SAPIENZA Università di Roma

Facoltà di Ingegneria

Tesina per il corso di Seminari di ingegneria del software

Anno accademico 2006/2007

DBLP Ontology

Germano Rocco

Relatore: Prof. Giuseppe De Giacomo

Indice

Capitolo 1

Introduzione

Questo progetto nasce come l'argomento principa della tesina del corso di Seminari di ingegneria del software. Lo scopo principale della tesina è stato quello di realizzare l'ontologia della bibliografia DBLP.

La bibliografia DBLP, a cura del Dr. Michael Ley, raccoglie tutte le pubblicazioni scientifiche riguardanti l'Informatica (Computer Science).

Di conseguenza, il dominio dell'ontologia riflette quello della bibliografia DBLP, cioè, pubblicazioni di carattere scientitico.

Dato che, dati della bibliografia sono strutturati in XML, è stato quindi necessario realizzare un'applicazione che effettuasse la lettura dei dati, e il successivo inserimento in un apposito database.

Si è proseguito realizzando l'ontologia. Inizialmente descritta nel linguaggio $DL - Lite_A$ e successivamente realizzata tramite il tool di gestione per ontologie *Protégé*.

Sono stati realizzati i mapping per trasformare i dati reazionali in oggetti dell'ontologia ed infine eseguite alcune query SPARQL.

Capitolo 2

DBLP Bibliograpy

2.1 Caratteristiche

La bibliografia DBLP (a cura del Dr. Michael Ley, docente di informatica al dipartimento di Computer Science dell'università di Trier, Germania) è consultabile attraverso il seguente link: http://www.informatik.uni-trier.de/ ley/db/.

Il lavoro di archiviazione nasce nel 1993 come hobby.

Inizialmente venivano raccolte solo le pubblicazioni riguardanti programmazione e database. E' nel 1997 che la bibliografia DBLP diventa un vero e proprio progetto, integrando tale lavoro con l' ACM SIGMOD Anthology.

Nel 1998 si sono aggiunti inoltre sponsor come MicroSoft Research, allargando l'interesse a tutte le aree della Compuer Science.

Nel 2006 la bibliografia DBLP comprendeva oltre 700000 record.

La Figura 2.1 mostra la sua evoluzione in termini del numero di record. Tramite la Figura 2.2 è possibile osservare il numero crescente di pubblicazioni scientifiche (il grafico è aggiornato al 12 gennaio 2006).

Uno dei problemi fondamentali è la qualità dei dati. Poichè DBLP continua ad esistere grazie ad un notevole sforzo umano, a lungo termine può essere giustificato solamente se la qualità dei dati è elevata.

In particolare, misuriamo tale qualità mediante quattro grandezze:

- Accuratezza
- Completezza
- Tempestività
- Rilevanza

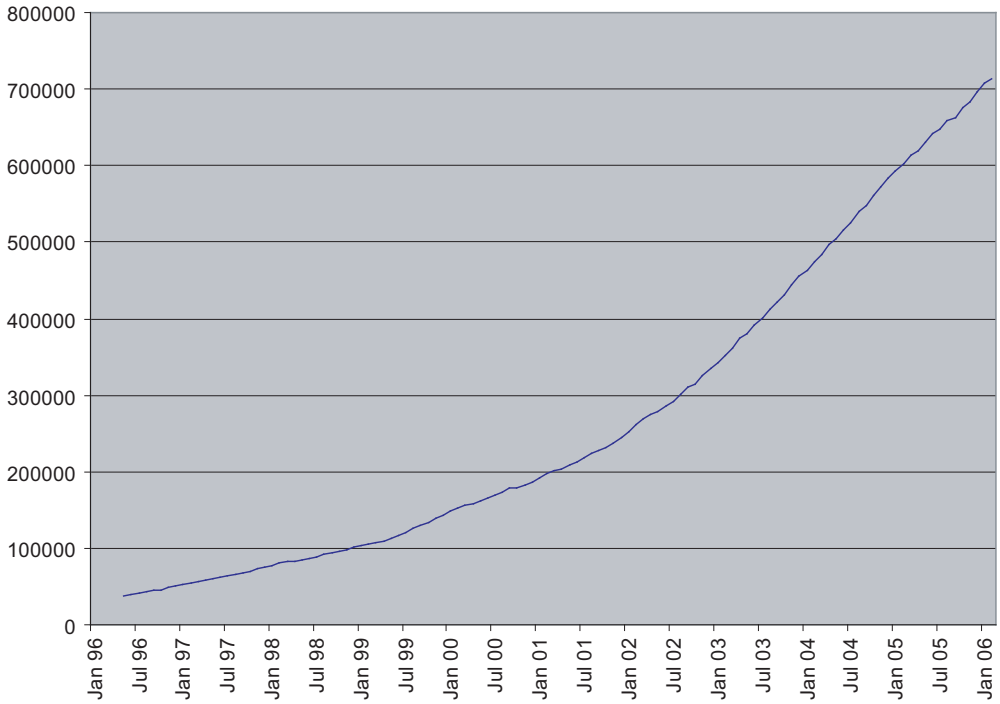

Figura 2.1: Andamendo del numero di record

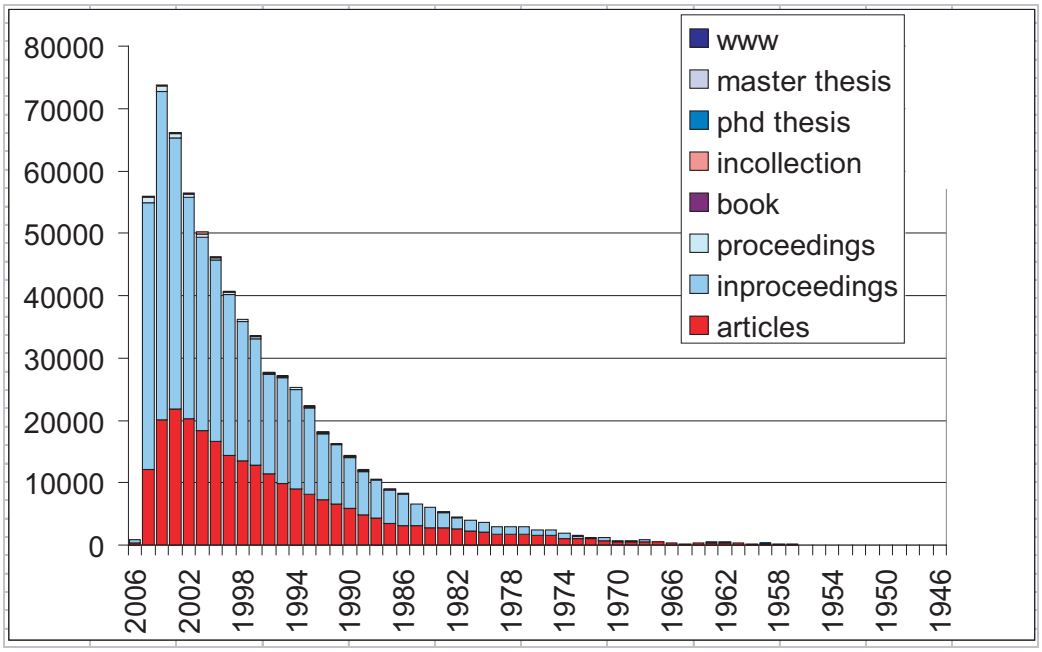

Figura 2.2: Tipi di record e quantità

Per quanto riguarda la memorizzazione dei nomi vengono concesse alcune libertà, come la possibilità di abbreviazioni, inversione di nome e cognome, nickname e accenti. Tuttavia sorgono alcuni problemi relativi alla corrispondenza biunivoca tra le persone e il relativo nome.

L'acquisizione è suddivisa in 2 step: inizialmente l'autore della pubblicazione formatta il documento secondo uno standard preciso.

Successivamente il Dr. Ley corregge eventuali errori e aggiunge ulteriori dati, se necessario. A questo segue una fase di normalizzazione dei nomi ed, inoltre, si cerca di verificare se tale persona sia già presente in DBLP (magari con un nome leggermente differente) . Mediante delle euristiche si individua il nome più vicino (in termini di somiglianza) a quello in esame.

2.2 Bibliographic Records

L'idea è stata quella di usare il formato BibTex unito alla sintassi XML.

Essa è costituita da 8 elementi principali:

article Un articolo da un giornale o da una rivista.

inproceedings Un articolo negli atti di una conferenza.

proceedings Gli atti di una conferenza.

book Un lavoro che è stampato e rilegato.

incollection La parte di un libro avente il proprio titolo.

phdthesis Una tesi di dottorato.

masterthesis Una tesi di laurea.

www Il sito internet di una persona.

Essi rappresentano le pubblicazioni che si intendono memorizzare, ognuna delle quali contiene attributi ed entità. Gli attributi sono i seguenti:

key - Required - La chiave: unica all'interno della bibliografia DBLP.

mdate - Implied - La data dell'ultima modifica.

Il record Article può disporre di due attributi supplementari:

reviewid - Implied - Chiave dell'ultima revisione.

rating - Implied - Una classificazione.

Le entità sono le seguenti:

author Il nome dell'autore della pubblicazione.

editor Il nome dell'editore di elementi di tipo Book, Incollection e Proceeding.

title Il titolo della pubblicazione.

booktitle Il titolo del libro di cui fanno parte elementi di tipo Incollection e Inproceeding.

pages Numero di pagine, separato da virgole o da doppi trattini (–).

month Il mese di pubblicazione (o, se inedito, il mese di creazione).

year L'anno di pubblicazione (o, se inedito, l'anno di creazione).

address Indirizzo dell'editore. Di solito solo la città, ma può anche essere l'indirizzo completo per gli editori meno noti.

journal La rivista nel quale l'articolo è stato pubblicato.

volume Il volume del giornale oppure del libro (se multivolume).

number Il numero del giornale, rivista o resoconto tecnico, se applicabile.

url Per Inproceedings, Article e Incollection il campo url contiene il link alla tabella in cui il volume è contenuto.

ee Contiene il collegamento verso un testo.

cdrom Usata solo per ACM SIGMOD Anthology.

cite Contiene le chiavi di altre pubblicazioni presenti nella bibliografia.

publisher Il nome della casa editrice.

note Offre la possibilità di inserire una nota.

crossref Descritto in seguito.

isbn Isbn di un libro, contenuto nell'elemento Book.

series La collana di libri all'interno della quale è stato pubblicato.

school L'istituto presso il quale è stata scritta la tesi.

chapter Il numero del capitolo.

Gli elementi cite e crossref

Il significato dell'elemento *crossref* è espresso nella Figura 2.3.

Esso contiene la chiave dell'elemento gerarchicamente superiore. In altre parole, un'elemento Inproceeding conterrà, all'interno dell'elemento crossref, la chiave (attributo *key*) dell'elemento Proceeding in cui esso è contenuto (cioè la conferenza in cui l'articolo è stato discusso).

In modo analogo può essere analizzato il significato dell'elemento *cite*. Esso contiene l'elemento gerarchicamente inferiore.

Un'elemento Proceeding, ad esempio conterrà, al suo interno, una serie di elementi cite: a loro volta essi contengono le chiavi di tutti gli elementi Inproceeding discussi in tale conferenza.

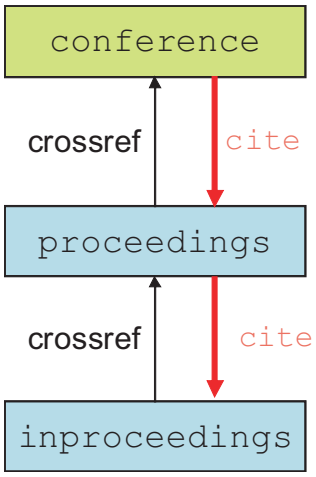

Figura 2.3: Elemento crossref

2.2.1 Record Article

```
<article mdate="2004-01-14" key="journals/amai/HaDVH98">
<author>Vu A. Ha</author>
<author>AnHai Doan</author>
<author>Van H. Vu</author>
<author>Peter Haddawy</author>
<title>Geometric Foundations for Interval-Based Probabilities.</title>
<pages>1-21</pages>
<year>1998</year>
<volume>24</volume>
<journal>Ann. Math. Artif. Intell.</journal>
```

```
<number>1-4</number>
<url>db/journals/amai/amai24.html#HaDVH98</url>
</article>
```

2.2.2 Record Inproceedings

```
<inproceedings mdate="2002-01-03" key="conf/adbt/AusielloDM82">
<author>Giorgio Ausiello</author>
<author>Alessandro D'Atri</author>
<author>Marina Moscarini</author>
<title>Minimal Coverings of Acyclic Database Schemata.</title>
<pages>27-51</pages>
<year>1982</year>
<booktitle>Advances in Data Base Theory</booktitle>
<url>db/conf/adbt/adbt82.html#AusielloDM82</url>
</inproceedings>
```

2.2.3 Record Proceedings

```
<proceedings mdate="2002-01-03" key="conf/adbt/79">
<editor>Herv&eacute; Gallaire</editor>
<editor>Jean-Marie Nicolas</editor>
<editor>Jack Minker</editor>
<title>Advances in Data Base Theory, Vol. 1, Based on the Proceedings of
the Workshop on Formal Bases for Data Bases, December 12-14, 1979, Centre
d'Etudes et de Recherches de l'Ecole Nationale Sup&eacute;rieure de
l'A&eacute;ronautique et de l'Espace de Toulouse (CERT),France</title>
<series>Advances in Data Base Theory</series>
<publisher>Plemum Press</publisher>
<address>New York</address>
<year>1981</year> <
isbn>0-306-40629-2</isbn>
<url>db/conf/adbt/adbt79.html</url>
</proceedings>
```

2.2.4 Record Book

```
<book mdate="2007-01-12" key="persons/Lions1996">
<author>John Lions</author>
<title>Lions' Commentary on UNIX 6th Edition, with Source Code</title>
<year>1996</year>
```

```
<publisher>Peer-to-Peer Communications</publisher>
<isbn>1-57398-013-7</isbn>
<url>http://www.lemis.com/grog/Documentation/Lions/index.html</url>
</book>
```

2.2.5 Record Incollection

```
<incollection mdate="2002-01-03" key="books/acm/kim95/AnnevelinkACFHK95">
<author>Jurge Annevelink</author>
<author>Rafiul Ahad</author>
<author>Amelia Carlson</author>
<author>Daniel H. Fishman</author>
<author>Michael L. Heytens</author>
<author>William Kent</author>
<title>Object SQL - A Language for the Design and Implementation of
Object Databases.</title>
<pages>42-68</pages>
<year>1995</year>
<booktitle>Modern Database Systems</booktitle>
<url>db/books/collections/kim95.html#AnnevelinkACFHK95</url>
</incollection>
```

2.2.6 Record Phdthesis

```
<phdthesis mdate="2002-01-03" key="phd/Seshadri96">
<author>Praveen Seshadri</author>
<title>Management of Sequence Data</title>
<year>1996</year>
<school>Univ. of Wisconsin-Madison, CS Department</school>
</phdthesis>
```

2.2.7 Record Masterthesis

```
<mastersthesis mdate="2002-01-03" key="phd/VanRoy84">
<author>Peter Van Roy</author>
<title>A Prolog Compiler for the PLM.</title>
<year>1984</year>
<school>University of California at Berkeley</school>
</mastersthesis>
```

2.2.8 WWW

```
<www mdate="2008-06-14" key="homepages/a/AliAmiri"> <author>Ali
Amiri</author>
<title>Home Page</title>
<url>http://spears.okstate.edu/directory/index.php?profile=amiri&action
=getprofile</url> <note>Oklohoma State University</note>
</www>
```

Capitolo 3

La base di dati

Una **base di dati** è una collezione di dati che descrivono le attività di una o più organizzazioni. Un sistema di gestione di basi di dati o **DBMS** è un software progettato per gestire grandi collezioni di dati.

L'utilizzo di un DBMS presenta molti vantaggi:

Indipendenza dei dati I programmi applicativi sono all'oscuro su come i dati siano fisicamente organizzati.

Accesso efficiente ai dati I DBMS usano tecniche avanzate per recuperare i dati in maniera efficiente.

Integrità e sicurezza dei dati Il DBMS può garantire vincoli di integrità e sicurezza.

Amministrazine dei dati I dati possono essere amministrati da sezioni apposite, in modo da diminuire la ridondanza e facilitare il reperimento.

Accesso concorrente e recupero da crash Il DBMS fa in modo che ogni utente pensi di accedere ai dati in maniera atomica. Fornisce anche delle funzionalità per il recovery

Riduzione del tempo di sviluppo delle applicazioni Un DBMS fornisce primitive comuni a molte applicazioni.

Tuttavia vi sono situazioni in cui non è consigliabile usare un DBMS, come esempio nel caso in cui vi siano vincoli di tempo ben definiti oppure quando una qualsiasi applicazione debba interagire con i dati in modo diretto.

Un **modello di dati** è una collezione di costrutti che rappresentano dati ad alto livello. Esso, inoltre, nasconde molti dei dettagli di memorizzazione

a basso livello. La maggior parte dei sistemi di gestione di basi di dati sono basati sul **modello di dati relazionale**. Una base di dati diventa allora una collezione di di una o più relazioni, dove per relazione si intende una tabella con righe e colonne. Una relazione è composta da due elementi: uno **schema relazionale** e l' **istanza della relazione**. L'istanza della relazione è una tabella, mentre lo schema relazionale descrive le intestazioni delle colonne di essa.

Lo schema specifica il nome della relazione, il nome e il dominio di ciascun campo. L'istanza, invece, è un'insieme di tuple (o record). Ciascuna tupla ha, a sua volta, lo stesso numero di campi dello schema relazionale.

Il modello Entità-Relazione ci permette di descrivere i dati coinvolti in termini di oggetti e delle loro relazioni. Esso ci consente di passare da una descrizione informale ad una più precisa e maggiormente dettagliata che può essere implementata in un DBMS.

3.1 Analisi dei requisiti

Il primo step si è concentrato sulla raccolta dei requisiti. Ciò significava comprendere come i dati di cui si dispone siano strutturati e quali di essi memorizzare.

Purtroppo la documentazione sulla struttura della bibliogradia DBLP risulta pressochè assente. Si è quindi deciso di analizzare il documento DTD[1] al fine di individuare record e relazioni tra essi.

Document Type Definition

In questa sezione verrà illustrato come l'analisi del *Document Type Definition* sia stata rilevante per la raccolta dei requisiti.

```
<!ELEMENT dblp (article|inproceedings|proceedings|book|incollection|
                phdthesis|mastersthesis|www)*>
```

Il codice precedente indica che gli elementi principali della bibliografia sono: Article, Inproceedings, Proceedings, Book, Incollection, Phdthesis, Mastersthesis, Www.

```
<!ENTITY % field "author|editor|title|booktitle|pages|year|address
|journal|volume|number|month|url|ee|cdrom|cite|publisher|note|crossref
|isbn|series|school|chapter">
```

[1] *Document Type Definition*: defisce le componenti ammesse nella costruzione di un documento XML.

Indica che ci sono una serie di Entity[2]. Tali entità memorizzano i dati relativi ad uno specifico elemento. La fase successiva è stata quella di individuare gli elementi e in che modo essi siano associati alle varie entità.

```
<!ELEMENT article       (%field;)*>
<!ATTLIST article
                        key CDATA #REQUIRED
                        reviewid CDATA #IMPLIED
                        rating CDATA #IMPLIED
                        mdate CDATA #IMPLIED
>
```

Il codice precedente mostra la struttura di un record Article. Esso può contenere un numero arbitrario delle entità precedentemente descritte. In più, ci sono quattro attributi: key, reviewid, rating ed mdate, tutti di tipo CDATA (lettere o numeri). Segue la dichiarazione delle entità:

```
<!ELEMENT title     (%titlecontents;)*>
<!ELEMENT booktitle (#PCDATA)>
<!ELEMENT pages     (#PCDATA)>
<!ELEMENT year      (#PCDATA)>
<!ELEMENT journal   (#PCDATA)>
<!ELEMENT volume    (#PCDATA)>
<!ELEMENT number    (#PCDATA)>
<!ELEMENT month     (#PCDATA)>
<!ELEMENT url       (#PCDATA)>
<!ELEMENT ee        (#PCDATA)>
<!ELEMENT cdrom     (#PCDATA)>
<!ELEMENT cite      (#PCDATA)>
<!ELEMENT school    (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ATTLIST publisher
                    href CDATA #IMPLIED
>
<!ELEMENT note      (#PCDATA)>
<!ATTLIST cite
                    label CDATA #IMPLIED
>
<!ELEMENT crossref  (#PCDATA)>
```

[2]Un token ENTITY serve a dichiarare un'entità, cioè una rappresentazione simbolica di un'informazione.

```
<!ELEMENT isbn      (#PCDATA)>
<!ELEMENT chapter   (#PCDATA)>
<!ELEMENT series    (#PCDATA)>
<!ATTLIST series
                    href CDATA #IMPLIED
>
```

Tale dichiarazione asserisce che i valori contenuti nelle entità sono di tipo PCDATA.

Inoltre viene descritta la struttura di due entità particolari: *cite* e *publisher*. Entrambe contengono un ulteriore attributo di tipo CDATA.

Il significato è il seguente:

- *cite*: contiene l'attributo *label*, un'etichetta per ragioni interne alla bibliografia DBLP.
- *publisher*: contiene l'attributo *href*, link ad una entry che contiene dati sulla casa editrice.

Esempio di record che contiene elementi di tipo Cite:

```
</book> - <book mdate="2002-01-03" key="books/aw/AbiteboulHV95">
  <author>Serge Abiteboul</author>
  <author>Richard Hull</author>
  <author>Victor Vianu</author>
  <title>Foundations of Databases.</title>
  <publisher>Addison-Wesley</publisher>
  <year>1995</year>
  <isbn>0-201-53771-0</isbn>
  <cdrom>AHV/Toc.pdf</cdrom>
  <cite label="Abi88">conf/icdt/Abiteboul88</cite>
  <cite label="Abi89">journals/ipl/Abiteboul89</cite>
  <cite label="Abr74">conf/ds/Abrial74</cite>
  <cite label="ABU79">journals/tods/AhoBU79</cite>
  <cite label="ABW88">books/mk/minker88/AptBW88</cite>
  <cite label="AC78">conf/vldb/AroraC78</cite>
  <cite label="AC89">conf/stoc/AfratiC89</cite>
  <cite label="ACO85">journals/tods/AlbanoCO85</cite>
  <cite label="ACY91">conf/pods/AfratiCY91</cite>
</book>
```

Esempio di record che contiene un'elemento di tipo Publisher:

```
</book> - <book mdate="2002-01-03" key="books/aw/ArnoldG96">
  <author>Ken Arnold</author>
  <author>James Gosling</author>
  <title>The Java Programming Language</title>
  <publisher href="db/publishers/aw.html">Addison-Wesley</publisher>
  <year>1996</year>
  <isbn>0-201-63455-4</isbn>
</book>
```

3.2 Progettazione concettuale della base di dati

Le informazioni raccolte nella fase di analisi dei requisiti vengono utilizzate per elaborare una descrizione ad alto livello dei dati da memorizzare.

Per sviluppare tale fase è stato usato il modello Entità-Relazione. Lo schema concettuale è riportato in Figura 3.1.

3.2.1 Le entità

Il primo passo è stato quello di determinare le entità coinvolte.

Un'entità è un oggetto del mondo reale che si distingue da altri oggetti; essa viene descritta servendosi di un insieme di attributi.

Sulla base delle informazioni raccolte nella fase di analisi, sono state, successivamente individuate le entità con i relativi attributi.

Gli attributi sottolineati rappresentano le chiavi delle relazioni che li contengono.

Authors Memorizza i nomi degli attori coinvolti nelle pubblicazioni. Si è deciso di considerare un'entità (rispetto al documento XML in cui è considerata un attributo) proprio perchè un'autore può partecipare a più pubblicazioni.

- *name* : Elemento *author* del record XML.

Editors Nomi degli editori.

- *name* : Elemento *editor* del record XML.

Publishers Case editrici e relativo indirizzo.

- *name* : Elemento *publisher* del record XML.

- *href* : Attributo *href* del record XML.
- *address* : Elemento *address* del record XML.

Articles Memorizza gli elementi di tipo Article.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *reviewid* : Attributo *reviewid* del record XML.
- *rating* : Attributo *rating* del record XML.
- *title* : Elemento *title* del record XML.
- *url* : Elemento *url* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *crossref* : Elemento *crossref* del record XML.
- *ee* : Elemento *ee* del record XML.
- *note* : Elemento *note* del record XML.

Journal Memorizza i dati di una rivista nella quale un determinato articolo è stato pubblicato.

- *name* : Elemento *journal* del record XML.
- *volume* : Elemento *volume* del record XML.
- *number* : Elemento *number* del record XML.

Proceedings Memorizza gli elementi di tipo Proceeding.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *url* : Elemento *url* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *isbn* : Elemento *isbn* del record XML.
- *ee* : Elemento *ee* del record XML.
- *note* : Elemento *note* del record XML.

Inproceedings Memorizza gli elementi di tipo Inproceeding.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *url* : Elemento *url* del record XML.
- *crossref* : Elemento *crossref* del record XML.
- *booktitle* : Elemento *booktitle* del record XML.
- *pages* : Elemento *pages* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *cdrom* : Elemento *cdrom* del record XML.
- *ee* : Elemento *ee* del record XML.
- *note* : Elemento *note* del record XML.

Books Memorizza gli elementi di tipo Book.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *isbn* : Elemento *isbn* del record XML.
- *cdrom* : Elemento *cdrom* del record XML.
- *url* : Elemento *url* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *note* : Elemento *note* del record XML.

Incollections Memorizza gli elementi di tipo Incollection.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *crossref* : Elemento *crossref* del record XML.
- *booktitle* : Elemento *booktitle* del record XML.

- *chapter* : Elemento *chapter* del record XML.
- *pages* : Elemento *pages* del record XML.
- *url* : Elemento *url* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *ee* : Elemento *ee* del record XML.
- *note* : Elemento *note* del record XML.

Thesis Essendo le tesi di laurea e di dottorato sottoclassi dell' entità Thesis, si è preferito effettuare una generalizzazione.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *school* : Elemento *school* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *note* : Elemento *note* del record XML.

Master_Thesis Memorizza gli elementi di tipo Master_thesis, generalizzato in Thesis.

Phd_Thesis Memorizza gli elementi di tipo Phd_thesis, generalizzato in Thesis.

WWW Memorizza gli elementi di tipo www.

- *id* : Attributo *key* del record XML.
- *mdate* : Attributo *mdate* del record XML.
- *title* : Elemento *title* del record XML.
- *url* : Elemento *url* del record XML.
- *month* : Elemento *month* del record XML.
- *year* : Elemento *year* del record XML.
- *note* : Elemento *note* del record XML.

Series Memorizza il nome di una Serie in cui gli atti di una conferenza Proceeding possono essere pubblicati.

- *name* : Elemento *series* del record XML.

Cites Un elemento può citare anche altre pubblicazioni. Di conseguenza è stata creata un'entità Cites che contiene le chiavi degli elementi citati e il testo corrispondente.

- *label* : Attributo *label* del record XML.
- *text* : Elemento *text* del record XML.

3.2.2 Le relazioni

Una relazione è un' associazione tra due o più entità. Nello schema E-R ad esempio, constatiamo che un'articolo è contenuto in una rivista mediante la relazione *In*.

Una relazione,inoltre, può anche avere attributi descrittivi. Essi vengono utilizzati per registrare informazioni sulla relazione piuttosto che sulle entità partecipanti.Ad esempio, la relazione *In* ha l'attributo *Pages*, quest'ultimo indica le pagine della rivista in cui è contenuto quel determinato articolo.

Sulla base delle informazioni raccolte nella fase di analisi sono state individuate le seguenti relazioni:

In - Journal (0,n) ↔ (1,1) Articles : asserisce che ogni articolo è contenuto in una specifica rivista in determinate pagine. Ogni rivista, in determinate pagine di una specifica rivista contiene quindi uno solo articolo.

- *pages* : Elemento *pages* del record XML.

In_a - Proceedings (0,1) ↔ (0,1) Series : asserisce che una conferenza può essere contenuta in uno specifico volume di una determinata serie. Ciascun volume di una serie contiene una sola conferenza.

- *volume* : Elemento *volume* del record XML.

Cited_1 - Cites (1,1) ↔ (0,n) Books : asserisce che un libro può avere un numero indeterminato di citazioni, mentre ogni citazione è riferita ad uno specifico libro.

Cited_2 - Cites (1,1) ↔ (0,n) Proceedings : asserisce che una conferenza può avere un numero indeterminato di citazioni, mentre ogni citazione è riferita ad una specifica conferenza.

Published_1 - Proceedings (0,1) ↔ (0,n) Publishers : asserisce che gli atti di una conferenza possono essere pubblicati da una sola casa editrice. D'altro canto, una casa editrice può pubblicare un numero indeterminato di atti di conferenze.

Published_2 - Books (0,1) ↔ (0,n) Publishers : asserisce che un libro può essere pubblicato da una sola casa editrice, ma, una casa editrice può pubblicare un numero indeterminato di libri.

Edited_1 - Proceedings (0,n) ↔ (0,n) Editors : Asserisce che una conferenza può essere editata da un numero indeterminato di editori, e che un'editore può editare un numero indeterminato di conferenze.

Edited_2 - Incollections (0,n) ↔ (0,n) Editors : Asserisce che la parte di un libro può essere editato da un numero indeterminato di editori, e che un'editore può editare un numero indeterminato di parti di libri.

Edited_3 - Books (0,n) ↔ (0,n) Editors : Asserisce che un libro può essere editato da un numero indeterminato di editori, e che un'editore può editare un numero indeterminato di libri.

By_1 - Articles (1,n) ↔ (0,n) Authors : Asserisce che ogni articolo deve avere almeno un'autore. Ogni autore può partecipare ad un numero indeterminato di articoli.

By_2 - WWW (1,n) ↔ (0,n) Authors : Asserisce che ogni sito web appartiene almeno ad un'autore. Ogni autore può possedere un numero indeterminato di siti web.

By_3 - Books (1,n) ↔ (0,n) Authors : Asserisce che ogni libro deve avere almeno un'autore. Ogni autore può partecipare ad un numero indeterminato di libri.

By_4 - Incollections (1,n) ↔ (0,n) Authors : Asserisce che un'entità Incollection deve avere almeno un'autore. Ogni autore può partecipare ad un numero indeterminato di entità Incollection.

By_5 - Thesis (1,n) ↔ (0,n) Authors : Asserisce che ogni tesi deve avere almeno un'autore. Ogni autore può partecipare ad un numero indeterminato di tesi.

By_6 - Inproceedings (1,n) ↔ (0,n) Authors : Asserisce che ogni articolo discusso in una conferenza deve avere almeno un'autore. Ogni autore può partecipare ad un numero indeterminato di articoli discussi in una conferenza.

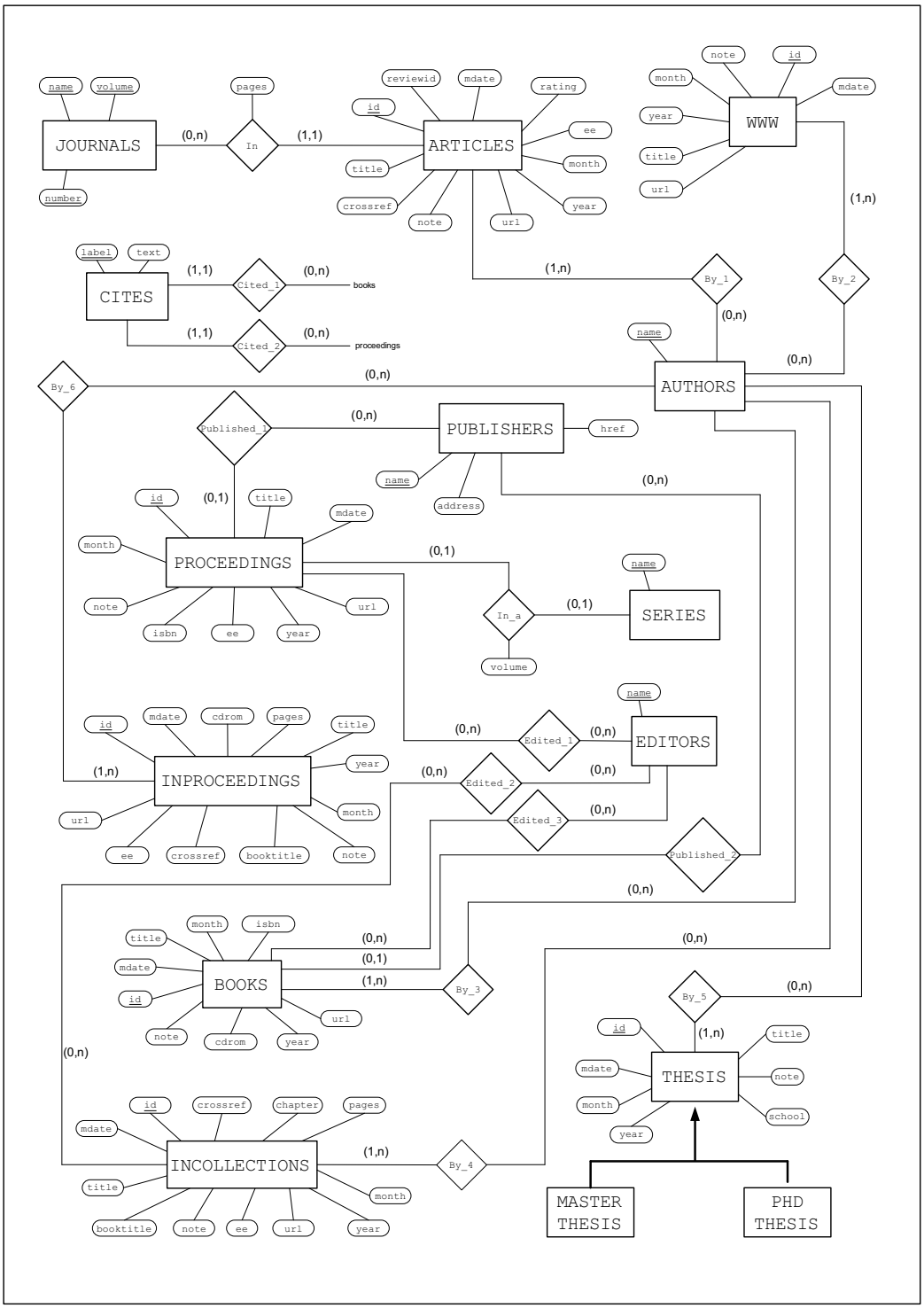

Figura 3.1: Schema concettuale

3.2.3 Osservazioni

Durante la progettazione concettuale sono state valutate possibili scelte alternative. Alcune si esse prevedevano accorgimenti che avrebbero dato maggiore qualità alla base di dati.

Una proposta era quella di mettere in relazione l'entità *Incollections* con l'entità *Book* mediante la relazione *In_book*.

In questo modo ogni tupla di Incollection era legata, mediante la relazione *In_Book*, al libro che effettivamente la contiene. Ciò avrebbe reso molto più naturale la definizione di Incollection. Inoltre, alcuni attributi di Incollections come **Bookname**, sarebbero stati superflui, e avrebbero diminuito di conseguenza la ridondanza dei dati.

Si è pensato, quindi, di ricorrere all'uso dell'elemento *crossref*. Mediante un'analisi approfondita dei dati, è emerso che l'elemento crossref non è sempre presente. Di conseguenza non è sempre possibile la referenziazione.

Un'altra alternativa era quella di cercare il libro relativo ad un'elemento Incollection, ciò è possibile sfruttando la caratteristica: l'attributo *Booktitle* dell'entità Incollection corrisponde all'attributo *Title* dell'entità Book.

Ciò è molto dispendioso dal punto di vista computazionale, ed è impossibile effettuare query in un documento sql[3].

3.3 Progettazione logica della base di dati

In questa fase viene scelto il DBMS per implementare il progetto, e convertire la progettazione concettuale in uno schema del modello del DBMS scelto. Il risultato è uno **schema logico** nel modello di dati relazionale.

Ogni insieme di entità viene tradotta in una relazione (o tabella) nel seguente modo: ogni attributo di un insieme di entità diventa un attributo della tabella. Un insieme di relazioni, così come le entità, viene tradotto in una relazione del modello relazionale. Gli attributi della relazione includono:

- Gli attributi della chiave primaria di ciascun insieme di entità partecipante, come campi di chiave esterne.
- Gli attributi descrittivi dell'insieme di relazioni.

3.3.1 Le relazioni

Di seguito verranno elencate le relazioni della base di dati con relative istruzioni SQL per la loro creazione.

[3]Nel caso in cui si scelga che l'output sia il documento di istruzioni sql

Authors(id, name) Poichè l'entità *Authors* è composta da un solo attributo, è stato scelto di integrare le relazioni *By_1, By_2, By_3, By_4, By_5, By_6* direttamente nella relazione *Authors*. Il vincolo di integrità imposto dalle cardinalità è rispettato per il fatto che sia **id** che **name** sono chiave per la relazione. In questo modo possono esistere 2 tuple con lo stesso autore, ma che differiscono per il campo id (un autore è responsabile di più pubblicazioni). Inoltre, una pubblicazione può avere più autori (stesso valore **id** ma diverso valore **name**). Non è invece possibile avere tuple il cui campo **id** è diverso da *null* ma il campo **name** è *null* (significherebbe che una pubblicazione non ha autori, violando quindi la cardinalità). Comandi SQL per la creazione della tabella:

```
CREATE TABLE authors(id VARCHAR(150),
                     name VARCHAR(100),
                     PRIMARY KEY (id, name))
                     ENGINE=InnoDB
```

Editors(id, name) Anche per la relazione Editors è stato scelto di integrare le relazioni *Edited_1*, *Edited_2* ed *Edited_3* direttamente nella relazione stessa. Le considerazioni sul rispetto dei vincoli di integrità sono identici al caso della relazione Authors, e non verranno discussi. Comandi SQL per la creazione della tabella:

```
CREATE TABLE editors(id VARCHAR(150),
                      name VARCHAR(100),
                      PRIMARY KEY (id, name))
                      ENGINE=InnoDB
```

Articles(id, mdate, reviewid, rating, title, url, month, year, crossref, ee, note) Tabella per la memorizzazione di articoli. Comandi SQL per la creazione della tabella:

```
CREATE TABLE articles(id VARCHAR(150),
                      mdate VARCHAR(20),
                      reviewid VARCHAR(20),
                      rating VARCHAR(10),
                      title VARCHAR(400),
                      url VARCHAR(100),
                      month VARCHAR(10),
```

```
                    year VARCHAR(10),
                    crossref VARCHAR(100),
                    ee VARCHAR(100),
                    note VARCHAR(20),
                    PRIMARY KEY(id))
                    ENGINE=InnoDB
```

Proceedings(id, mdate, title, url, month, year, isbn, ee, note) Tabella per la memorizzazione di elementi Proceeding. Comandi SQL per la creazione della tabella:

```
CREATE TABLE proceedings(id VARCHAR(150),
                         mdate VARCHAR(20),
                         title VARCHAR(500),
                         url VARCHAR(200),
                         month VARCHAR(10),
                         year VARCHAR(10),
                         isbn VARCHAR(100),
                         ee VARCHAR(100),
                         note VARCHAR(20),
                         PRIMARY KEY(id))
                         ENGINE=InnoDB
```

Inproceedings(id, mdate, title, url, crossref, booktitle, pages, month, year, cdrom, ee, note) Tabella per la memorizzazione di elementi Inproceeding. Comandi SQL per la creazione della tabella:

```
CREATE TABLE inproceedings(id VARCHAR(150),
                           mdate VARCHAR(20),
                           title VARCHAR(400),
                           url VARCHAR(200),
                           crossref VARCHAR(30),
                           booktitle VARCHAR(250),
                           pages VARCHAR(10),
                           month VARCHAR(10),
                           year VARCHAR(10),
                           cdrom VARCHAR(20),
                           ee VARCHAR(100),
                           note VARCHAR(20),
```

```
                                PRIMARY KEY(id))
                                ENGINE=InnoDB
```

Incollections(<u>id</u>, mdate, title, crossref, booktitle, chapter, pages, url, month, year, ee, note) Tabella per la memorizzazione di elementi Incollection. Comandi SQL per la creazione della tabella:

```
CREATE TABLE incollections(id VARCHAR(150),
                           mdate VARCHAR(20),
                           title VARCHAR(400),
                           crossref VARCHAR(100),
                           booktitle VARCHAR(250),
                           chapter VARCHAR(10),
                           pages VARCHAR(10),
                           url VARCHAR(200),
                           month VARCHAR(10),
                           year VARCHAR(10),
                           ee VARCHAR(100),
                           note VARCHAR(300),
                           PRIMARY KEY(id))
                           ENGINE=InnoDB
```

Master_Thesis(<u>id</u>, mdate, title, school, month, year, note) Tabella per la memorizzazione di tesi di laurea. Comandi SQL per la creazione della tabella:

```
CREATE TABLE master_thesis(id VARCHAR(150),
                           mdate VARCHAR(20),
                           title VARCHAR(400),
                           school VARCHAR(150),
                           month VARCHAR(10),
                           year VARCHAR(10),
                           note VARCHAR(300),
                           PRIMARY KEY(id))
                           ENGINE=InnoDB
```

Phd_Thesis(<u>id</u>, mdate, title, school, month, year, note) Tabella per la memorizzazione di tesi di dottorato. Comandi SQL per la creazione della tabella:

```
CREATE TABLE phd_thesis(id VARCHAR(150),
                        mdate VARCHAR(20),
                        title VARCHAR(400),
                        school VARCHAR(150),
                        month VARCHAR(10),
                        year VARCHAR(10),
                        note VARCHAR(300),
                        PRIMARY KEY(id))
                        ENGINE=InnoDB
```

Books(id, mdate, title, isbn, cdrom, url, month, year, note) Tabella per la memorizzazione di libri. Comandi SQL per la creazione della tabella:

```
CREATE TABLE books(id VARCHAR(150),
                   mdate VARCHAR(20),
                   title VARCHAR(400),
                   isbn VARCHAR(20),
                   cdrom VARCHAR(20),
                   url VARCHAR(200),
                   month VARCHAR(10),
                   year VARCHAR(10),
                   note VARCHAR(300),
                   PRIMARY KEY(id))
                   ENGINE=InnoDB
```

Publishers(name, address, href) Tabella per la memorizzazione delle case editrici. Comandi SQL per la creazione della tabella:

```
CREATE TABLE publishers(name VARCHAR(100),
                        address VARCHAR(50),
                        href VARCHAR(50),
                        PRIMARY KEY(name))
                        ENGINE=InnoDB
```

WWW(id, mdate, title, url, month, year, note) Tabella per la memorizzazione di elementi www. Comandi SQL per la creazione della tabella:

```
CREATE TABLE www(id VARCHAR(150),
                 mdate VARCHAR(20),
                 title VARCHAR(400),
                 url VARCHAR(200),
                 month VARCHAR(10),
                 year VARCHAR(10),
                 note VARCHAR(300),
                 PRIMARY KEY(id))
                 ENGINE=InnoDB
```

Cites(id, label, text) E' stato scelto di integrare le relazioni *Cited_1* e *Cited_2* dello schema E-R direttamente nella tabella Cites. L E stato aggiunto un'ulteriore campo, che corrisponde all'attributo **id** delle relazioni *Books* e *Proceedings*. Le cardinalità sono rispettate dalla condizione che sia **id** che **label** sono chiave per la relazione. Comandi SQL per la creazione della tabella:

```
CREATE TABLE cites(id VARCHAR(150),
                   label VARCHAR(100),
                   text VARCHAR(300),
                   PRIMARY KEY(id, label))
                   ENGINE=InnoDB
```

Published_1(id, name) Tabella per la memorizzazione della relazione *Published_1*. Sia **id** che **name** sono chiave per la relazione. Il campi **id** referenzia Proceedings mentre il campo **name** referenzia publisher. In questo modo viene garantito il vincolo di integrità imposto dalle cardinalità: gli elementi di una conferenza possono essere pubblicati da una sola casa editrice. Comandi SQL per la creazione della tabella:

```
CREATE TABLE published_1(id VARCHAR(150),
                         name VARCHAR(100),
                         PRIMARY KEY (id, name),
                         FOREIGN KEY (name)
                         REFERENCES Publishers(name),
                         FOREIGN KEY (id)
                         REFERENCES Proceedings(id))
                         ENGINE=InnoDB
```

Published_2(<u>id</u>, <u>name</u>) Tabella per la memorizzazione della relazione *Published_2*. Stesse considerazioni della tabella Published_1. Comandi SQL per la creazione della tabella:

```
CREATE TABLE published_2(id VARCHAR(150),
                         name VARCHAR(100),
                         PRIMARY KEY (id, name),
                         FOREIGN KEY (name)
                         REFERENCES Publishers(name),
                         FOREIGN KEY (id)
                         REFERENCES Books(id))
                         ENGINE=InnoDB
```

In_a(<u>id</u>, <u>name</u>, volume) Poichè l'entità *Series* ha un solo attributo, è stato optato di integrare l'entità in questione con la relazione *In_a*. Le chiavi della relazione sono *id* (contiene la chiave di un'entry *Proceedings*) e *name* (nome della serie). Le cardinalità sono rispettate poichè gli atti di una conferenza sono contenuti in un volume di una sola serie (non possono esistere due entry con stessi valori di id e name). Comandi SQL per la creazione della tabella:

```
CREATE TABLE in_a(id VARCHAR(150),
                  name VARCHAR(100),
                  volume VARCHAR(20),
                  PRIMARY KEY (id, name),
                  FOREIGN KEY (id)
                  REFERENCES Proceedings(id),
                  FOREIGN KEY (name)
                  REFERENCES Series(name))
                  ENGINE=InnoDB
```

In_1(<u>id</u>, <u>name</u>, <u>volume</u>, <u>number</u>, pages) Anche in questo caso è stato scelto di integrare l'entità *Journal* con la relazione *In_1*. Tale scelta è motivata dal fatto che la relazione tra Journals e Articlesè 1 ad 1, ciò significa che nella relazione In_1 non esisteranno 2 tuple con stessi valori. Di conseguenza, il vantaggio di modellare una relazione dello schema E-R come tabella veniva a cadere. Modellare **id**, **name**, **volume** e **number** come chiavi, implica che, in determinate pagine di un volume di una rivista è contenuto un solo articolo (rispettando quindi le cardinalità). Comandi SQL per la creazione della tabella:

```
CREATE TABLE in_1(id VARCHAR(150),
                  name VARCHAR(150),
                  volume VARCHAR(20),
                  number VARCHAR(30),
                  pages VARCHAR(30),
                  PRIMARY KEY (id, name, volume, number))
                  ENGINE=InnoDB
```

Capitolo 4

L'applicazione DBLPConverter

La seconda fase è stata quella di progettare e sviluppare un'applicazione che permettesse la creazione e il popolamento del database, a partire dei dati contenuti nella bibliografia DBLP. A questa applicazione è stato dato il nome di *DBLPConverter*.

4.1 Requisiti

I requisiti dell'applicazione sono:

1. Linguaggio indipendente dalla piattaforma hardware/software.
2. Apertura e successiva lettura del file XML della bibliografia.
3. Due modalità di output:
   - Creazione di un file contenente i comandi SQL per la creazione e popolamento del database con i dati della bibliografa.
   - Connessione ad un DBMS, creazione delle tabelle e popolamento del database con i dati della bibliografia.
4. Nel caso di connessione al DBMS, possibilità di interagire con esso (creazione tabelle, drop tabelle e cancellazione tuple).
5. Enumerazione di elementi, attributi e caratteri contenuti nella bibliografia.
6. Mostrare lo stato del processo di conversione (progress bar) e il tempo trascorso.

7. Possibilità di scegliere quale tipo di elementi aggiungere al database (Article, Proceedings, Inproceeding, Book, Incollection, Master_thesis, Phd_thesis, WWW).

4.2 Architettura

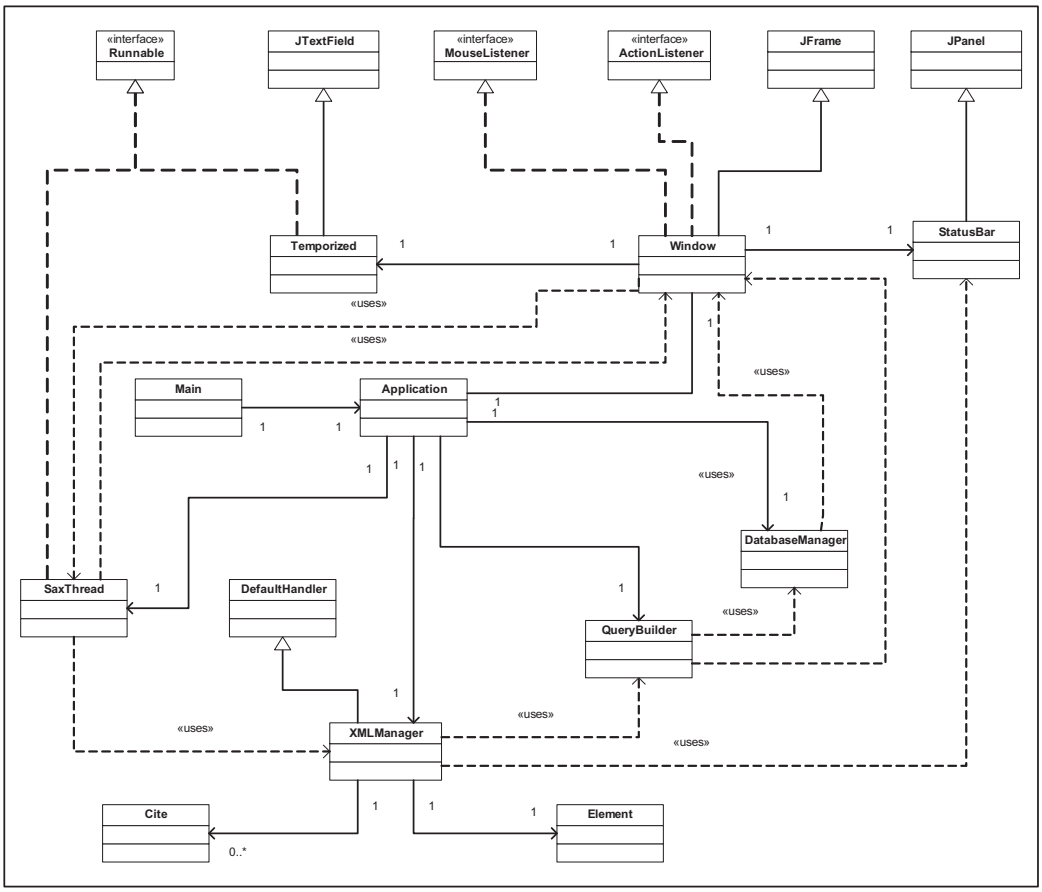

Figura 4.1: Class Diagram

La figura Figura 4.1 mostra il diagramma delle classi dell'applicazione *DBLPConverter*. Essa fa uso di alcune classi ed interfaccie esterne.

4.2.1 Descrizione delle classi

Classe Main

Classe obbligatoria, viene instanziata nel momento in cui si avvia l'applicazione. Tale classe si occupa di instanziare la classe *Application*.

Classe Application

Classe principale dell'applicazione. Si occupa di instanziare le classi: *Window*, *XMLManager*, *SaxThread*, *XMLManager*, *DatabaseManager* e *QueryBuilder*.

Classe SaxThread

Classe che si occupa di gestire *Xerces-C++ XML Parser*. Essa estende l'interfaccia *Runnable*, facendo in modo che, una volta eseguito il metodo Run, tale oggetto diviene un thread separato dall'applicazione. Questo è obbligatorio nel caso si vogliano gestire risorse esterne (nel nostro caso Sax2Count e Sax2Print) senza compromettere il funzionamento dell'applicazione.

Classe XMLManager

La classe in esame si occupa della gestione degli elementi XML.

Classe Element

Tale classe è sostanzialmente una struttura dati per la memorizzazione dei vari elementi XML contenuti in un elemeto principale. Tali elementi vengono poi passati alla classe che si occupa di gestire il database. Tale classe contiene inoltre un vettore, per la memorizzazione degli oggetti di tipo Cite.

Classe Cite

Classe che si occupa di memorizzare un elemento di tipo Cite.

Classe QueryBuilder

Classe che si occupa di creare le query SQL. Tali query vengono generate dinamicamente in base all'elemento e ai valori da aggiungere.

Classe DatabaseManager

Classe che offre le primitive a basso livello per poter effettuare interrogazioni sul database, oppure scrivere sul file .sql .

Classe Window

Classe che implementa la GUI (Graphics User Interface). Mette a disposizione verso l'utente tutti i comandi per poter interagire con l'applicazione *DBLPConverter*.

Classe StatusBar

Classe per la gestione della barra di avanzamento per indicare lo stato del progresso durante l'esecuzione. Essa rapporta il numero di elementi analizzati con quelli ancora da analizzare (calcolati tramite l'applicazione SaxCounter).

Classe Temporized

Classe che si occupa di gestire il timer. Come per la classe *SaxThread*, anche questa implementa l'interfaccia Runnable, per rendere autonomo l'avanzamento del timer.

4.2.2 Analisi dell'esecuzione

In questa sezione viene descritta in dettaglio l'esecuzione del task di analisi e memorizzazione di un elemento. Consideriamo a titolo di esempio il seguente elemento article:

```
<article mdate="2004-01-14" key="journals/amai/GrumbachL97">
<author>Zo&egrave; Lacroix</author>
<title>On Non-Determinism in Machines and Languages.</title>
<pages>169-213</pages>
<year>1997</year>
<volume>19</volume>
<journal>Ann. Math. Artif. Intell.</journal>
<number>1-2</number>
<url>db/journals/amai/amai19.html#GrumbachL97</url>
</article>
```

La Figura 4.2 mostra in che modo l'applicazione interagisce con i dati, e tutte le classi coinvolte. Senza perdere di generalità, non vengono considerate tutte le classi auiliarie, come ad esempio *Window* oppure *StatusBar*.

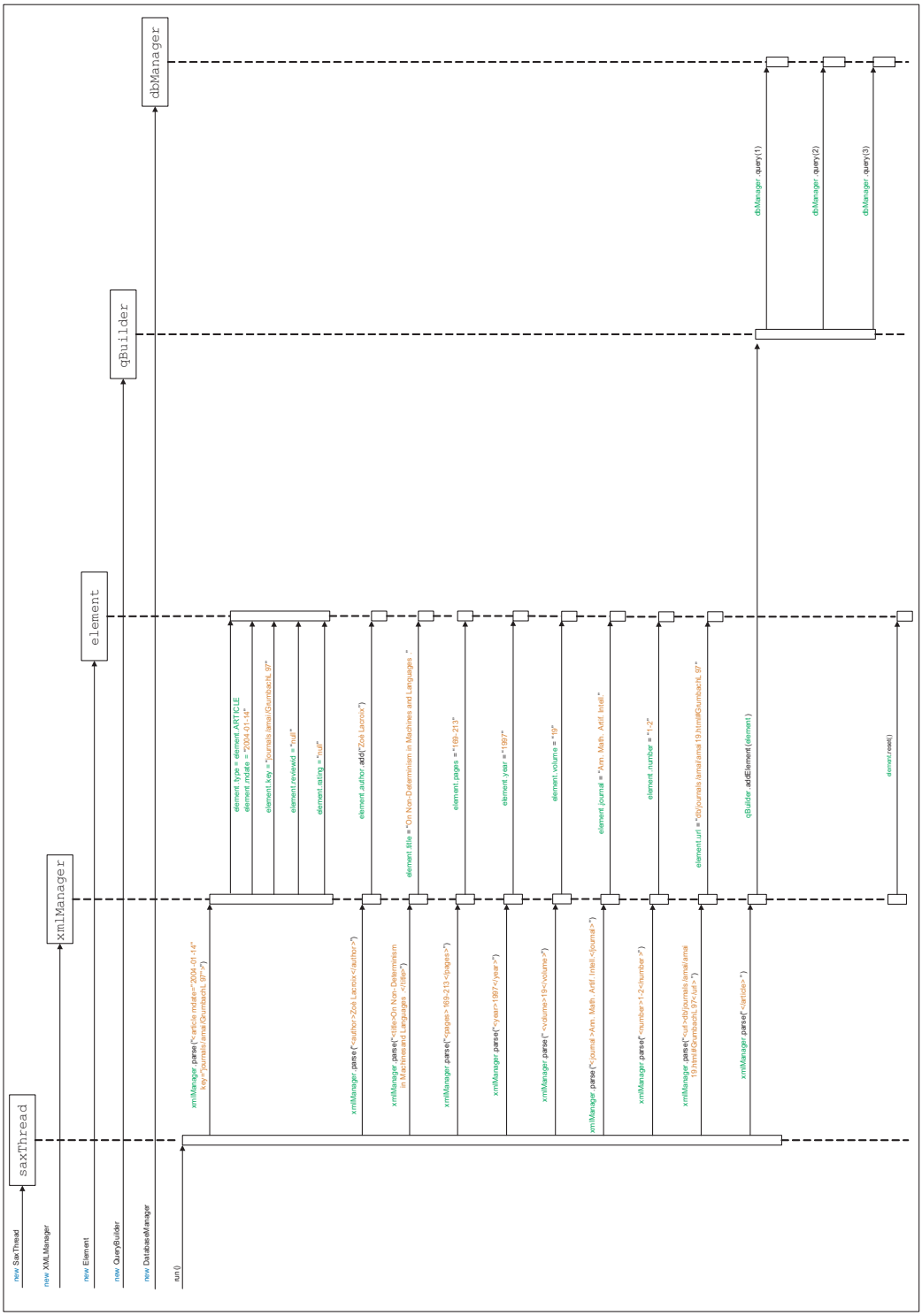

Figura 4.2: Esecuzione dell'applicazione

Inizialmente la classe *Application* instanzia le classi: *SaxThread*, *XMLManager*, *Element*, *QueryBuilder* e *DatabaseManager*. Una volta specificato il documento XML e ricevuto il comando di *parse()* l'oggetto saxThread (che a sua volta usa l'applicazione *SaxPrint*) inizia la lettura del file per elemento. Ogni elemento letto viene passato all'oggetto xmlManager, che a sua volta esegue il parsing vero e proprio, riconoscendo così gli attributi dell'elemento article (key, mdate, reviewid e rating) e gli elementi in esso contenuti (author, title, pages, year, volume, journal, number e url). I valori di questi elementi vengono memorizzati nell'oggetto *element*, che provvederà a tenerli temporaneamente memorizzati. Una volta che *xmlManager* rileva il tag di chiusura dell'elemento principale(</article>) provvederà a passare l'oggetto *element* all'oggetto *qBuilder*, che a sua volta genererà dinamicamente le query a partire dai valori contenuti in *element*. L'oggetto *qBuilder* userà i metodi messi a disposizione dall'oggetto *dbManager* per l'aggiunta dei valori al database, o per la scrittura delle query su file.

Per questioni di spazio, la Figura 4.2 non contiene le query sql (ad esse è assegnato per semplicità un numero), verranno quindi riportate di seguito:

1 :

```
INSERT INTO articles(id, mdate, reviewid, rating, title,
url, month, year, crossref, ee, note)
VALUES('journals/amai/GrumbachL97', '2004-01-14', 'null',
'null', On Non-Determinism in Machines and Languages.',
'db/journals/amai/amai19.html#GrumbachL97', 'null',
'1997', 'null', 'null', 'null')
```

2 :

```
INSERT INTO in_1(id, name, volume, number, pages)
VALUES('journals/amai/GrumbachL97',
'Ann. Math. Artif. Intell.', '19',
'1-2', '169-213')
```

3 :

```
INSERT INTO authors(id, name)
VALUES('journals/amai/GrumbachL97', 'Zoè Lacroix')
```

Da aggiungere infine che Java effettua automaticamente il riconoscimento delle lettere accentate.

4.3 Descrizione dei componenti

In questa sezione verranno discussi tutti i componenti messi a disposizione all'utente. La Figura 4.3 mostra l'interfaccia grafica dell'applicazione.

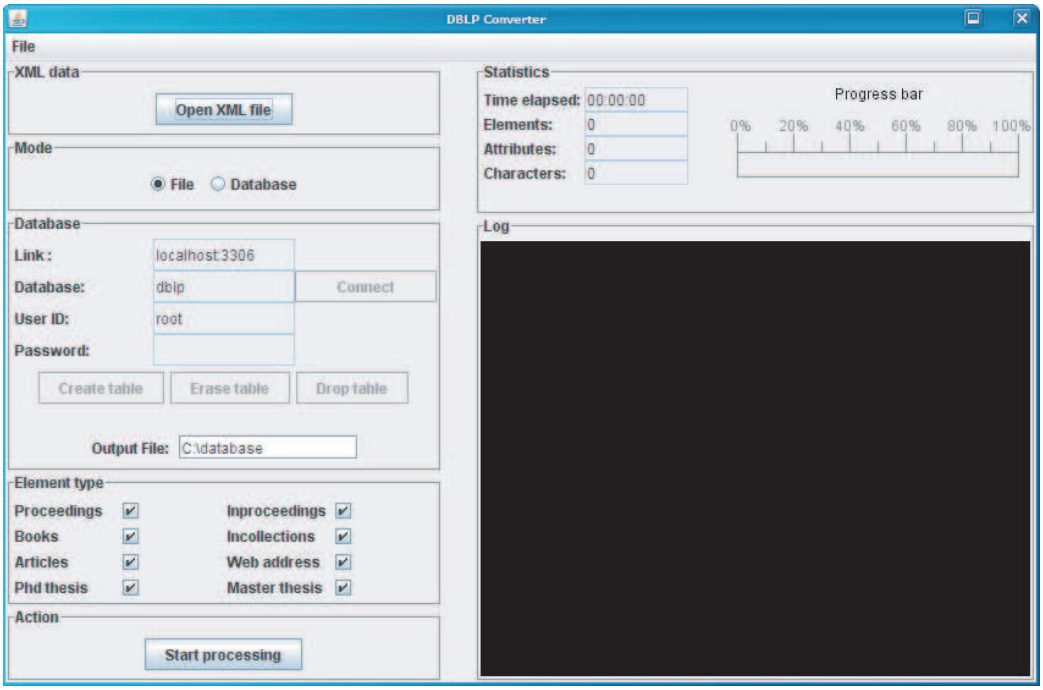

Figura 4.3: Interfaccia grafica dell'applicazione

Verranno elencati tutti i pannelli con le relative funzionalità:

XML Data Pannello che permette di specificare il nome e la posizione del file XML.

Mode Questo pannello offre la possibilità di scegliere l'output dell'applicazione: database o documento sql.

Database Pannello dedicato alla gestione del database. E' possibile specificare: indirizzo del DBMS, nome del database, nome utente e password. Sono disponibili inoltre alcuni pulsanti che permettono di Creare le tabelle, cancellare le tuple delle tabelle (erase) e rimozione delle tabelle (drop). Vi è infine un text field per specificare il percorso del file sql.

Element type In queso pannello è possibile specificare i tipi degli elementi da aggiungere al database.

Action Pannello che contiene il pulsante per avviare o interrompere il processo di parsing.

Statistics Pannello dedicato alle informazioni analitiche: numero di elementi, numero d attributi, numero di caratteri e stato di avanzamento del processo.

Log Log dell'applicazione.

La Figura 4.4 mostra l'esecuzione dell'applicazione DBLPConverter.

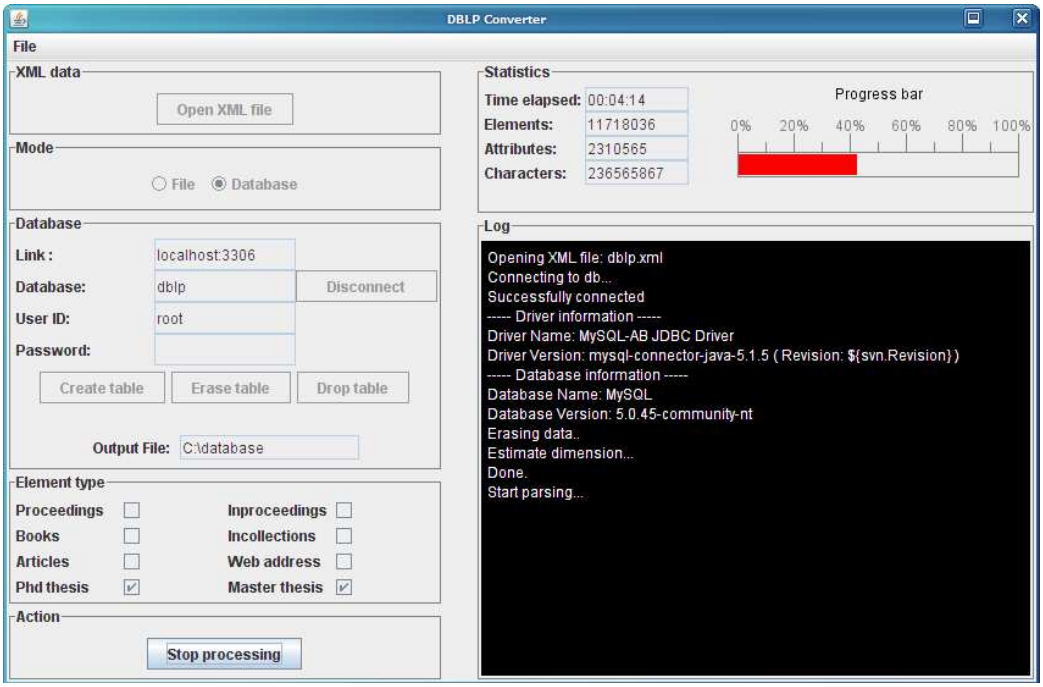

Figura 4.4: Esecuzione dell'applicazione BLPConverter

4.4 Problematiche riscontrate

In questa sezione verrà discusso per quale motivo si è ricorso all'uso degli applicativi Sax2Count e Sax2Print.

Inizialmente, l'applicazione DBLPConverter utilizzava i metodi messi a disposizione dal linguaggio Java per l'apertura e la gestione del file XML. Durante il parsing del file, è però emerso un problema relativo all'occupazione di memoria centrale, che, andava lentamente saturandosi. Questo rendeva

quindi impossibile analizzare completamente il file. Si è ricorso allora agli applicativi citati precedentemente, essi gestiscono l'apertura del file e la lettura (quest'ultima avviene per elemento) delegando a Java la gestione degli elementi. La Figura 4.5 mostra l'output dell'applicazione Sax2Count con input la bibliografia DBLP. Mentre la Figura 4.6 mostra l'output dell'applicazione Sax2Print con il medesimo input di Sax2Count.

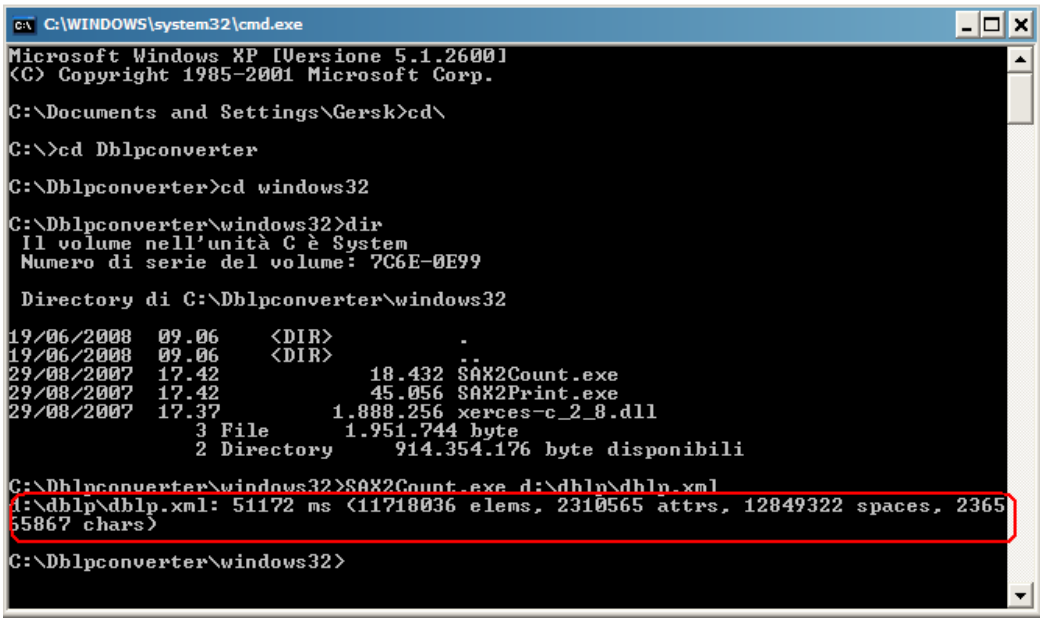

Figura 4.5: Esecuzione di Sax2Count

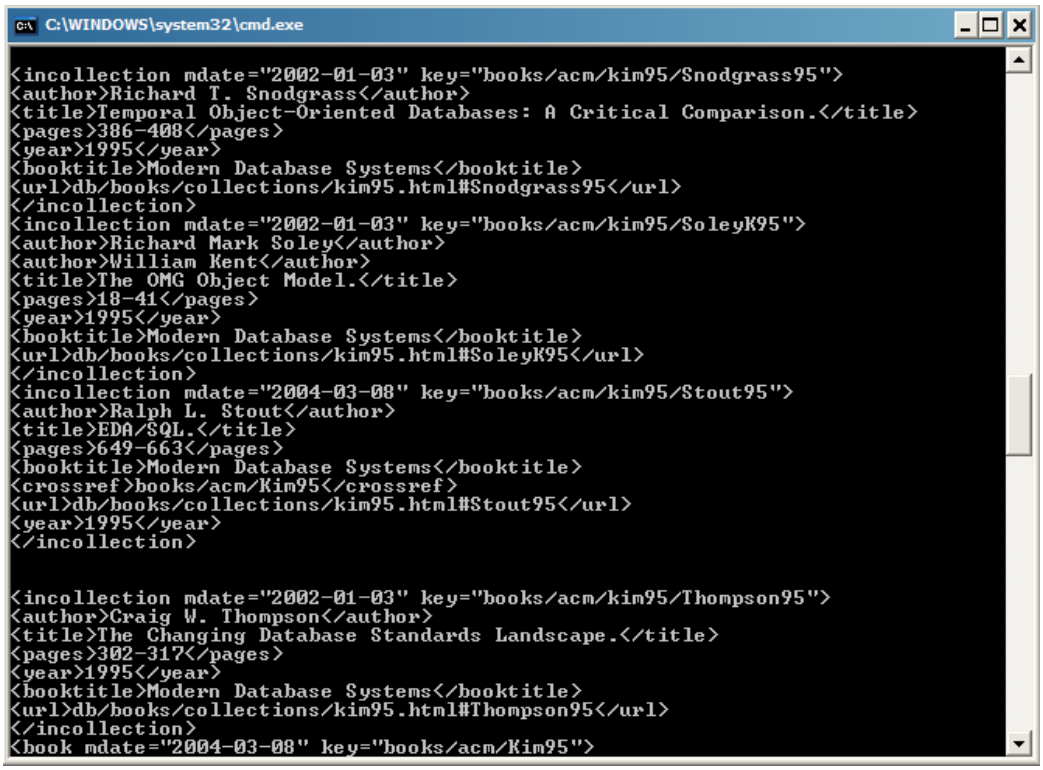

Figura 4.6: Esecuzione di Sax2Printl

4.5 Tecnologie adottate

L'applicazione è stata sviluppata utilizzando il linguaggio *Java*; la versione della JDK[1] è la 1.5. L'utilizzo di Java ha permesso di rispettare il requisito 1. Il DBMS utilizzato è *MySQL* versione 1.2.12. Per permettere la connessione tra applicazione e DBMS viene usata la libreria *Connector/J* versione 5.1.5 fornita da MySQL. Per l'analisi e il parsing del file XML è stato usato *Xerces-C++ XML Parser* versione 2.8.0.

[1]Java Development Kit

Capitolo 5

Ontologia

Un ontologia è una descrizione formale esplicita dei concetti di un dominio. Tramite un ontologia è possibile rappresentare informazioni da un punto di vista concettuale, in termini di classi, concetti, entità e relazioni tra essi.

Un ontologia viene espressa in Logica:

- Logica del primo ordine.
- Logica descrittiva: un'arricchimento della logica del primo ordine per esprimere la conoscenza (in termini di classi e relazioni).

L'accesso ai dati basato sull'ontologia permette di nascondere all'utente dove e come i dati sono fisicamente memorizzati. Fornisce una vista concettuale, corredata di un formalismo semantico.

5.1 Ontologia della bibliografia DBLP

In questa sezione verrà descritta l'ontologia della bibliografia DBLP, mostrata in Figura 5.1.

Essa riflette quello che è il significato della bibliografia, e cioè, la memorizzazione di tutte le pubblicazioni riguardanti o inerenti l'informatica. Nell'ontologia, queste informazioni sono rappresentare tramite concetti, attributi di concetto e ruoli tra concetti, riflettendo quindi il dominio di interesse.

L'ontologia è stata descritta utilizzando il linguaggio $DL - Lite_A$, e successivamente realizzata tramite il tool di gestione per ontologie *Protégé*.

Successivamente sono stati creati i mappings per trasformare i dati relazionali in oggetti dell'ontologia. Per poter inserire tali mapping nel progetto in Protégé, è stato necessario integrare quest'ultimo con il plugin ODBA che, permette anche di effettuare la connessione con il reasoner e la possibilità di

definire query SPARQL. Il reasoner utilizzato è il DIGServer. Questo è infatti un server che wrappa QuOnto/Mastro e implementa l'interfaccia DIG 1.1, permettendo così a QuOnto/Mastro di interagire con qualsiasi client che implementi tale interfaccia (nel nostro caso il plugin per Protégé).

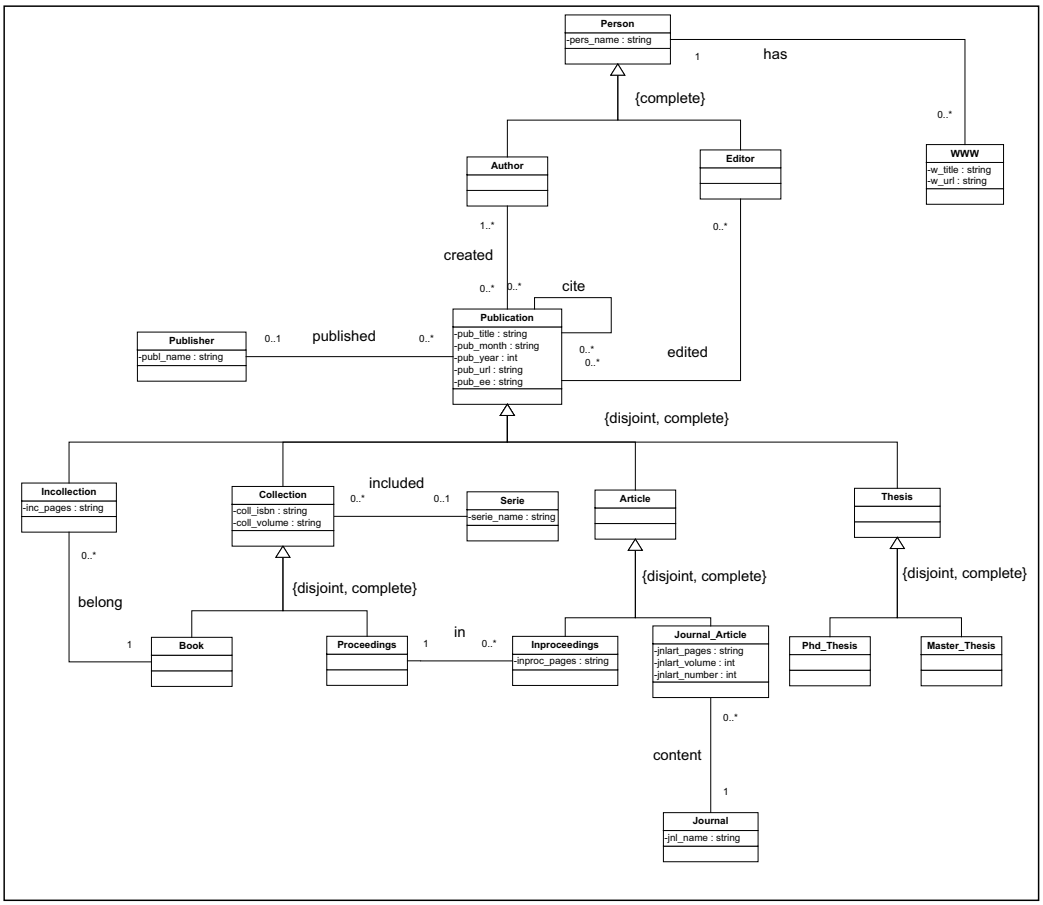

Figura 5.1: Ontologia della bibliografia DBLP

5.2 Il linguaggio $DL - Lite_A$

$DL-Lite_A$ è un linguaggio della famiglia delle logiche descrittive proposto dal DIS[1]. Tale linguaggio è pensato per modellare linguaggi base per ontologie, modelli concettuali e formalismi orientati agli oggetti. Il linguaggio $DL - Lite_A$ fa una distinzione tra oggetti e valori, distinguendo inoltre i concetti dai valori di dominio. Dove un concetto è un'astrazione per un insieme di

[1]Dipartimento di Informatica e Sistemistica dell'università Sapienza

oggeti e un valore di dominio denota un insieme di valori per i dati. Vengono inoltre distinti gli attributi dai ruoli, dove un ruolo denota una relazione binaria tra due oggetti mentre un attributo denota una relazione binaria tra oggetti e valori. Verranno di seguito descritte le regole per modellare: attributi, associazioni, classi, relazioni is-a e disgiunzioni tra classi.

Gli attributi

δ(attributo) $\subseteq$ classe
ρ(attributo) $\subseteq$ xsd: tipo

Il simbolo δ indica il dominio dell attributo, cioè la classe a cui esso appartiene, mentre con il simbolo ρ si indica il range ovvero il tipo di dato. Può inolre essere modellata la cardinalità dell'attributo:

(funct attributo)
indica una molteplicità del tipo 0..1, mentre

classe $\subseteq$ δ(attributo)
indica una molteplicità del tipo 1..* .

Associazioni

Un'associazione tra due classi A e B viene tradotta nel seguente modo:

$\exists$ associazione $\subseteq$ Classe A
$\exists$ associazione$^-$ $\subseteq$ classe B
che identifica le due classi appartenenti all'associazione. Per definire le molteplicità vi sono le seguenti regole:

classe A $\subseteq$ associazione
per indicare una molteplicità del tipo 0..* $\rightarrow$ 1..* .

(funct associazione)
per indicare una molteplicità del tipo 0..* $\rightarrow$ 0..1 .

classe B $\subseteq$ associazione$^-$
per indicare una molteplicità del tipo 1..* $\rightarrow$ 0..* .

(funct associazione$^-$)
per indicare una molteplicità del tipo 0..1 $\rightarrow$ 0..* .

Relazioni Is-A

classe B $\subseteq$ classe A
specifica che la classe B è una sottoclasse della classe A.

Disgiunzioni tra classi

classe B $\subseteq \neg$classe A
indica che le classi A e B sono disgiunte, non hanno cioè elementi in comune.

5.3 Traduzione dell'ontologia in $DL - Lite_A$

5.3.1 Isa tra classi

Author $\subseteq$ Person
Editor $\subseteq$ Person

Incollection $\subseteq$ Publication
Collection $\subseteq$ Publication
Article $\subseteq$ Publication
Thesis $\subseteq$ Publication

Book $\subseteq$ Collection
Proceeding $\subseteq$ Collection

Inproceeding $\subseteq$ Article
Journal_article $\subseteq$ Article

Phd_Thesis $\subseteq$ Thesis
Master_Thesis $\subseteq$ Thesis

5.3.2 Disgiunzione tra classi

Incollection $\subseteq \neg$ Collection
Incollection $\subseteq \neg$ Article
Incollection $\subseteq \neg$ Thesis
Collection $\subseteq \neg$ Article
Collection $\subseteq \neg$ Thesis
Article $\subseteq \neg$ Thesis

Book $\subseteq \neg$ Proceedings

Inproceedings $\subseteq \neg$ Journal_article

Phd_thesis $\subseteq \neg$ Master_thesis

5.3.3 Classi

Classe Person

δ(pers_name) $\subseteq$ Person
ρ(pers_name) $\subseteq$ xsd: string
(funct pers_name)
Person $\subseteq \delta$(pers_name)

$\exists$ HAS $\subseteq$ Person
$\exists$ HAS$^-$ $\subseteq$ WWW
WWW $\subseteq$ HAS$^-$

Classe Author

Classe Editor

Classe Publisher

δ(publ_name) $\subseteq$ Publisher
ρ(publ_name) $\subseteq$ xsd: string
(funct publ_name)
Publisher $\subseteq \delta$(publ_name)

Classe WWW

δ(w_title) $\subseteq$ WWW
ρ(w_title) $\subseteq$ xsd: string
(funct w_title)
WWW $\subseteq \delta$(w_title)

δ(w_url) $\subseteq$ Web_address
ρ(w_url) $\subseteq$ xsd: string
(funct w_url)
WWW $\subseteq$ δ(w_url)

Classe Publication

δ(pub_title) $\subseteq$ Publication
ρ(pub_title) $\subseteq$ xsd: string
(funct pub_title)
Publication $\subseteq$ δ(pub_title)

δ(pub_month) $\subseteq$ Publication
ρ(pub_month) $\subseteq$ xsd: string
(funct pub_month)

δ(pub_year) $\subseteq$ Publication
ρ(pub_year) $\subseteq$ xsd: string
(funct pub_year)

δ(pub_url) $\subseteq$ Publication
ρ(pub_url) $\subseteq$ xsd: string
(funct pub_url)

δ(pub_ee) $\subseteq$ Publication
ρ(pub_ee) $\subseteq$ xsd: string
(funct pub_ee)

$\exists$ CREATED $\subseteq$ Publication
$\exists$ CREATED$^-$ $\subseteq$ Author
Publication $\subseteq$ CREATED

$\exists$ EDITED $\subseteq$ Publication
$\exists$ EDITED$^-$ $\subseteq$ Editor

$\exists$ CITE $\subseteq$ Publication
$\exists$ CITE$^-$ $\subseteq$ Publication

$\exists$ PUBLISHED $\subseteq$ Publication
$\exists$ PUBLISHED$^-$ $\subseteq$ Publisher

(funct PUBLISHED)

Classe Incollection

δ(inc_pages) $\subseteq$ Incollection
ρ(inc_pages) $\subseteq$ xsd: string
(funct inc_pages)
Incollection $\subseteq$ δ(inc_pages)

$\exists$ BELONG $\subseteq$ Incollection
$\exists$ BELONG$^-$ $\subseteq$ Book
(funct BELONG)
Incollection $\subseteq$ BELONG

Classe Collection

δ(coll_isbn) $\subseteq$ Collection
ρ(coll_isbn) $\subseteq$ xsd: string
(funct coll_isbn)

δ(coll_volume) $\subseteq$ Collection
ρ(coll_volume) $\subseteq$ xsd: string
(funct coll_volume)

$\exists$ INCLUDED $\subseteq$ Collection
$\exists$ INCLUDED$^-$ $\subseteq$ Series
(funct INCLUDED)

Classe Book

Classe Proceeding

Classe Series

δ(serie_name) $\subseteq$ Series
ρ(serie_namee) $\subseteq$ xsd: string

(funct serie_name)
Series $\subseteq \delta$(serie_name)

Classe Article

Classe Inproceeding

δ(inproc_pages) $\subseteq$ Inproceedings
ρ(inproc_pages) $\subseteq$ xsd: string
(funct inproc_pages)
Inproceedings $\subseteq \delta$(inproc_pages)

$\exists$ IN $\subseteq$ Inproceedings
$\exists$ IN$^-$ $\subseteq$ Proceedings
(funct IN)
Inproceeding $\subseteq$ IN

Classe Journal_article

δ(jnlart_pages) $\subseteq$ Journal_article
ρ(jnlart_pages) $\subseteq$ xsd: string
(funct jnlart_pages)
Journal_article $\subseteq \delta$(jnlart_pages)

δ(jnlart_number) $\subseteq$ Journal_article
ρ(jnlart_number) $\subseteq$ xsd: string
(funct jnlart_number)
Journal_article $\subseteq \delta$(jnlart_number)

δ(jnlart_volume) $\subseteq$ Journal_article
ρ(jnlart_volume) $\subseteq$ xsd: string
(funct jnlart_volume)
Journal_article $\subseteq \delta$(jnlart_volume)

$\exists$ CONTENT $\subseteq$ Journal_article
$\exists$ CONTENT$^-$ $\subseteq$ Journal
(funct CONTENT)
Journal_article $\subseteq$ CONTENT

Classe Journal

δ(jnl_name) $\subseteq$ Journal
ρ(jnl_name) $\subseteq$ xsd: string
(funct jnl_name)
Journal $\subseteq$ δ(jnl_name)

5.4 Specifica della sorgente dei dati

Il database utilizzato come sorgente di dati per l'ontologia non è quello discusso nei capitoli precedenti. Data la sua simiglianza con la struttura dell'ontologia, si è scelto di usare un database di bassa qualità, composto da 3 sole relazioni. Questo per enfatizzare uno degli scopi dell'ontologia: l'integrazione tra diverse sorgenti di dati, offrendo all'utente una vista concettuale indipendente da come i dati sono fisicamente organizzati. Le relazioni della base di dati sono mostrate nella Figura 5.2.

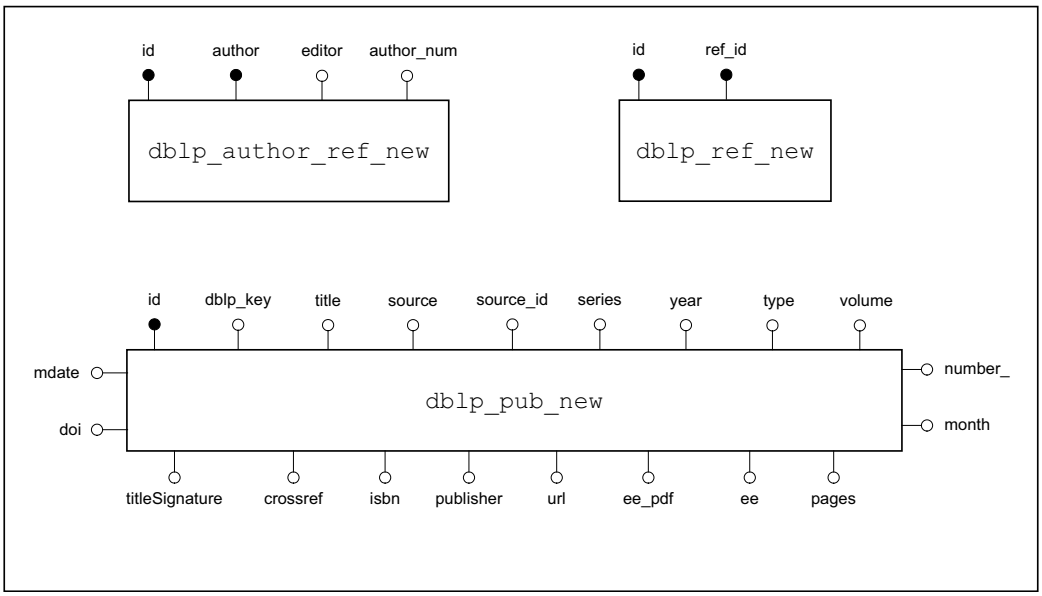

Figura 5.2: Struttura del db utilizzato dall'ontologia DBLP

Il DBMS utilizzato è stato *Oracle 10g*. Si è scelto di utilizzare tale prodot-

to per alcuni motivi, tra cui l'incapacità da parte di MySQL di eseguire sia il *Check KB Consistency*, sia semplici query SPARQL.

La fase successiva è stata quella di creare i mappings, delle asserzioni che permettessero di correlare i dati ralazionali con i concetti dell'ontologia. Tali mappings sono necessari per ovviare al *impedance mismatch problem*, cioè, la diversità dei dati presenti nel db e gli oggetti dell'ontologia.

I mappings vengono divisi in due sottoinsiemi:

- *data-to-object mappings*: asserzioni usate per mappare i dati presenti nel database, con i concetti, ruoli e attributi che costituiscono l'ongologia
- *typing mappings*: asserzioni necessarie per associare i tipi di dato ai valori del database. Necessario in tutti quei casi dove i tipi di dato tra database e ontologia non corrispondono direttamente.

Per effettuare i mappings è necessario ricorrere al concetto di *funtore*. Esso è una funzione che, prende in input i valori provenienti dai campi del database, e identificano quindi le varie istanze dell'ontologia.

Vengono riportate di seguito le regole per la creazione dei mappings.

Attributo

attr(**funct**(VAR), VAR_ATTR)
per mappare un'attributo di concetto bisogna usare la relazione binaria *attr* tra l'istanza della classe (identificata tramite il funtore *funct*) e il campo del database, VAR_ATTR (attributo da mappare).

Concetto

Class(**funct**(VAR))
Nel caso si voglia mappare un concetto dell'ontologia, si passa al funtore *(funct)* il campo del database (VAR).

Associazione

ASSOC(**funct**(VAR), **funct2**(VAR2)) Per mappare un'associazione tra due istanze, identificate dai funtori *funct* e *funct2*, si utilizza un'associazione binaria tra i funtori.

5.4.1 Mapping per le tabelle del database (*typing mappings*)

Mapping per la tabella dblp_author_ref_new

```
CREATE TABLE "GERSK"."DBLP_AUTHOR_REF_NEW" (
    "ID" NUMBER(8,0) NOT NULL ENABLE,
    "AUTHOR" VARCHAR2(70 BYTE) NOT NULL ENABLE,
    "EDITOR" NUMBER(1,0) DEFAULT 0 NOT NULL ENABLE,
    "AUTHOR_NUM" NUMBER(3,0) NOT NULL ENABLE,
    CONSTRAINT "DBLP_AUTHOR_REF_NEW_PK" PRIMARY KEY ("ID", "AUTHOR")
    USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
    TABLESPACE "USERS"  ENABLE )
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

COMMENT ON COLUMN "GERSK"."DBLP_AUTHOR_REF_NEW"."ID" IS 'Our internal
    database key in dblp_pub_new';

COMMENT ON COLUMN "GERSK"."DBLP_AUTHOR_REF_NEW"."AUTHOR" IS 'The
    author name';

COMMENT ON COLUMN "GERSK"."DBLP_AUTHOR_REF_NEW"."EDITOR" IS 'Bool being
    true when the author is editor of the book';

COMMENT ON COLUMN "GERSK"."DBLP_AUTHOR_REF_NEW"."AUTHOR_NUM" IS 'The
    author number (from the implicit order in the dblp xml file)';

COMMENT ON TABLE "GERSK"."DBLP_AUTHOR_REF_NEW"  IS 'References between
    publication/collection and authors';
```

M_{t1} : SELECT id FROM dblp_author_ref_new $\rightsquigarrow$ `xsd:string`
M_{t2} : SELECT author FROM dblp_author_ref_new $\rightsquigarrow$ `xsd:string`
M_{t3} : SELECT editor FROM dblp_author_ref_new $\rightsquigarrow$ `xsd:string`
M_{t4} : SELECT author_num FROM dblp_author_ref_new $\rightsquigarrow$ `xsd:string`

Mapping per la tabella dblp_pub_new

```
CREATE TABLE "GERSK"."DBLP_PUB_NEW" (
    "ID" NUMBER(8,0) NOT NULL ENABLE,
    "DBLP_KEY" VARCHAR2(150 BYTE) NOT NULL ENABLE,
    "TITLE" VARCHAR2(4000 BYTE) NOT NULL ENABLE,
    "SOURCE" VARCHAR2(150 BYTE) DEFAULT null,
    "SOURCE_ID" VARCHAR2(50 BYTE) DEFAULT null,
    "SERIES" VARCHAR2(100 BYTE) DEFAULT null,
    "YEAR" NUMBER(4,0) DEFAULT 0 NOT NULL ENABLE,
    "TYPE" VARCHAR2(20 BYTE) DEFAULT 0 NOT NULL ENABLE,
    "VOLUME" VARCHAR2(50 BYTE) DEFAULT null,
    "NUMBER_" VARCHAR2(20 BYTE) DEFAULT null,
    "MONTH" VARCHAR2(30 BYTE) DEFAULT null,
    "PAGES" VARCHAR2(100 BYTE) DEFAULT null,
    "EE" VARCHAR2(200 BYTE) DEFAULT null,
    "EE_PDF" VARCHAR2(200 BYTE) DEFAULT null,
    "URL" VARCHAR2(150 BYTE) DEFAULT null,
    "PUBLISHER" VARCHAR2(250 BYTE) DEFAULT null,
    "ISBN" VARCHAR2(25 BYTE) DEFAULT null,
    "CROSSREF" VARCHAR2(50 BYTE) DEFAULT null,
    "TITLESIGNATURE" VARCHAR2(255 BYTE) DEFAULT null,
    "DOI" VARCHAR2(255 BYTE) DEFAULT null,
    "MDATE" DATE NOT NULL ENABLE,
    CONSTRAINT "DBLP_PUB_NEW_PK" PRIMARY KEY ("ID")
    USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
    TABLESPACE "USERS"  ENABLE )
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."ID" IS 'The internal key
    in the database';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."DBLP_KEY" IS 'The key
    in the xml file';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."TITLE" IS 'Title of
```

```
    the publication';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."SOURCE" IS 'Name to
    the publication source, i.e. Conference, Journal, etc.;
    for collections, the booktitle is stored here';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."SOURCE_ID" IS 'Reference
    to the publication source (first part of the dblp_key)';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."SERIES" IS 'Reference to
    the publication series (books and proceedings only)';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."YEAR" IS 'The year of the
    publication';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."TYPE" IS 'Type of publication,
    i.e. article, proceedings, etc.';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."VOLUME" IS 'Volume of the
    source where the publication was published';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."NUMBER_" IS 'Number of the
    source where the publication was published';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."MONTH" IS 'Month(s) when
    the publication was published';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."PAGES" IS 'Pages in the
    source, i.e. for example the journal';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."EE" IS 'external URL to
    the electronic edition of the publication';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."EE_PDF" IS 'external URL to
    the PDF version of the electronic edition of the publication';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."URL" IS 'DBLP-internal URL
    (starting with db/...) where a web-page for that publication can
    be found on DBLP';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."PUBLISHER" IS 'Name of the
```

```
    publisher of the publication; school for theses; affiliation
    for homepages';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."ISBN" IS 'ISBN number
    of the collection';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."CROSSREF" IS 'dblpkey
    crossreference to one other publication (book, proceeding, in the
    dblp_collections table), in which this publication was published';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."TITLESIGNATURE" IS 'Title
    string without space and some common characters like !?,.
    for comparing the title with citeseer titles';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."DOI" IS 'The DOI of
    the publication';

COMMENT ON COLUMN "GERSK"."DBLP_PUB_NEW"."MDATE" IS 'The last
    modification date of the entry';

COMMENT ON TABLE "GERSK"."DBLP_PUB_NEW"  IS 'Stores dblp
    publications of all types';
```

M_{t1} : SELECT id FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t2} : SELECT dblp_key FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t3} : SELECT title FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t4} : SELECT source FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t5} : SELECT source_id FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t6} : SELECT series FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t7} : SELECT year FROM dblp_pub_new $\rightsquigarrow$ `xsd:int`
M_{t8} : SELECT type FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t9} : SELECT volume FROM dblp_pub_new $\rightsquigarrow$ `xsd:int`
M_{t10} : SELECT number_ FROM dblp_pub_new $\rightsquigarrow$ `xsd:int`
M_{t11} : SELECT month FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t12} : SELECT pages FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t13} : SELECT ee FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t14} : SELECT ee_pdf FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t15} : SELECT url FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t16} : SELECT publisher FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`

M_{t17} : SELECT isbn FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t18} : SELECT crossref FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t19} : SELECT titlesignature FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t20} : SELECT doi FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`
M_{t20} : SELECT mdate FROM dblp_pub_new $\rightsquigarrow$ `xsd:string`

Mapping per la tabella dblp_ref_new

```
CREATE TABLE "GERSK"."DBLP_REF_NEW" (
    "ID" NUMBER(8,0) NOT NULL ENABLE,
    "REF_ID" VARCHAR2(150 BYTE) NOT NULL ENABLE,
    CONSTRAINT "DBLP_REF_NEW_PK" PRIMARY KEY ("ID", "REF_ID")
    USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
    STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
    PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
    TABLESPACE "USERS"  ENABLE )
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

COMMENT ON COLUMN "GERSK"."DBLP_REF_NEW"."ID" IS 'Our internal
    database key in dblp_pub_new';

COMMENT ON COLUMN "GERSK"."DBLP_REF_NEW"."REF_ID" IS 'DBLP key of the
    publication being cited (not crossreferenced) by source';

COMMENT ON TABLE "GERSK"."DBLP_REF_NEW"  IS 'Holds all citations
    (no cross-references) between dblp publication';
```

M_{t1} : SELECT id FROM dblp_ref_new $\rightsquigarrow$ `xsd:string`
M_{t2} : SELECT ref_id FROM dblp_ref_new $\rightsquigarrow$ `xsd:string`

5.4.2 Mapping per i concetti dell'ontologia (*data-to-object mappings*)

Mapping per il concetto Editor

M_{m1} : **SELECT**
author, is AS publication
FROM
dblp_author_ref_new
WHERE
editor = '1'
$\rightsquigarrow$ **Editor**(**pers**($author)),
pers_name(**pers**($author),$author),
edited(**pub**($publication),**pers**($author))

Mapping per il concetto Author

M_{m1} : **SELECT**
author, is AS publication
FROM
dblp_author_ref_new
WHERE
editor = '0'
$\rightsquigarrow$ **Author**(**pers**($author)),
pers_name(**pers**($author),$author),
created(**pub**($publication),**pers**($author))

Mapping per il concetto Publisher

M_{m1} : **SELECT**
id as publication, publisher
FROM
dblp_pub_new
WHERE
publisher is not null
$\rightsquigarrow$ **Publisher**(**pb**($publisher)),
publ_name(**pb**($publisher),$publisher),
published(**pub**($publication),**pb**($publisher))

Mapping per il concetto WWW

M_{m1} : **SELECT**
a.author as person,
p.id as www,
p.title, p.ee as url
FROM
dblp_author_ref_new a,
dblp_pub_new p
WHERE
p.type = 'www' and
a.id = p.id
$\rightsquigarrow$ **WWW**(**ww**($www)),
w_name(**ww**($www),$title),
w_url(**ww**($www),$url),
has(**pers**($person),**ww**($www))

Mapping per il concetto Incollection

M_{m1} : **SELECT**
a.id as incollection,
a.title,
b.id as book
FROM
dblp_pub_new
WHERE
type = 'incollection' and
a.crossref = b.dblp_key
$\rightsquigarrow$ **Incollection**(**pub**($incollection)),
pub_title(**pub**($incollection),$title),
belong(**pub**($incollection),**pub**($book))

M_{m2} : **SELECT**
a.id as incollection,
pages,
FROM
dblp_pub_new
WHERE
type = 'incollection' and
pages is not null
$\rightsquigarrow$ **inc_pages**(**pub**($incollection),$pages)

Mapping per il concetto Collection

M_{m1} : **SELECT**
id as collection, isbn
FROM
dblp_pub_new
WHERE
(type = 'book' or
type = 'proceedings')
and isbn is not null
$\rightsquigarrow$ **coll_isbn**(**pub**($collection),$isbn)

M_{m2} : **SELECT**
id as collection, volume
FROM
dblp_pub_new
WHERE
(type = 'book' or
type = 'proceedings')
and volume is not null
$\rightsquigarrow$ **coll_volume**(**pub**($collection),$volume)

M_{m3} : **SELECT**
id as collection, series
$\rightsquigarrow$ **Serie**(**ser**($series)),
serie_name(**ser**($series),$series),

FROM
dblp_pub_new
WHERE
(type = 'book' or
type = 'proceedings')
and series is not null

included(**pub**($collection),**ser**($series))

Mapping per il concetto Book

M_{m1} : **SELECT**
id as book, title
FROM
dblp_pub_new
WHERE
type = 'book'

$\rightsquigarrow$ **Book**(**pub**($book)),
pub_title(**pub**($book),$title)

Mapping per il concetto Proceedings

M_{m1} : **SELECT**
id as proceedings, title
FROM
dblp_pub_new
WHERE
type = 'proceedings'

$\rightsquigarrow$ **Proceedings**(**pub**($proceedings)),
pub_title(**pub**($proceedings),$title)

Mapping per il concetto Inproceedings

M_{m1} : **SELECT**
a.id as inproceedings,
a.title,
b.id as proceedings
FROM
dblp_pub_new
WHERE
a.type = 'inproceedings' and
a.crossref = b.dblp_key

$\rightsquigarrow$ **Inroceedings**(**pub**($inproceedings)),
pub_title(**pub**($inproceedings),$title)
in(**pub**($inproceedings),**pub**($proceedings))

M_{m2} : **SELECT**
a.id as inproceedings,
pages,

$\rightsquigarrow$ **inproc_pages**(**pub**($inproceedings),$pages)

FROM
dblp_pub_new
WHERE
type = 'inproceedings' and
pages is not null

Mapping per il concetto Journal_article

M_{m1} :
SELECT
id as article,
title,
source as journal
FROM
dblp_pub_new
WHERE
type = 'article'

$\rightsquigarrow$ **Journal_article**(**pub**($article)),
pub_title(**pub**($article),$title),
Journal(**jnl**($journal)),
jnl_name(**jnl**($journal),$journal),
content(**pub**($article),**jnl**($journal))

M_{m2} :
SELECT
a.id as article,
pages,
FROM
dblp_pub_new
WHERE
type = 'article' and
pages is not null

$\rightsquigarrow$ **jnlart_pages**(**pub**($article),$pages)

M_{m3} :
SELECT
a.id as article,
volume,
FROM
dblp_pub_new
WHERE
type = 'article' and
volume is not null

$\rightsquigarrow$ **jnlart_volume**(**pub**($article),$volume)

M_{m2} :
SELECT
a.id as article,
number,
FROM
dblp_pub_new
WHERE
type = 'article' and
number is not null

$\rightsquigarrow$ **jnlart_number**(**pub**($article),$number)

Mapping per il concetto Phd_thesis

M_{m1} : **SELECT**
id as p_thesis, title
FROM
dblp_pub_new
WHERE
type = 'phdthesis'

$\rightsquigarrow$ **Phd_thesis**(**pub**($p_thesis)),
pub_title(**pub**($p_thesis),$title)

Mapping per il concetto Master_thesis

M_{m1} : **SELECT**
id as m_thesis, title
FROM
dblp_pub_new
WHERE
type = 'masterthesis'

$\rightsquigarrow$ **Master_thesis**(**pub**($m_thesis)),
pub_title(**pub**($m_thesis),$title)

Mapping per il concetto Publication

M_{m1} : **SELECT**
id as publication, month
FROM
dblp_pub_new
WHERE
(type = 'incollection' or
type = 'book' or
type = 'proceedings' or
type = 'inproceedings' or
type = 'article' or
type = 'phdthesis' or
type = 'masterthesis') and
month is not null

$\rightsquigarrow$ **pub_month**(**pub**($publication),$month)

M_{m2} : **SELECT**
id as publication, year
FROM
dblp_pub_new
WHERE
(type = 'incollection' or

$\rightsquigarrow$ **pub_year**(**pub**($publication),$year)

type = 'book' or
type = 'proceedings' or
type = 'inproceedings' or
type = 'article' or
type = 'phdthesis' or
type = 'masterthesis') and
year is not null

M_{m3} : **SELECT**
id as publication, url
FROM
dblp_pub_new
WHERE
(type = 'incollection' or
type = 'book' or
type = 'proceedings' or
type = 'inproceedings' or
type = 'article' or
type = 'phdthesis' or
type = 'masterthesis') and
url is not null
$\rightsquigarrow$ **pub_url**(**pub**($publication),$url)

M_{m4} : **SELECT**
id as publication, ee
FROM
dblp_pub_new
WHERE
(type = 'incollection' or
type = 'book' or
type = 'proceedings' or
type = 'inproceedings' or
type = 'article' or
type = 'phdthesis' or
type = 'masterthesis') and
ee is not null
$\rightsquigarrow$ **pub_ee**(**pub**($publication),$ee)

M_{m5} : **SELECT**
a.id as cite,
b.id as cited
FROM
dblp_pub_new a,
$\rightsquigarrow$ **cite**(**pub**($cite),**pub**($cited))

dblp_pub_new b,
dblp_ref_new c
WHERE
a.id = c.id and
c.ref_id = b.dblp_key

GUI di Protégé

La Figura 5.3 mostra la sezione di Protégé relativa alla definizione della sorgente di dati ed ai mappings.

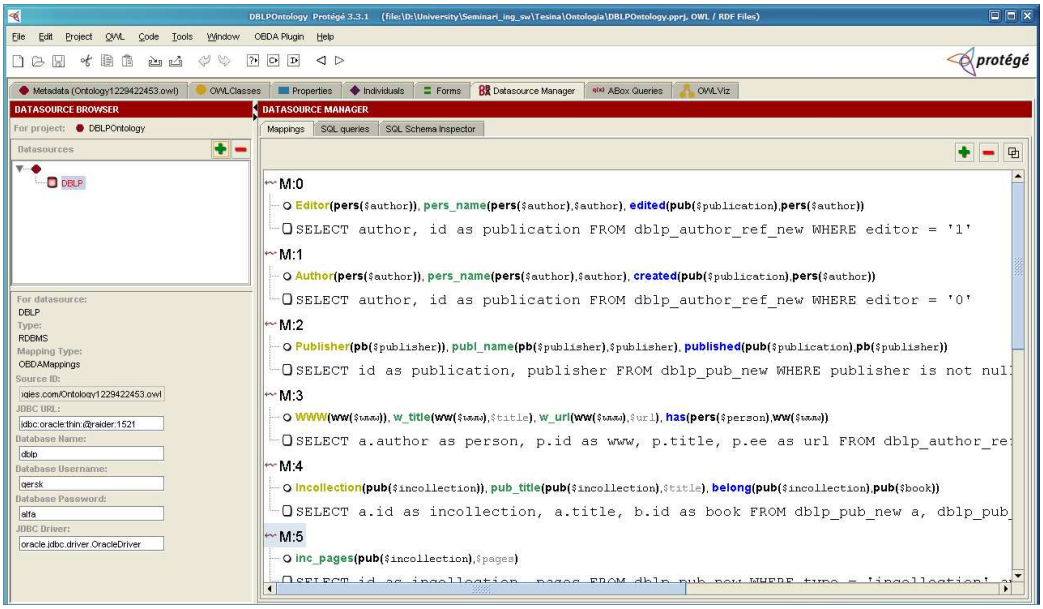

Figura 5.3: Datasource Manager di Protégé

5.5 Query sull'ontologia

Per valutare sia la correttezza dell'ontologia, sia i tempi di risposta, sono state definite ed eseguite alcune query. In questa sezione verranno riportate le query stesse, i risultati ed il report del database.

5.5.1 Query n.1

La query richiede il titolo di tutte le pubblicazioni di tipo Incollection create dal professor De Giacomo, il libro in cui esse sono contenute e le relative pagine.

```
SELECT $w $v $h
WHERE {
      $x rdf:type 'Person'.
      $x pers_name 'Giuseppe De Giacomo'.
      $y rdf:type 'Incollection'.
      $y created $x.
      $y pub_title $w.
      $y inc_pages $v.
      $z rdf:type 'Publication'.
      $y belong $z.
      $z pub_title $h}
```

Espansione della UCQ:

```
q(w,v,h) :- belong(y,z), pub_title(z,h), created(y,x),
pers_name(x,'Giuseppe De Giacomo'), pub_title(y,w),
inc_pages(y,v).
```

Unfolding della UCQ:

```
SELECT DISTINCT alias_0.term2, alias_3.term2, alias_5.term2
FROM ((SELECT DISTINCT CONCAT('pub(',CONCAT(book,')'))
AS term1,title AS term2 FROM (SELECT id as book, title
FROM dblp_pub_new WHERE type = 'book') DummyTable) UNION
(SELECT DISTINCT CONCAT('pub(',CONCAT(proceedings,')'))
AS term1,title AS term2 FROM (SELECT id as proceedings,
title FROM dblp_pub_new WHERE type = 'proceedings')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',CONCAT
(article,')')) AS term1,title AS term2 FROM (SELECT id
as article, title, source as journal FROM dblp_pub_new
WHERE type = 'article') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(m_thesis,')')) AS term1,title AS term2
FROM (SELECT id as m_thesis, title FROM dblp_pub_new WHERE
type = 'masterthesis') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(p_thesis,')')) AS term1,title AS term2
FROM (SELECT id as p_thesis, title FROM dblp_pub_new WHERE
type = 'phdthesis') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(incollection,')')) AS term1,title AS
term2 FROM (SELECT a.id as incollection, a.title, b.id as
book FROM dblp_pub_new a, dblp_pub_new b WHERE a.type =
'incollection' and a.crossref = b.dblp_key) DummyTable)
```

```
UNION(SELECT DISTINCT CONCAT('pub(',CONCAT(inproceedings,
')')) AS term1,title AS term2 FROM (SELECT a.id as
inproceedings, a.title, b.id as proceedings FROM
dblp_pub_new a, dblp_pub_new b WHERE a.type =
'inproceedings' and a.crossref = b.dblp_key) DummyTable))
alias_0 , ((SELECT DISTINCT CONCAT('pers(',CONCAT(author,
')')) AS term1,author AS term2 FROM (SELECT author, id as
publication FROM dblp_author_ref_new WHERE editor = '1')
DummyTable) UNION (SELECT DISTINCT CONCAT('pers(',CONCAT
(author,')')) AS term1,author AS term2 FROM (SELECT author,
id as publication FROM dblp_author_ref_new WHERE editor = '0')
DummyTable)) alias_1 , (SELECT DISTINCT CONCAT('pub(',
CONCAT(publication,')')) AS term1,CONCAT('pers(',
CONCAT(author,')'))AS term2 FROM (SELECT author, id
as publication FROMdblp_author_ref_new WHERE editor = '0')
DummyTable) alias_2 ,(SELECT DISTINCT CONCAT('pub(',
CONCAT(incollection,')')) ASterm1,pages AS term2
FROM (SELECT id as incollection, pages FROM dblp_pub_new
WHERE type = 'incollection' and pages is not null )
DummyTable) alias_3 , (SELECT DISTINCT CONCAT('pub
(',CONCAT(incollection,')')) AS term1,CONCAT('pub
(',CONCAT(book,')')) AS term2 FROM (SELECT a.id as
incollection, a.title, b.id as book FROM dblp_pub_new a,
dblp_pub_new b WHERE a.type = 'incollection'
and a.crossref = b.dblp_key) DummyTable) alias_4 ,
((SELECT DISTINCT CONCAT('pub(',CONCAT(book,')')) AS
term1,title AS term2 FROM (SELECT id as book, title
FROM dblp_pub_new WHERE type = 'book') DummyTable)
UNION (SELECT DISTINCT CONCAT('pub(',CONCAT(proceedings,')'))
AS term1,title AS term2 FROM (SELECT id as proceedings,
title FROM dblp_pub_new WHERE type = 'proceedings')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',
CONCAT(article,')')) AS term1,title AS term2 FROM
(SELECT id as article, title, source as journal FROM
dblp_pub_new WHERE type= 'article') DummyTable) UNION
(SELECT DISTINCT CONCAT('pub(',CONCAT(m_thesis,')
')) AS term1,title AS term2 FROM (SELECT id as m_thesis,
title FROM dblp_pub_new WHERE type = 'masterthesis')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',
CONCAT(p_thesis,')')) AS term1,title AS term2 FROM
(SELECT id as p_thesis, title FROM dblp_pub_new WHERE
```

```
type = 'phdthesis') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(incollection,')')) AS term1,title
AS term2 FROM (SELECT a.id as incollection, a.title, b.id as
book FROM dblp_pub_new a, dblp_pub_new b WHER
E a.type = 'incollection' and a.crossref = b.dblp_key)
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',
CONCAT(inproceedings,')')) AS term1,title AS term2 FROM
(SELECT a.id as inproceedings, a.title, b.id as proceedings
FROM dblp_pub_new a, dblp_pub_new b WHERE a.type =
'inproceedings' and a.crossref = b.dblp_key)
DummyTable)) alias_5 WHERE alias_1.term2='Giuseppe De
Giacomo'  AND alias_0.term1=alias_2.term1  AND
alias_1.term1=alias_2.term2  AND alias_2.term1=alias_3.term1
AND alias_3.term1=alias_4.term1  AND alias_4.term2=alias_5.term1
```

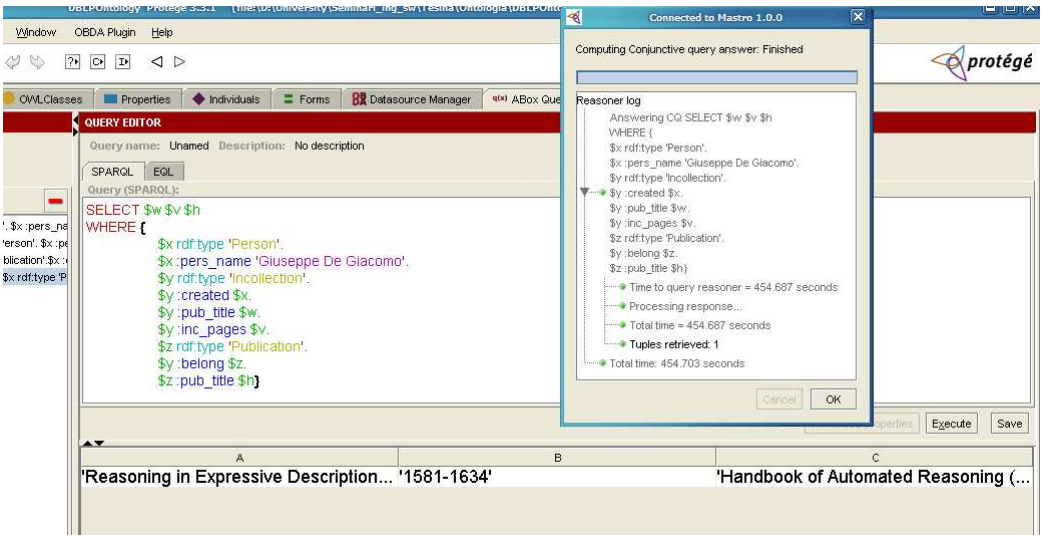

Figura 5.4: Risultato della query n.1

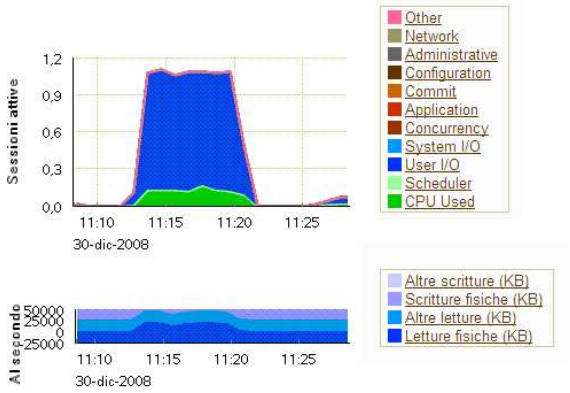

Figura 5.5: Report del db per la query n.1

5.5.2 Query n.2

La query richiede tutti gli articoli scientifici creati dal professor De Giacomo, con relativa rivista che li contiene.

```
SELECT $w $a $b $c $h
WHERE {
      $x rdf:type 'Person'.
      $x pers_name 'Giuseppe De Giacomo'.
      $y rdf:type 'Journal_article'.
      $y created $x.
      $y pub_title $w.
      $y jnlart_pages $a.
      $y jnlart_number $b.
      $y jnlart_volume $c.
      $y content $z.
      $z jnl_name $h}
```

Espansione della UCQ:

```
q(w,a,b,c,h) :- jnl_name(z,h), content(y,z), created(y,x),
pers_name(x,'Giuseppe De Giacomo'), jnlart_number(y,b), pub_title(y,w),
jnlart_pages(y,a), jnlart_volume(y,c)."/>
```

Unfolding della UCQ:

```
SELECT DISTINCT alias_0.term2, alias_2.term2, alias_6.term2,
```

```
alias_7.term2, alias_4.term2 FROM ((SELECT DISTINCT
CONCAT('pub(',CONCAT(book,')')) AS term1,titleAS term2 FROM
(SELECT id as book, title FROM dblp_pub_new WHERE type = 'book')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',
CONCAT(proceedings,')')) AS term1,title AS term2 FROM
(SELECT id as proceedings, title FROM dblp_pub_new WHERE
type = 'proceedings') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(article,')')) AS term1,title AS term2 FROM
(SELECT id as article, title, source asjournal FROM
dblp_pub_new WHERE type = 'article') DummyTable) UNION (SELECT
DISTINCT CONCAT('pub(',CONCAT(m_thesis,')')) AS term1,title AS
term2 FROM (SELECT id
 as m_thesis, title FROM dblp_pub_new WHERE type = 'masterthesis')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',CONCAT(p_thesis,')'))
AS term1,title AS term2 FROM (SELECT id as p_thesis, title
FROM dblp_pub_new WHERE type = 'phdthesis') DummyTable) UNION
(SELECT DISTINCT CONCAT('pub(',CONCAT(incollection,')')) AS term1,title
AS term2 FROM (SELECT a.id as incollection, a.title, b.id as book FROM
dblp_pub_new a, dblp_pub_new b WHERE a.type = 'incollection' and
a.crossref =b.dblp_key) DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(inproceedings,')')) AS term1,title AS term2
FROM (SELECT a.id as inproceedings, a.title, b.id as proceedings
FROM dblp_pub_new a, dblp_pub_new b WHERE a.type = 'inproceedings'
and a.crossref = b.dblp_key) DummyTable)) alias_0 , ((SELECT DISTINCT
CONCAT('pers(',CONCAT(author,')')) AS term1,author AS term2 FROM
(SELECT author, id as publication FROM dblp_author_ref_new WHERE
editor = '1') DummyTable) UNION (SELECT DISTINCT
CONCAT('pers(',CONCAT(author,')')) AS term1,author AS term2 FROM
(SELECT author, id as publication FROM dblp_author_ref_new WHERE
editor = '0') DummyTable)) alias_1 , (SELECT DISTINCT
CONCAT('pub(',CONCAT(article,')')) AS term1,pages AS term2 FROM
(SELECT id as article, pages FROM dblp_pub_new WHERE type = 'article'
and pages is not null) DummyTable) alias_2 , (SELECT DISTINCT
CONCAT('pub(',CONCAT(publication,')')) AS term1,CONCAT('pers(',
CONCAT(author,')')) AS term2 FROM (SELECT author, id as publication
FROM dblp_author_ref_new WHERE editor = '0') DummyTable) alias_3 ,
(SELECT DISTINCT CONCAT('jnl(',CONCAT(journal,')')) AS term1,journal
AS term2 FROM (SELECT id as article, title, source as journal FROM
dblp_pub_new WHERE type = 'article') DummyTable) alias_4 ,
(SELECT DISTINCT CONCAT('pub(',CONCAT(article,')')) AS term1,
CONCAT('jnl(',CONCAT(journal,')')) AS term2 FROM (SELECT id as article,
```

```
title, source as journal FROM dblp_pub_new WHERE type = 'article')
DummyTable) alias_5 , (SELECT DISTINCT CONCAT('pub(',CONCAT(article,')'))
AS term1,num AS term2 FROM (SELECT id as article, number_ as num FROM
dblp_pub_new WHERE type = 'article' and number_ is not null) DummyTable)
alias_6 , (SELECT DISTINCT CONCAT('pub(',CONCAT(article,')')) AS term1,
volume AS term2 FROM (SELECT id as article, volume FROM dblp_pub_new
WHERE type = 'article' and volume is not null) DummyTable) alias_7
WHERE alias_1.term2='GiuseppeDe Giacomo'  AND alias_0.term1=
alias_2.term1  AND alias_2.term1=alias_3.term1  AND alias_1.term1=
alias_3.term2  AND alias_3.term1=alias_5.term1  AND alias_4.term1=
alias_5.term2  AND alias_5.term1=alias_6.term1  AND alias_6.term1=
alias_7.term1
```

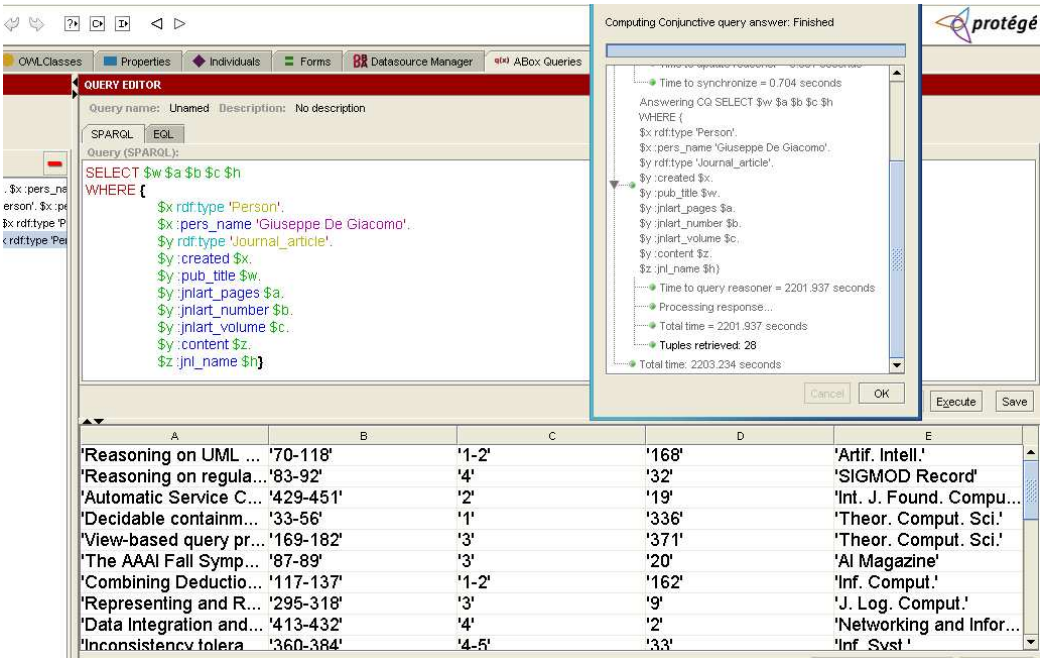

Figura 5.6: Risultato della query n.2

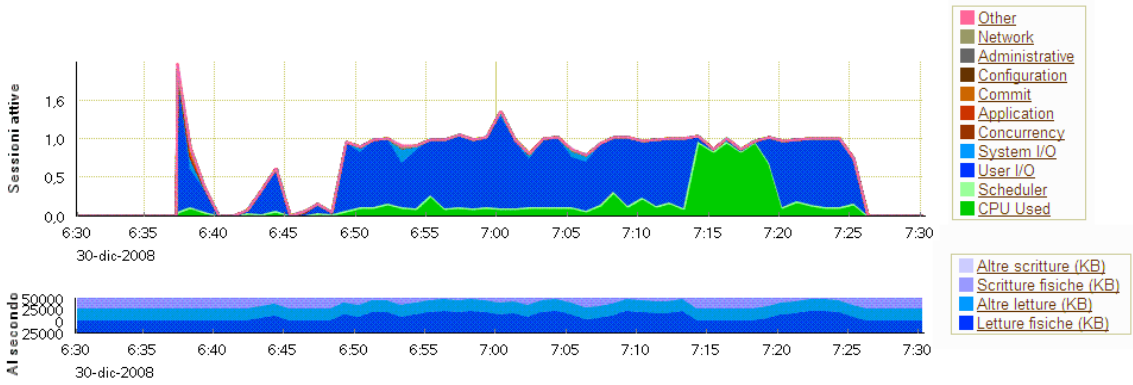

Figura 5.7: Report del db per la query n.2

5.5.3 Query n.3

La query richiede i titoli sia delle pubblicazioni citanti che di quelle citate.

```
SELECT $v $w
WHERE {
      $x rdf:type 'Publication'.
      $y rdf:type 'Publication'.
      $x cite $y.
      $x pub_title $v.
      $y pub_title $w}
```

Espansione della UCQ:

```
q(x) :- cite(x,y), pub_title(y,w), pub_title(x,v).
```

Unfolding della UCQ:

```
SELECT DISTINCT alias_2.term1 FROM ((SELECT DISTINCT
CONCAT('pub(',CONCAT (book,')')) AS term1,title
AS term2 FROM (SELECT id as book, title FROM
dblp_pub_new WHERE type = 'book') DummyTable)
UNION (SELECT DISTINCT CONCAT('pub(',CONCAT
(proceedings,')')) AS term1,title AS term2 FROM
(SELECT id as proceedings, title FROM dblp_pub_new
WHERE type = 'proceedings') DummyTable) UNION (SELECT
DISTINCT CONCAT('pub(',CONCAT(article,')')) AS term1,
title AS term2 FROM (SELECT id as article, title, source
```

```
as journal FROM dblp_pub_new WHERE type = 'article')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(', CONCAT
(m_thesis,')')) AS term1,title AS term2 FROM (SELECT id
as m_thesis, title FROM dblp_pub_new WHERE type =
'masterthesis') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(p_thesis,')')) AS term1,title
AS term2 FROM (SELECT id as p_thesis, title FROM
dblp_pub_new WHERE type = 'phdthesis') DummyTable)
UNION (SELECT DISTINCT CONCAT('pub(',CONCAT(incollection,
')')) AS term1,title AS term2 FROM (SELECT a.id as
incollection, a.title, b.id as book FROM dblp_pub_new a,
dblp_pub_new b WHERE a.type = 'incollection' and a.crossref =
b.dblp_key) DummyTable) UNION (SELECT DISTINCT CONCAT
('pub(',CONCAT(inproceedings,')')) AS term1,title AS term2
FROM (SELECT a.id as inproceedings, a.title, b.id as
proceedings FROM dblp_pub_new a, dblp_pub_new b WHERE
a.type = 'inproceedings' and a.crossref = b.dblp_key)
DummyTable)) alias_0 , (SELECT DISTINCT CONCAT('pub
(',CONCAT(cite,')')) AS term1,CONCAT('pub(',CONCAT
(cited,')')) AS term2 FROM (SELECT a.id as cite, b.id
as cited FROM dblp_pub_new a, dblp_pub_new b,
dblp_ref_new c WHERE a.id= c.id and c.ref_id = b.dblp_key)
DummyTable) alias_1 , ((SELECT DISTINCT CONCAT
('pub(',CONCAT(book,')')) AS term1,title AS term2 FROM
(SELECT id as book, title FROM dblp_pub_new WHERE type =
'book') DummyTable) UNION (SELECT DISTINCT CONCA
T('pub(',CONCAT(proceedings,')')) AS term1,title AS
term2 FROM (SELECT id as proceedings, title FROM
dblp_pub_new WHERE type = 'proceedings') DummyTable)
UNION(SELECT DISTINCT CONCAT('pub(',CONCAT(article,')'))
AS term1,title AS term2 FROM (SELECT id as article, title,
source as journal FROM dblp_pub_new WHERE type ='article')
DummyTable) UNION (SELECT DISTINCT CONCAT('pub(',
CONCAT(m_thesis,')')) AS term1,title AS term2 FROM
(SELECT id as m_thesis, title FROM dblp_pub_new WHERE
type = 'masterthesis') DummyTable) UNION (SELECT DISTINCT
CONCAT('pub(',CONCAT(p_thesis,')')) AS term1,title AS
term2 FROM (SELECT id as p_thesis, title FROM
dblp_pub_new WHERE type = 'phdthesis') DummyTable)
UNION (SELECT DISTINCT CONCAT('pub(',CONCAT
(incollection,')')) AS term1,title AS term2 FROM
```

```
(SELECT a.id as incollection, a.title, b.id as
book FROM dblp_pub_new a, dblp_pub_new b WHERE
a.type = 'incollection' and a.crossref =
b.dblp_key) DummyTable) UNION (SELECT D
ISTINCT CONCAT('pub(',CONCAT(inproceedings,')'))
AS term1,title AS term2 FROM (SELECT a.id as
inproceedings, a.title, b.id as proceedings FROM
dblp_pub_new a, dblp_pub_new b WHERE a.type =
'inproceedings' and a.crossref = b.dblp_key)
DummyTable)) alias_2 WHERE alias_0.term1=
alias_1.term2  AND alias_1.term1=alias_2.term1
```

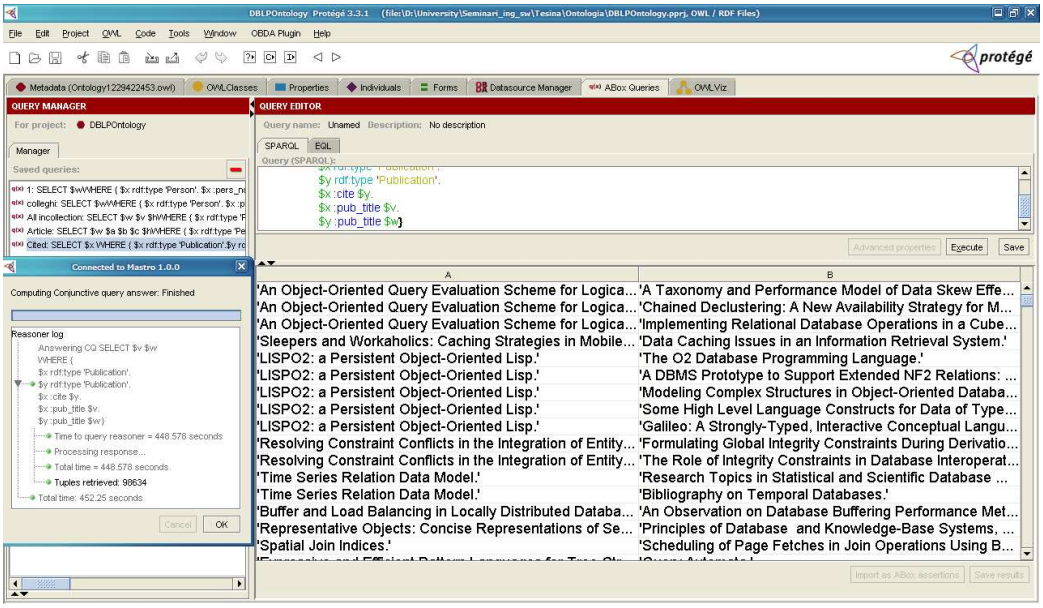

Figura 5.8: Risultato della query n.3

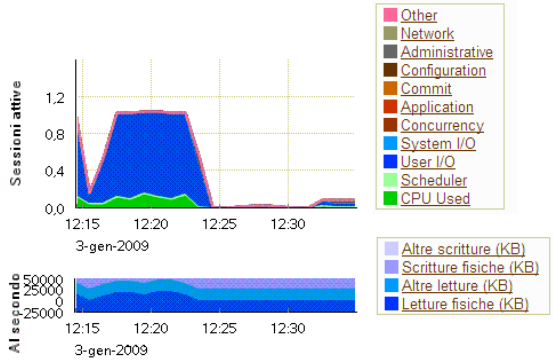

Figura 5.9: Report del db per la query n.3

5.5.4 Query n.4

La query richiede tutti i coautori del prof. De Giacomo.

```
SELECT $w
WHERE {
      $x rdf:type 'Person'.
      $x pers_name 'Giuseppe De Giacomo'.
      $y rdf:type 'Publication'.
      $y created $x.
      $y created $z.
      $z pers_name $w}
```

Espansione della UCQ:

```
q(w) :- pers_name(z,w), created(y,x),
pers_name(x,'Giuseppe De Giacomo'), created(y,z).
```

Unfolding della UCQ:

```
ELECT DISTINCT alias_2.term2 FROM (SELECT DISTINCT
CONCAT('pub(',CONCAT(publication,')')) AS term1,
CONCAT('pers(',CONCAT(author,')')) AS term2 FROM
(SELECT author, id as publication FROM
dblp_author_ref_new WHERE editor = '0') DummyTable)
alias_0 , ((SELECT DISTINCT CONCAT('pers(',
CONCAT(author,')')) AS term1,author AS term2 FROM
(SELECT author, id as publication FROM
```

```
dblp_author_ref_new WHERE editor = '1') DummyTable)
UNION (SELECT DISTINCT CONCAT('pers(',CONCAT(author,')'))
AS term1,author AS term2 FROM (SELECTauthor, id as
publication FROM dblp_author_ref_new WHERE editor = '0')
DummyTable)) alias_1 , ((SELECT DISTINCT CONCAT('pers(',
CONCAT(author,')')) AS term1,author AS term2 FROM
(SELECT author, id as publication FROM dblp_author_ref_new
WHERE editor = '1') DummyTable) UNION (SELECT DISTINCT
CONCAT('pers(',CONCAT(author,')')) AS term1,author AS term2
FROM (SELECT author, id as publication FROM dblp_
author_ref_new WHERE editor = '0') DummyTable)) alias_2 ,
(SELECT DISTINCT CONCAT('pub(',CONCAT(publication,')'))
AS term1,CONCAT('pers(',CONCAT(author,')')) AS term2
FROM (SELECT author, id as publication FROM
dblp_author_ref_new WHERE editor = '0') DummyTable)
alias_3 WHERE alias_1.term2='Giuseppe De Giacomo'
AND alias_0.term2=alias_2.term1
AND alias_0.term1=alias_3.term1
AND alias_1.term1=alias_3.term2
```

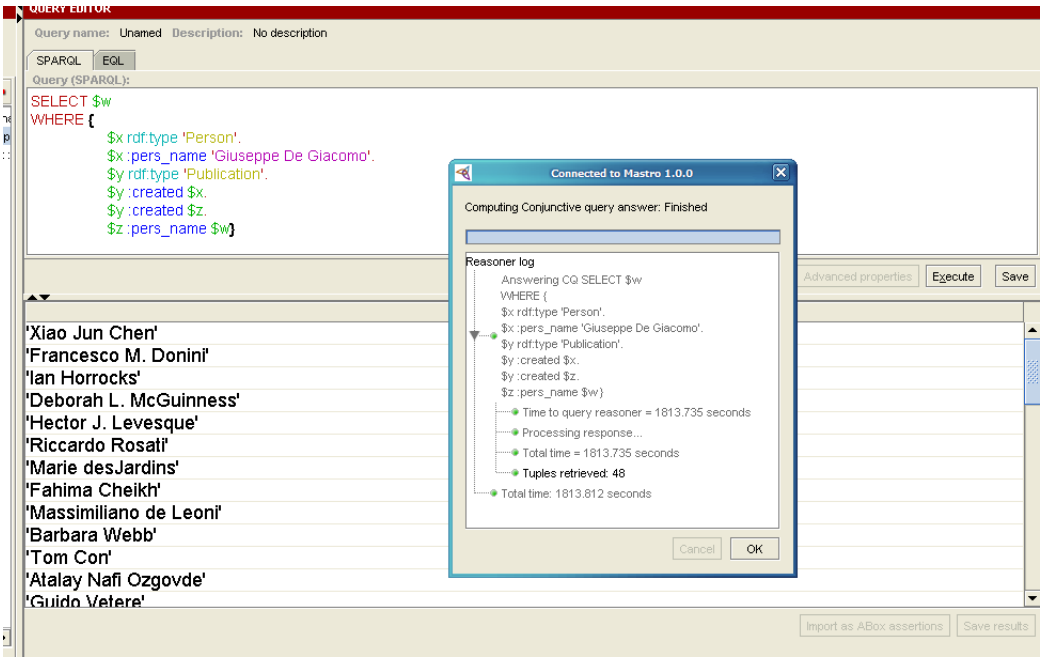

Figura 5.10: Risultato della query n.4

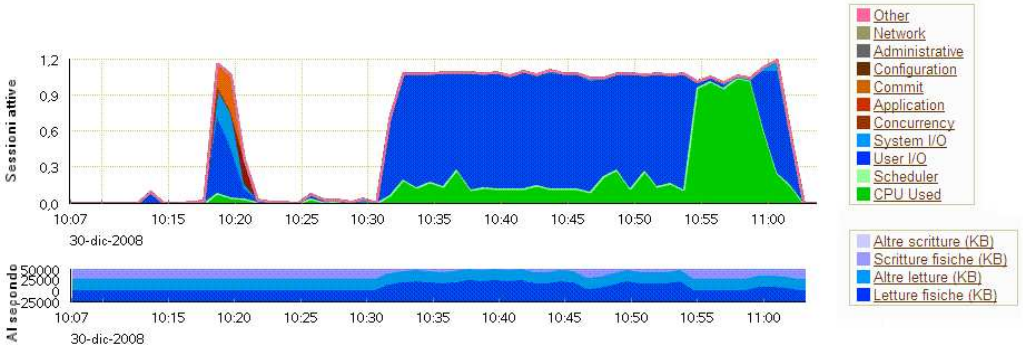

Figura 5.11: Report del db per la query n.4

Bibliografia

[1] K. Carey and S. Blatnik. *Guida a XML (XML: Content and Data)*. chapter 6 - 7. McGraw-Hill, 2002.

[2] G. De Giacomo. Dispense del corso di *Seminari di ingegneria del software*. `http://www.dis.uniroma1.it/ degiacomo`.
anno accademico 2006-2007.

[3] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. *Linking Data to Ontologies*.

[4] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. *MASTRO-I: Efficient integration of relational data through DL ontologies*.

[5] M. Ley. Materiale presente sul sito:
`http://www.informatik.uni-trier.de/ ley/db/`.

[6] R. Ramakrishnan and J. Gehrke. *Sistemi di basi di dati*. McGraw-Hill.

[7] P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone. *Basi di dati: modelli e linguaggi di interrogazione*

[8] Marc Baudoin. *Apprends* LaTeX. 1994-1998