurf per essere eseguito. Nella pagina seguente è riportato un file di configurazione tipico per questa libreria.

Figura 6.2: Creazione Risorsa MM - 2



Per esempio, un file di configurazione tipo sarà:

```
windsurf.db.driver=com.mysql.jdbc.Driver
windsurf.db.url=jdbc:mysql://127.0.0.1:3306/windsurf_rs_es_1
windsurf.db.passwd=<pwd>
windsurf.db.user=<usr>
windsurf.db.host=localhost
windsurf.db.port=3306
windsurf.db.dbname=windsurf_rs_es_1
windsurf.feat_table_name=feat_immagini
windsurf.rd_table_name=rd_immagini
windsurf.host=localhost
windsurf.basePath=C:\\...assoluto...\\rs_es_1\\
windsurf.temporaryFilesPath=tmp/
windsurf.imagesPath=images_db/
windsurf.imageIndexName=index/imageIndex.m3
windsurf.regionIndexName=index/regionIndexName.m3
windsurf.CVSFile=rs_es_1.csv
windsurf.CVSFilePK=IMG_NAME
windsurf.CVSNameTab=EXTRA_DB
windsurf.CVSNameDB=windsurf_rs_es_1
```

Possiamo vedere come i primi elementi si riferiscono al DBMS utilizzato da Windsurf per funzionare, quindi i dati per effettuare la connessione (url, username, password, porta, ecc..). Successivamente dobbiamo inserire il nome delle due tabelle che la libreria utilizza per funzionare (una che contiene i riferimenti alle immagini ed una

che contiene, o meglio conterrà, i dati sulle features), il path base della risorsa, i path relativi ad alcune cartelle fondamentali a partire dal percorso base precedentemente fornito, i nomi dei file indici che dovranno esser popolati e per concludere le informazioni riguardanti il file CSV contenente dati extra relativi agli elementi multimediali (ricordiamo che questo file non è obbligatorio, in quanto se esso non viene trovato, in automatico il sistema ignora questa informazione inserita).

Come abbiamo precedentemente detto, nel caso in cui si possiedono delle informazioni aggiuntive legate agli elementi multimediali, esse dovranno esser inserite all'interno di un file CSV. Per riuscire ad interpretare correttamente queste informazioni, esse devono essere obbligatoriamente strutturate seguendo questo formato:

1. La prima riga deve contenere i *nomi* delle colonne.

2. La seconda riga deve contenere il *tipo* dei dati contenuti, come se si stesse lavorando in JAVA.

3. Dalla terza riga (compresa) in poi avremo i dati veri e propri.

Nel file di configurazione spiegato in precedenza, possiamo vedere che viene indicato all'atto della configurazione il campo Primary Key, necessario per la creazione della tabella nel database. Si preferisce trasferire tutti i dati nel DBMS direttamente all'atto della creazione della risorsa, in quanto le performance di una ricerca in DB sono estremamente superiori rispetto a quelle di ricerca in un file, oltretutto sottolineiamo ancora una volta come l'azione che noi chiamiamo "*Creazione della risorsa multimediale*" in realtà è semplicemente il "Popolare con dati estratti dalle risorse multimediali fornite un Database" e quindi "Creare degli indici" da sfruttare tramite Windsurf. Finita questa procedura, è stata creata quella che nel paragrafo 4.3 abbiamo chiamato **WLS_EXTRA**.

A questo punto, bisogna configurare correttamente il file che permette all'utente di eseguire il software Windsurf assieme a tutte le librerie ad esso necessarie per il suo corretto funzionamento. Nel nostro caso, lavorando su un calcolatore con sistema operativo Microsoft Windows, si è utilizzato il file "windsurf_automated.bat" (file batch[1]), invece per i computer basati su distribuzioni GNU-Linux si dovrà utilizzare il file "windsurf_automated.sh".

Questi file contengono degli script che una volta lanciati, offrono un modo guidato per sfruttare le funzioni dal programma offerte, come appunto la possibilità di creare una risorsa multimediale. È possibile modificare lo script per far eseguire il comando "*HELP*" che offre una guida su quali comandi sono messi a disposizione dalla libreria Windsurf.

Il file BAT (quindi il file per Windows) è strutturato come segue:

[1] `http://it.wikipedia.org/wiki/File_batch`

- La prima parte contiene una piccola guida che spiega come configurare ed utilizzare il file.

```
:---------------------MINI-GUIDE-------::---> SANTOMARCO
::put windsurf.jar in a EMPTY DIRECTORY
::put the windsurf_lib dir (or where you want, but inside this dir)
          next to the jar
::create and configurate the windsurf project in a file with the
          prefixed structure ( input config.file)
::change path on this file
::change command
:: HINT: execute jar file with help command to see what kind of
          function it offers!!
:: way to execute
::java -classpath <classpth> >className> <command> <parameter/s>
```

- La seconda parte invece contiene le direttive che devono essere invocate per il corretto caricamento della libreria.

```
TITLE WINDSURF
ECHO We are ready
CLS

SET dir="C:\Users\Marco\...\risorse_mm\rs_es_1"

SET className="it.unimo....windsurf.WindsurfShell"

 @echo off
 setLocal EnableDelayedExpansion
 set classpth="
 for /R %dir% %%a in (*.jar) do (
   set classpth=!classpth!;%%a
 )
 set classpth=!classpth!"
 echo !classpth!

CLS
```

```
ECHO WINDSURF AUTOMATED BAT FILE
```

- Per esempio, se avessimo configurato il path in modo corretto ed inserito in coda al file bat il comando:

```
java -classpath %classpth% %className% help
```

avremmo avuto in output il suggerimento di come possa essere utilizzata la libreria:

```
WINDSURF AUTOMATED BAT FILE
 -
 This is a WINDSURF command line class:
    WindSurfShell command [option list]
 --- Commands ---
 help             display this text
 createIndex propertiesFile numberOfImage
                  Index the image directory according the configuration
                  set in the propertiesFile
                  numberOfImage to add , 0 if you want all of them
                  and also remove all the existing
 queryDBImage propertiesFile imageID
                  given an existing (in the DB) image ID
                  return the 4 most similar images
 queryImageDir propertiesFile imgFolder
                  for each image in the given image folder (imgFolder)
                  return the 4 most similar images
 queryAllImageDir propertiesFile imgFolder limit
                  for each image in the given image folder (imgFolder)
                  return the n (limit)  most similar images.
                                   Limit = 0 means NO_LIMIT
 queryImage propertiesFile imageFullPathName [limit]
                  return the N (limit) most similar images
                  Limit default is 4
                                   Limit = 0 means NO_LIMIT
```

Quindi, nel nostro caso per popolare il BD e quindi per la creazione della risorsa multimediale di tipo Windsurf, si inserirà nel file .bat/.sh il comando:

```
createIndex  propertiesFile  numberOfImage
```

Che farà leggere alla libreria dalla directory corrente il file di configurazione chiamato "propertiesFile", sapendo che "numberOfImage" è il numero delle immagini su cui calcolare l'indice (0 prende l'intero contenuto della directory contenente i file multimediali, come spiegato dal comando HELP).

Una volta eseguito lo script, Windsurf mostra a video l'esito dell'operazione.

```
C:/Users/Marco/.../rs_es_1/images_db/
Processo il file :art1026.jpg
...
Fineshed Feature Extraction. Error count = 0
Load DLL for WindsurfRegMtreeWrapper...
Load DLL for WindsurfImgMtreeWrapper...
Finished Index Creation.
DataBase Created
FILE-----C:\Users\Marco\...\rs_es_1\rs_es_1.csv
read
 1:CF,ETA,STILE,DESCRIPTION,IMG_NAME,DATA
 2:String,String,String,String,String,Date
 ...
 Loaded info from CVS
```

In questo modo è stata correttamente creata una risorsa multimediale di tipo Windsurf, che è possibile utilizzare all'interno di MOMIS.

6.3 Esecuzione di MOMIS-Windsurf

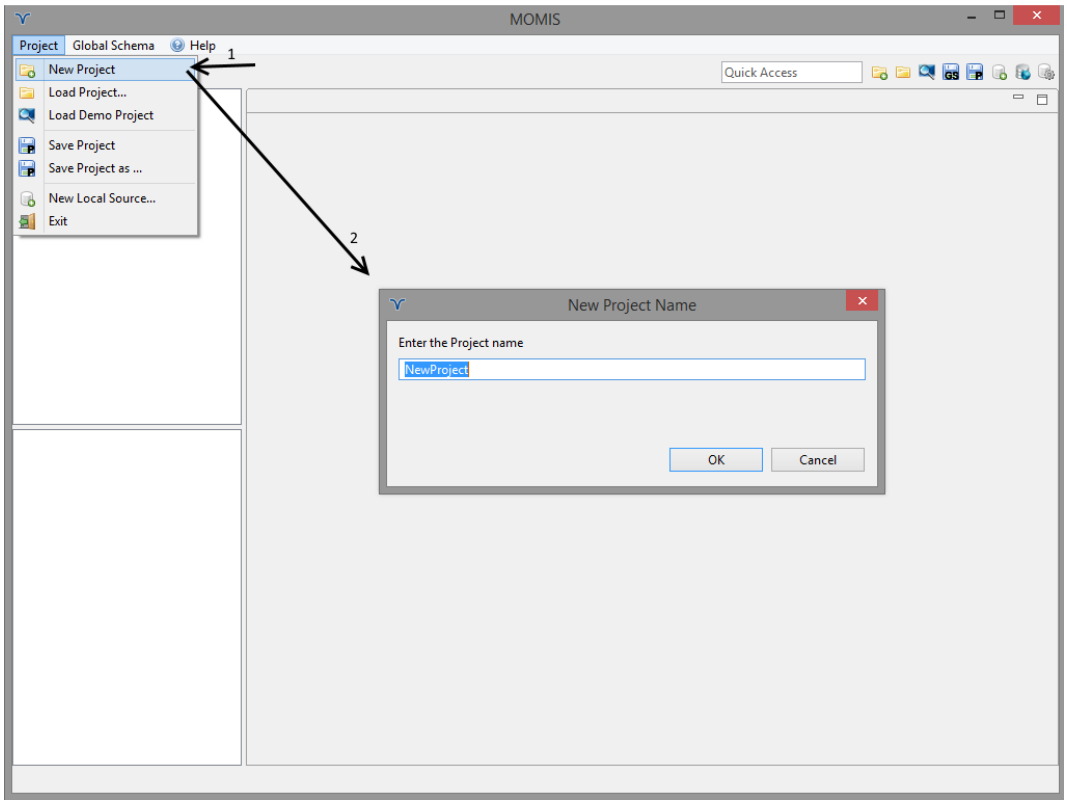
La finestra di avvio dell'applicazione MOMIS-Windsurf è mostrata in figura 6.3.

Figura 6.3: Esecuzione di MOMIS

6.3.1 Creazione in MOMIS di un nuovo progetto

Si vada nel menù “Project”, poi “New Project”, si inserisca il nome del Progetto

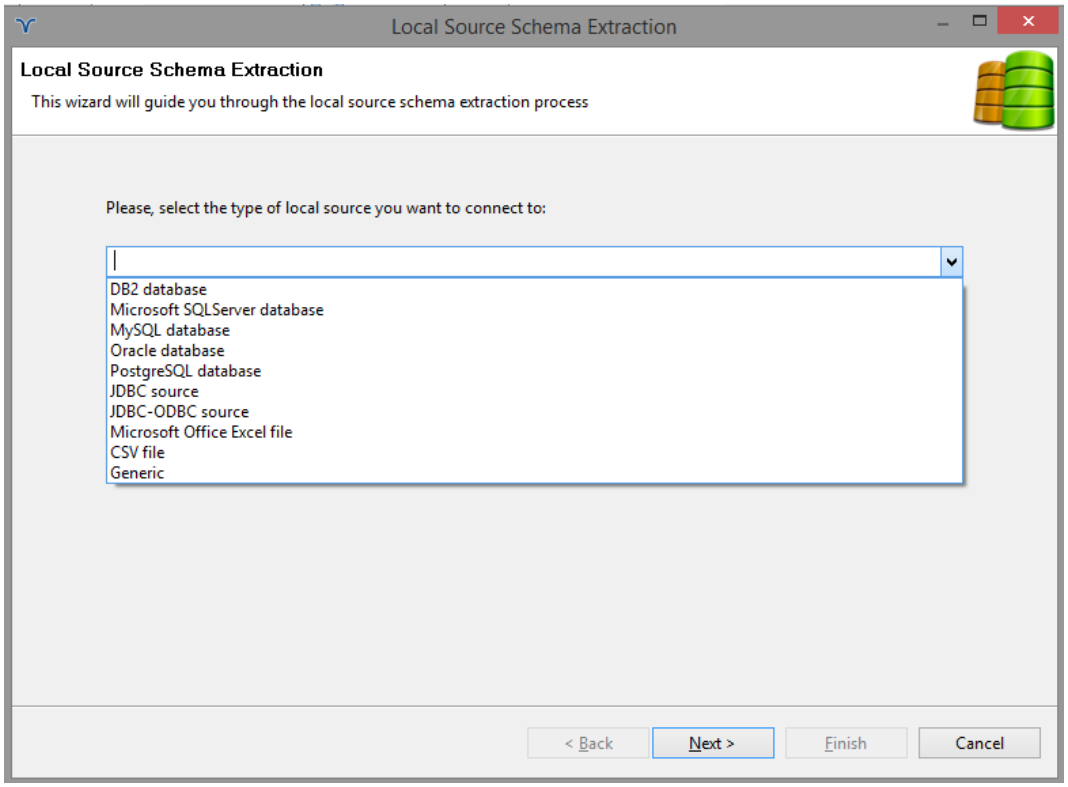
Figura 6.4: New Project



6.3.2 Import di Sorgenti Tradizionali

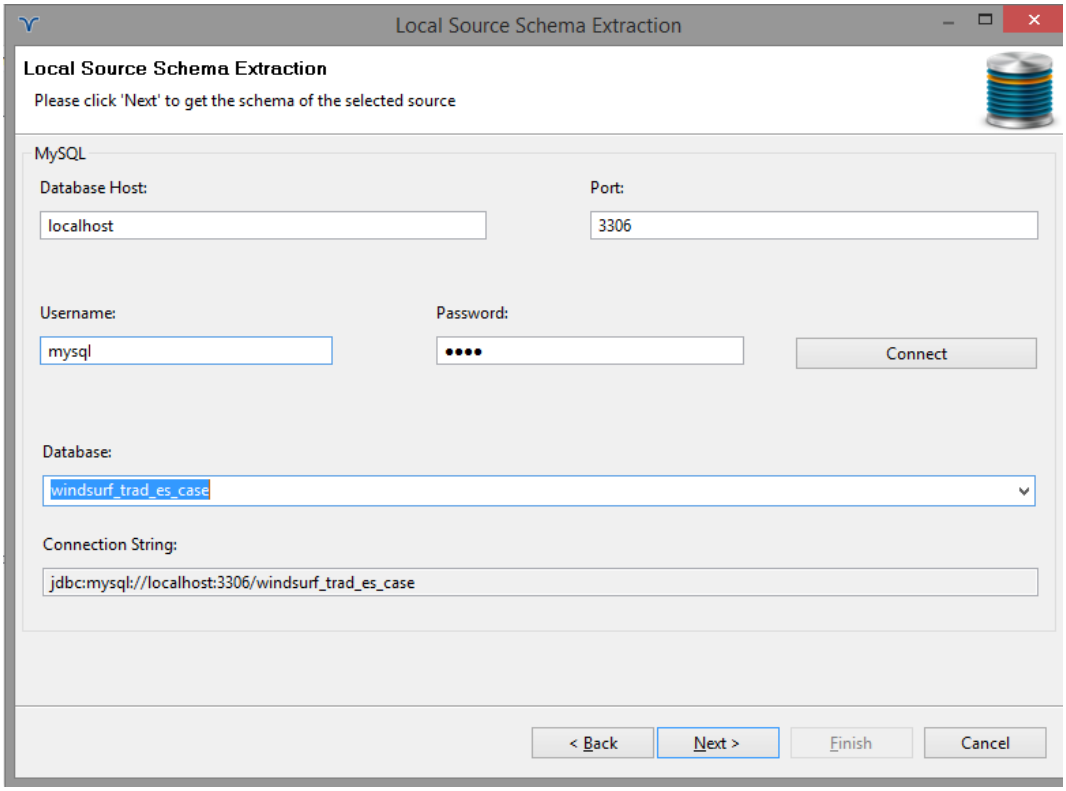
MOMIS suggerisce di importare le sorgenti locali per l’estrazione del loro schema. Selezioniamo il tipo dal menù a tendina e inseriamo il nome con il quale vogliamo importare la risorsa. In questo esempio importeremo una risorsa di tipo MySQL (figura 6.5).

Figura 6.5: Import Source - 1



Nella pagina di configurazione successiva (presente in figura 6.6) ci vengono chiesti i campi relativi alla connessione; configuriamo e siamo in grado di raggiungere la prossima schermata.

Figura 6.6: Import Source - 2



Arrivati nella schermata in figura 6.8, dovremo selezionare le colonne della risorsa da importare in MOMIS, inoltre il software offre la possibilità di visualizzare una preview dei dati contenuti nelle colonne selezionate (Figura 6.7). Clicchiamo su "Finish" per concludere l'import della risorsa.

Figura 6.7: Import Source - Data Preview

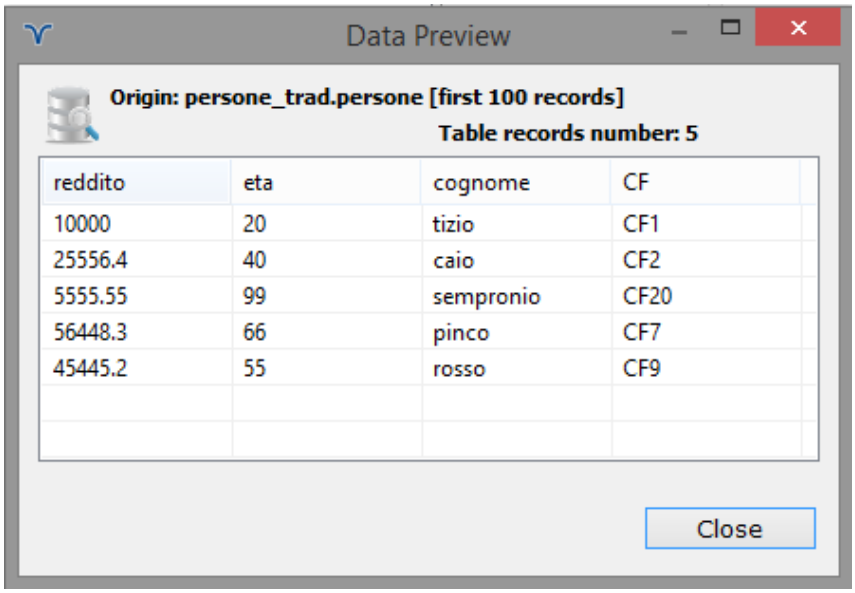
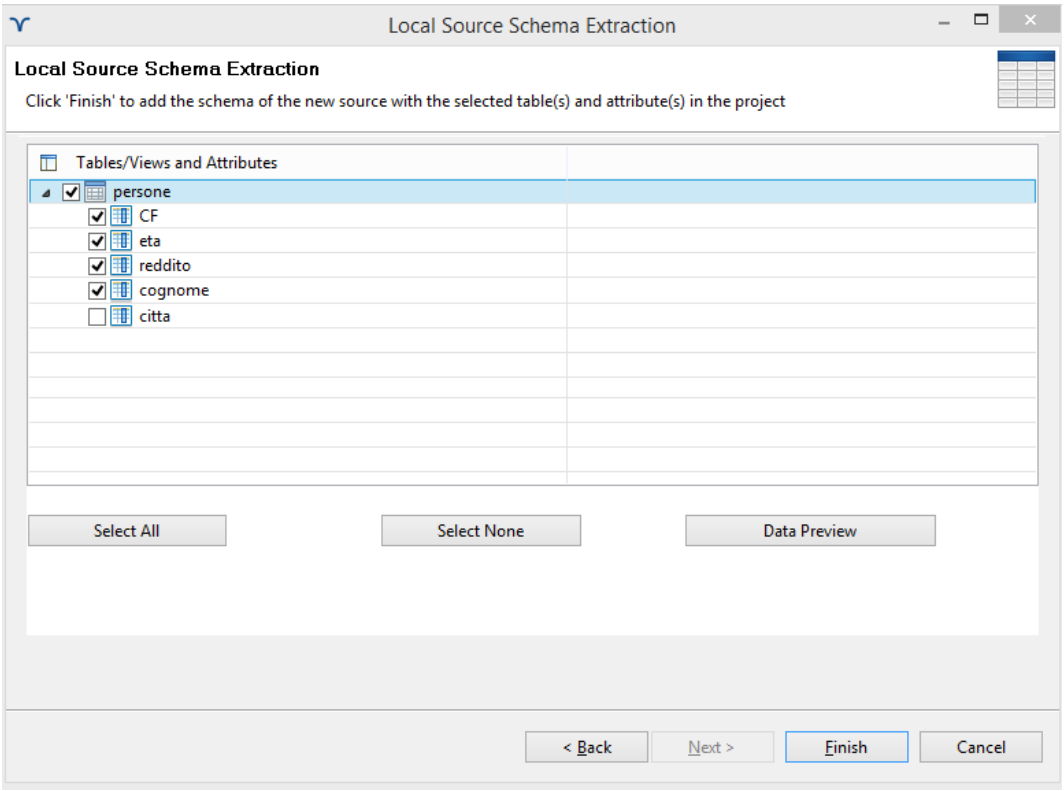
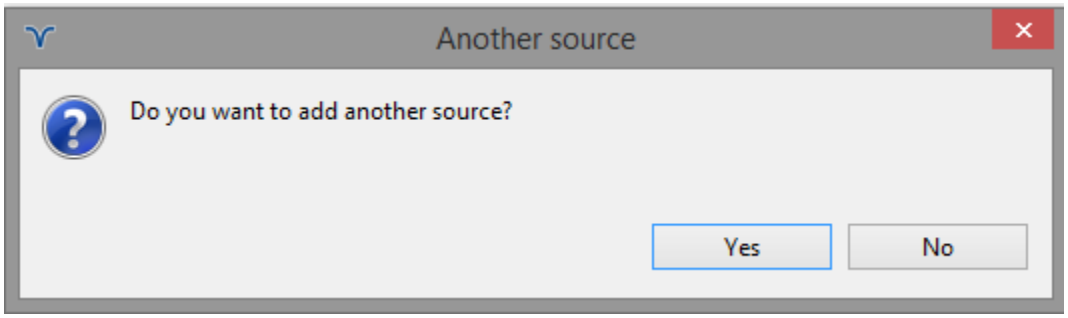


Figura 6.8: Import Source - 3



Al termine di questo wizard ci viene chiesto di importare altre risorse, Nel nostro caso selezioniamo “SI” ed importiamo anche una risorsa di tipo multimediale.

Figura 6.9: Import Source - 5



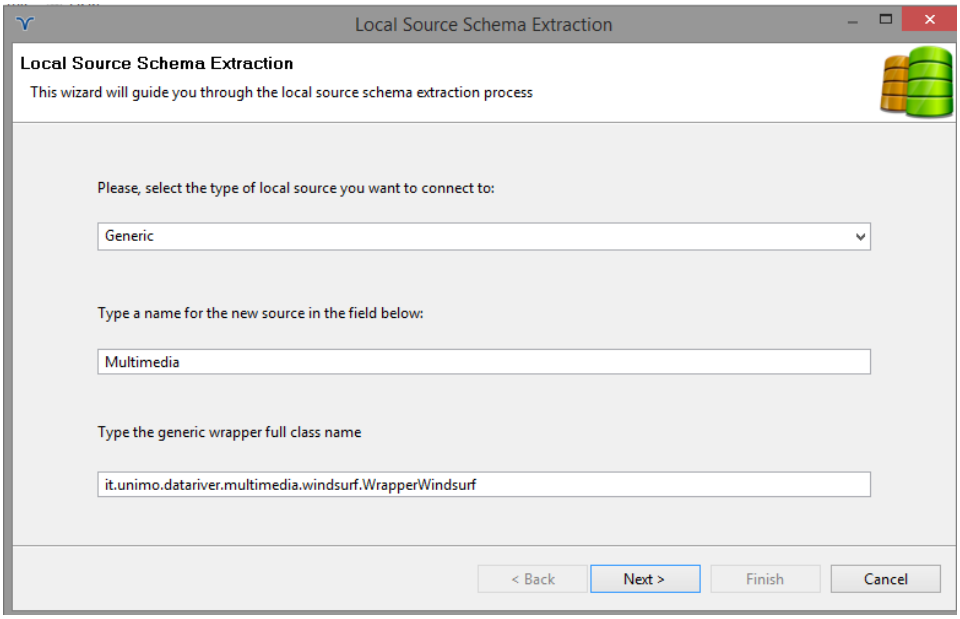
6.3.3 Import della Sorgente Multimediale (sorgente Windsurf)

Per importare una risorsa multimediale dovremo scegliere “Generic” (in futuro potremo importare direttamente una risorsa Windsurf tramite un wizard creato ad-hoc, ma per l’attuale implementazione di MOMIS si è voluto lasciar slegati i due progetti). Inseriamo il nome della risorsa (definito e scelto dall’utente) ed inseriamo

il "*Full Class Name*" del wrapper (ovvero il nome intero della classe, comprendete il percorso dei package in cui essa è contenuta).

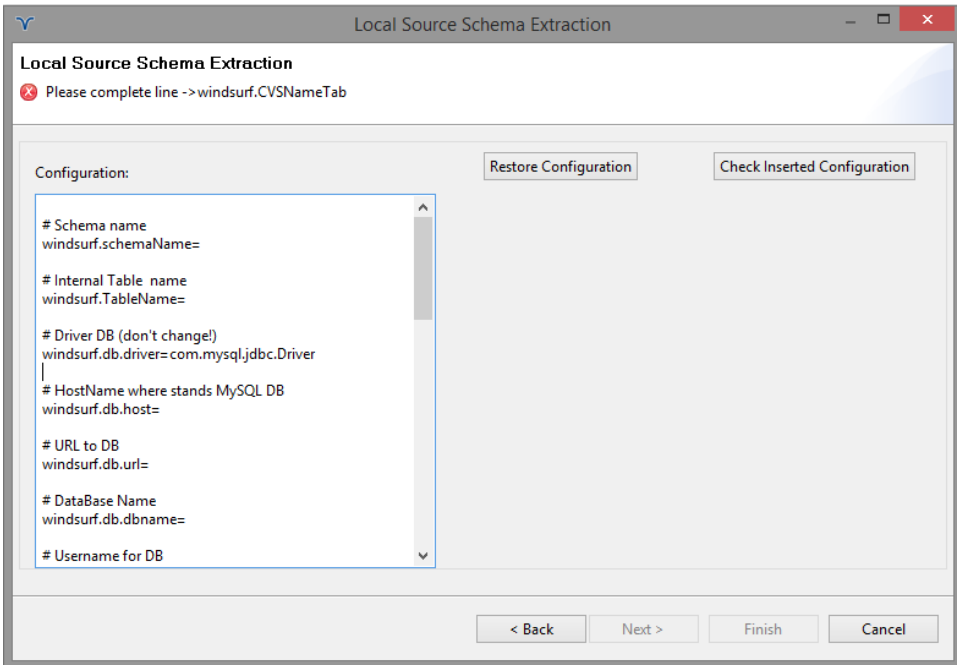
Per disaccoppiare MOMIS da Windsurf abbiamo fatto si che questo wrapper venga inserito all'interno delle librerie esterne di MOMIS: in particolare la libreria JAR contenente il wrapper dovrà esser inserita all'interno della cartella "lib" prima che Windsurf venga invocato. Questa directory viene automaticamente creata all'interno del Workspace di MOMIS (deciso dall'utente all'atto del primo avvio del software).

Figura 6.10: Import Source MM - 1



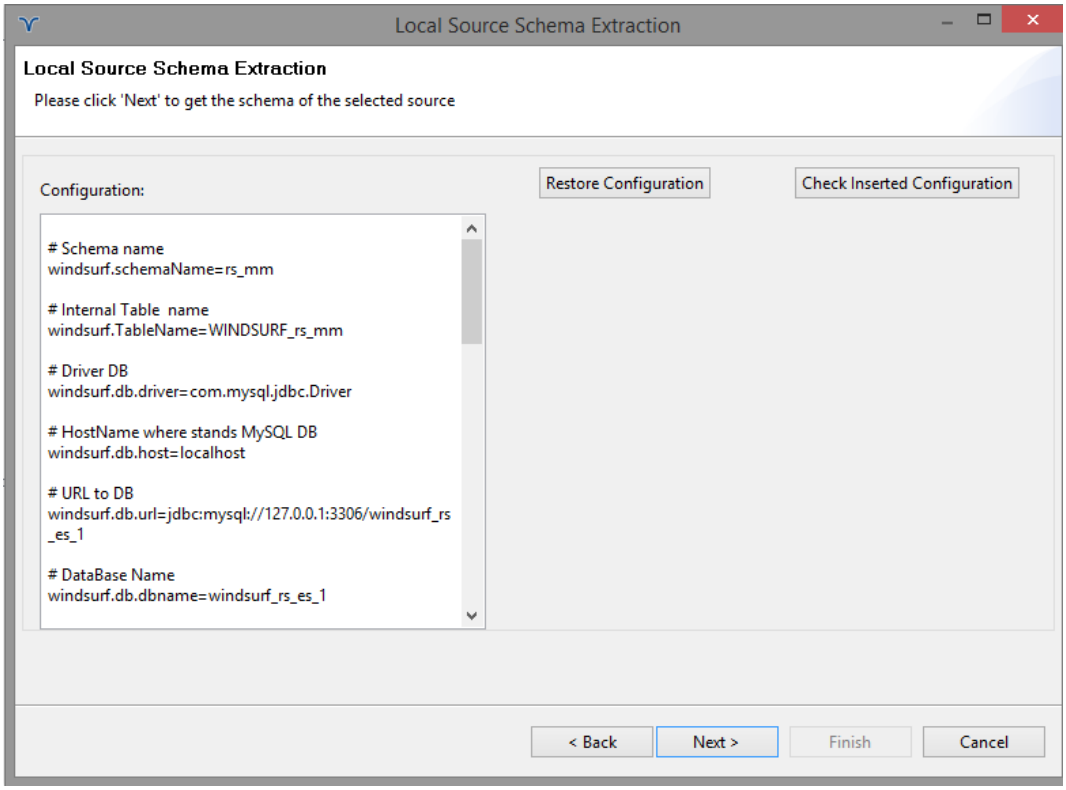
Nella schermata raggiunta tramite il pulsante "Next", il wizard di MOMIS mostra a video la descrizione di ciò che il wrapper specificato nella pagina precedente (quindi nel nostro caso quello di Windsurf) vuole per essere configurato.

Figura 6.11: Import Source MM - 2



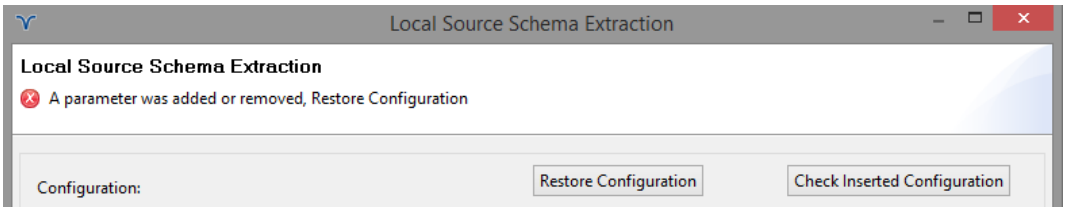
Una volta inseriti tutti i valori richiesti dal wrapper (di cui molti sono già stati spiegati all'atto della configurazione della risorsa al capitolo 6.2), si prema sul pulsante di "Check Inserted Configuration", che invoca una funzione che analizza la configurazione inserita dall'utente. In caso che il controllo abbia dato un responso positivo (quindi configurazione completa) sarà possibile selezionare il pulsante "Next", in caso contrario verrà mostrato a video un messaggio d'errore esaustivo sul problema occorso.

Figura 6.12: Import Source MM - 3



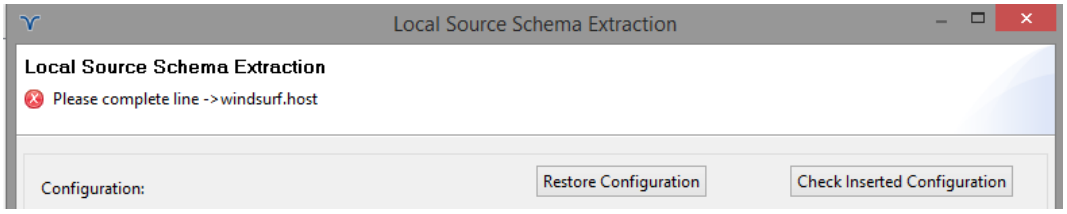
Per esempio, la figura 6.13 mostra il caso in cui un parametro è stato completamente cancellato.

Figura 6.13: Import Source MM - Errore 1



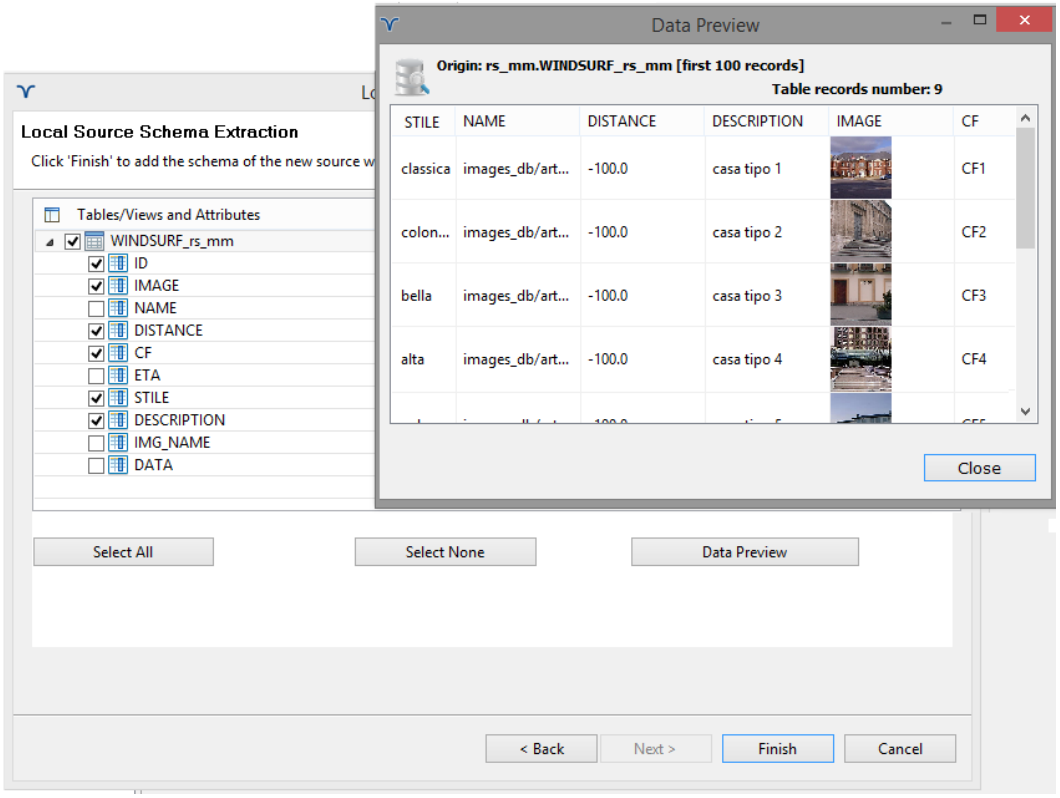
Invece in figura 6.14 è mostrato il caso in cui un parametro non è stato completato (non è stato inserito il valore).

Figura 6.14: Import Source MM - Errore 2



Infine selezioniamo le colonne di nostro interesse. Possiamo notare che, come precedentemente spiegato, questa risorsa verrà vista da MOMIS come una unica tabella. Anche in questo caso è possibile visualizzare l'anteprima di quanto selezionato, compresi gli elementi multimediali.

Figura 6.15: Import Source MM - 4

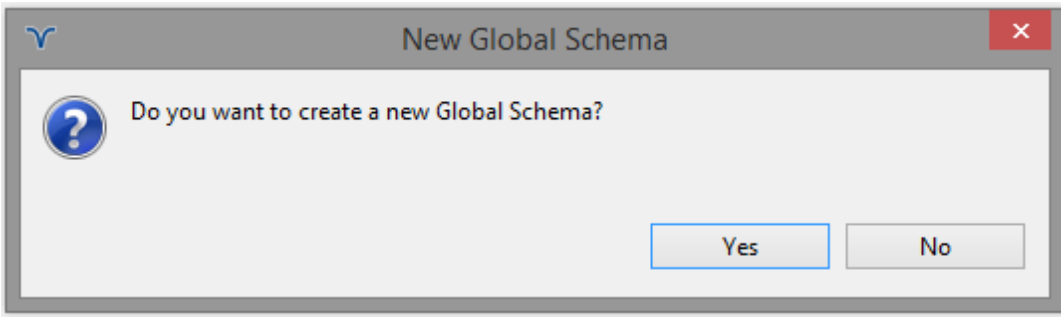


Premendo il pulsante "Finish" abbiamo concluso l'import della risorsa multimediale all'interno di un progetto MOMIS.

6.3.4 Creazione di un nuovo Global Schema

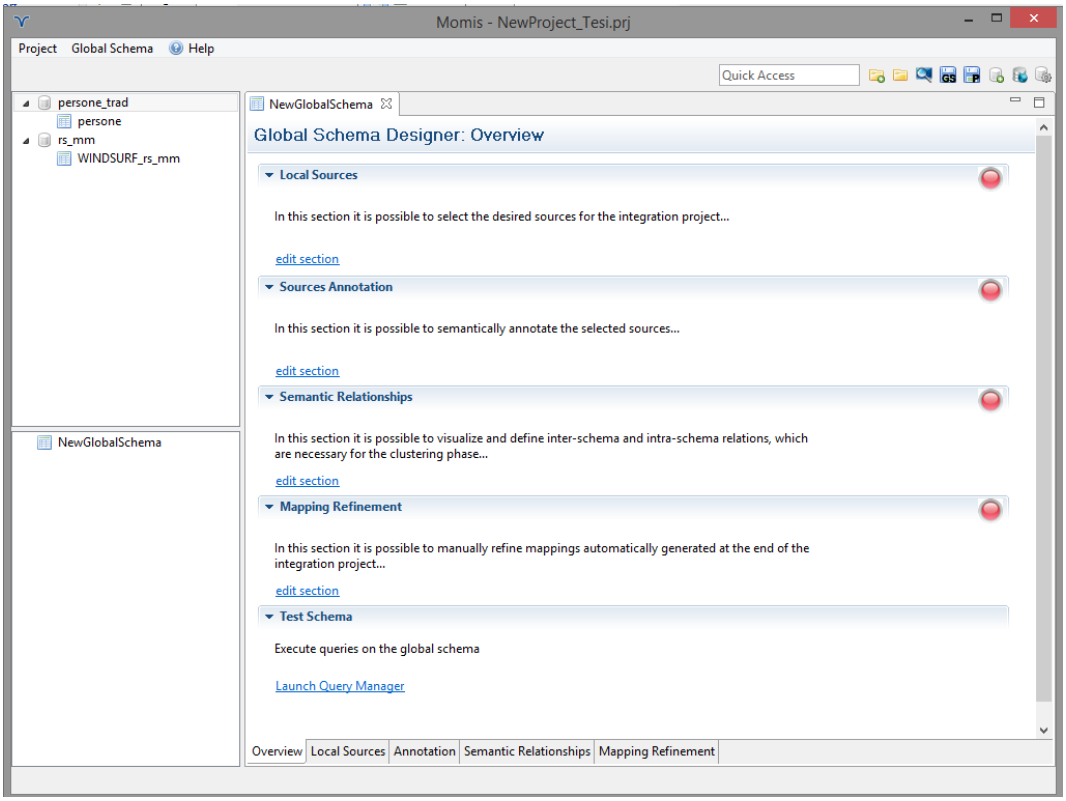
Dopo aver finito di importare le sorgenti di nostro interesse nel progetto di MOMIS, ci verrà chiesto di creare un nuovo Global Schema.

Figura 6.16: Global Schema - 1



Impostiamo il nome e quindi ci troviamo davanti alla schermata (guidata) per la definizione di questo Global Schema:

Figura 6.17: Global Schema - 2



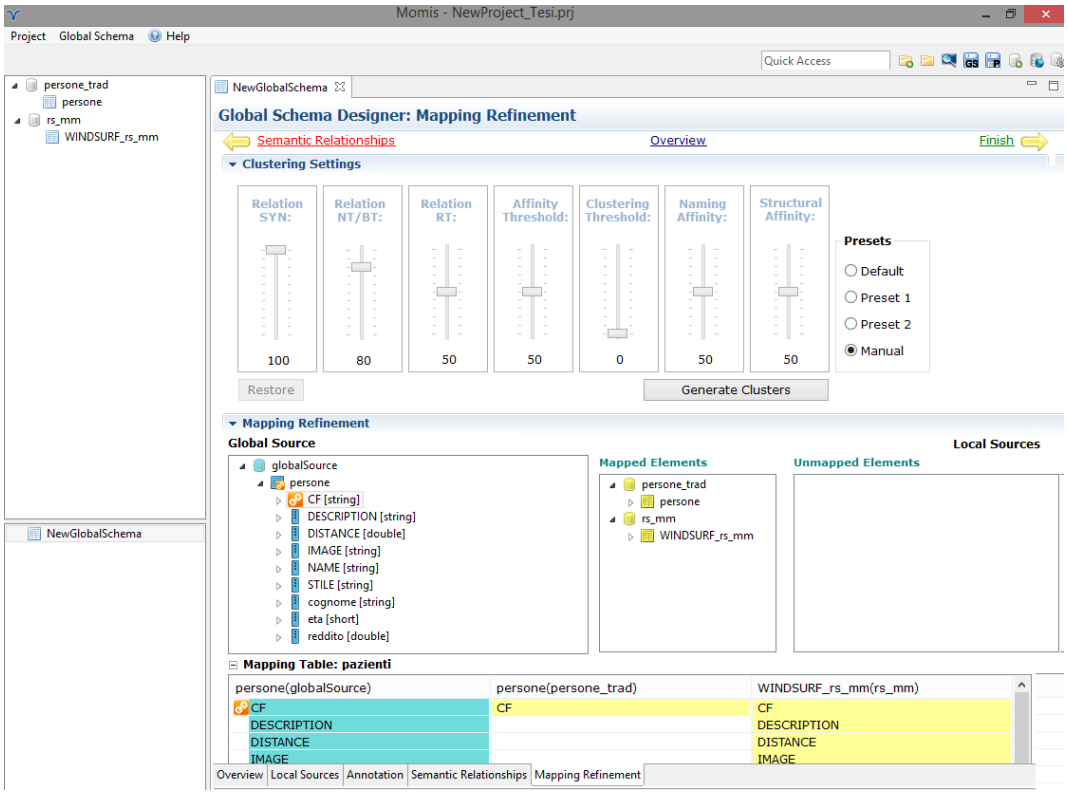
Passi per la definizione del GS

Questa fase segue precisamente tutto ciò che è stato varie volte definito nei documenti che analizzano il funzionamento di MOMIS e del Global Schema, quindi si eviterà di analizzarla.

Join attribute e Resolution function

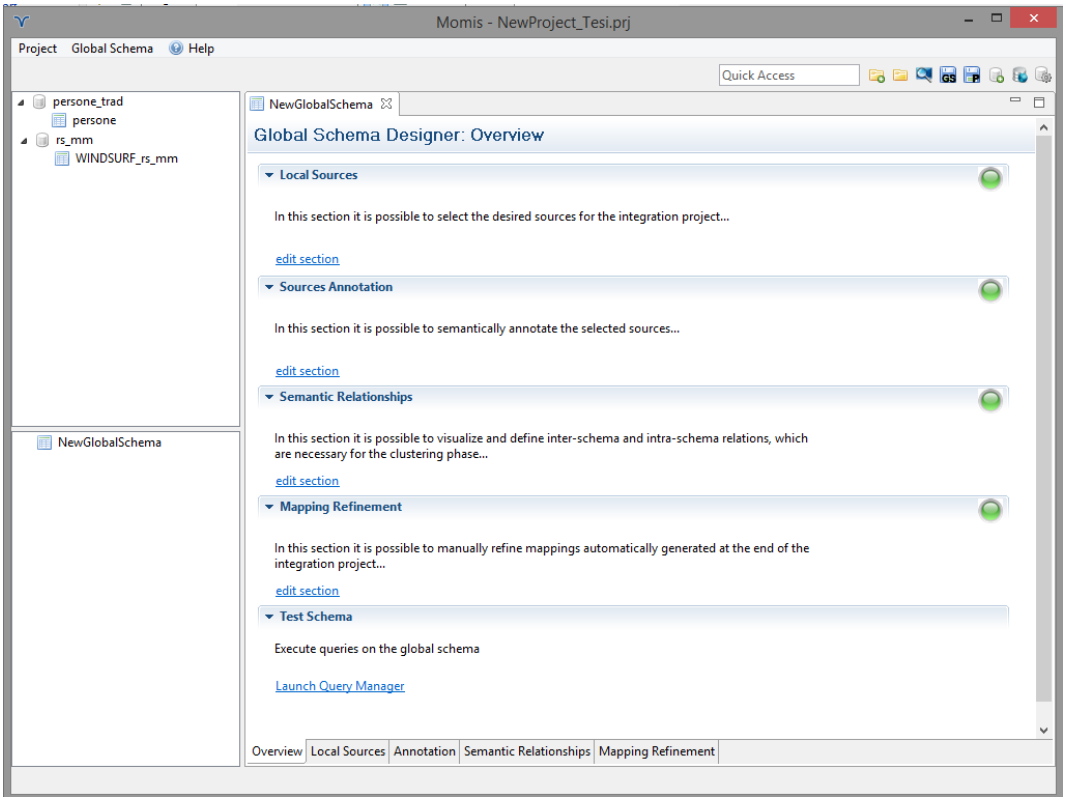
Anche questa fase non differisce da quelle standard di MOMIS, ma sottolineiamo come si necessiti di specificare un elemento sul quale MOMIS si dovrà basare per effettuare il join atto ad unificare le risorse. Si può visualizzare la Global Class creata, e quindi si possono anche modificare gli attributi e le funzioni di risoluzione ad essi legati.

Figura 6.18: Global Schema - 3



In questo modo abbiamo correttamente definito il Global Schema e siamo pronti per eseguire le query sulla nostra sorgente globale integrata. Il sistema mostra visivamente lo stato di completamento del Global Schema tramite sfere di colore verde, quindi nel nostro caso risulta essere completamente e correttamente definito.

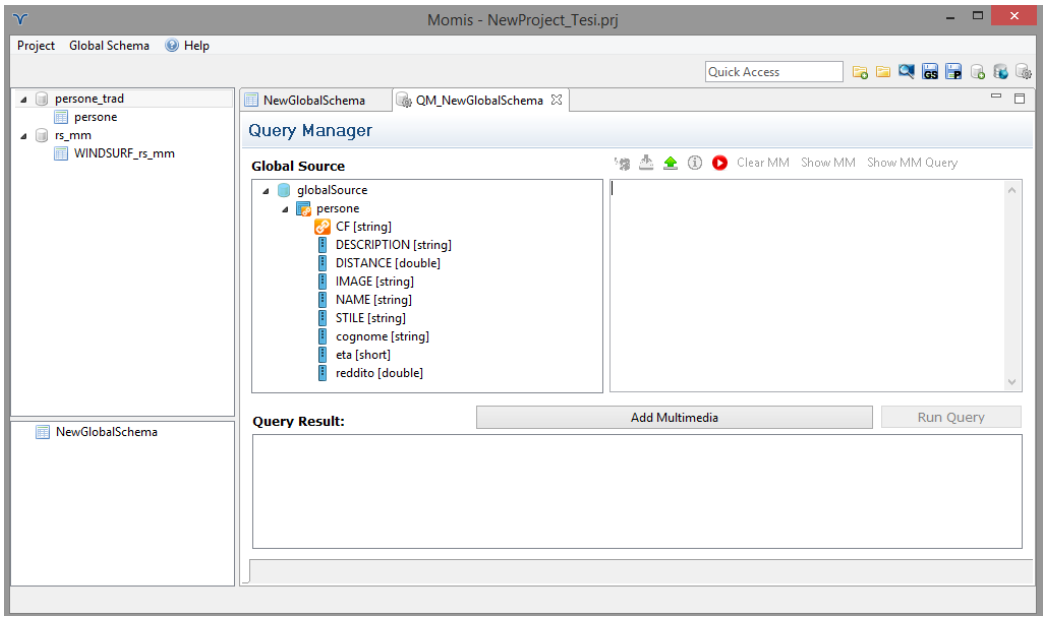
Figura 6.19: Global Schema Correttamente Configurato



6.4 Interrogazioni Multimediali sul Global Schema

Dopo che l'utente ha correttamente definito un Global Schema, possono essere effettuate query su di esso. L'interfaccia offerta in caso sia presente nel GS (almeno) una risorsa Multimediale è mostrata in figura 6.20.

Figura 6.20: Global Query Interface



6.4.1 Formulazione della Query Globale

L'utente scriverà una query globale, ricordandoci che la sorgente multimediale avrà anche dei campi tradizionali (es. colonne contenenti nomi, ID, numeri, ecc) sulle quali possono esser imposti vincoli "tradizionali". Per aggiungere alla query un elemento "Multimediale" da comparare con quelli presenti nella sorgente Windsurf, bisogna utilizzare l'apposito pulsante "Add Multimedia"; con il click su questo pulsante si aprirà un file chooser che permette di selezionare l'elemento multimediale di interesse. L'immagine inserita sarà considerata l'immagine di riferimento nei predicati multimediali inseriti nella query globale, che ricordiamo possono essere:

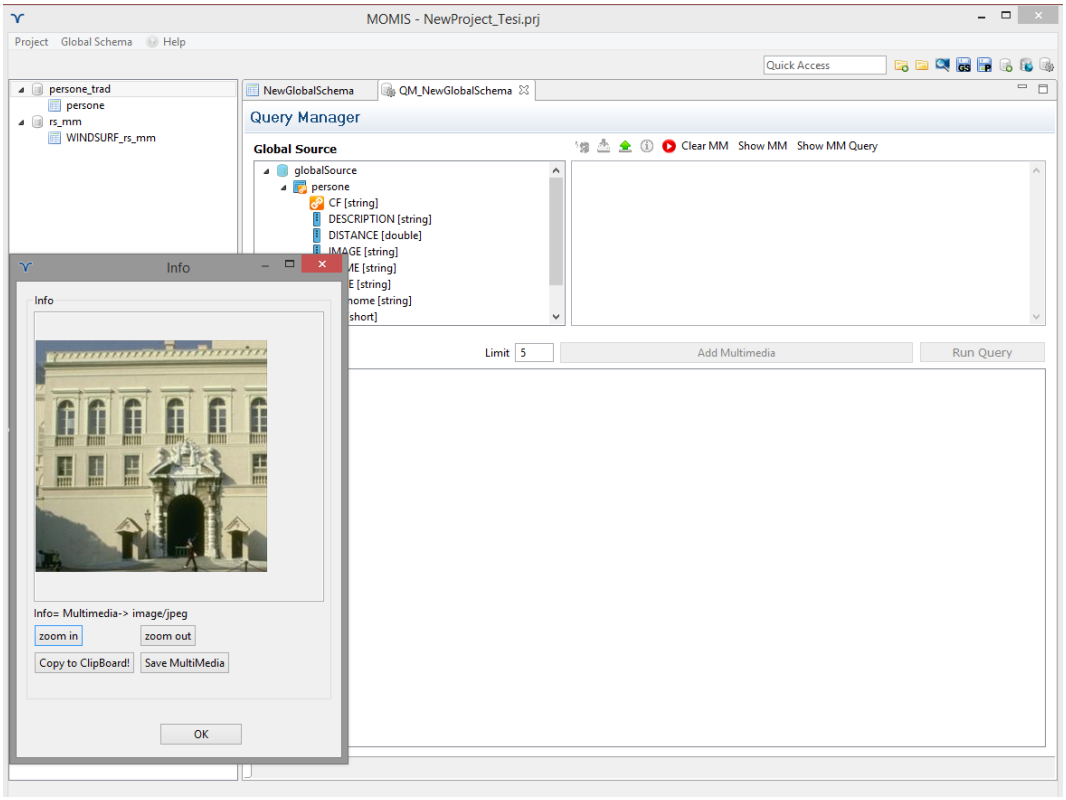
- RANGE: dammi gli elementi con una similarità X o compresi fra due valori rispetto all'elemento che è passato in input
- Top-K: dammi i K elementi più simili a quello che ti ho inviato

È possibile visualizzare l'elemento multimediale inserito nella query tramite il bottone "Show MM" ed è possibile rimuoverlo tramite "Clear MM". Si noti che dopo averlo rimosso, non abbiamo più alcuna immagine di riferimento per la query e quindi i predicati multimediali eventualmente aggiunti non avranno alcun effetto, ovvero risulteranno sempre veri.

Il predicato Top-k dovrà essere manualmente inserito nel campo "Limit" che diventa visibile solamente dopo aver aggiunto l'elemento multimediale alla query. Di default è impostato a 5 (Top-$K = 5$) ma può esser settato anche con il valore 0 che significa NO LIMIT.

La parte multimediale della query è stata "nascosta" alla vista dell'utente ma è possibile visualizzarla tramite il bottone "*Show Multimedia Query*". Come mostrato in Figura 6.21, vengono visualizzate le informazioni legate alla parte multimediale, in particolare si nota che tutti gli elementi saranno caratterizzati dall'essere racchiusi dalla keyword " JSON FOR MULTIMEDIA..."

Figura 6.21: Global Query Interface - Show MM



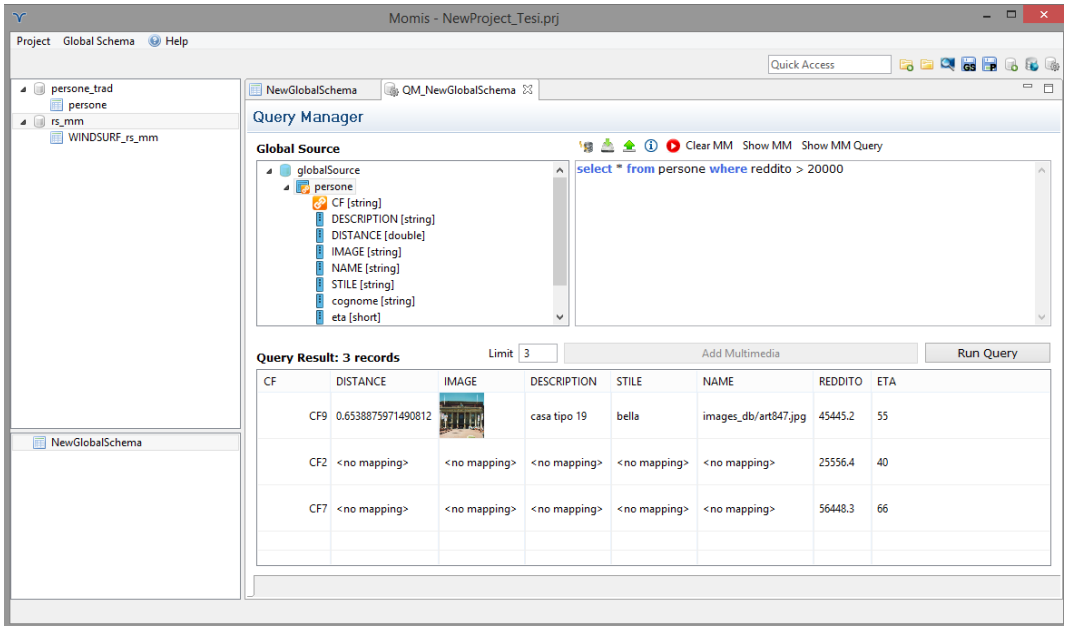
Nel seguito si mostreranno due semplici esempi di query multimediale, la prima di tipo **Top-***K* e la seconda di **Range**.

Nella prima query è stata aggiunta l'immagine di riferimento ed una specifica Top-*k*, con K=3

```
SELECT *
FROM persone
WHERE reddito > 20000
```

Di cui vediamo i risultati in figura:

Figura 6.22: Global Query Top-K



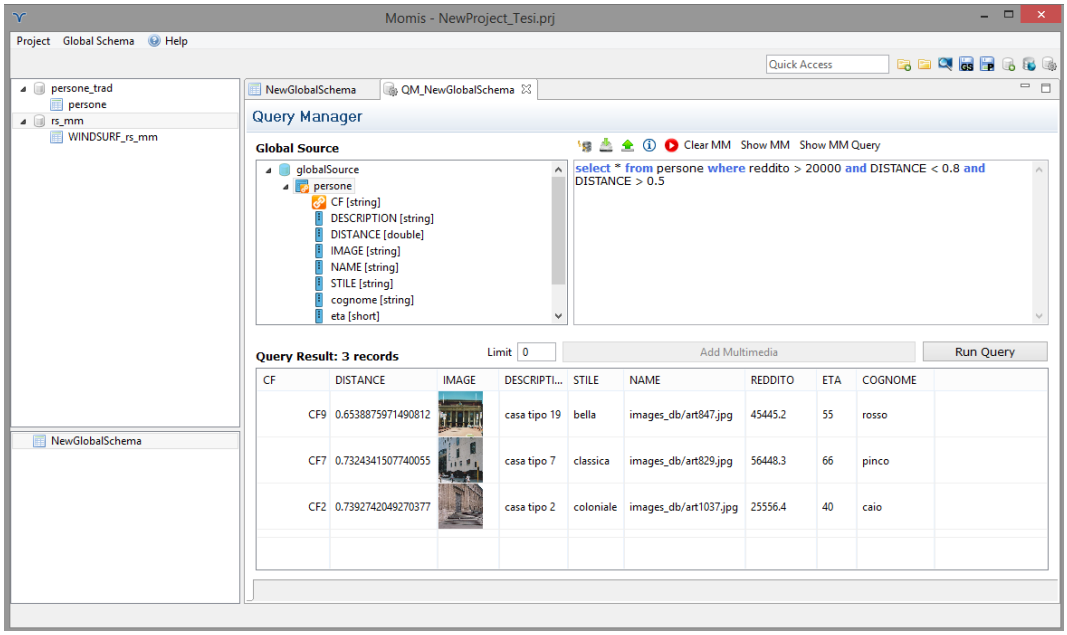
Possiamo notare il problema derivante dalla presenza contemporanea di un predicato Top-k con uno tradizionale, analizzato in dettaglio nel capitolo 3.3.2.

Nella seconda query, il predicato di RANGE invece dovrà essere manualmente scritto all'interno della query dall'utente; in particolare dovrà avere come oggetto il campo DISTANCE, sempre presente nella risorsa multimediale.

```
SELECT *
FROM persone
WHERE reddito > 20000
AND DISTANCE < 0.8 AND DISTANCE > 0.5
```

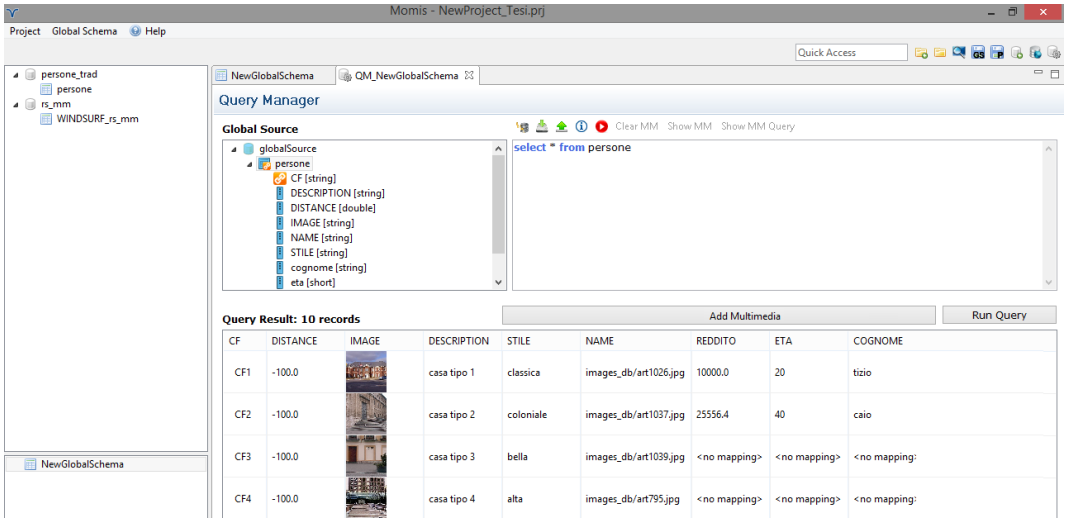
Dobbiamo anche ricordarci di disattivare il predicato Top-k (quindi limit lo imposteremo pari a 0, in quanto di default ha valore 5), ed in questo modo otteniamo i risultati della query di range:

Figura 6.23: Range Global Query



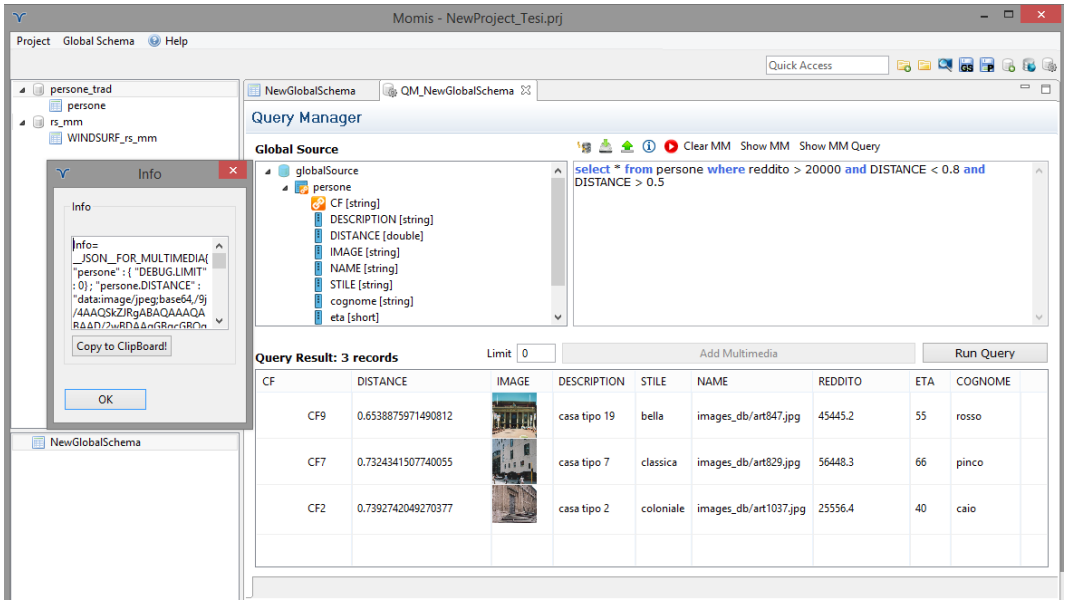
Ricordiamo che nel caso di query *non* multimediali, il campo DISTANCE assumerà il valore non significativo "-100".

Figura 6.24: Query non Multimediale



In qualunque momento l'utente potrà controllare la struttura completa della query sottomessa al QM, che include anche la visualizzazione dell'elemento multimediale sia come immagine vera e propria sia come immagine convertita in stringa.

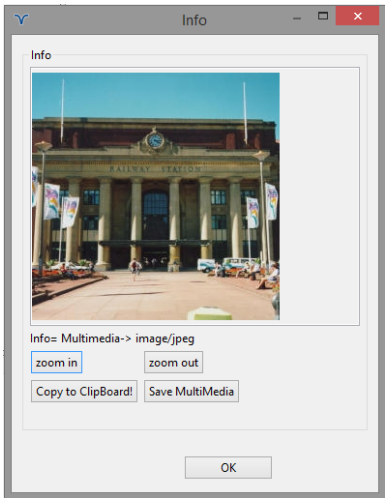
Figura 6.25: Global Query Info



6.4.2 Esecuzione delle Query e Risultati

Una volta che l'utente ha formulato e scritto la query da inviare e la sottomette tramite il pulsante "*Run Query*", essa è eseguita dal Query Manager ed il risultato mostrato a video. Ogni elemento di una qualsiasi colonna che ha la caratteristica di iniziare per determinati caratteri (se risulta essere un elemento multimediale codificato tramite Data URI) verrà convertito e mostrato nella sua forma originale (quindi, se è per esempio un immagine JPEG ed inizia per "..." essa verrà mostrata come una piccola immagine) con la possibilità di analizzarlo nel dettaglio tramite doppio click sull'oggetto.

Figura 6.26: Dialog Info Image



In futuro si potrà aggiungere la possibilità di visualizzare video, GIF animate,

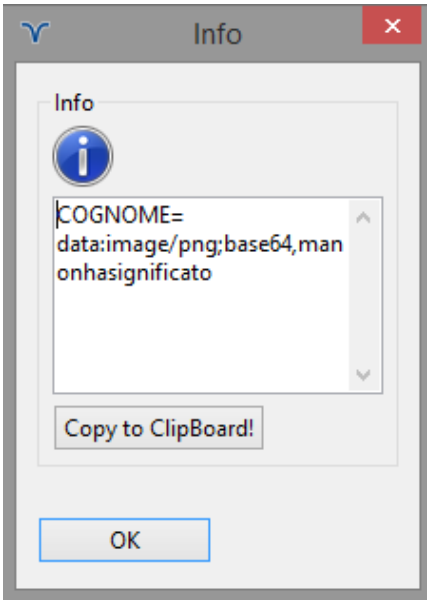
immagini diverse da quelle JPEG, audio, ecc ecc.. dove ogni file verrà interpretato correttamente e mostrato con i giusti comandi (se è un video saranno presenti i comandi di play/stop, per file audio lo stesso, un'immagine necessiterà di salva, zoom-in/out, ecc..).

Se il sistema dovesse fallire nella conversione da stringa Data URI a elemento MM, per esempio sfortuna vuole che la combinazione di caratteri:

```
‘‘...’’
```

sia il contenuto di una colonna di tipo stringa testuale senza significato multimediale, il contenuto originale verrà comunque mostrato all'utente sotto forma di stringa. Questa soluzione permette di non aver quindi nessun problema di mancanza di risultati visualizzati a causa di cattiva interpretazione del dato. In questo modo i dati saranno sempre visualizzabili e utilizzabili.

Figura 6.27: Dialog Info



Capitolo 7

Framework MOMIS-Windsurf: Implementazione

In questa sezione verranno esposti i dettagli implementativi per la realizzazione del Framework MOMIS-Windsurf, svolti durante la tesi.

L'implementazione realizzata può essere schematizzata nella seguente struttura:

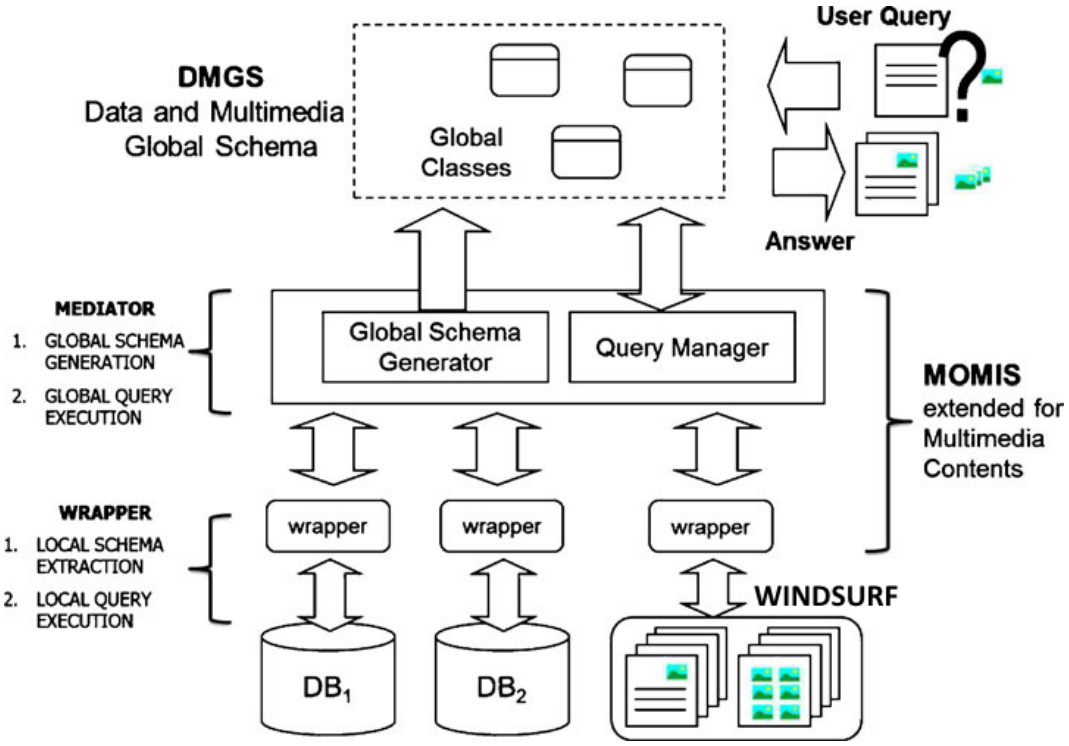
- Modifiche a Windsurf
- Modifiche a MOMIS
- Creazione wrapper Windsurf per MOMIS

Ognuna di queste classi di modifiche verrà analizzata nel dettaglio.

Per maggiore chiarezza e semplicità, l'architettura del Framework MOMIS-Windsurf riportata in figura 4.6 è qui ripetuta in figura 7.1. Il wrapper di Windsurf deve essere un modulo separato da MOMIS, quindi si è tentato di mantenerli quanto più disaccoppiati possibile.

Prima di iniziare la spiegazione, presentiamo gli strumenti che sono stati utilizzati per la realizzazione del progetto.

Figura 7.1: Framework MOMIS-Windsurf: architettura complessiva



7.1 Strumenti utilizzati

Per la realizzazione del wrapper, necessario per mettere in comunicazione MOMIS con Windsurf, ed una versione funzionante di Windsurf capace di essere "Stand Alone", sono stati usati i seguenti elementi:

- JDK java 1.8 32bit : l'utilizzo della versione solo a 32 bit è dovuta al limite di JAI (java advanced image) che è implementata in hardware solo per macchine a 32bit, in particolare la parte C sulla quale si basa è stata compilata solo per macchine a 32bit.
- Eclipse Kepler (4.3): non è stato possibile usare Eclipse Luna (4.4) in quanto MOMIS non risulta essere compatibile con alcune funzioni non più disponibili nella nuova versione dell'IDE.
- IA-32: l'architettura "Intel 32bit" è necessaria in quanto l'implementazione dell'indice M-Tree utilizzata da Windsurf non è mai stata ne testata ne compilata per macchine a 64bit.
- MOMIS 1.2 : ultima versione disponibile.
- MySQL Server 5.6.
- Visual Studio 2013 e Visual Studio 2015 Tech. Preview

- Repository di codice condiviso su server DataRiver, sfruttando Mercurial, software per controllo di versione distribuito.

7.2 Modifiche a Windsurf

Tecnicamente la libreria Windsurf disponibile all'inizio del mio lavoro di tesi e tirocinio (in modo gratuito sul sito del progetto Windsurf) doveva risultare capace di eseguire determinate operazioni e di essere utilizzata "as is", ma gli sviluppatori non hanno mai reso disponibile nessun progetto per il suo test ed utilizzo, quindi la prima parte del lavoro è stata abbondantemente impiegata per creare un progetto facilmente importabile e utilizzabile.

Per prima cosa sono state compilate per i differenti sistemi operativi le librerie contenenti gli indici M-Tree[1], necessari per l'esecuzione di Windsurf. Esse sono state compilate per sistemi Apple Macintosh, Microsoft Windows (32bit), Linux 32 e 64bit.

La prima modifica introdotta in Windsurf è legata appunto al fatto che la corretta libreria per gli indici M-Tree venga caricata in modo completamente automatico all'atto dell'inizializzazione del componente su un qualunque calcolatore. In particolare è stata modificata la funzione per il caricamento della DLL presente all'interno della classe "WindsurfImgMtreeWrapper", contenuta nel package "IndexManager".

```
public static void loadIndexMTreeDll() {
    ...
     extra_info_dll=System.getProperty("os.name")+
                    System.getProperty("sun.arch.data.model");

     String fiName = "indexMtree_" + extra_info_dll+".dll";
     String fiFullName = pwd+"src/main/resources/bin/"+fiName;
     File fi= new File(fiFullName);
     if(fi.exists()){
        System.load(fiFullName);
     }else{
        //DLL in jar
        //extract DLL from jar and then load
        }
     }
     System.out.println("Loaded DLL> "+s);
  }
```

[1] http://www-db.deis.unibo.it/Mtree/

Questo metodo è stato impostato STATICO, in modo tale che in tutti i casi in cui è richiesto caricare la libreria legata all'M-Tree, si utilizzi sempre questa funzione, senza introdurre ridondanza di codice.

Possiamo notare che nel caso in cui questa funzione venga invocata da Windsurf utilizzato come un modulo esterno/stand alone, quindi avente la DLL inserita nello stesso pacchetto, per il corretto funzionamento, la libreria M-Tree deve essere estratta e salvata in un file temporaneo (operazione ora fatta in automatico), per poi successivamente essere caricata in memoria.

Dopo aver fatto caricare al sistema la DLL corretta (il codice rileva le informazioni sulla macchina su cui esso è eseguito, quindi in relazione a ciò carica la libreria compilata ad hoc), verranno eseguite le altre operazioni richieste.

Le altre modifiche che sono state apportate sul codice di Windsurf sono legate alla gestione e all'invocazione delle funzioni. È stata messa a disposizione dell'utilizzatore una shell che permette di invocare le varie funzioni della libreria.

La classe WindsurfShell è stata inserita assieme agli altri file relativi al wrapper all'interno del package "*it.unimo.datariver.multimedia*", in modo da mantenere separato questo "spazio" rispetto a quello legato strettamente alla logica di Windsurf. Si è usato questo path in quanto si tenta di mantenere coerenza con la logica dei nomi di packaging già presenti nel progetto MOMIS.

Analizzando la classe "WindsurfShell", possiamo notare la presenza di un oggetto che astrae la gestione delle feature degli elementi multimediali di Windsurf sul database (WrapperFeatureWindsurfMySQL).

```
public class WindsurfShell {
   // fields
   private String module_path;
   private Properties properties ;
   private WrapperFeatureWindsurfMySQL wf;
```

Sono presenti in questa classe due metodi per effettuare l'inizializzazione della shell. Il primo necessita di un file di configurazione dotato di struttura:

```
<PARAMETRO>=<valore>
```

Ed il secondo, invece, viene invocato passandogli una variabile contenente le proprietà a Windsurf necessarie.

```
   /**
    * Init the shell from configuration file
    * @param propertyFile: absolute file name of config file of windsurf
    * @throws Exception
    */
```

```
public void init(String propertyFile) throws Exception{
   ...
}

/**
 * Init the shell from properties
 * @param prop
 * @throws Exception
 */
public void init(Properties prop) throws Exception{
   ...
}
```

In entrambi i casi l'oggetto WrapperFeatureWindsurfMySQL, quindi quello per la gestione sulla base di dati delle feature, è inizializzato:

```
    wf = new WrapperFeatureWindsurfMySQL(properties);
```

Il metodo **Close()** chiude la comunicazione con il database contenente tutti i dettagli sugli elementi multimediali (features) ed effettua il reset del parametro contenente le proprietà.

```
/**
 * close the shell and free all the variables
 */
public void close(){
   if(wf!=null)
   wf.close();
   if(properties!=null)
   properties.clear();
   module_path="";
}
```

Il metodo *static main* della shell, serve ad invocare le funzioni che Windsurf mette a disposizione, in base al parametro che riceve in standard input.

```
public static void main(String args []) throws Exception{
   WindsurfShell obj = new WindsurfShell();
   try {
   ... invocazione metodi su obj ...
   } catch (ArrayIndexOutOfBoundsException e) {
      e.printStackTrace(System.err);
      obj.usage();
   } catch (Exception e) {
      e.printStackTrace(System.err);
```

```
    }
    obj.close();
  }
```

Fra le funzioni che la shell di Windsurf mette a disposizione, c'è quella che permette di popolare il database. Per prima cosa viene effettuata la connessione al database indicato nel file di configurazione, vengono create le tabelle (se l'utente del database fornito ha sufficienti privilegi) e se ciò ha avuto successo, vengono recuperati e processati i file multimediali dal folder (indicato anch'esso nei parametri di config.).

Per ogni file viene invocata la funzione:

```
id = wf.extractImage(imageFileName);
```

Dove **wf** è l'elemento di Windsurf che permette di gestire sia l'estrazione dal file di feature che il loro salvataggio nella base di dati, ed **imageFileName** è il nome (path assoluto) del file multimediale. **ID** è un oggetto di classe ImageID, che contiene l'identificativo dell'immagine estratta; esso risulterà essere >0 se l'estrazione e l'inserimento sono andati a buon fine. In realtà nel caso si tentasse di inserire un immagine già presente sulla base di dati, l'inserimento non avverrà, ma il valore di ID restituito è il valore ID dell'immagine (identica) presente nel Database.

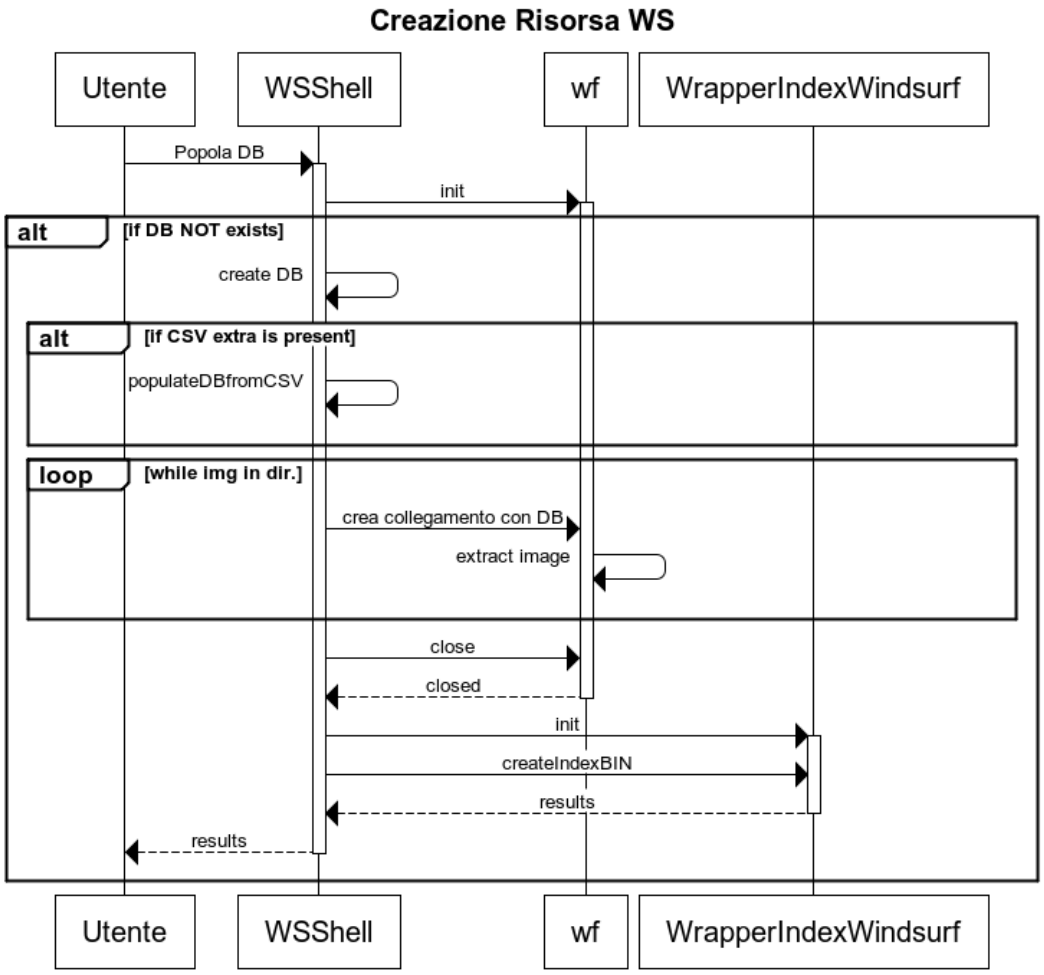
Nelle istruzioni successivamente riportate, verranno creati gli indici delle immagini e delle regioni, la classe **WrapperIndexWindsurf** carica le DLL ed inizializza le variabili necessarie per crearli a salvarli su file.

```
WrapperIndexWindsurf.init(imageIndexName, regionIndexName, wf);

MySQL2Txt script = new MySQL2Txt(regfile);
script.createRegionIndexFile((WindsurfFeatureManagerMySQL) wf.getFM());
script.createImageIndexFile((WindsurfFeatureManagerMySQL) wf.getFM());
```

Il codice realizzato per motivi di spazio non viene riportato nella sua interezza, ma viene offerto sotto forma di pseudo Sequence Diagram per spiegare la logica di funzionamento.

Figura 7.2: Creazione Risorsa MM - Sequence



Se entriamo nel dettaglio della funzione invocata all'atto della creazione della risorsa, quando è presente il file extra (introdotto nella sezione 4.3 e 6.2) contenente valori tradizionali legati agli elementi multimediali, possiamo vedere come essa effettui semplicemente il parsing del file ed esegua l'inserimento di tutti i dati nella base di dati già utilizzata da Windsurf. Anche per descrivere quest'azione è stato riportato un sequence diagram (Figura 7.3)

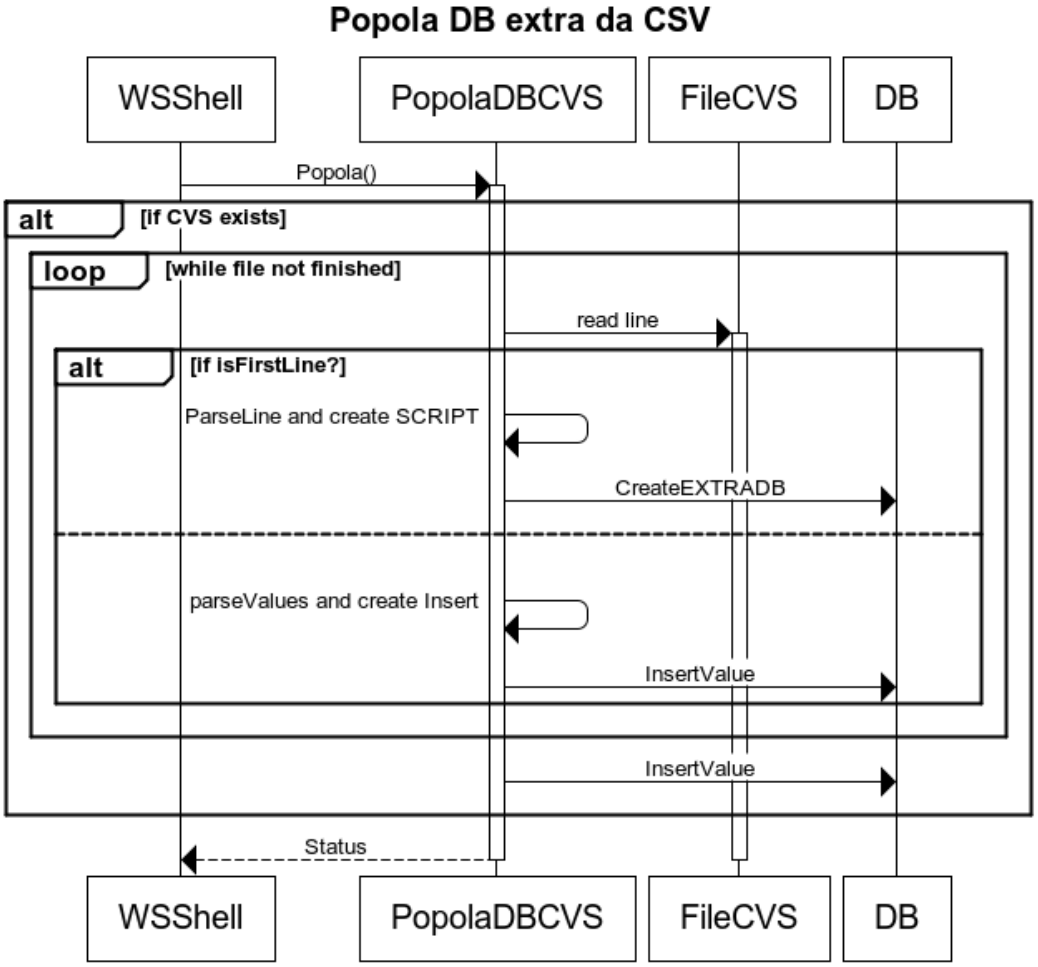
Il file CSV per essere correttamente compreso dal sistema, deve essere strutturato nel seguente modo:

```
<Nomecampo1>,<Nomecampo2>,<Nomecampo3>,<Nomecampo4>,...,<NomecampoN>
<TipoCampo1>,<TipoCampo2>,<TipoCampo3>,<TipoCampo4>,...,<TipoCampoN>
<valore1_1>,<valore1_2>,<valore1_3>,<valore1_4>,...,<valore1_N>
<valore2_1>,<valore2_2>,<valore2_3>,<valore2_4>,...,<valore2_N>
...
```

Un esempio di un file esistente:

```
CF,ETA,ANALISI_A,DESCRIPTION,IMG_NAME,DATA
String,String,String,String,String,Date
CF1,60,radiografia,malattia brut,images_db/art1026.jpg,2014-10-20
....
```

Figura 7.3: Creazione Risorsa MM - Sequence CSV



Dopo aver introdotto la funzione per la "creazione" della risorsa multimediale, ora analizziamo la prima delle funzioni che permettono di effettuare query multimediali.

Nella funzione **queryDBImage** viene inglobata la gestione delle richieste di query multimediali di tipo Top-K a partire da un ID di un immagine presente nel Database. In particolare, per prima cosa viene istanziato un oggetto che permette di calcolare la distanza, in questo caso si è sfruttato quello che si basa sulla *Earth Mover's Distance*[2]:

[2] `http://en.wikipedia.org/wiki/Earth_mover%27s_distance`

```
ScoringFunction sf=new ScoringFunctionEMD();
```

Si estrae dal DB il valore dell'immagine di cui effettuare la ricerca di somiglianza, partendo dall'ID che essa possiede

```
WindsurfImage query=(WindsurfImage) wf.retrieveImage(query_id);
```

Infine viene creato un oggetto che che basandosi sull'indice M-Tree, permette di navigare i risultati. Esso prende in input:

- Il valore dell'immagine di cui si vuole fare la ricerca nel DB
- La Scoring Function precedentemente decisa ed inizializzata
- Il path assoluto concatenato al nome del file indice

```
WindsurfQuerySFMtree alg=new WindsurfQuerySFMtree(query,
                    sf, imageIndexName);
```

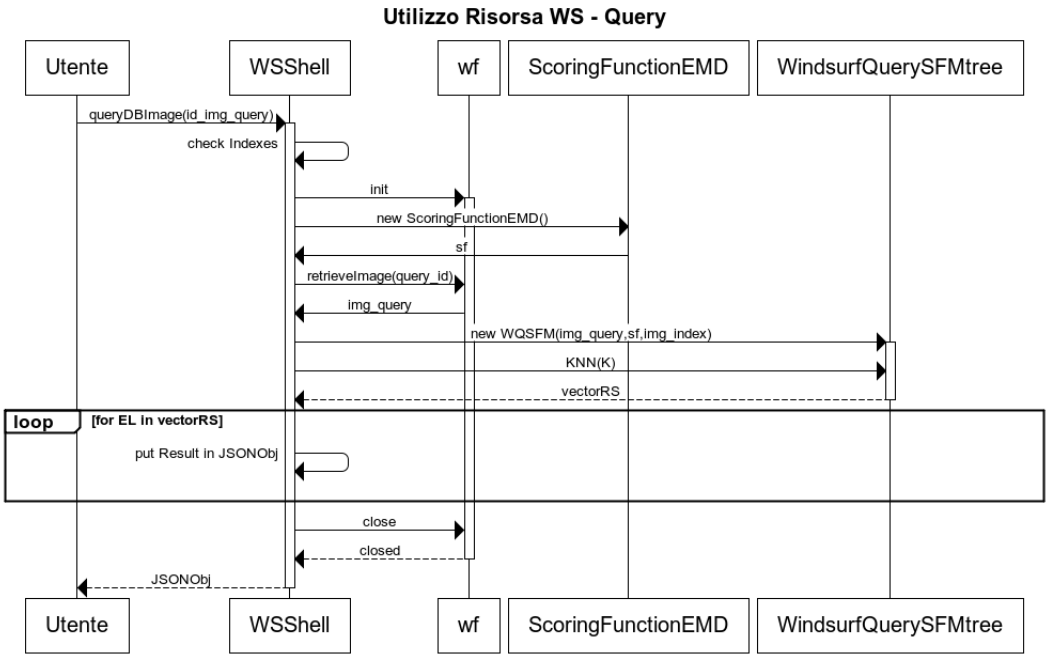
Invocando la funzione *KNN(K)* su quest'oggetto, ottengo i K risultati più simili all'immagine passata:

```
Vector<ScoredResultImage> v=alg.kNN(k);
```

In questo caso i risultati che vengono estratti sono inseriti in una struttura JSONObject, permettendo di mantenere l'ordine degli elementi selezionati, cosa molto importante in quanto li vogliamo in base al valore DISTANCE crescente (ricordandoci che essa è calcolata rispetto all'immagine data come query).

Il seguente sequence diagram ne spiega il funzionamento:

Figura 7.4: Query da immagine in DB - Sequence



Ora analizziamo come viene effettuata una query multimediale partendo da un immagine *non* presente all'interno della base di dati multimediali. La funzione che fa ciò è **queryImage/queryImageGetCursor**, analizzata in alcuni punti a partire dalla linea successiva a quella in cui è stata imposta classica funzione per il calcolo della distanza.

```
ScoringFunction sf = new ScoringFunctionEMD();
```

A questo punto viene inizializzato l'estrattore delle feature dell'immagine.

```
    WindsurfFeatureExtractor f_extractor = new WindsurfFeatureExtractor();
```

Ricordo che serve il path assoluto del File multimediale da usare per la ricerca nella sorgente Windsurf.

```
query_imageFileName = "file://" + module_path + image_query_rel_path
    +file.getName();
```

Vengono estratte le feature dall'immagine passata: il comando seguente effettua l'estrazione delle feature che vengono inserite nel DB e restituisce l'immagine in formato WindsurfImage.

```
WindsurfImage query = (WindsurfImage)
    f_extractor.extract(query_imageFileName, query_id);
```

Quindi successivamente viene creato l'oggetto che permette di trovare gli elementi

più simili a quello dato in input, sfruttando l'indice M-Tree:

```
WindsurfQuerySFMtree alg = new WindsurfQuerySFMtree(query, sf,
    module_path+properties.getProperty("windsurf.imageIndexName"));
```

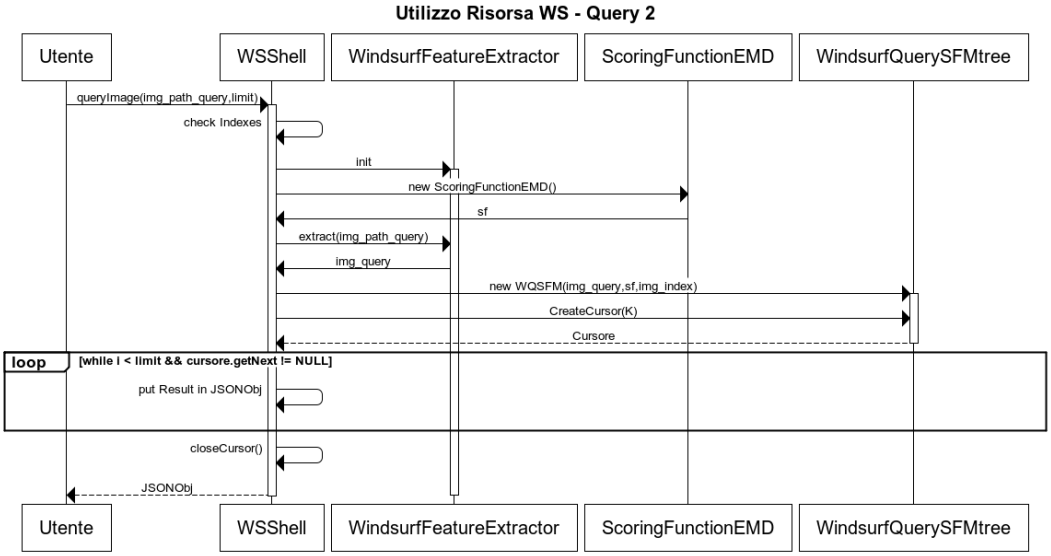
Infine itero sui risultati e ne estraggo K (Top-k):

```
if (alg.createCursor()) {
   for(int i=0; i<k; i++) {
      ScoredResultImage res = alg.getNext();
      if(res==null) break;
      ....
      }
   }
```

Utilizzo la funzione *getNext()* dell'oggetto **alg** per ottenere il successivo elemento, essa restituisce un valore $NULL$ se il successivo non è presente.

Il principio di funzionamento è riassunto nel seguente diagramma (figura 7.5).

Figura 7.5: Query da immagine - Sequence



7.2.1 Problema: caso più risorse Windsurf

Si è visto che la libreria Windsurf si basa su molte classi statiche, e ciò causa l'impossibilità di utilizzare la risorsa di tipo Windsurf in molteplici istanze. Per risolvere questo problema si dovrebbe intervenire pesantemente sul codice, ma nel nostro caso si è trovata una soluzione (momentanea) grazie alle funzionalità di MOMIS.

Sulla macchina si registra una istanza di Windsurf che nella sua implementazione stub/skeleton, permette di invocare le funzioni su oggetti remoti come se essi fossero

locali. In questo modo lanciando le istanze di Windsurf su più macchine virtuali Java, registrando sulla macchina un oggetto di tipo Windsurf ed importando in MOMIS una risorsa di tipo WSDL[3], si bypassa il problema. Per la registrazione della risorsa in una differente macchina virtuale, su macchine Microsoft è stato utilizzato Cygwin[4], un emulatore di shell Linux-like.

7.3 Modifiche a MOMIS

La prima delle modifiche apportate in MOMIS è quella legata all'import di una risorsa multimediale, ricordando che si deve mantenere quanto più disaccoppiato possibile MOMIS da Windsurf.

Ricordando il funzionamento tramite wizard per la configurazione semi-automatizzata e guidata della risorsa in MOMIS, è possibile scegliere la tipologia di risorsa da importare tramite un semplice menù a tendina che elenca le tipologie gestite dal software.

Per la gestione delle risorse multimediali senza dover aggiungere riferimenti ad una libreria esterna in modo "*hard-coded*", si è deciso quindi di creare un import "Generic", dove oltre al campo per la scelta del nome della risorsa da importare (da utilizzare dentro lo schema globale), bisogna indicare il Full Class Name sfruttato dal codice per caricare a runtime la classe del wrapper indicata.

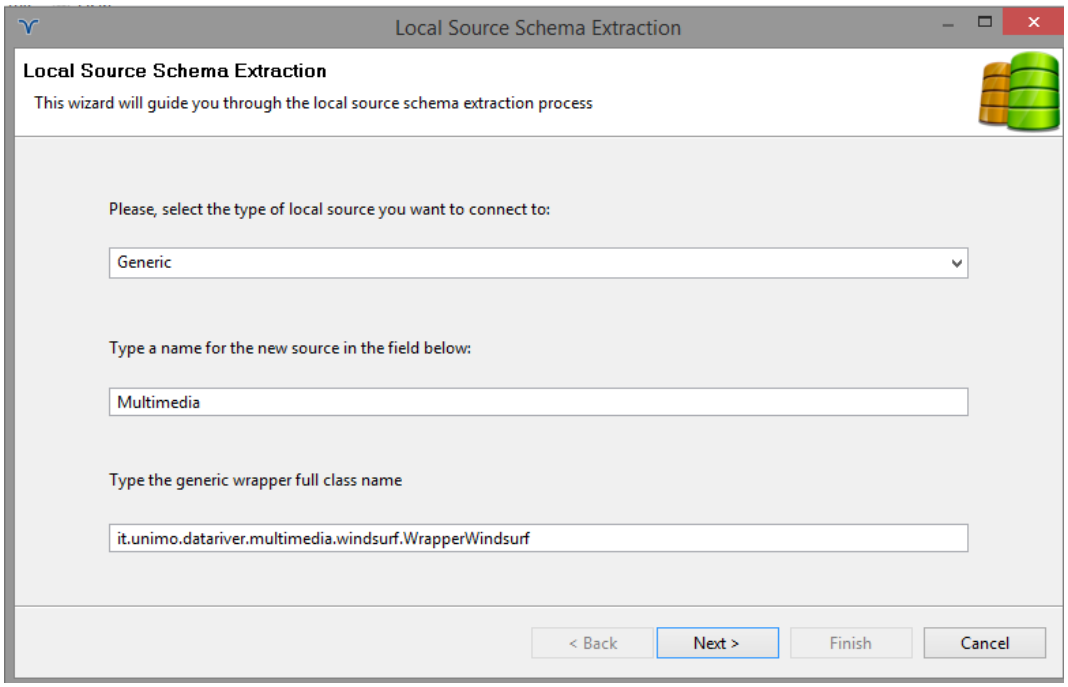
```
try{
..
classToLoad= Class.forName(fullNameOfWrapper, true,
    Activator.getCustomClassLoader());
            wrapper = (WrapperCore )classToLoad.newInstance();
            configuration.setText(wrapper.getParametersAsPropertiesTemplate());
            btnSubmit.setEnabled(true);
            btnRestore.setEnabled(true);
            if(isActived)updatePageComplete();
            isActived=true;
        }catch (Exception e1){
            configuration.setText("ERROR");
            updatePageComplete();
            setErrorMessage("Error : Class not found and no Schema
                obtained");
```

Tutto ciò funziona correttamente grazie alla presenza dell'oggetto statico **Activator**, che tramite il metodo *getCustomClassLoader()* carica dinamicamente le librerie contenute nella directory "lib" presente nel workspace di MOMIS . Se il

[3] http://en.wikipedia.org/wiki/Web_Services_Description_Language
[4] https://www.cygwin.com/

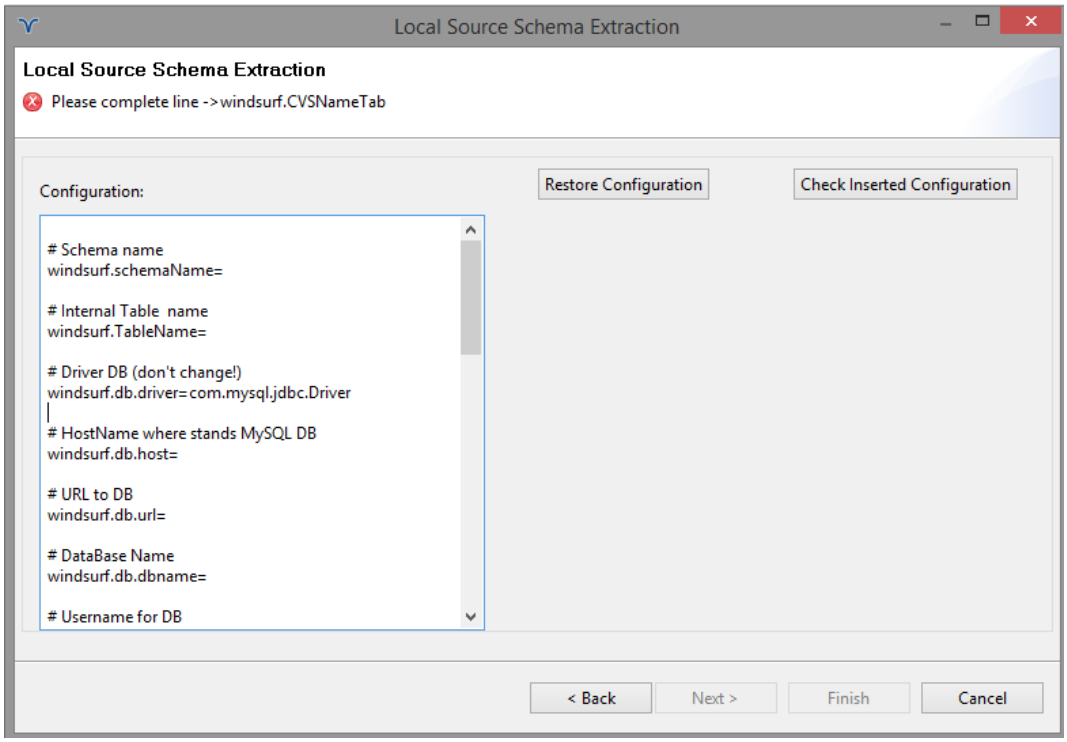
caricamento della classe **fullNameOfWrapper** inserita dall'utente non da esito positivo, il wizard non prosegue, in quanto non conosce la struttura della risorsa da importare. In caso positivo invece, verranno abilitati i pulsanti per andare avanti nella configurazione.

Figura 7.6: Import Source MM - 1



Nella schermata successiva raggiunta tramite il pulsante "**Next**", il wizard di MOMIS mostra a video la descrizione di ciò che vuole il wrapper (di Windsurf nel nostro caso) per essere configurato.

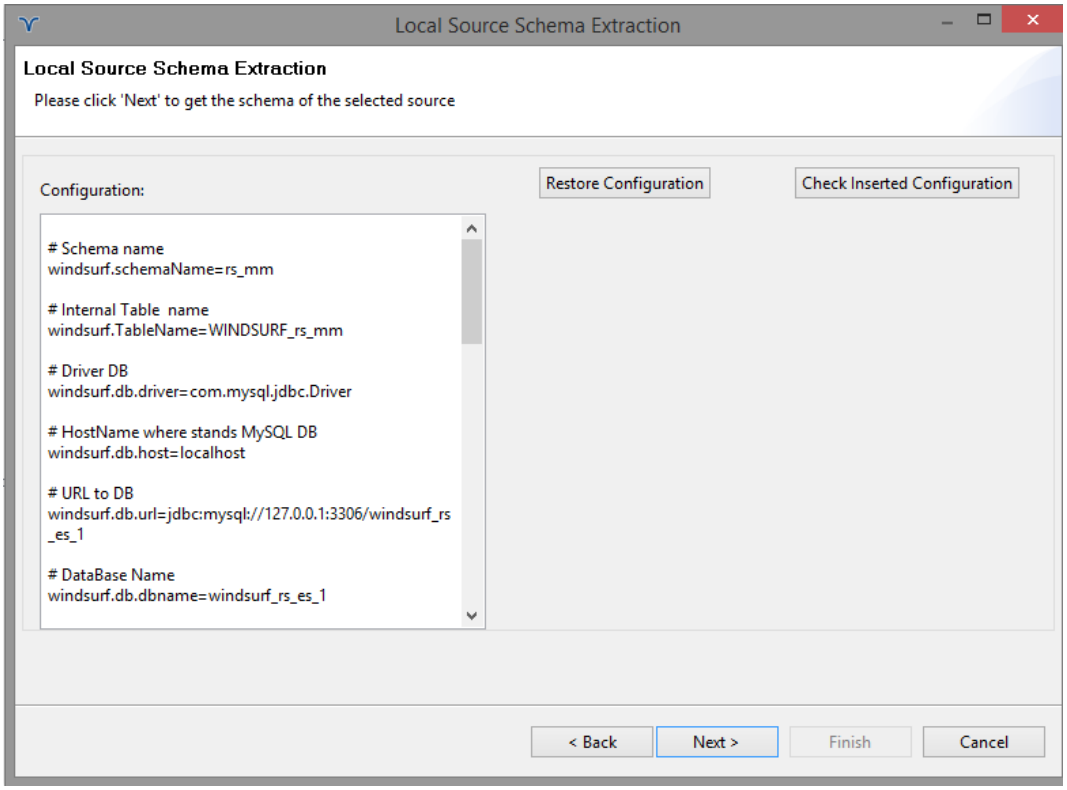
Figura 7.7: Import Source MM - 2



Il wizard di MOMIS per ottenere questa descrizione invoca il comando ***public String getParametersAsPropertiesTemplate()*** del wrapper, che descriverà quali sono i parametri ad esso necessari per la corretta configurazione della risorsa che gestisce.

Dopo che quanto richiesto è stato inserito, per procedere oltre bisognerà premere il bottone di "**Check Inserted Configuration**" . Questo comando controlla se sono stati completati i parametri necessari. Solo se tutti i parametri saranno stati inseriti, sarà possibile cliccare sul pulsante "Next", in caso contrario verrà mostrato a video un messaggio d'errore esaustivo sul problema occorso.

Figura 7.8: Import Source MM - 3



Nella schermata successiva è possibile selezionare le colonne di interesse. Questo è fatto iterativamente: ad ogni selezione di un elemento della lista mostrata a video, verrà inviato al wrapper un vettore di stringhe contenente i nomi delle colonne selezionate.

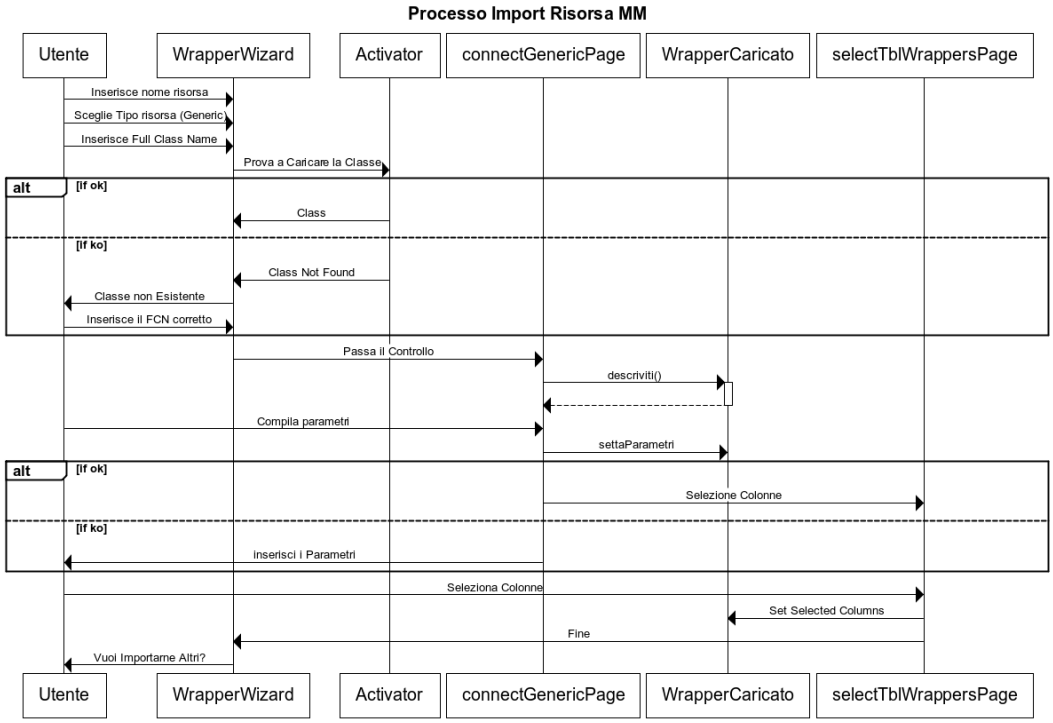
Il bottone "Data Preview" non fa altro che chiedere al wrapper la generica Query:

```
SELECT  TOP 100 <selected_columns>
from Wrapper
```

Quindi è data la possibilità all'utente di vedere interattivamente ciò che è stato selezionato.

In figura 7.9 è riportato un sequence diagram che riassume il funzionamento dell'import della risorsa multimediale

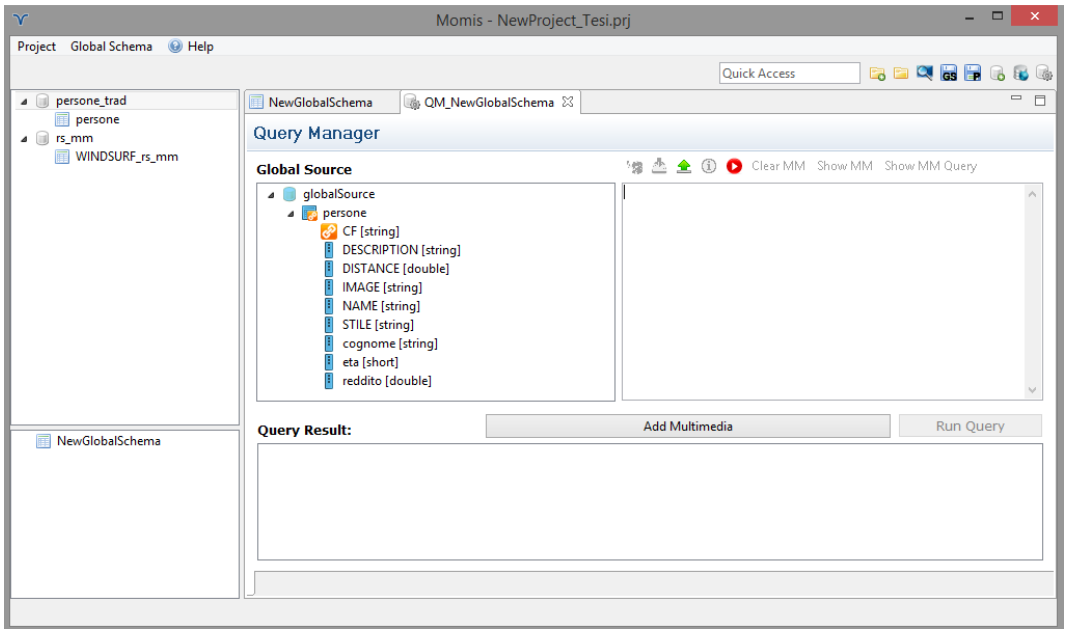
Figura 7.9: Import Source Sequence



Dopo aver introdotto la funzionalità del caricamento dinamico delle librerie di MOMIS, utile non soltanto per questo lavoro di tesi ma anche per qualunque futuro sviluppatore di nuovi wrapper per MOMIS, ed analizzato il "Generic Import", sono state apportate varie modifiche come l'introduzione della query multimediale in MOMIS ed alcuni miglioramenti nella gestione delle query.

Per introdurre la multimedialità nella gestione della query, sono state effettuate modifiche nella classe **QueryManagerEditorPage**. Il cambiamento più evidente è a livello di GUI: sono stati aggiunti alcuni pulsanti visualizzati solo se nel Global Schema è presente almeno una risorsa di tipo multimediale, che permettono appunto di gestire il predicato multimediale.

Figura 7.10: Query Editor Page - MM



Quindi tramite il Pulsante "**Add Multimedia**", viene aperto un *file chooser* che permette di scegliere un file multimediale: esso verrà convertito in stringa tramite la codifica Data URI. Per fare ciò si è sfruttata la classe **ImageUriDataTools** e la sua funzione ***getImageAsDataUri(file)*** che dato in input un path assoluto di un file multimediale, lo converte nella corrispettiva stringa in codifica Data URI. La parte della query contenente l'elemento multimediale sarà in un formato pseudo-JSON e la struttura segue la seguente logica:

```
__JSON__FOR_MULTIMEDIA{
        "GT_MM_NAME":{"DEBUG.LIMIT":"VAL_TOP_K"};
        "GT_MM_NAME.GA_MULTIMEDIA":"IMAGE_DATAURI"
}
```

Dove GT_MM_NAME è il nome della risorsa a cui ci riferiamo, GA_MULTIMEDIA è il nome dell'attributo multimediale (il modo per effettuare la query è appunto passare l'immagine abbinata all'attributo multimediale della risorsa a cui ci riferiamo).

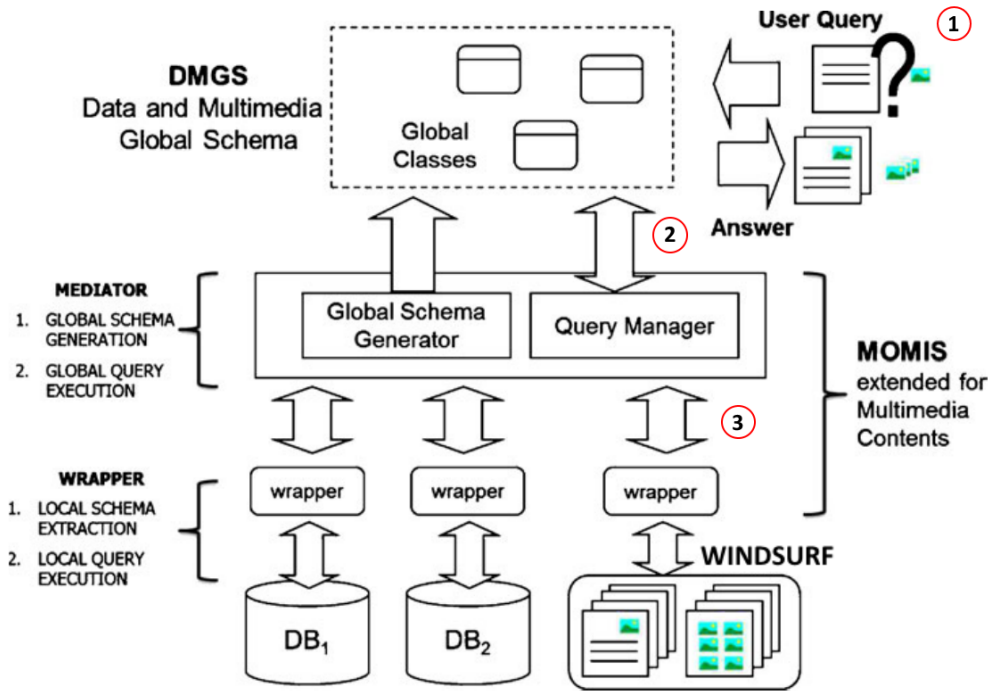
Questa struttura è mantenuta nascosta all'utente, ma può essere visualizzata tramite gli appositi pulsanti messi a disposizione.

Una volta che l'utente ha formulato e scritto la query da inviare e la sottomette tramite il pulsante "Run Query", essa viene pre-parsata dal Query Manager per rimuovere tutta la componente legata alla multimedialità (come da specifica discussa nella sezione 4.4.1).

Questo elemento estratto verrà successivamente mappato sulla risorsa locale ed aggiunto in coda alla (local) query indirizzata al wrapper multimediale.

Riportiamo ancora una volta l'architettura funzionale, mostrando in aggiunta le fasi di esecuzione di una query (figura 7.11).

Figura 7.11: Query Execution



Dove :

1. Sottomissione dell'intera Query
2. Rimozione della parte multimediale prima che in QM effettui il parsing della Query
3. Inserimento della parte multimediale (con mapping corretto) nella Query indirizzata verso la risorsa (multimediale)

Una volta che MOMIS ha scomposto in sotto-query la global query (processo di *Query Unfolding* analizzato sezione 1), dopo aver recuperato i wrapper in gioco ed averne analizzato la tipologia, se uno di essi implementa l'interfaccia "*Wrapper-Multimedia*", elemento che permette di distinguere un wrapper multimediale da uno tradizionale, il QM effettuerà la traduzione da Global Attribute in Local Attribute degli elementi nella parte multimediale (se presente). Dopo di ciò questa porzione di query verrà re-inserita in coda a quella diretta al wrapper multimediale e per concludere il flusso di esecuzione passerà ed essi.

Dopo che ogni wrapper (non solo quello di WS che verrà analizzato in dettaglio nella prossima sezione) ha finito la sua esecuzione ed ha restituito il suo ResultSet (RS), MOMIS effettua il processo di "Merge dei risultati".

Nel nostro caso questa fusione avviene sempre per mezzo del Full Outer Join calcolato dal motore SQL integrato in MOMIS. Se sono presenti predicati residui da applicare, essi verranno applicati in questa fase.

La classe **QueryManagerEditorPage** sarà anche incaricata di mostrare i risultati a video. È stata aggiunta la funzionalità che se il RS ricevuto contiene elementi rilevati come multimediali[5], questi al posto che essere mostrarti a video come stringhe, sono trasformati e visualizzati come "icone" contenenti una preview del dato multimediale. Inoltre è stata aggiunta la possibilità che tramite un doppio click sul record, si apra un *dialog* che mostra a video le informazioni contenute nella cella selezionata.

Questo dialog varia a seconda della tipologia del contenuto della cella: se viene rilevato un elemento di tipo immagine, verrà aperto il dialog che mostra l'immagine ed i comandi ad essa relativi, come salvare il contenuto su un file, zoom-in o zoom-out.

In futuro si potrà aggiungere la possibilità di visualizzare video, GIF animate, immagini diverse da quelle JPEG, audio, ed altri, dove ogni file verrà interpretato correttamente e mostrato con i giusti comandi (come per esempio per un video saranno presenti i comandi di play/stop, per file audio lo stesso, ecc..).

Se il sistema dovesse fallire nella conversione da stringa Data URI a elemento multimediale, come per esempio nel caso che la combinazione di caratteri:

```
‘‘...’’
```

sia il contenuto di una colonna di tipo stringa testuale senza significato multimediale, il contenuto originale verrà comunque mostrato all'utente sotto forma di stringa.

In figura 7.12 verrà mostrato il dettaglio dell'esecuzione di una query, ad ora analizzato solamente nell'esecuzione interna a MOMIS.

7.4 Creazione Wrapper Windsurf per MOMIS

Per riuscire ad integrare la risorsa multimediale Windsurf all'interno di MOMIS si è dovuto realizzare un wrapper. Il wrapper, come precedentemente detto, è un'entità capace di mettere in comunicazione una risorsa verso un'altra, nascondendo la sua reale implementazione.

[5]Elementi stringhe che iniziano con un formato che segue le regole della codifica Data URI, per esempio un immagine JPEG con codifica Data URI inizierà per "...".

Per realizzare ciò si è deciso di inserire all'interno di un package (dedicato) della libreria Windsurf tutte le classi ad esso necessarie. Il package in questione è "***it.unimo.datariver.multimedia.windsurf***", il cui nome è stato scelto sulla base della regola usata per definire i nomi dei package in MOMIS (che iniziano per "*it.unimo.datariver....*").

All'interno del progetto di MOMIS è stata creata l'interfaccia **WrapperMultimediaQuery**, utile per capire se il wrapper che MOMIS sta utilizzando è di tipo multimediale, in quanto si avrà in questo caso la necessità di inviare dati extra. Il wrapper di Windsurf espone esattamente le stesse funzioni di un qualunque altro wrapper per risorse di MOMIS. Essendo un wrapper di tipologia diversa dagli altri, alla richiesta della sua descrizione esso risponderà di essere del tipo "***MULTIMEDIA***". Gli elementi classici del setup del wrapper rimangono inalterati a livello di logica. All'atto dell'invocazione della query (metodo *runQuery*(*String query*)), il wrapper esegue un primo parsing e controlla se la query ricevuta contiene elementi multimediali.

La prima azione intrapresa dal wrapper Windsurf (WS abbreviato) è quella di controllare se è presente la parte multimediale:

```
''__JSON__FOR_MULTIMEDIA{...}''
```

Se questa keyword contenente la porzione multimediale della query risulta essere presente, il wrapper ne analizzerà il contenuto.

Il parsing nello stato attuale è effettuato all'interno della classe "**OggettoDatiwrapperWindsurf**", ed è eseguito semplicemente tramite una serie di controlli sulla struttura dei predicati. In un secondo momento si implementerà un parser che verrà creato usando JavaCC[6] (Java Compiler Compiler). Dopo di ciò il wrapper invoca la sua funzione ***runQueryFinal*** che in input riceve la stringa query (senza la parte multimediale) e l'oggetto che si è auto-popolato con i parametri di interesse (l'oggetto che ha effettuato il parsing della query, istanza della classe OggettoDatiwrapperWindsurf). Questa funzione restituisce un ResultSet, che sarà di tipo **WindsurfRSWindsurf** e che deriverà dalla classe WrapperRSInternal, avente le funzioni base che deve offrire un tipico ResultSet.

Analizzando questa classe di tipo WrapperRSWindsurf possiamo notare che tutto il reale calcolo dei risultati è da essa effettuato.

Come primo comando nella sua inizializzazione, viene istanziato correttamente un oggetto di tipo **WindsurfShell** sfruttando la configurazione presente nel wrapper, in modo tale che si possono utilizzare tutti i comandi che essa mette a disposizione.

[6] `http://it.wikipedia.org/wiki/JavaCC`

Successivamente, il wrapper chiede all'oggetto "Parser" (OggettoDatiwrapper-Windsurf) se la query è multimediale, e quindi se contiene un immagine (o genericamente un elemento multimediale). In caso affermativo essa viene recuperata, trasformata in elemento immagine (ricordando che in MOMIS è gestita come una Stringa codificata in Data URI) e salvata in un file temporaneo.

A questo punto si recuperano i nomi dei parametri che devono essere restituiti (parametri presenti nella clausola SELECT della query) e viene inizializzato l'array di risultati.

Come avevamo precedentemente detto, oltre alle classiche colonne di WS si potrebbe avere la necessità di un attributo deciso dall'utente che possa essere utilizzato per effettuare l'object fusion. Questo elemento è necessario nel caso in cui sia presente il file EXTRA contenente dati aggiuntivi relativi agli elementi multimediali, ricordando che essi sono stati preventivamente trasferiti in un database. Si fa presente che questi dati non saranno gestiti da Windsurf stesso ma dal wrapper, che oltre ad invocare le funzioni MM offerte dalla shell di WS, effettuerà anche il join fra i risultati così estratti con i dati extra. In questo modo si riescono ad aggiungere agli elementi "puramente" multimediali, i valori extra specificati dall'utente in un file CSV all'atto della creazione della risorsa multimediale.

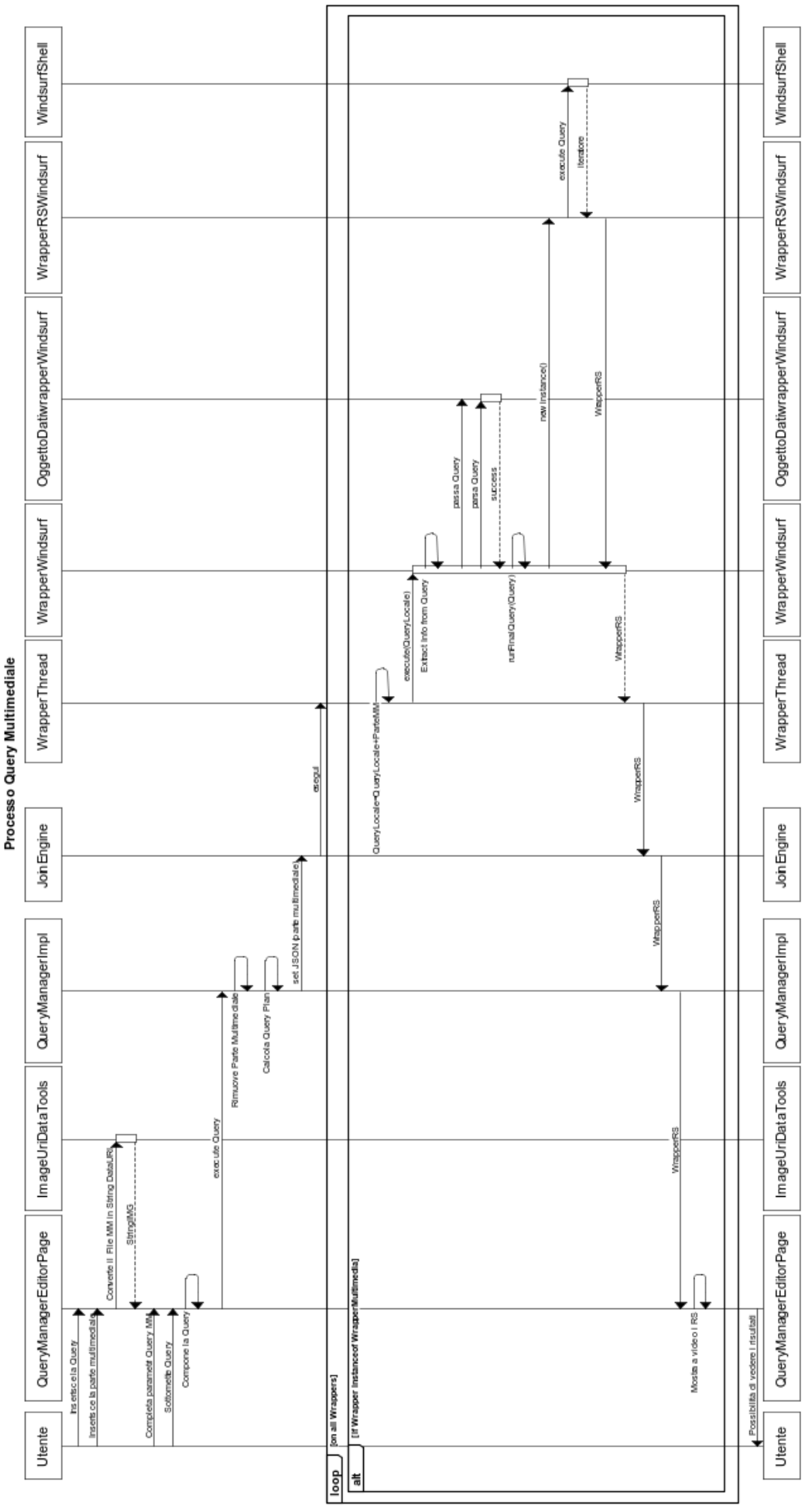
Riassumendo, se la query non è multimediale, il wrapper accede in modo diretto alla base di dati, per estrarne i dettagli (non multimediali, come nome degli elementi) e quindi se necessario mette i risultati in join con i dati extra; se invece la query è multimediale, vengono sfruttate tutte le funzionalità offerte tramite invocazione diretta sull'oggetto di tipo **WindsurfShell**, come è stato precedentemente spiegato.

Questo WrapperRSWindsurf conterrà al suo interno un contatore, che servirà per scorrere il ResultSet; esso offrirà i classici metodi *next()*, *GET<Tipo>()*, tipici degli elementi iterabili. Il **WrapperRSWindsurf** in questo modo permette di non dover caricare interamente in memoria una lista probabilmente grande, contenente elementi di dimensioni notevoli.

Nel momento in cui MOMIS invocherà *next()* e poi *GET<Tipo>()* sul ResultSet, la classe WrapperRSWindsurf accederà nel database per integrare il risultato multimediale con i possibili dati extra.

Il seguente schema in figura 7.12 è uno pseudo sequence diagram che riassume il funzionamento del wrapper.

Figura 7.12: Query Sequence Diagram



Conclusioni

Questa tesi ha trattato la realizzazione (progettazione ed implementazione) di un Sistema di Integrazione tra dati tradizionali e multimediali, in grado di costruire uno Schema Globale integrato rappresentativo di entrambe le tipologie di sorgenti, in modo da consentire all'utente finale di specificare interrogazioni che contengano congiuntamente sia criteri di ricerca tradizionali che criteri di ricerca per similarità (in particolare Top-K query) sui dati multimediali. Il sistema deve essere quindi in grado di effettuare la fusione dei dati che provengono dalle sorgenti locali, sia tradizionali che multimediali, per restituire all'utente una risposta unificata. La tesi deriva dalle attività svolte durante un tirocinio di sei mesi presso DataRiver, spin-off universitario e si colloca nell'ambito del sistema di Integrazione Dati MOMIS, sviluppato dal DBGroup dell'Università di Modena e Reggio Emilia e distribuito in versione open source da DataRiver.

Nella presente tesi, dopo un'analisi delle soluzioni già presentate in letteratura, in particolare di quella precedentemente sviluppata nel contesto del Sistema MOMIS, viene sviluppata ed implementata una nuova soluzione per la fusione di dati tradizionali e multimediali, la cui caratteristica fondamentale è quella di utilizzare ancora il Query Manager tradizionale del Sistema MOMIS, basato solo su operazioni standard del linguaggio SQL. In altri termini, nella soluzione proposta ed implementata i criteri di ricerca per similarità su dati multimediali sono tutti gestiti dal Sistema Windsurf, sviluppato dall'Università di Bologna, che permette di effettuare ricerche efficienti in sorgenti multimediali; a livello di Sistema MOMIS avviene quindi la fusione con dati tradizionali tramite un "SQL-engine" standard.

Gli obiettivi sono stati raggiunti, in quanto è stato realizzato il wrapper ed effettuate le modifiche che hanno reso MOMIS uno strumento funzionante che consente di interrogare congiuntamente sorgenti tradizionali e multimediali. Dal punto di vista dell'utilizzo, si è analizzato come esso possa essere utilizzato nel campo medico, in particolare per l'integrazione e ricerca in esami PET (con i relativi valori SUV). MOMIS-Windsurf permette ai medici di avere una vista integrata di queste analisi, che verrà sfruttata per cercare casi analoghi a quello in esame, per avere un supporto

alla decisione derivante dallo storico delle procedure intraprese. Il progetto realizzato in questo tirocinio potrebbe essere un buon punto di partenza per successivi sviluppi.

Concludendo, si è potuto apprezzare la facilità di estensione del sistema MOMIS realizzato dal DBGroup e da DataRiver, caratteristica non presente invece nel progetto Windsurf, dove l'introduzione nella libreria di una nuova tipologia di risorsa multimediale ha richiesto una modifica sostanziale al progetto. Sottolineiamo che per realizzare un progetto funzionante nei mesi di tirocinio, ci sono stati compromessi lato "numero di funzionalità" realmente introdotte nel sistema integrato, per questo motivo si può notare che alcuni elementi della tesi non sono stati dettagliati, fornendo la motivazione e analizzando nel capitolo "Sviluppi Futuri" le possibili implementazioni studiate ma non realizzate.

Sviluppi futuri

Il progetto realizzato soddisfa gli obiettivi che erano stati originariamente fissati, ovvero quello di a realizzare un integrazione completa con più sorgenti, fra cui una multimediale. Uno dei principali sviluppi futuri è quello di estendere il framework per integrare un generico numero di sorgenti multimediali. Da un punto di vista teorico, l'approccio è facilmente estendibile al caso di più sorgenti multimediali. Invece, da un punto di vista implementativo, ci sono alcune limitazioni che impediscono il suo utilizzo per la gestione di più sorgenti multimediali in quanto il software Windsurf si basa su varie classi statiche e su altri elementi che impediscono l'utilizzo della libreria in più istanze. Per risolvere efficacemente il problema, si dovrebbero rendere dinamiche le "classi *core*" di tale software, e quindi modificando profondamente il codice Windsurf.

La soluzione "parziale" che è stata implementata senza stravolgere il codice Windsurf è stata quella di sfruttare la gestione di risorse remote tramite i web service, come descritto in sezione 7.2.1.

Con il Case Study analizzato in questa tesi abbiamo verificato che la gestione di elementi multimediali differenti da quelli attualmente previsti in Windsurf (in sostanza le sole immagini JPEG), ha richiesto un intervento non indifferente a livello di logica di quest'ultimo. Un possibile sviluppo futuro riguarda il sistema Windsurf per renderlo più facilmente estendibile verso la gestione di nuovi elementi multimediali.

In definitiva, per rendere tutto il framework MOMIS-Windsurf facilmente estendibile, il componente Windsurf dovrebbe essere modificato per raggiungere i requisiti di estendibilità che invece sono già soddisfatti dal componente MOMIS (come si è potuto verificare appunto nello svolgimento di tale tesi, in cui MOMIS è stato esteso per gestire elementi multimediali).

Un elemento che potrebbe essere migliorato nel wrapper di MOMIS-Windsurf è il parser della query locale. Attualmente esso per riuscire a comprendere i parametri che sono di interesse, quelli da restituire e le clausole WHERE, si basa su un analisi effettuata tramite un parsing rudimentale composto da una cascata di if. Inoltre le clausole WHERE gestite correttamente in modo diretto dal wrapper sono poche,

come per esempio quelle sull' ID, sul NAME (dell'elemento multimediale) e sulla DISTANCE. Il sistema funziona in quanto i predicati sono applicati una seconda volta da MOMIS all'atto della Final Query, quindi sugli elementi unificati che erano stati restituiti dalle varie risorse locali. L'obiettivo per il futuro è quello creare un parser completamente funzionante posto all'interno del wrapper Windsurf, usando il JavaCC.

Concludiamo con quello che è il punto più importante da affrontare come sviluppo futuro, ovvero il problema relativo alle Top-K query discusso in sezione 3.3. Ricordiamo sinteticamente che nella soluzione attuale una Top-K query, in presenza di predicati tradizionali, restituisce in genere un numero di risultati inferiore a K.

Le soluzioni delineate sono in sostanza due: la prima, più di carattere teorico, è basata sulla stima delle selettività dei predicati; tale soluzione potrebbe utilizzare le stime di selettività già disponibili nei DBMS (soprattutto quelli commerciali) e non dovrebbe richiedere uno stravolgimento del Query Manager di MOMIS.

La seconda soluzione è invece basata su una nuova logica di funzionamento del Query Manager di MOMIS, ovvero richiederebbe di riprogettare la logica, in quanto necessita di inoltrare al wrapper della risorsa multimediale una query che contiene al suo interno un risultato (parziale) della query tradizionale.

Appendice A

Soluzione con MILOS - Medrank

A.1 Medrank

In questa appendice verrà sintetizzato l'articolo [15] che mostra il principio di funzionamento di MEDRANK e si mostreranno i limiti e i pregi di questa implementazione.

Medrank [8] è un algoritmo che permette di ottenere in modo altamente efficiente l'unificazione di liste ordinate di elementi. Esso può restituire un singolo record per volta e permette il calcolo delle query Top-K, esso lavora a livello degli oggetti. Medrank scansiona la lista ordinata ottenuta dall'esecuzione della query sulla risorsa multimediale. Il primo valore di ID (inteso come identificatore del record, elemento univoco) che è presente in più della metà delle liste è considerato come un Top ID, si procede nello stesso modo per calcolare i successivi elementi (per esempio, nell'esecuzione della Top-K questo processo sarà eseguito K volte). Per il calcolo di questo algoritmo non è necessario avere accesso random alla lista ordinata ma nel nostro caso, dal momento che la fusione deve essere effettuata a livello di record, abbiamo la necessità di avere accesso diretto al record dato un ID restituito da Medrank. Ciò è necessario sia per verificare se esso soddisfa tutti i predicati imposti che per ottenere tutti gli attributi della global query.

La formalizzazione dell'algoritmo, che noi chiameremo MEDRANKA, utilizza l'algoritmo originale MEDRANK come una black-box ed è come segue:

- MEDRANK.Init , è l'inizializzazione della procedura di MEDRANK
- MEDRANK.Next , è la procedura che ritorna :
  - id : il valore ID
  - RL : il set degli indici della query dalla quale l'ID è stato ottenuto

– RSET : il set dei record locali correnti dalla quale l'ID è ottenuto

L'accesso random alla lista ordinata (ranked-classificata) LQ(ID,A_1,...,A_n) è implementata dalla funzione:

$$RandomAccess(LQ, id) := \begin{cases} r[A_1, ..., A_n] & \text{se } LQ \text{ contiene r con r[ID]=id,} \\ r_\perp[A_1, ..., A_n] & \text{altrimenti} \end{cases}$$

Dove $r_\perp$ è il record nullo, i cui attributi sono riempiti con il valore NULL. Inoltre, dati due record r_1 e r_2 con set di attributi disgiunto, concat(r_1, r_2) creerà un nuovo record che contiene gli attributi r_1 e r_2. Questo algoritmo è mostrato in figura A.1 e il suo input è formato da:

- un set di query locali LQ_1,...,LQ_n
- RX1: il set di indici di query locali con almeno un predicato atomico.
- RX2 : il set di indici di query locali contenenti almeno un attributo di query locale.

La procedura di MEDRANKA.init chiama a sua volta la procedura init di MEDRANK. Medranka.next è l'implementazione di MEDRANKA che restituisce un record per invocazione, e a sua volta si basa sul NEXT di MEDRANK.

Figura A.1: Feature Extraction

1. Procedure $MEDRANKRA.Init(\{LQ_1, \ldots, LQ_n\}, RX1, RX2)$
2. $\quad MEDRANK.Init(\{LQ_1, \ldots, LQ_n\})$

3. Function $r = MEDRANKRA.Next()$
4. $\quad$ Set *record_found=false*
5. $\quad$ While not *record_found*
6. $\quad\quad$ Set $[RL, id, RSET] = MEDRANK.Next()$
7. $\quad\quad$ If id = NULL Return NULL
8. $\quad\quad$ Set r[ID] = id
9. $\quad\quad$ Set $RC = \left(\{1, \ldots, n\} \setminus RL\right) \cap RX1$
10. $\quad\quad$ Set $RA = \left(\{1, \ldots, n\} \setminus RL\right) \cap RX2$
11. $\quad\quad$ For Each $ri \in RSET$
12. $\quad\quad\quad$ Set $r = concat(r, ri)$
13. $\quad\quad$ End For Each
14. $\quad\quad$ Set *record_found=true*
15. $\quad\quad$ For Each $i \in RC$
16. $\quad\quad\quad$ Set $rx = RandomAccess(LQ_i, id)$
17. $\quad\quad\quad$ If rx is not $NULL$
18. $\quad\quad\quad\quad$ Set $r = concat(r, rx)$
19. $\quad\quad\quad\quad$ Set $RA = RA \setminus \{i\}$
20. $\quad\quad\quad$ Else
21. $\quad\quad\quad\quad$ Set *record_found=false*
22. $\quad\quad\quad\quad$ break
23. $\quad\quad\quad$ End If
24. $\quad\quad$ End For Each
25. $\quad\quad$ If *record_found=true*
26. $\quad\quad\quad$ For Each $i \in RA$
27. $\quad\quad\quad\quad$ Set $rx = RandomAccess(LQ_i, id)$
28. $\quad\quad\quad\quad$ Set $r = concat(r, rx)$
29. $\quad\quad\quad$ End For Each
30. $\quad\quad$ End If
31. $\quad$ End While
32. $\quad$ Return r

Glossario e Acronimi

API : Application Programming Interface

CBIR : Content Based Image Retrieval

CSV : Comma-Separated Values

DBMS : Database Management System

DLL : Dynamic-Link Library

DMGS : Data and Multimedia Global Schema

DMS : Data and Multimedia Source

ETL : Extract Transform Load

GA : Global Attribute

GAV : Global-As-View

GS : Global Schema

GUI : Graphical User Interface

GVV : Global Virtual View

IETF : Internet Engineering Task Force

IR : Information Retrieval

JAI : Java Advanced Imaging

JAR : Java Archive

JPEG : Joint Photographic Experts Group

MM : Multimediale

MT : Mapping Table

ODL : Object Definition Language

OQL : Object Query Language

PET : Positron Emission Tomography

QM : Query Manager

QMT : Query Manager Tradizionale

RBIR : Region Based Image Retrieval

RDBMS : Relational Database Management System

RFC : Request for Comments

RM/RMN : Risonanza Magnetica Nucleare / Nuclear Magnetic Resonance

RS : Result Set

SQL : Structured Query Languag

TC/TAC : Tomografia (Assiale) Computerizzata / Computed Tomography

WLS : Windsurf Local Source

WS : Windsurf, **W**avelet-based **IND**exing of Image**S** **U**sing **R**egion **F**ragmentation

WSDL : Web Services Description Language

Bibliografia

[1] Giuseppe Amato, Claudio Gennaro, Fausto Rabitti, and Pasquale Savino. Milos: A multimedia content management system for digital library applications. In Rachel Heery and Liz Lyon, editors, *Research and Advanced Technology for Digital Libraries*, volume 3232 of *Lecture Notes in Computer Science*, pages 14–25. Springer Berlin Heidelberg, 2004.

[2] Domenico Beneventano and Sonia Bergamaschi. Semantic search engines based on data integration systems. *Semantic Web Services: Theory, Tools and Applications*, pages 317–341, 2007.

[3] Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, 7(5):42–51, 2003.

[4] Domenico Beneventano, Sonia Bergamaschi, and Carlos Rodrigue Nana Mbinkeu. Full outer join optimization techniques in integration information systems. Technical report, Dipartimento di Ingegneria dell'Informazione, 2006. http://www.dbgroup.unimo.it/prototipo/paper/cleandb.pdf.

[5] Domenico Beneventano, Claudio Gennaro, Sonia Bergamaschi, and Fausto Rabitti. A mediator-based approach for integrating heterogeneous multimedia sources. *Multimedia Tools and Applications*, 62(2):427–450, 2013.

[6] Domenico Beneventano, Claudio Gennaro, Sonia Bergamaschi, and Fausto Rabitti. A mediator-based approach for integrating heterogeneous multimedia sources. *Multimedia Tools and Applications*, 62(2):427–450, 2013.

[7] Sonia Bergamaschi, Domenico Beneventano, Francesco Guerra, and Mirko Orsini. Data integration. In *Handbook of Conceptual Modeling: Theory, Practice and Research Challenges*. Springer Verlag, 2011.

[8] Sonia Bergamaschi, Domenico Beneventano, Francesco Guerra, and Mirko Orsini. Data integration. In D. W. Embley and B. Thalheim, editors, *Handbook of Conceptual Modeling: Theory, Practice and Research Challenges*. Springer Verlag, 2011.

[9] Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. Semantic integration of semistructured and structured data sources. *ACM Sigmod Record*, 28(1):54–59, 1999.

[10] Sonia Bergamaschi, Silvana Castano, Maurizio Vincini, and Domenico Beneventano. Semantic integration of heterogeneous information sources. *Data Knowl. Eng.*, 36(3):215–249, 2001.

[11] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Comput. Surv.*, 41(1):1–41, 2008.

[12] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[13] Giovanni Esposito. Momis e open data: Integrazione di dati aziendali con sorgenti dati pubblici. diploma thesis, Università di Modena e Reggio Emilia.

[14] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 301–312, New York, NY, USA, 2003. ACM.

[15] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, SIGMOD '03, pages 301–312, New York, NY, USA, 2003. ACM.

[16] Claudio Gennaro, Rita Lenzi, Federica Mandreoli, Riccardo Martoglia, Matteo Mordacchini, Wilma Penzo, and Simona Sassatelli. A unified multimedia and semantic perspective for data retrieval in the semantic web. *Information Systems*, 36(2):174 – 191, 2011. Special Issue: Semantic Integration of Data, Multimedia, and Services.

[17] Bernhard Haslhofer and Wolfgang Klas. A survey of techniques for achieving metadata interoperability. *ACM Comput. Surv.*, 42(2):7:1–7:37, March 2010.

[18] Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. *Data Quality and Record Linkage Techniques*. Springer Publishing Company, Incorporated, 1st edition, 2007.

[19] Marco Patella Ilaria Bartolini and Guido Stromei. The windsurf library for the efficient retrieval of multimedia hierarchical data. In *Proceedings of the Interna-*

tional Conference on Signal Processing and Multimedia Applications SIGMAP 2011, pages 139–148, Seville, Spain, July 2011.

[20] Paolo Ciaccia Ilaria Bartolini and Marco Patella. Query processing issues in region-based image databases. *Knowledge and Information Systems (KAIS)*, 25(2):389–420, 2010.

[21] Entela Kazazi. Il componente query manager del sistema momis: testing ed analisi delle performance. Master's thesis, Università di Modena e Reggio Emilia.

[22] A. Knoll, C. Altenschmidt, J. Biskup, H.-M. Blüthgen, I. Glöckner, S. Hartrumpf, H. Helbig, C. Henning, R. Lüling, B. Monien, T. Noll, and N. Sensen. An integrated approach to semantic evaluation and content-based retrieval of multimedia documents. In *Research and Advanced Technology for Digital Libraries*, volume 1513 of *Lecture Notes in Computer Science*, pages 409–428. Springer Berlin Heidelberg, 1998.

[23] Vitaveska Lanfranchi, Fabio Ciravegna, and Daniela Petrelli. Semantic web-based document: Editing and browsing in aktivedoc. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *The Semantic Web: Research and Applications*, volume 3532 of *Lecture Notes in Computer Science*, pages 623–632. Springer Berlin Heidelberg, 2005.

[24] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

[25] Zhuhua Liao, Jing Yang, Chuan Fu, and Guoqing Zhang. Integrating web videos for faceted search based on duplicates, contexts and rules. In Zhongzhi Shi; Sunil Vadera; Agnar Aamodt; David Leake, editor, *Intelligent Information Processing V*, volume 340 of *IFIP Advances in Information and Communication Technology*, pages 203–212. Springer, 2010.

[26] Rodrigue Carlos Nana Mbinkeu. *Query Optimization and Quality-Driven Query Processing for Integration Systems*. PhD thesis, UNIVERSITY OF MODENA, 2011.

[27] Phivos Mylonas, Thanos Athanasiadis, Manolis Wallace, Yannis Avrithis, and Stefanos Kollias. Semantic representation of multimedia content: Knowledge representation and semantic indexing. *Multimedia Tools and Applications*, 39(3):293–327, 2008.

[28] Felix Naumann, Johann Christoph Freytag, and Ulf Leser. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.

[29] Felix Naumann and Matthias Häussler. Declarative data merging with conflict resolution. In *IQ*, pages 212–224, 2002.

[30] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

[31] Kosmas Petridis, Dionysios Anastasopoulos, Carsten Saathoff, Norman Timmermann, Yiannis Kompatsiaris, and Steffen Staab. M-ontomat-annotizer: Image annotation linking ontologies and multimedia low-level features. In Bogdan Gabrys, RobertJ. Howlett, and LakhmiC. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4253 of *Lecture Notes in Computer Science*, pages 633–640. Springer Berlin Heidelberg, 2006.

[32] Lorenzo Pirazzini. Estensione del framework windsurf per la gestione e il ritrovamento per confronto di esami pet tramite valori suv, 2014-2015.

[33] M.T. Roth and P. Scharz. Don't Scrap It, Wrap it! A Wrapper Architecture for Legacy Data Sources. In *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, March 1995.

[34] Umberto Straccia and Giulio Visco. G.: Dlmedia: An ontology mediated multimedia information retrieval system. In *In: Proc. DL-2007 (2007).*

[35] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity search: the metric space approach*, volume 32. Springer, 2006.