SAPIENZA
UNIVERSITÀ DI ROMA

Laurea specialistica in Ingegneria Informatica

a.a. 2008/2009

Corso di Seminari in Ingegneria del Software

Prof. Giuseppe de Giacomo

"Traduzione dei diagrammi ER in DL-Lite$_A$ ed in sintassi OWL DL, query answering sulle ontologie risultanti."

Flavio Refrigeri

Indice

1 Introduzione

Il seguente elaborato ha come obbiettivo la traduzione dei diagrammi *ER* relativi ad alcuni testi di esame dell'esame di Basi di Dati in sintassi funzionale *OWL* ed in sintassi *OWL DL*, la realizzazione del query answering sulle ontologie risultanti e l'analisi dei limiti espressivi di tali linguaggi, il lavoro si suddivide in due parti:

1 Definizione delle ontologie mediante la sintassi funzionale *OWL*, utilizzo del toolkit *Matro/QuoOnto* per effettuare le queries dei compiti d'esame direttamente sull'ontologia precedentemente definita utilizzando il linguaggio *SparSQL*.
La sintassi funzionale *OWL* non permette la definizione di tutti i vincoli o specifiche presenti nei diagrammi *ER*, tali vincoli o specifiche non esprimibili dovranno essere messi in evidenza e, nel caso sia possibile, si dovranno utilizzare delle metodologie alterative per garantire che l'ontología risultante li rispetti.

2 Una volta tradotti gli schemi in sintassi funzionale *OWL* si dovrà utilizzare l'editor Protègè 4.0 per tradurre l'ontologia in un file .owl espresso mediante il linguaggio *OWL DL*.A questo punto si dovrà utilizzare il toolkit *Proteègè 3.1* de in particolare il *plugin OBDA* per effettuare il mapping tra l'ontologia e la base di dati ed effettuare le queries dei compiti d'esame direttamente sull'ontologia mettendo in evidenza i vincoli che non possono essere espressi.

1.1 Introduzione al Web Semantico

Ogni anno che passa e' sempre più evidente che la tecnologia entra a far parte integrante della nostra vita,supportandoci alla gestione delle nostre attività, sostituendoci a volte, permettendoci di comunicare con estrema facilità e di accedere ad una enorme quantità di informazioni, cosa che già solo trent'anni fa era impensabile

Basti pensare che 'solo' venti anni fa veniva prestato presso il CERN (*Conseil Européen pour la Recherche Nucléaire)* di Ginevra da Tim Berners Lee il documento "*Information Management a Proposal"*, che descriveva un progetto di elaborare un software per la condivisione di documentazione scientifica in formato elettronico indipendente dalla piattaforma utilizzata, con l'obbiettivo di ottimizzare la comunicazione e la cooperazione dei ricercatori dell'istituto.

In pochi il *World Wide Web* si e' sviluppato ad un ritmo frenetico ed inevitabilmente e' diventato parte integrante della nostra vita. Oggi però non possiamo affermare che lo sviluppo del Web sia terminato, dato che rimane ancora una grande raccolta on-line di pagine *HTML* statiche. Il problema e' che tale struttura e' congegnale agli umani, perché facilmente comprensibile, pero non e' altrettanto efficiente per essere elaborata. Vediamo il seguente esempio:

<?xml version="1.0"?>
<catalogo>
<sedia> <legno> ciliegio < legno /> <prezzo> 10 </ prezzo > </ sedia >
< tavolo > < legno > noce < legno /> < prezzo > 100 </ prezzo > </tavolo>
</catalogo>

Come vediamo per noi e' immediato comprendere che si tratti di un Catalogo che contiene informazioni su determinati prodotti, ma per un computer e' altrettanto chiaro?

Come potrebbe un elaboratore comprendere che catalogo e' sinonimo di Lista di Prezzi e rendere piu' completa un ipotetica query effettuata mediante un motore di ricerca?Quello che manca affinché tali informazioni possano essere utilizzate ed elaborate automaticamente e' una struttura che definisca in maniera rigorosa le relazioni tra le risorse e il significato delle risorse stesse. Questa e' una delle idee che stanno alla base del Web Semantico, proposto, ancora una volta da Tim Berners Lee.

A questo punto appare e' chiaro che per realizzare il *Web Semantico* abbiamo bisogno di linguaggi che permettano la rappresentazione formale di queste informazioni.

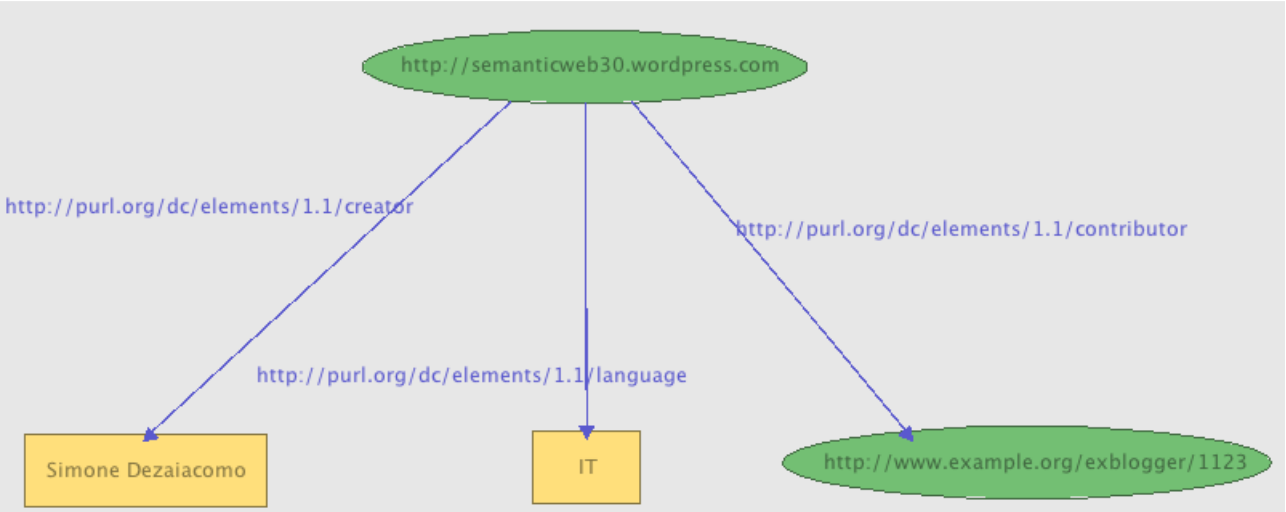
Il *W3C* ha definito alcuni linguaggi che consentono la descrizione di tali "Meta-informazioni" di seguito elencherò i piu importanti in relazione al contenuto della tesina.

2 Linguaggi per la definizione di ontologie

2.1 The Resource Description Framework (RDF)

E' un linguaggio che permette di descrivere in maniera concettuale le risorse, è uno standard delW3C e la sua struttura si costituisce di triple :

- **Soggetto(Risorsa):** una risorsa fisica o concettuale identificata mediante un URI.
- **Predicato(Proprietà):**Descrive la relazione tra Soggetto ed Oggetto.
- **Oggetto(Valore):** E' il valore della relazione predicato, questo può essere un valore o una risorsa. Vediamo un esempio di un grafico *RDF* ottenuto con il tool IsaViz:



2.2 Ontology Web Lenguage(OWL)

Il linguaggio ontologico *OWL* è stato progettato per permettere alle applicazioni che lo utilizzano di elaborare il contenuto delle informazioni che il linguaggio stesso descrive.*OWL* agevola una maggiore interpretabilità dei contenuti del Web da parte delle macchine rispetto a quanto permettano *XML,RDF e RDF Schema.*

La realizzazione del Web Semantico ha l'obbiettivo di permettere alle macchine di elaborare ed integrare in maniera automatica le informazioni disponibili nel Web ed *OWL* è stato progettato per rispondere a queste necessità, permettendo, a differenza di *RDF* di avere un' espressività maggiore,

consentendo la descrizione di proprietà come le relazioni tra classi, la cardinalità(per esempio "esattamente uno"), l'uguaglianza, la simmetria, etc..

Chiaramente avere un linguaggio per la definizione di ontologie molto espressivo non ha solo aspetti positivi, tale espressività infatti si "paga" in termini di efficienza computazionale, caratteristica importantissima se pensiamo che quando lavoriamo ed elaboriamo dati provenienti dal Web possiamo avere milioni di istanze, in questi casi la efficienza computazionale diventa indispensabile per garantire tempi di risposta accettabili.

Risulta quindi chiaro che nell'adottare un linguaggio di definizione di ontologie si deve trovare un giusto compromesso tra espressività del linguaggio ed efficienza computazionale, per venire incontro a questa esigenza il *W3C* ha definito dei sottolinguaggi di *OWL* che si differenziano proprio in funzione del potere espressivo:

- OWL *Full:* destinato agli utenti che necessitano della massima espressività, non è però decidibile.
- OWL *DL:* aiuta gli utenti che vogliono ottime caratteristiche di espressività, mantenendo però la completezza computazionale(non garantita con OWL *Full*) e la decidibilità(tutte le computazioni finiscono in un tempo finito). Il nome del linguaggio (OWL Description Logic) è tale per la sua rispondenza con la logica descrittiva, in seguito vedremo in dettaglio le corrispondenze.
- OWL *Lite:* pensato per gli utenti che hanno bisogno di una gerarchia di classificazione e di semplici restrizioni sulle risorse, ad esempio permette solo valori di cardinalità 0 e 1.

2.3 Description Logic

Dalle loro origini alla fine del 1970 in cui erano utilizzate per risolvere problemi di rappresentazione della conoscenza con reti semantiche, le *Description Logic* sono cresciute fino diventare l'unica vera chiave di volta nella storia della rappresentazione della conoscenza.
Il cuore della rappresentazione della conoscenza mediante la Logica Descrittiva è il linguaggio dei concetti, mediante il quale si rappresenta il dominio di conoscenza mediante un insieme di costrutti per denotare classi e relazioni tra classi:

- **Concetti:** sono usati per rappresentare le classi, che sono intese come insieme di individui.
- **Ruoli:** sono relazioni vinarie usate per specificare le proprietà o gli attributi delle classi.

Componenti principali che caratterizzano la logica descrittiva:

- **Un linguaggio descrittivo:** deve permettere la definizione di Concetti e Ruoli

    1. Si parte da un alfabeto finito di Concetti e Ruoli atomici.
    2. Successivamente si applicano ei costruttori mediante i quali possiamo costituire concetti e ruoli complessi partendo dai rispettivi concetti e ruoli atomici.

- **Un meccanismo che permetta di specificare la conoscenza come concetti e ruoli (TBox) mediante asserzioni sugli stessi:**
    1. Asserzione di inclusione tra concetti: $C_1 \sqsubseteq C_2$
    2. Asserzione di inclusione tra ruoli: $R_1 \sqsubseteq R_2$
    3. Ecco alcune asserzioni di proprietà sui ruoli atomici:
        a. *(transitive P)*
        b. *(symmetric P)*
        c. *(domain P C)*
        d. *(functional P)*
        e. *(reflexive P)*
        f. *(range P C)*

- **Un meccanismo che permette di specificare le proprietà degli oggetti(ABox):**

    1. Per i concetti: A(c)
    2. Per i ruoli: P(c1, c2)

- **In insieme di servizi di inferenza come fare ragionamento su una base di conoscenza.**

L'Ontologia risultante è quindi formata da una coppia $O = <T,A>$, ,questa distinzione è utile perché differenzia la visione del dominio dal punto di vista astratto con la visione del dominio dal punto di vista delle entità che lo compongono.

2.3.1 Complessità delle queries eseguite sulle Ontologie espresse in DL

Dato il numero di istanze dei concetti che costituiscono il livello estensionale delle ontologie, risulta di fondamentale importanza analizzare la complessità del "query answering" utilizzando come parametro di misura la dimensione dei dati.
Andiamo ad analizzare la complessità rispetto alla dimensione dei dati relativa all'esecuzione di Union of Conjunctive Queries :

- Il costo è *LogSpace* per "plain databases".
- Il costo è *coNP-hard* con la disgiunzione nelle TBox.
- Il costo è *coNP-complete* per *Description Logics* molto espressive.

Sorge a questo punto il problema di trovare un sottoinsieme delle *DLs* per permettere una risoluzione più efficiente delle queries sulle ontologie.

2.4DL-Lite $_A$

Come detto in precedenza, quando si parla di linguaggi per la descrizione di ontologie si deve tenere in conto che l'obbiettivo non è solo la massima espressività ma piuttosto il giusto compromesso tra quest'ultima ed un accettabile costo computazionale.
L'evidente vantaggio di questo frammento della Description Logic è che permette di eseguire il query answering sull'ontologia in LogSpace rispetto alla complessità dei dati.
La base di conoscenza in *DL-Lite* $_A$ è una coppia <T,A> dove devono essere soddisfatte le seguenti condizioni:

1. Per ogni ruolo atomico o inverso di un ruolo atomico Q che appare in un concetto della forma $\exists Q.C$, l' asserzione (funct Q) e (funct Q^-) non può essere espressa in T;
2. Per ogni asserzione di inclusione tra ruoli $Q \sqsubseteq R$ in T. dove R è un ruolo atomico o l' inverso di un ruolo atomico, l' asserzione (funct R) e (funct R^-) non appartiene a T;
3. Per ogni asserzione di inclusione di attributivi concetto $U_c \sqsubseteq V_c$ in T, dove V_c è un attributo di concetto atomico, l' asserzione (funct V_c) non appartiene a T;.
4. Per ogni asserzione di inclusione che riguarda attributi di ruolo, $U_R \sqsubseteq V_R$ in T dove V_R è un attributo di un ruolo. l'asserzione (funct V_R) non può appartenere a T.

2.4.1 Query answering in DL-Lite A

Effettuare il query answering sulle ontologie si reduce alla realizzazione di query del primo ordine su un database relazionale rappresentato dall'ABox.La query del primo ordine si ottiene riformulando la query originale in funzione dei vincoli presenti nella TBox, vediamone un esempio:

TBox: Pilota $\sqsubseteq \exists$conduce

$\exists$conduce$^- \sqsubseteq$ Auto

Query: q(x) ← conduce(x,y), Auto(y)

Riformulazione: q(x) ← conduce(x,y), Auto(y)

q(x) ← conduce(x,y),conduce(_,y)

q(x) ← conduce(x,_)

q(x) ← Pilota(x)

ABox: conduce(Schumacher, Ferrari)

Pilota(Senna)

E' facile capire che nonostante le informazioni dell'ABox siano incomplete, la risposta della query, sfruttando la riformulazione massimizza i risultati sfruttando la base di conoscenza (TBox).

Un altro aspetto importante è la separazione che viene effettuata tra le diverse asserzioni dela TBox, in particolare:

- **Positive inclusion(PIs) assertion:** sono le inclusioni che hanno un concetto/ruolo/attributo di concetto/attributo di ruolo positivo nella parte destra dell'asserzione stessa, vediamone un esempio

  $Cl \sqsubseteq A \mid \exists Q$

  $Q_1 \sqsubseteq Q_2$

- **Negative inclusion(NIs) assertion:** sono le inclusioni che hanno un concetto/ruolo/attributo di concetto/attributo di ruolo negato nella parte destra dell'asserzione stessa, vediamone un esempio

  $Cl \sqsubseteq \neg A \mid \neg \exists Q$

  $Q_1 \sqsubseteq \neg Q_2$

- **Functionality assertion:** Asserzioni della forma (funct φ), dove φ è un ruolo/inverso di un ruolo/attributo di un concetto atomico/attributo di ruolo atomico.

Una volta inserite nella TBox tutte le inclusioni negative logicamente implicite della TBox stessa, viene effettuato il query answering considerando separatamente gli insiemi PIs,NIs e fuctionally assertions.Piu' precisamente le NIs assertions e le functionality assertion sono rilevanti per la soddisfacibilità della base di conoscenza, inoltre in una base di conoscenza soddisfacibile, solo le PIs sono rilevanti per la risposta a UCQs.

Complessità del reasoning nelle DL-Lite A

Andiamo ora ad analizzare la complessità del reasoning su ontologie scritte utilizzando DL-LiteA, ponendo in evidenza la complessità richiesta per valutare la soddisfacibilità di un ontologia e la complessità del query answering:

- Valutare la soddisfacibilità di un ontologia è:
    1. *PTime* rispetto alla dimensione dell'ontologia (combined complexty).
    2. *LogSpace* rispetto alla dimensione dell'ABox (data complexity).
- La complessità del query answering è:
    1. *NP-complete* rispetto alla dimensione della query e dell' ontologia (combined complexity)
    2. *PTime* rispetto alla dimensione dell'ontologia.
    3. *LogSpace* rispetto alla dimensione dell'ABox (data complexity)

Possiamo a questo punto fare alcune considerazioni sui vantaggi offerti dalla *DL-Lite* $_A$, come abbiamo già precedentemente detto, definire un ontologia significa esprimere in un linguaggio elaborabile da una macchina i vincoli e le relazioni che sussistono tra i dati.
Se da un lato si vorrebbe la massima espressività di un linguaggio ontologico, al fine da poter esprimere tutte le relazioni che esistono tra i dati, dall'altro dobbiamo considerare che nel caso in cui la nostra ontologia sia costituita da un gran numero di istanze, l'espressività del linguaggio si paga in termini di complessità computazionale nel momento in cui si esegue il reasoning.
La *DL-Lite* $_A$ cerca di soddisfare entrambe le esigenze appena descritte, permettendo da un lato di esprimere buona parte delle caratteristiche di un modello concettuale dei dati, dall'altro offre delle ottime caratteristiche a livello di complessità computazionale.
Quest'ultima caratteristica è di estrema importanza, soprattutto perché molto spesso nelle ontologie si lavora con milioni di istanze, in questi casi è fondamentale che la complessità computazionale sia bassa per permettere tempi di risposta soddisfacenti e, nel caso delle *DL-Lite* $_A$ come abbiamo è visto la complessità del reasoning, effettuando il query answering è *LogSpace* rispetto alla dimensione della *ABox*.

2.4.2 La sintassi funzionale

La sintassi funzionale *OWL* permette di esprimere i vincoli del linguaggio mediante una sintassi facile da essere letta ed utilizzata dagli esseri umani,questo permette mediante costrutti testuali di rendere molto piu' intuitivo e semplice il compito dell'utente mentre si accinge a creare la *TBox* e *l'ABox* corrispondenti,le quali saranno date in pasto ad un software che all'interno contiene un ragionatore che tradurrà gli statement espressi in sintassi funzionale in un espressione in logica descrittiva, il tutto viene eseguito in maniera totalmente trasparente all'utente.

2.4.3 1.1.7 La sintassi tedesca

A differenza della sintassi funzionale, che come abbiamo detto permette di definire l'ontología mediante dei costrutti testuali, la sintassi tedesca si basa su dei costrutti logico/matematici che permettono la descrizione della TBox.

3 Strumenti per la definizione e l'interrogazione di ontologie

3.1 Il Tool Mastro

Mastro è un tool sviluppato in Java per effettuare l'integrazione semantica dei dati,è basato sul QuOnto che vedremo in seguito, Mastro è un sistema che ed è stato progettato ed implementato dal *Dipartimento di Informatica e Sistemistica* dell'università *La Sapienza di Roma*. Permette la definizione di Ontologie utilizzando il linguaggio *DL-Lite* di effettuare il mapping con le sorgenti di dati e di valutare le query mediante la riformulazione delle stesse in queries espresse sulle sorgenti di dati.

3.2Il reasoner QuOnto

QuOnto è un reasoner che opera su Ontologie scritte con linguaggi appartenenti alla famiglia *DL-Lite* e nel nostro caso *Dl-Lite* $_A$, è stato progettato ed implementato dal *Dipartimento di Informatica e Sistemistica* dell'università *La Sapienza di Roma*, sviluppato in *Java*.Permette di gestire fino ad alcuni milioni d'istanza in memoria secondaria mantenendo una complessità di risposta alle query logaritmica (*LOGSPACE*) rispetto alla dimensione dei dati.

Alla base del ragionamento, sta l'idea di riscrivere le query usando le asserzioni che sono presenti nella *T-Box* (livello intensionale dell'ontología).

3.3 Query answering sulle Ontologie e la OWA

Quando parliamo di Ontologie e di Web Semantico, dobbiamo tener presente che molto spesso ci troviamo ad dover operare con dati incompleti, per far fronte a questo problema nelle ontologie si fa l'assunzione di mondo aperto (*OWA*), secondo la quale, tutto ciò che è nell'ontologia è vero, mentre tutto ciò che non è contenuto in essa non può essere definito né vero né falso.Come possiamo vedere il concetto di Open World Assumption(OWA) è contrastante con l'assunzione di mondo chiuso (Closed World Assumption) che tipicamente si adotta nelle basi di dati, nelle quali tutto cio' che non e' espresso viene considerato come falso.

Supponiamo ad esempio che si voglia fare un' integrazione di dati mediante le Ontologie, come spesso accade non tutte le informazioni sono presenti nel database, possiamo avere valori mancanti, dei dati incompleti, per questo nelle ontologie si lavora sotto la OWA, vediamone un esempio:

Consideriamo che la ABox sia costituita dalle seguenti tuple:

A= { Conduce(Schumacher, Ferrari), Conduce(Senna, McLaren), Conduce(Biasion, DeltaS4) }

Supponiamo che la interrogazione sia la seguente:

Chi ha al massimo una macchina?

Q(x): - $\exists$ y.Conduce(x,y) $\rightarrow$ $\neg$ $\exists$ z.(Conduce(x,z) $\wedge$ y $\neq$ z)

Risposta alla query sotto OWA: {}

Se invece avessimo interrogato una base di dati che aveva le stesse tuple, sotto una **CWA** avremmo ottenuto il seguente risultato: {Shumacher, Senna, Biasion}.

Assumendo quindi che le informazioni presenti nella nostra KB siano incomplete, dobbiamo interrogare le ontologie con un' ipotesi di mondo aperto, in questo caso però si presenta un problema aggiuntivo:

Le queries FOL sono indecidibili con OWA.

Per questo si utilizzano frammenti decidibili della FOL rappresentabili mediante *CQs* e *UCQs*, che comunque hanno delle limitazioni di espressività rispetto alla FOL in quanto non hanno l'operatore "not".Per poter sopperire a questa limitazione espressiva si dovrebbe effettuare una chiusura dinamica della conoscenza che vedremo nei prossimi due paragrafi.

3.4 Linguaggio SPARQL

Il linguaggio *SPARQL* (*SPARQL Protocol and RDF Query Lenguege*) è un linguaggio di query per dati standard *W3C*.Permette di effettuare queries con graph patterns, congiunzioni, disgiunzioni e pattern opzionali, il risultato di un query SPARQL può essere un insieme di risultati o grafi RDF.

Sintassi di una query SPARQL

Vediamo come è strutturata la sintassi di una query *SPARQL*

PREFIX dichiara prefissi e namespace.

SELECT definisce le variabili di ricerca da prendere in considerazione nel risultato.

FROM specifica il set di dati su cui dovrà operare la query.

WHERE definisce il criterio di selezione specificando tra parentesi graffe uno o più "triple patterns" separati dal punto.

3.5 Il linguaggio SparSQL

SparSQL è il linguaggio implementativo corrispondente a *EQL-Lite(UCQ)* , in cui il linguaggio di query immerso Q è costituito da *UCQ*.SparSQL nasce per soddisfare l'esigenza d avere un linguaggio di query più espressivo possibile, decidibile e con una complessità computazionale accettabile.

Sintassi di SparSQL

SparqSQL = SQL + SPARQL.

La sua sintassi quindi si basa sulla sintassi *SQL STANDARD* e la sintassi *SPARQL*, la sua struttura non è altro che la struttura di una query *SQL* con all'interno della clausola *FROM* una o più tabelle che sono il risultato di una o piu' query *SPARQL* valutate sull'ontologia.

Ecco la struttura:

SELECT *ListaAttributiOEspressioni*

FROM (sparqltable (<QuerySparql >) alias)+

[**where** *CondizioniSemplici*]

[**group by** *ListaAttributiDiRaggruppamento*]

[**having** *CondizioniAggregate*]

[**order by** *ListaAttributiDOrdinamento*]

Ad ogni sparqltable deve essere assegnato un alias che verrà utilizzato nell'ambito della query SparSQL per far riferimento a quella tabella.

Per recuperare il potere espressivo della *FOL SparSQL* effettua una chiusura dinamica della conoscenza, grazie alla quale è possibile esprimere l'operatore "not"che non può essere espresso nelle *UCQs.*

3.6 Protégé

Protégé e' una piattaforma open-source, basata in Java, che permette l'esportazione delle Ontologie in vari formati: *RDFS, XML Schema e OWL.*Puo' essere esteso utilizzando diversi tipi di plug-in e API per costruire tool ed applicazioni.

Pernette di utilizzare due differenti metodologie per la creazione di ontologie:

- **Protégé-Frames editor:** permette agli utenti di costruire e popolare ontologie che sono frame-based, in accordo con il protocollo Open Knowledge Base Connectivity (OKBC). In questo modello un ontología consiste in un set di classi organizzate secondo una gerarchia di sussunzione per rappresentare i concetti fondamentali del dominio, un insieme di slots associata a classi per descrivere le loro proprietà e relazioni, e un set di istanze di queste classi, esemplari individuali di concetti che hanno valori specifici per le loro proprietà.
- **Protégé-OWL editor:** permette all'utente di creare un ontología per il Web Semantico, in particolare in OWL, un ontología scritta in OWL può includere descrizioni di classi, proprietà e le loro istanze.

Nel nostro caso si è utilizzato Protégé-OWL editor per definire le ontologie, nei paragrafi successivi vedremo i dettagli di realizzazione delle ontologie stesse.

3.6.1 Protègè 4.0 per convertire la TBox in sintassi funzionale in una TBox OWL DL.

Come abbiamo già detto le ontologie sono state espresse in sintassi funzionale, per poter effettuare il mapping ed il query answering sulle ontologie utilizzando *Protege 3.3.1*, è necessario prima convertire le ontologie scritte in sintassi funzionale OWL in ontologie scritte mediante il linguaggio *RDF/XML*.Per covertire l'ontología nel formato *RDF/XML* ci siamo serviti della funzionalità di conversione offerta da Protege 4.0, che prende in input l'ontologia espressa in sintassi funzionale e la traduce in un ontologia espressa in linguaggio *RDF/XML*.

Per poter fornire in input la *TBox* in sintassi funzionale è stato necessario apportare alcune modifiche alla *TBox* stassa :

- LA TBox espressa in sintassi funzionale deve essere racchiusa dalla seguente intestazione :

  Namespace(=<http://NomeQualsiasi#>)
  Namespace(xsd=<http://www.w3.org/2001/XMLSchema#>)
  Namespace(rdfs=<http://www.w3.org/2000/01/rdf-schema#>)
  Namespace(owl=<http://www.w3.org/2002/07/owl#>)
  Namespace(rdf=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
  Ontology(http://NomeQualsiasi

  TBox espressa in sintassi funzionale

)

- La dichiarazione di range di ogni attributo va sostituita con la seguente dichiarazione di range:

  ✓ ogni attributo di tipo intero va scritto con la clausola finale *"xsd:int"*

  es: DataPropertyRange(num xsd:int)

- ✓ ogni attributo di tipo intero va scritto con la clausola finale string con xsd:string;

  es: DataPropertyRange(nome xsd:string)

- ✓ ogni attributo di tipo intero va scritto con la clausola finale date con xsd:date;

  es: DataPropertyRange(data xsd:date)

- ✓ ogni attributo di tipo intero va scritto con la clausola finale float con xsd:float;

  es: DataPropertyRange(media xsd:float

- La dichiarazione di cardinalità massima di ogni attributo va sostituita con la seguente dichiarazione:

  SubClassOf(nomeConcetto DataSomeValuesFrom(nomeAttributo rdf:XMLLiteral))

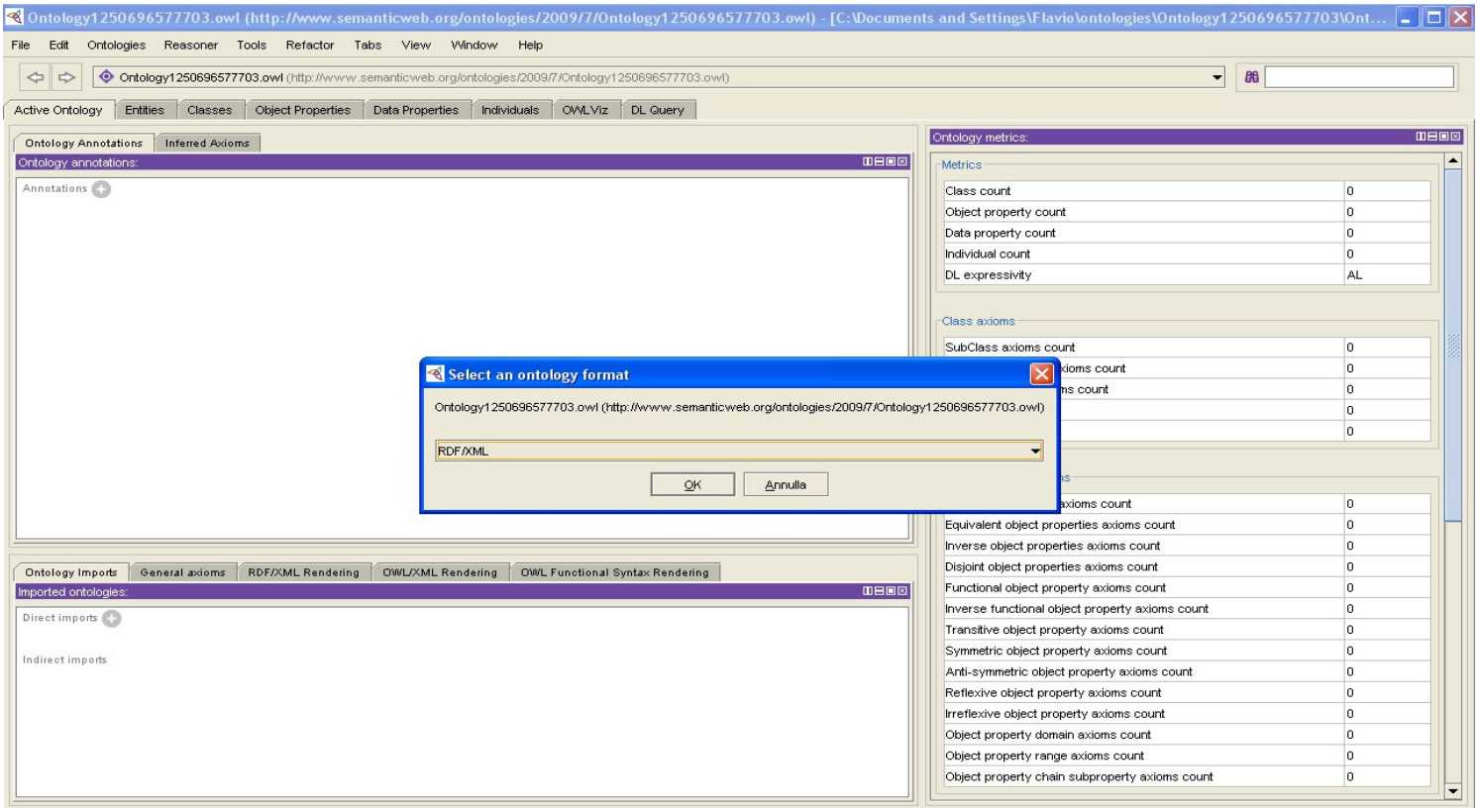
- La clausola *InverseObjectPropertyOf* va riscritto in *InverseObjectProperty*.

  FunctionalObjectProperty(InverseObjectProperty(NomeRuolo))

Una volta modificato la *TBox* scritta in sintassi funzionale come sopra-descritto si può eseguire *Protègè 4.0*, a questo punto dalla schermata principale di *Protègè* una volta aperta l'ontlogia, selezionando il file corrispondente cliccando su :
File -> Open

A questo punto su può generare il file *XML/RDF* cliccando su :
File -> Save as
Selezionando come formato *RDF/XML.*

Vediamo lo screenshot corrispondente:



3.6.2 Il Plugin OBDA per Protégé 3.3.1

L'Ontology Based Data Access è un' area di ricerca che ha come obbiettivo l'accesso a risorse eterogenee mediante l'utilizzo delle ontologie.Mediante la conoscenza intensionale fornita dalle ontologie, si rende più efficace la ricerca delle informazioni grazie alla capacità che ha il sistema di inferire nuova conoscenza sui dati. verificarne la integrità, effettuare l'integrazione semantica,etc.

OBDA plugin per Protégé e' un tool che permette di modellare alcuni elementi chiave di un sistema OBDA , permettendo all'utente di:

- Descrivere le sorgenti di dati di un sistema *OBDA*
- Descrivere il mapping che permette la connessione tra le sorgenti dei dati e le entità dell'ontologia
- Inviare la descrizione di queste componenti a un reasoner *OBDA*
- Effettuare queries a un reasoner *OBDA* e poter visualizzare i risultati.

Query answering con il plugin OBDA

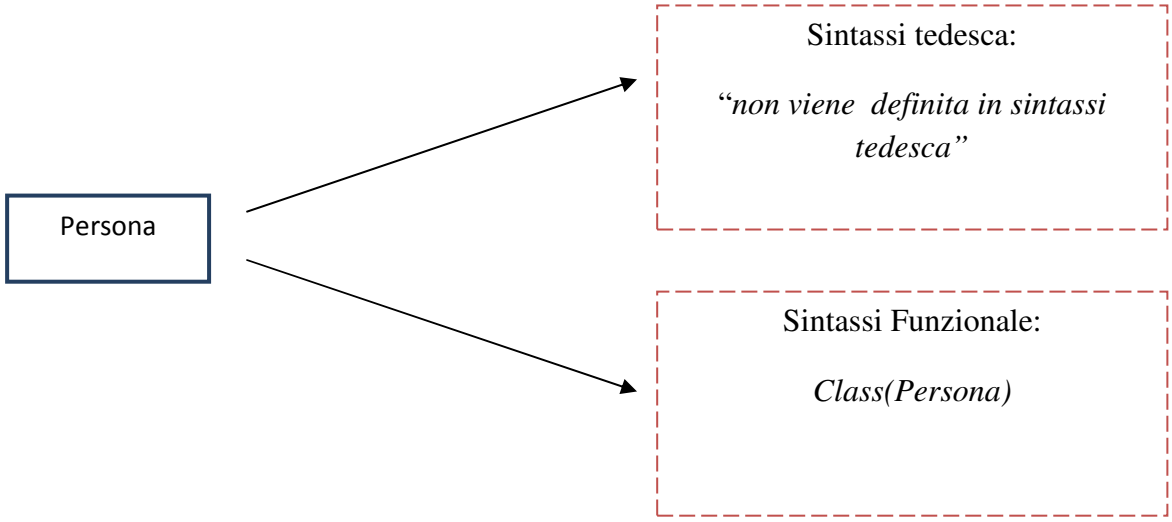
Mediante il plugin si può accedere alla schermata che permette di effettuare *UCQs SPARQL* sull'ontologia, una volta mandata in esecuzione la query, il reasoner DIG-Mastro sulla base dell'ontologia, le sorgenti di dati ed i mapping implementa una tecnica di riscrittura della query trasformando la UCQ in input in un insieme di query sulle sorgenti. I risultati vengono poi visualizzati dall'utente e il plugin permette ulteriori manipolazioni su di essi, come l'esportazione ed il salvataggio.

4 Dai diagrammi ER alla sintassi tedesca

Come già anticipato in precedenza, la sintassi tedesca permette la definizione della conoscenza intensionale (T-Box) mediante simboli logico/matematici. Le T-Box relative alle ontologie sono state realizzate a partire dagli schemi ER dei compiti relative a i compiti di Basi di Dati riportati in dettaglio nell'indice. In seguito verranno mostrate le traduzioni dei concetti le relazioni ed vincoli dei diagrammi ER in sintassi tedesca.

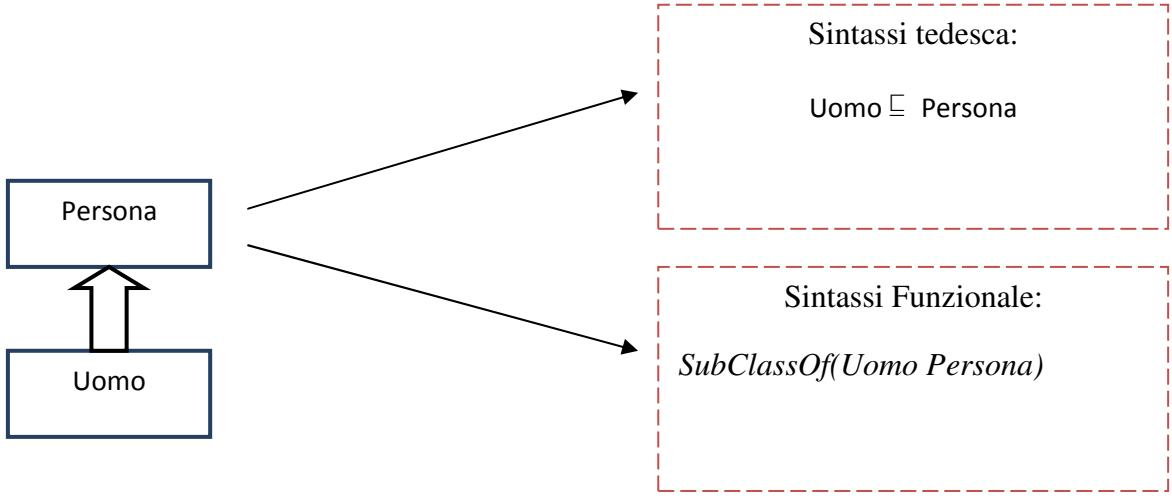
Entità

Diagramma ER:



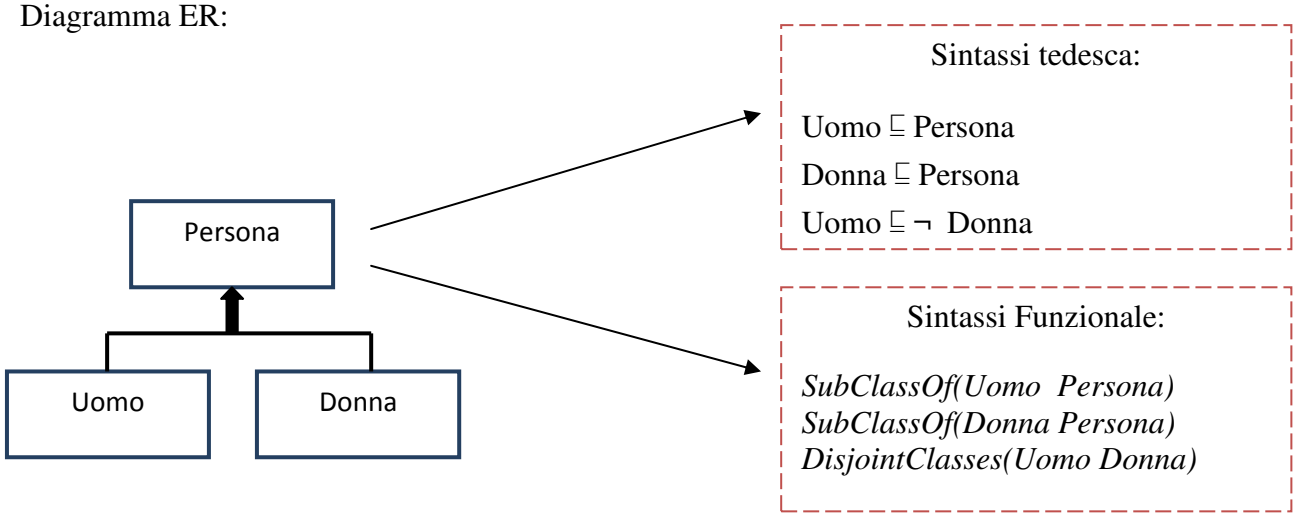
Ogni entità dello schema ER verrà tradotta in un concetto per l'ontologia.

ISA tra entità



Generalizzazione tra entità

Diagramma ER:



Come possiamo vedere è stato possibile esprimere la disgiunzione tra i due concetti (Uomo e Donna) non è però possibile esprimere la completezza, verrà successivamente utilizzata una query booleana per far rispettare tale vincolo.

Attributi di concetto

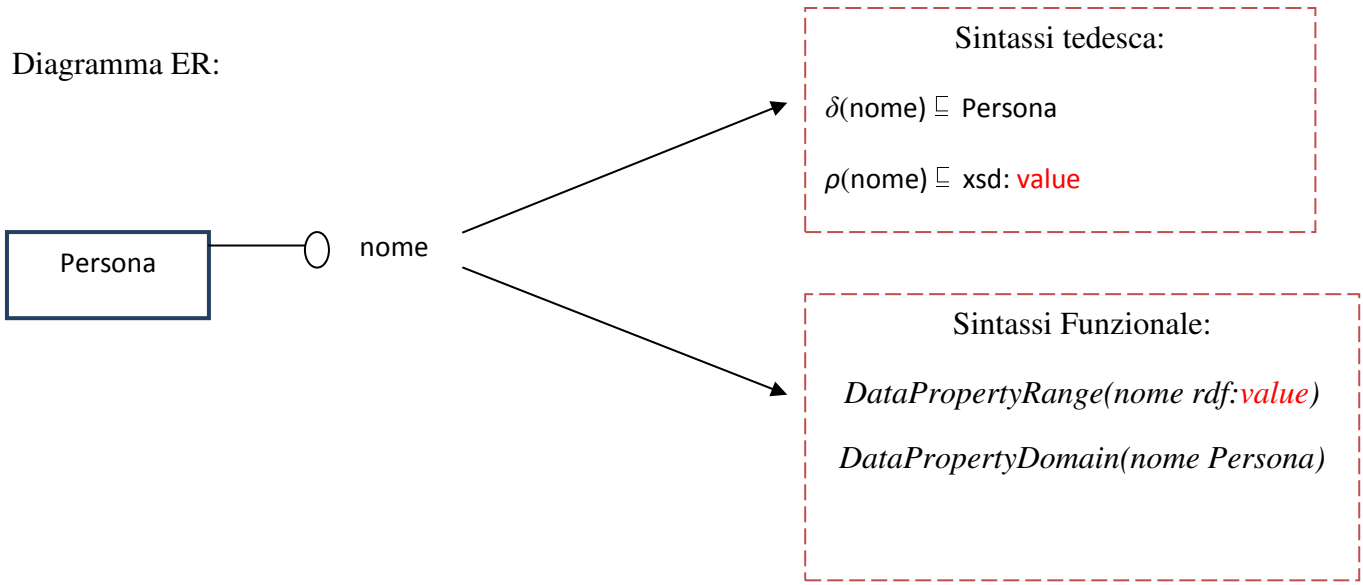
Mentre negli schemi ER se non viene specificata la cardinalità implicita degli attributi è (1,1), nelle ontologie la cardinalità implicita se non specificata è (0,n), è quindi necessario definire tali vincoli esplicitamente nel caso la molteplicità dell'attributo sia diversa da (0,n).

Per ogni attributo di concetto vanno definiti il dominio(il concetto a cui l'attributo si riferisce) e il relativo range(i valori che assume l'attributo.

Esistono inoltre delle limitazioni, in DL-Lite A infatti è possibile solo esprimere i vincoli di molteplicità per gli attributi pari a (0,n) (0,1) (1,1) (1,n), i restanti vincoli (quelli con molteplicità minima diversa da uno) devono essere verificati mediante la valutazione di query booleane sull'ontologia.

Dichiarazione di attributo di concetto

Diagramma ER:

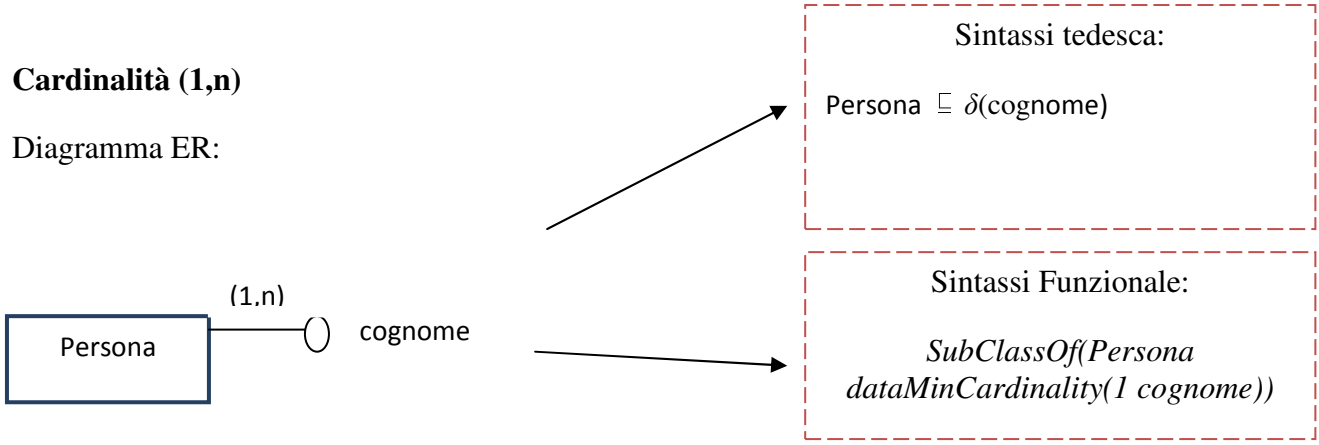


Cardinalità (0,n)

Come detto non deve essere specificata, in quanto è implicita con la dichiarazione dell'attributo

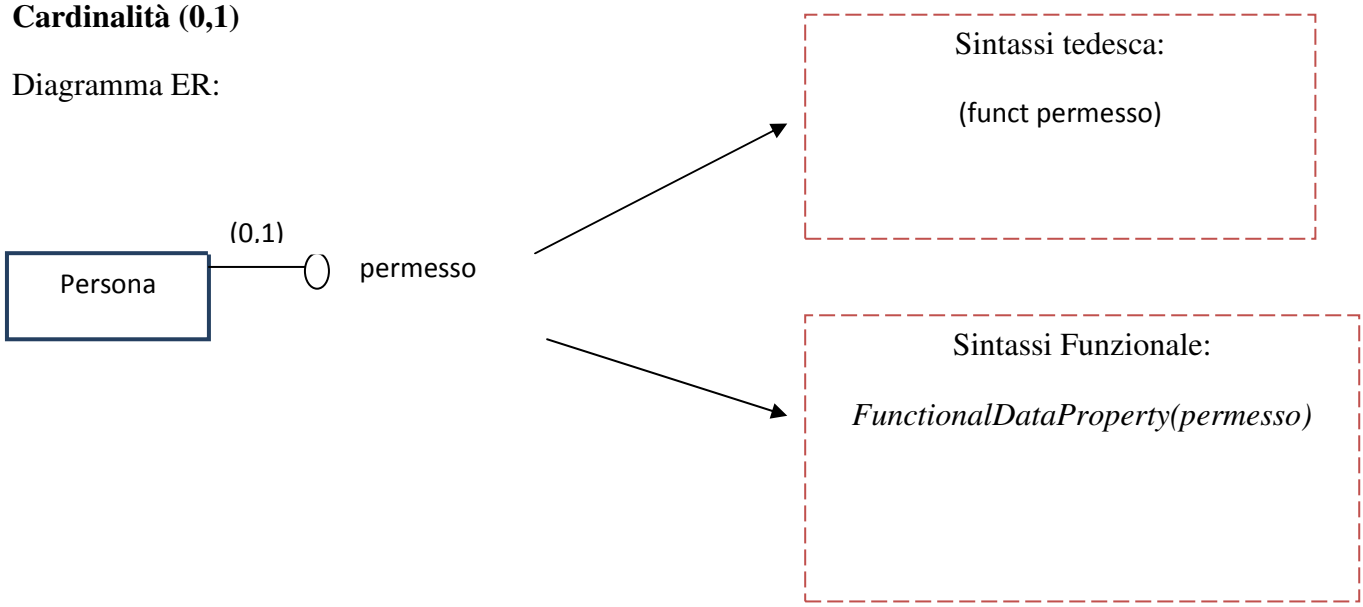
Cardinalità (1,n)

Diagramma ER:



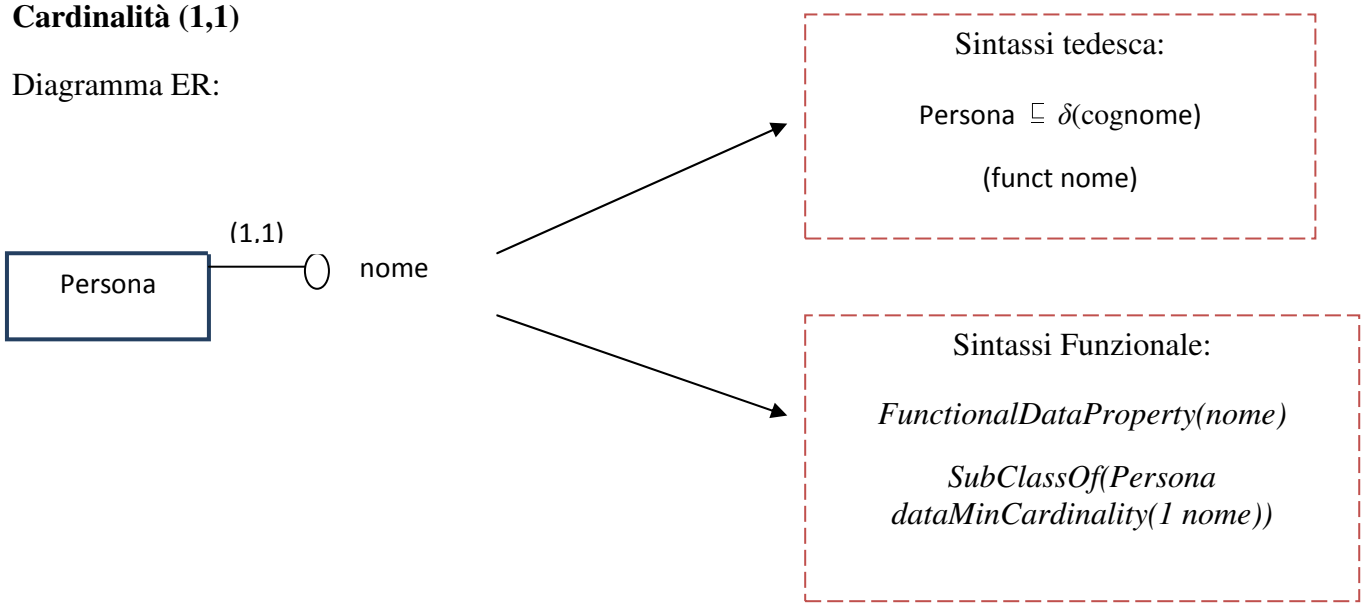
Cardinalità (0,1)

Diagramma ER:



Cardinalità (1,1)

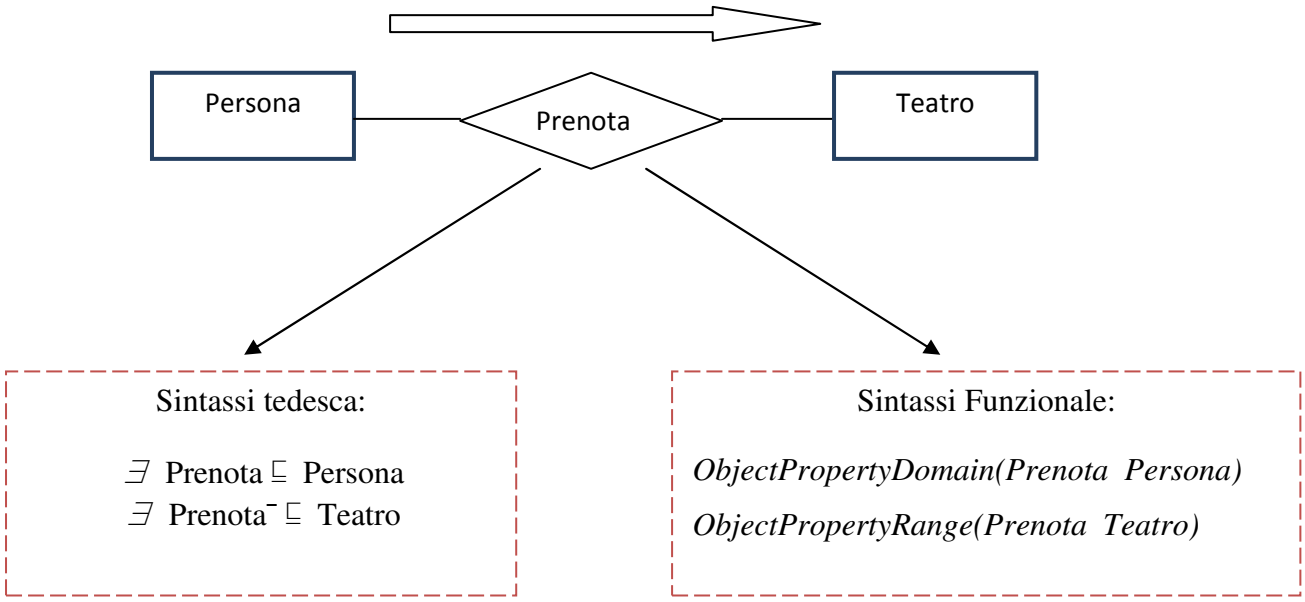
Diagramma ER:



Relazioni

Una relazione del diagramma ER si tradurrà in un ruolo dell'ontologia.Ogni ruolo dell'ontologia avrà a sua volta un Dominio(o concetto sorgente del ruolo) ed un Range(o concetto target del ruolo), per poter definire Dominio e Range di un ruolo si deve definire il relativo verso di percorrenza del ruolo stesso.

Diagramma ER

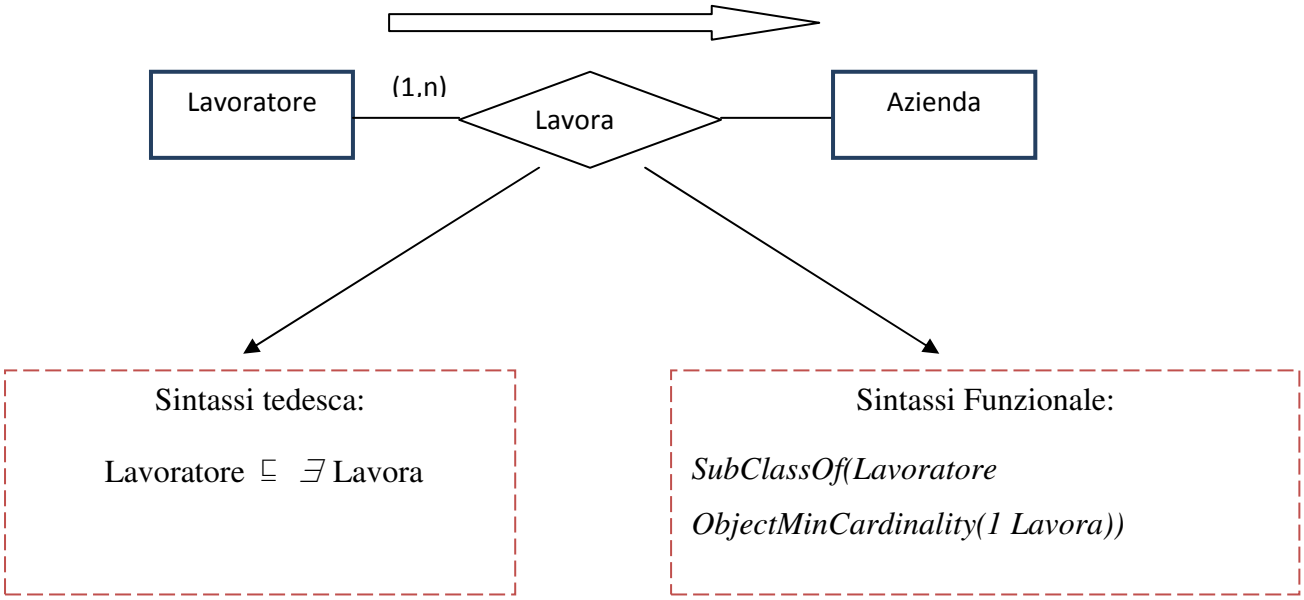


Relazioni (cardinalità 0-n)

Se non esplicitato, il vincolo di cardinalità implicito di un ruolo è (0,n) .

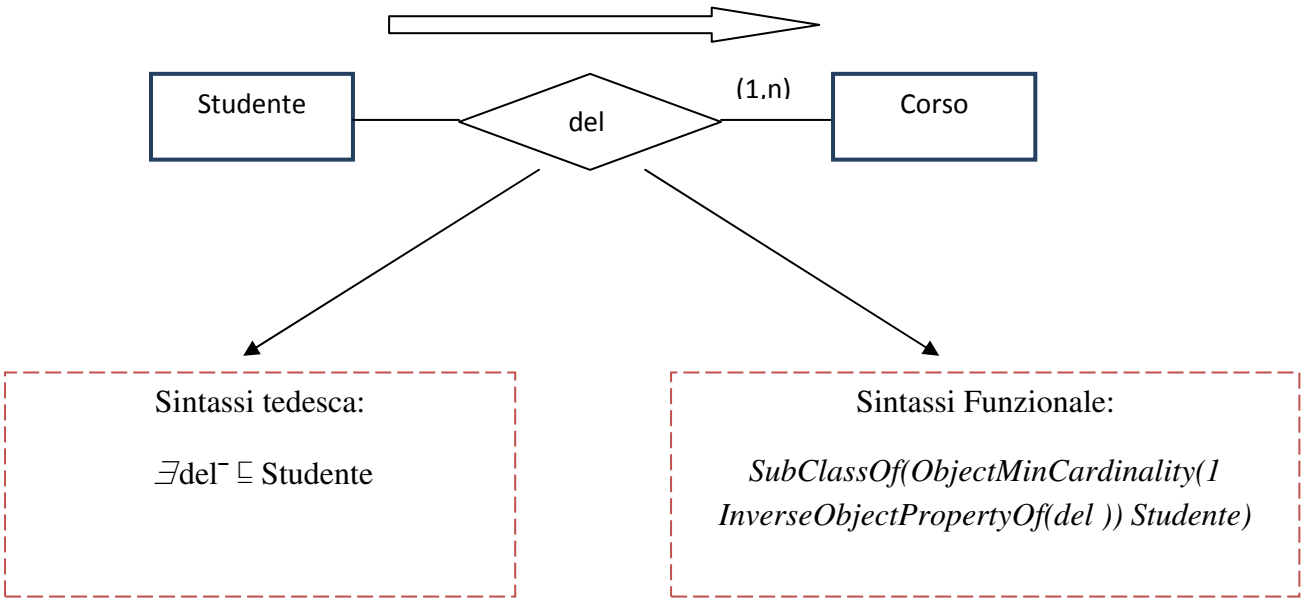
Relazioni (concetto sorgente ha cardinalità 1-n)

Diagramma ER



Relazioni (concetto target ha cardinalità 1-n)

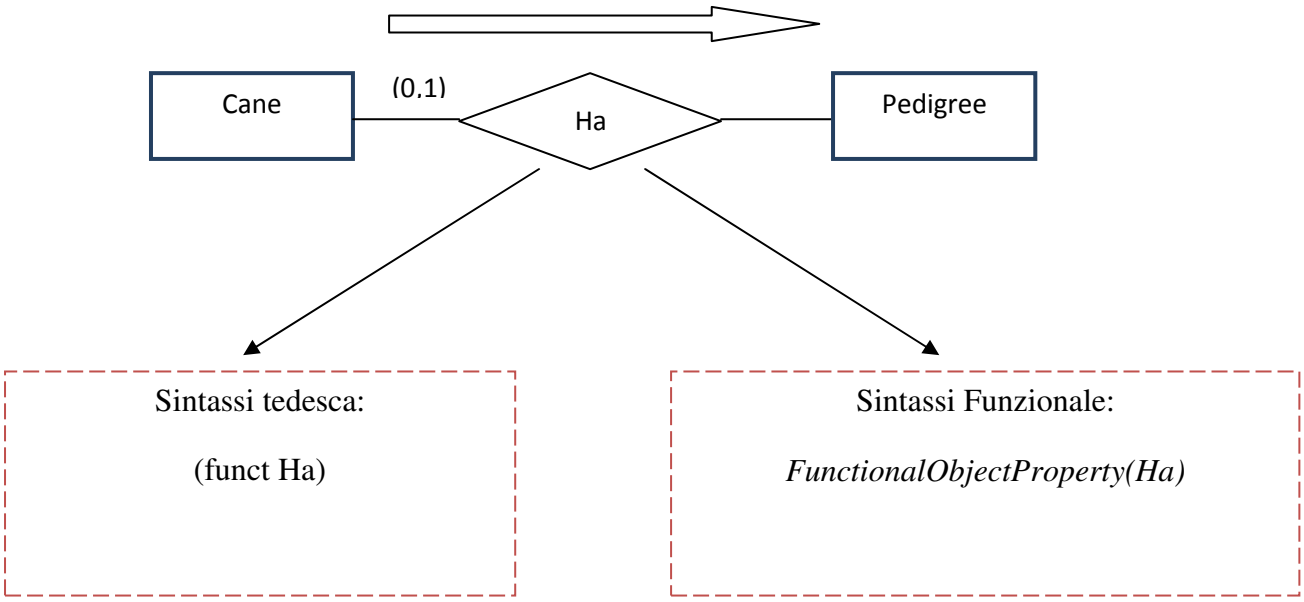
Diagramma ER



Relazioni (concetto sorgente ha cardinalità 0,1)

In questo caso dato che la cardinalità minima uguale a zero è implicita, dobbiamo solo esplicitare la cardinalità massima.

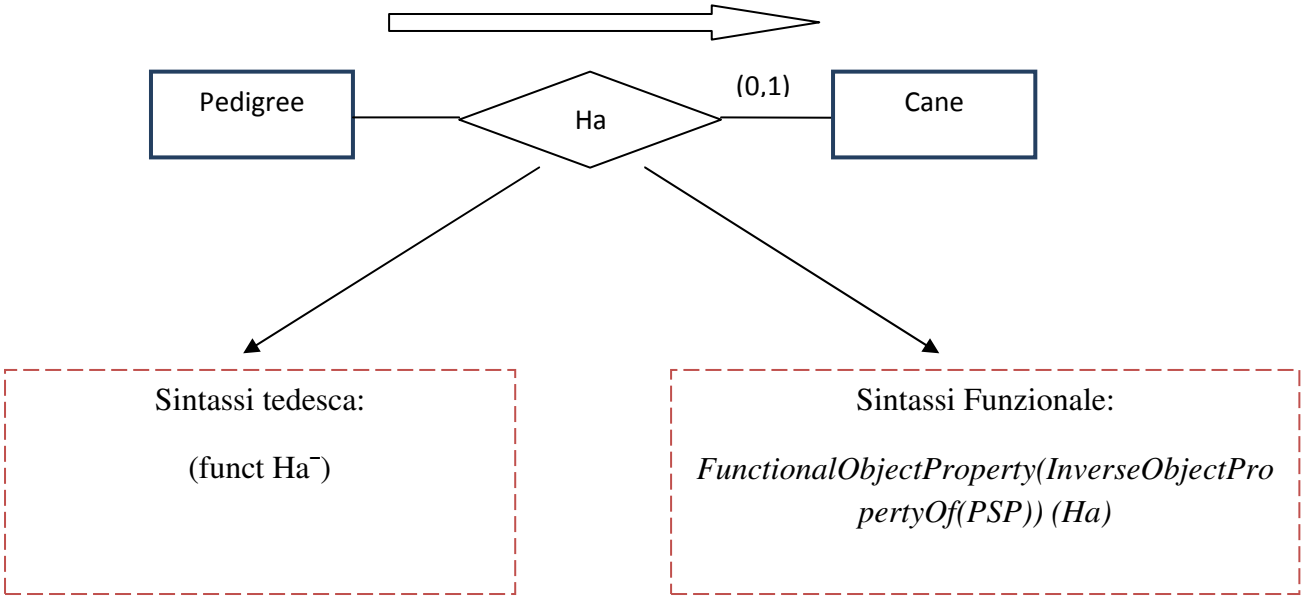
Diagramma ER



Relazioni (concetto target ha cardinalità 0,1)

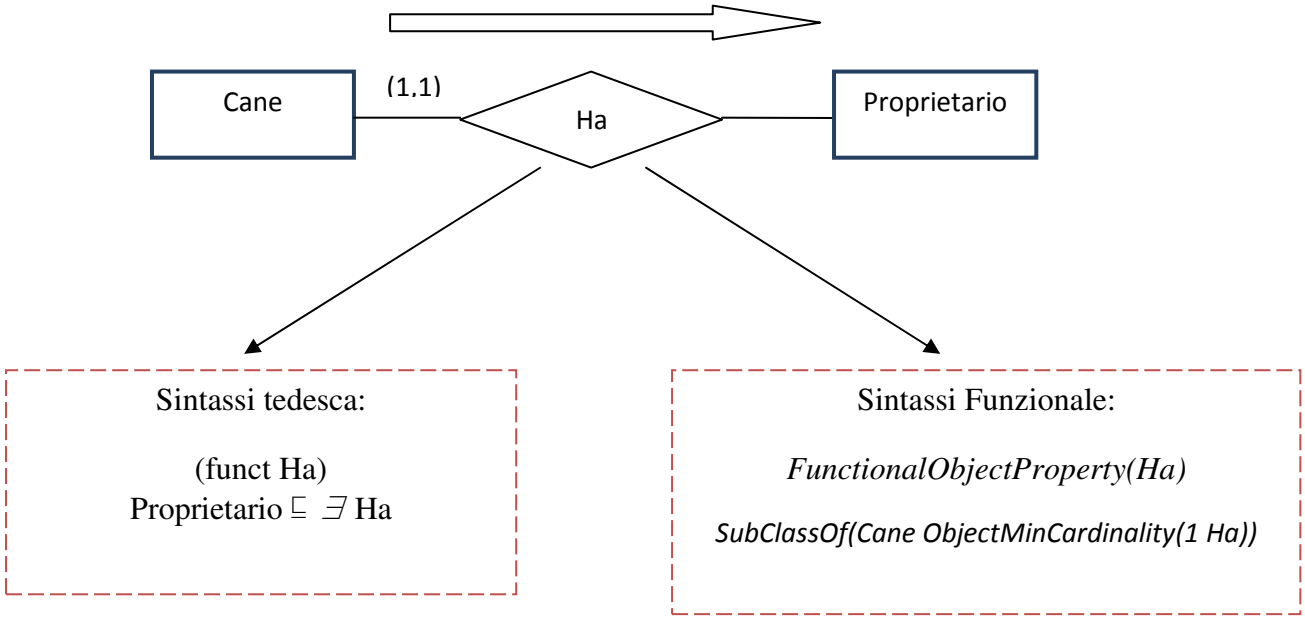
In questo caso dato che la cardinalità minima uguale a zero è implicita, dobbiamo solo esplicitare la cardinalità massima.

Diagramma ER



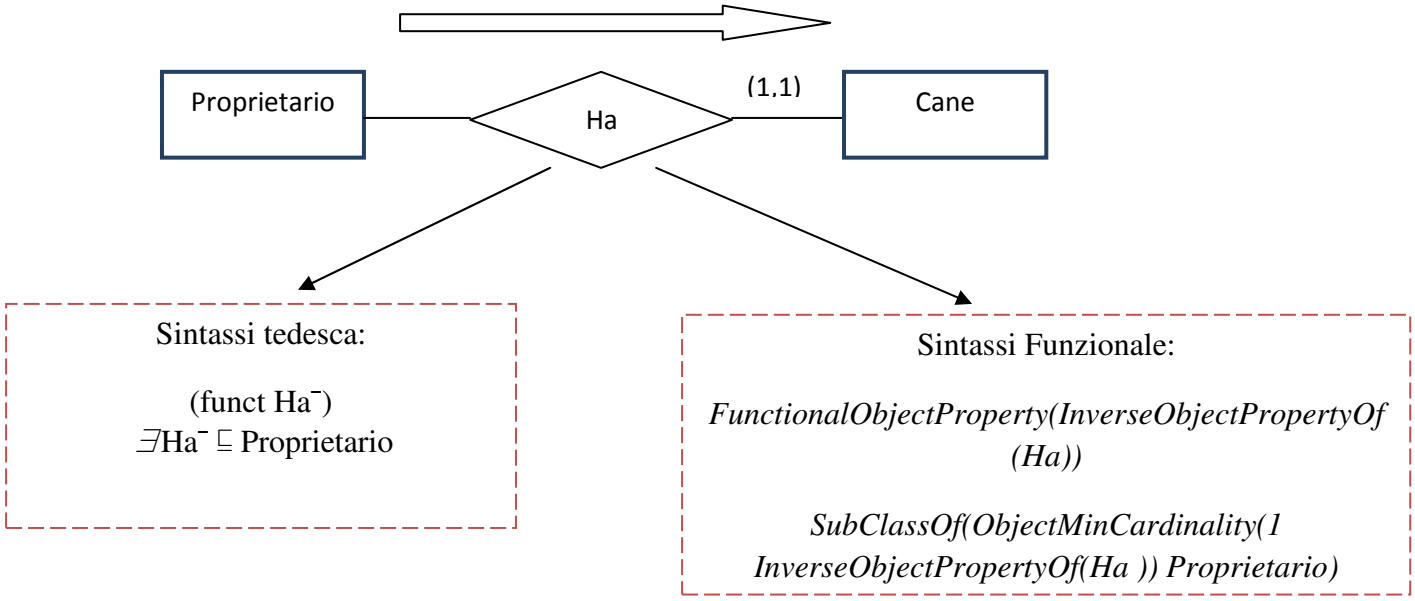
Relazioni (concetto sorgente ha cardinalità 1,1)

Diagramma ER



Relazioni (concetto target ha cardinalità 1,1)

Diagramma ER



Dichiarazione attributi di ruolo

Ogni attributo di una relazione dello schema ER va tradotto in un attributo di ruolo dell'ontología corrispondente. Come per il caso degli attribuiti di concetto, l'assunzione implicita che si adotta è di

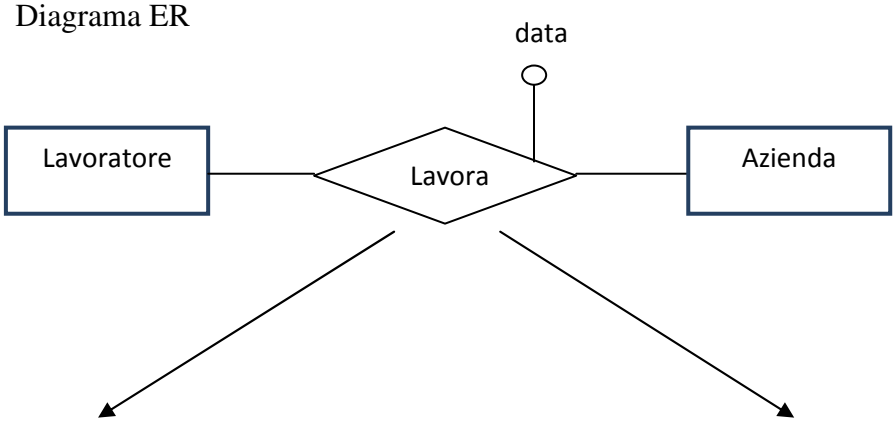
cardinalità (0,n) alla dichiarazione dell'attributo, pertanto se l'attributo di ruolo ha una cardinalità differente questa va definita esplicitamente.

Anche per quanto riguarda la limitazione sui vincoli, valgono le stesse considerazioni fatte per gli attributi di concetto,, tutti i vincoli diversi da (0,1) (1,1), (0,n), (1,n) vanno verificati mediante una query booleana SparSQL.

Per ogni attributo di ruolo vanno definiti il dominio(il ruolo a cui l'attributo si riferisce) e il relativo range(i valori che assume l'attributo.

Dichiarazione attributi di ruolo

Diagrama ER



Sintassi tedesca:

Lavora $\sqsubseteq \delta$(data)

δ(data) $\sqsubseteq$ Lavora

Sintassi Funzionale:

DataPropertyRange(data rdf: date)

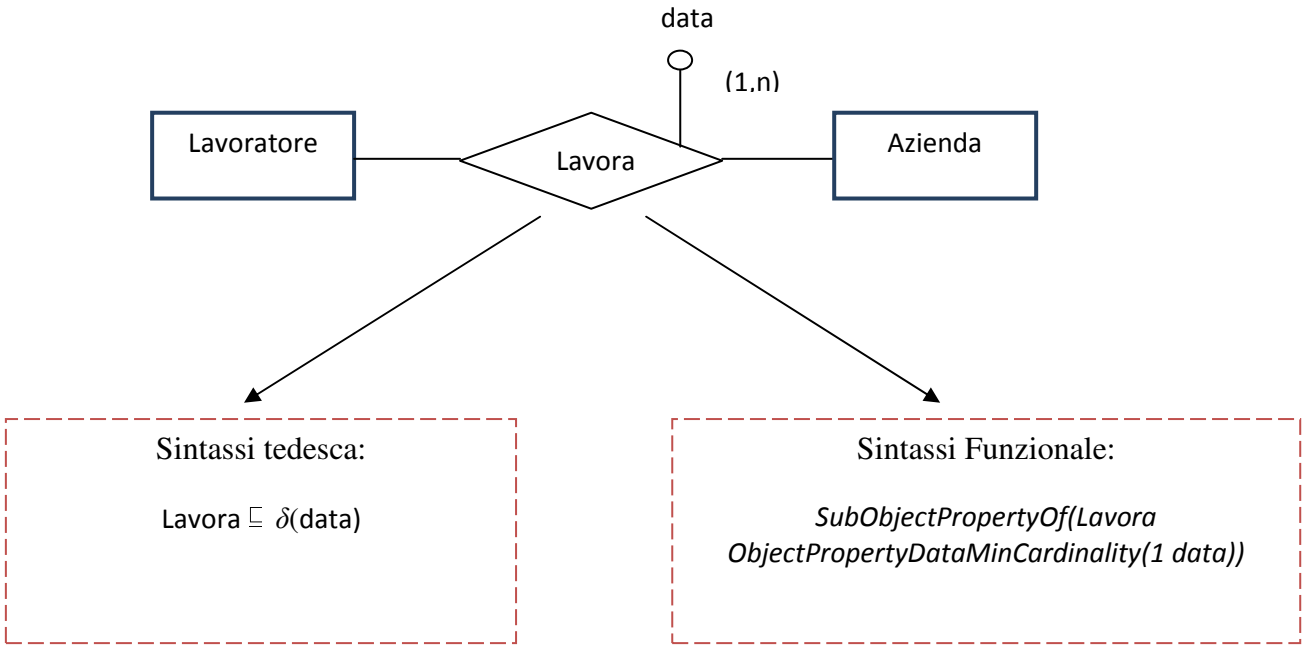
DataPropertyDomain(data Lavora)

Cardinalità (0,n)

Come già detto e' implicita momento in cui si dichiara l'attributo

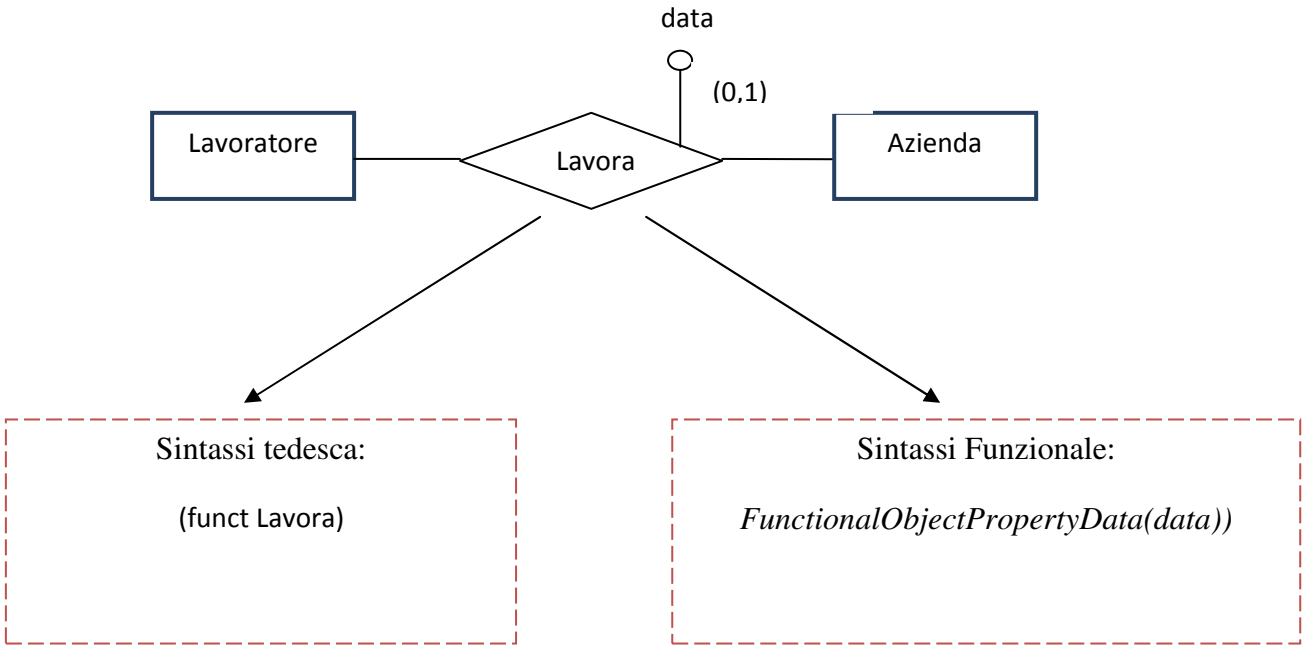
Cardinalità (1,n)

Diagramma ER:



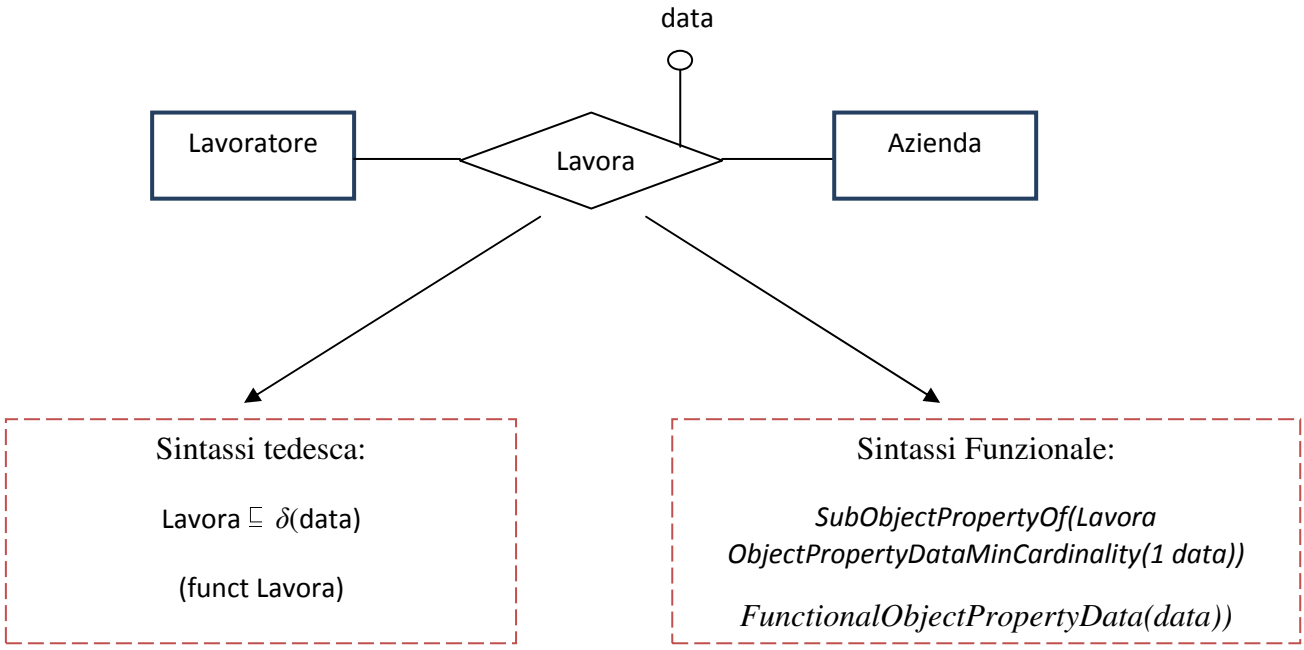
Cardinalità (0,1)

Diagramma ER:



Cardinalità (1,1)

Diagramma ER:



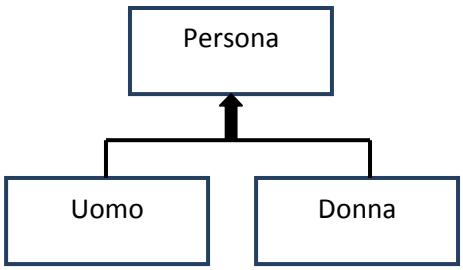
4.1Vincoli non esprimibili in DL-LiteA

Come detto in precedenza alcuni vincoli del diagramma concettuale non sono esprimibili direttamente in sintassi funzionale, per valutare comunque che i dati soddisfino tali vincoli, è possibile interrogare l'ontologia risultante mediante delle queries booleane espresse in SparSQL, che verificano l'integrità dell'ontologia rispetto a tali vincoli

Vincolo di completezza della generalizzazione

Per poter verificare che i concetti implicati nella generalizzazione rispettino il vincolo rispetto al concetto che estendono, si dovrà effettuare una query booleana che verifica che ogni istanza dell'entità padre corrisponda a una delle entità figlie e viceversa.

Diagramma ER:



Query booleana SparSQL

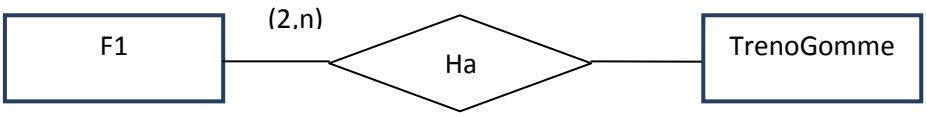
```
VERIFY not exists(
        SELECT per.x
        FROM sparqltable( SELECT ?x
                          WHERE
                         {
                          ?x rdf:type 'Persona'.
                         }
                         )per
WHERE per.x not in (
                         SELECT u.x
                         FROM sparqltable( SELECT ?x
                                           WHERE{
                                                 ?x rdf:type 'Uomo'.
                                                 }
                                          )u
                         UNION
```

```
        SELECT d.x
                FROM sparqltable( SELECT ?x
                                  WHERE{
                                        ?x rdf:type 'Donna'.
                                      }
                                  )d
            ))
)
```

Vincolo di cardinalità minima >1 di un attributo di concetto

In questo caso la query booleana conta le volte che ogni istanza del concetto partecipa alla relazione oggetto di verifica e ne verifica la coerenza rispetto alla cardinalità minima.

Diagramma ER



Query booleana SparSQL

```
VERIFY not exists(
SELECT f.x
FROM sparqltable( SELECT ?x ?w
                  WHERE{
                        ?x rdf:type 'F1'.
                        ?x :Ha ?w.
                        ?w rdf:type 'TrenoGomme'.
                        })f
GROUP BY (f.x)
HAVING COUNT(*) < 2
)
```

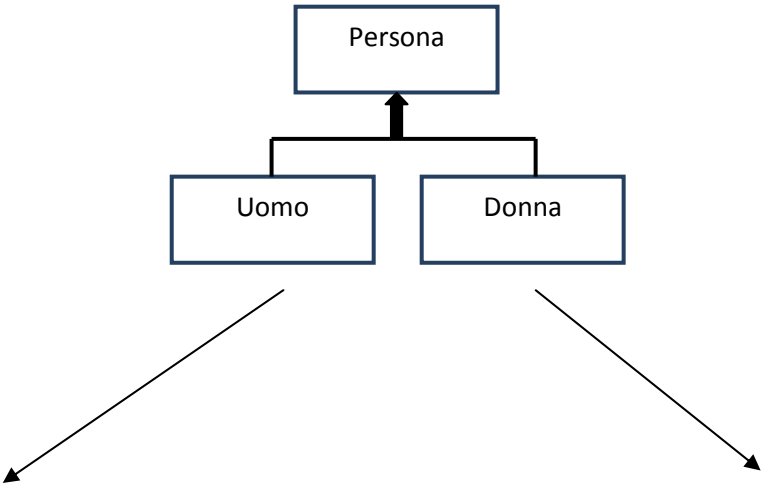
5 Dalla sintassi funzionale OWL alla sintassi OWL DL

Vediamo come avviene la traduzione dell'ontologia espressa in sintassi funzionale OWL in un ontologia espressa in linguaggio RDF/OWL.

Concetti-> Classes

(Definizione di concetti, generalizzazione tra concetti e disgiunzione)

Diagramma ER:



Sintassi RDF/XML

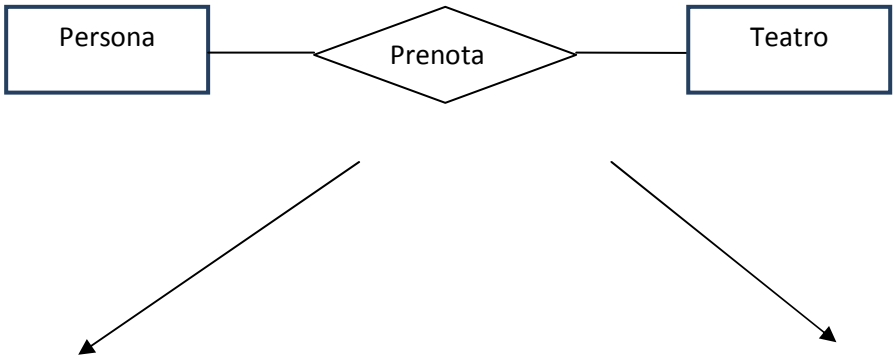
```
owl:Class rdf:ID="Uomo">
   <rdfs:subClassOf>
    <owl:Class rdf:ID="Persona"/>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
   </rdfs:subClassOf>
   <owl:disjointWith>
    <owl:Class rdf:ID="Donna"/>
   </owl:disjointWith>
  </owl:Class>
<owl:Class rdf:about="# Donna ">
   <rdfs:subClassOf rdf:resource="#
Persona "/>
   </rdfs:subClassOf>
```

Sintassi Funzionale:

SubClassOf(Uomo Persona)
SubClassOf(Donna Persona)
DisjointClasses(Uomo Donna)

Ruoli -> Object Properties

Un ruolo espresso in sintassi funzionale OWL viene tradotto in un ObjectProperties in sintassi OWL DL



Sintassi OWL DL:

<owl:ObjectProperty rdf:ID=" Prenota ">

<rdfs:range rdf:resource="# Teatro "/>

<rdfs:domain rdf:resource="# Persona "/>

</owl:ObjectProperty>

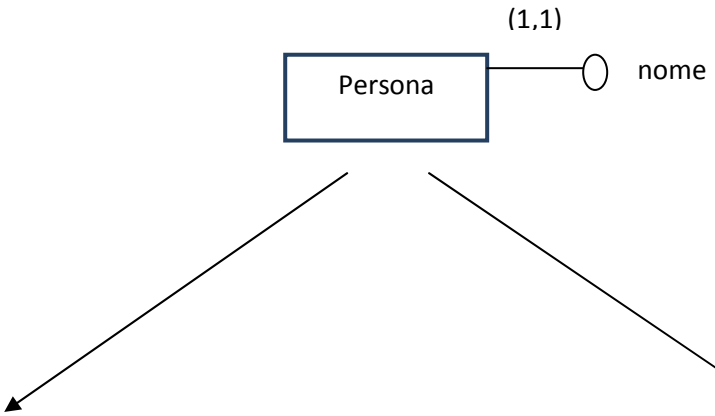
Sintassi Funzionale:

ObjectPropertyDomain(Prenota Persona)

ObjectPropertyRange(Prenota Teatro)

Attributi di concetto -> Datatype Properties

Per i datatype l'unica restrizione di cardinalità che puo' essere associata è di tipo functional.



Sintassi RDF/XML:

```
<rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty>
            <owl:FunctionalProperty
            rdf:ID="capSociale"/>
        </owl:onProperty>
        <owl:someValuesFrom
        rdf:resource="http://www.w3.org/1999/02/22-
        rdf-syntax-ns#XMLLiteral"/>
    </owl:Restriction>
<owl:FunctionalProperty rdf:about="#codFis">
        <rdf:type
        rdf:resource="http://www.w3.org/2002/07/
        owl#DatatypeProperty"/>
        <rdfs:domain rdf:resource="#Committente"/>
        <rdfs:range rdf:resource=
        "http://www.w3.org/2001/XMLSchema#string"
        />
</owl:FunctionalProperty>
```

Sintassi Funzionale:

DataPropertyRange(nome rdf:value)

DataPropertyDomain(nome Persona)

FunctionalDataProperty(nome)

SubClassOf(Persona dataMinCardinality(1 nome))

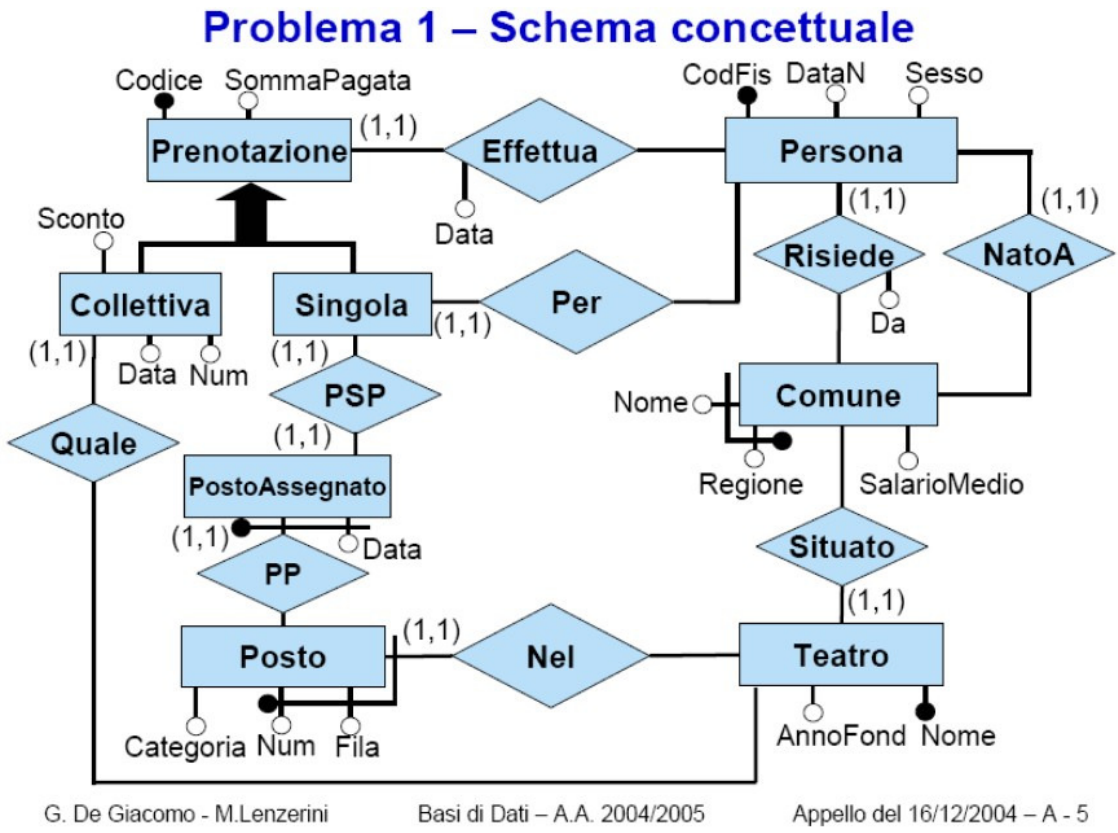
Attrubuti di ruolo in sintassi OWL XML/RDF

Non è possibile creare attributi di ruolo in quanto bisognerebbe creare una *datatype property* come *subproperty* di un *object property*, che non è esprimibile in OWL.

6 Casi di Studio

Traduzione compito A del 16/12/2004

Diagramma Er:



Traduzione dello schema in sintassi tedesca:

Generalizzazione e vincolo di disgiunzione:

Collettiva $\sqsubseteq$ Prenotazione

Singola $\sqsubseteq$ Prenotazione

Collettiva $\sqsubseteq \neg$ Singola

Definizione attributi di concetto:

Prenotazione $\sqsubseteq \delta$(codice Prenotazione)

δ(codice Prenotazione) $\sqsubseteq$ Prenotazione

ρ(codice Prenotazione) $\sqsubseteq$ xsd: integer

(funct codice Prenotazione)

Prenotazione $\sqsubseteq$ δ(sommaPagata)

δ(sommaPagata) $\sqsubseteq$ Prenotazione

ρ(sommaPagata) $\sqsubseteq$ xsd: integer

(funct sommaPagata)

Persona $\sqsubseteq$ δ(codFis)

δ(codFis) $\sqsubseteq$ Persona

ρ(codFis) $\sqsubseteq$ xsd:String

(funct codFis)

Persona $\sqsubseteq$ δ(dataN)

δ(dataN) $\sqsubseteq$ Persona

ρ(dataN) $\sqsubseteq$ xsd:String

(funct dataN)

Persona $\sqsubseteq$ δ(sesso)

δ(sesso) $\sqsubseteq$ Persona

ρ(sesso) $\sqsubseteq$ xsd:String

(funct sesso)

Comune $\sqsubseteq$ δ(nomeComune)

δ(nomeComune) $\sqsubseteq$ Comune

ρ(nomeComune) $\sqsubseteq$ xsd:String

(funct nomeComune)

Comune $\sqsubseteq$ δ(regione)

δ(regione) $\sqsubseteq$ Comune

ρ(regione) $\sqsubseteq$ xsd:String

(funct regione)

Comune $\sqsubseteq$ δ(salarioMedio)

δ(salarioMedio) $\sqsubseteq$ Comune

ρ(salarioMedio) $\sqsubseteq$ xsd:integer

(funct salarioMedio)

Teatro $\sqsubseteq$ δ(nomeTeatro)

δ(nomeTeatro) $\sqsubseteq$ Teatro

ρ(nomeTeatro) $\sqsubseteq$ xsd:String

(funct nomeTeatro)

Teatro $\sqsubseteq$ δ(annoFond)

δ(annoFond) $\sqsubseteq$ Teatro

ρ(annoFond) $\sqsubseteq$ xsd:int

(funct annoFond)

Posto $\sqsubseteq$ δ(num)

δ(num) $\sqsubseteq$ Posto

ρ(num) $\sqsubseteq$ xsd:integer

(funct num)

Posto $\sqsubseteq$ δ(fila)

δ(fila) $\sqsubseteq$ Posto

ρ(fila) $\sqsubseteq$ xsd:integer

(funct fila)

Posto $\sqsubseteq$ δ(categoria)

δ(categoria) $\sqsubseteq$ Posto

ρ(categoria) $\sqsubseteq$ xsd:String

(funct categoria)

PostoAssegnato $\sqsubseteq$ δ(datoPosto)

δ(datoPosto) $\sqsubseteq$ PostoAssegnato

ρ(datoPosto) $\sqsubseteq$ xsd:date

(funct datoPosto)

Collettiva $\sqsubseteq$ δ(sconto)

δ(sconto) $\sqsubseteq$ Collettiva

ρ(sconto) $\sqsubseteq$ xsd:integer

(funct sconto)

Collettiva $\sqsubseteq$ δ(dataCollettiva)

δ(dataCollettiva) $\sqsubseteq$ Collettiva

ρ(dataCollettiva) $\sqsubseteq$ xsd:date

(funct dataCollettiva)

Collettiva $\sqsubseteq$ δ(numCollettiva)

δ(numCollettiva) $\sqsubseteq$ Collettiva

ρ(numCollettiva) $\sqsubseteq$ xsd:integer

funct(numCollettiva)

Definizione dei ruoli e cardinalità

∃ effettua ⊑ Prenotazione

∃effettua ⁻ ⊑ Persona

Prenotazione ⊑ ∃effettua

(funct effettua)

∃ risiede ⊑ Persona

∃ risiede ⁻ ⊑ Comune

Persona ⊑ ∃ risiede

(funct risiede)

∃ natoA ⊑ Persona

∃ natoA ⁻ ⊑ Comune

Persona ⊑ ∃ natoA

(funct natoA)

∃ situato ⊑ Comune

∃ situato⁻ ⊑ Teatro

∃ situato⁻ ⊑ Comune

(funct situato⁻)

∃ nel ⊑ Posto

∃ nel ⁻⊑ Teatro

Posto ⊑ ∃ nel

(funct nel)

$\exists$ $PP \sqsubseteq$ PostoAssegnato

$\exists$ $PP^{-} \sqsubseteq$ Posto

PostoAssegnato $\sqsubseteq \exists$ PP

(funct PP)

$\exists$ $PSP \sqsubseteq$ Singola

$\exists$ $PSP^{-} \sqsubseteq$ Posto Assegnato

Singola $\sqsubseteq \exists$ PSP

(funct PSP)

$\exists$ $PSP^{-} \sqsubseteq$ Singola

(funct PSP^{-})

$\exists$ per $\sqsubseteq$ Singola

$\exists$ $per^{-} \sqsubseteq$ Persona

Singola $\sqsubseteq$ $\exists$ per

(funct per)

$\exists$ quale $\sqsubseteq$ Collettiva

$\exists$ $quale^{-} \sqsubseteq$ Teatro

Collettiva $\sqsubseteq \exists$ quale

(funct quale)

Definizione attributi di ruolo:

Effettua$\sqsubseteq$ δ(data)

δ(data) $\sqsubseteq$ Effettua

ρ(data) $\sqsubseteq$ xsd:date

(funct data)

Traduzione dello schema in sintassi funzionale

Generalizzazione e vincolo di disgiunzione:

SubClassOf(Collettiva Prenotazione)

SubClassOf(Singola Prenotazione)

DisjointClasses(Collettiva Singola)

Definizione dei ruoli

ObjectPropertyDomain(effettua Persona)

ObjectPropertyRange(effettua Prenotazione)

ObjectPropertyDomain(risiede Persona)

ObjectPropertyRange(risiede Comune)

ObjectPropertyDomain(natoA Persona)

ObjectPropertyRange(natoA Comune)

ObjectPropertyDomain(situato Teatro)

ObjectPropertyRange(situato Comune)

ObjectPropertyDomain(nel Posto)

ObjectPropertyRange(nel Teatro)

ObjectPropertyDomain(PP PostoAssegnato)

ObjectPropertyRange(PP Posto)

ObjectPropertyDomain(PSP Singola)

ObjectPropertyRange(PSP PostoAssegnato)

ObjectPropertyDomain(per Singola)

ObjectPropertyRange(per Persona)

ObjectPropertyDomain(quale Collettiva)

ObjectPropertyRange(quale Teatro)

Definizione delle cardinalità delle relazioni

SubClassOf(Persona ObjectMinCardinality(1 natoA))

SubClassOf(Persona ObjectMinCardinality(1 risiede))

SubClassOf(Teatro ObjectMinCardinality(1 situato))

SubClassOf(Persona ObjectMinCardinality(1 per))

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(effettua)) Prenotazione)

SubClassOf(Posto ObjectMinCardinality(1 nel))

SubClassOf(Collettiva ObjectMinCardinality(1 quale))

SubClassOf(Singola ObjectMinCardinality(1 PSP))

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(PSP)) PostoAssegnato)

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(effettua)) Prenotazione)

SubClassOf(PostoAssegnato ObjectMinCardinality(1 PP))

FunctionalObjectProperty(per)

FunctionalObjectProperty(situato)

FunctionalObjectProperty(PP)

FunctionalObjectProperty(PSP)

FunctionalObjectProperty(InverseObjectPropertyOf(PSP))

FunctionalObjectPropertyData(dataAnno)

Definizione del dominio degli attributi di concetto

DataPropertyDomain(dataPosto PostoAssegnato)

DataPropertyDomain(codicePrenotazione Prenotazione)

DataPropertyDomain(sommaPagata Prenotazione)

DataPropertyDomain(numCollettiva Collettiva)

DataPropertyDomain(dataCollettiva Collettiva)

DataPropertyDomain(sconto Collettiva)

DataPropertyDomain(codFis Persona)

DataPropertyDomain(dataN Persona)

DataPropertyDomain(sesso Persona)

DataPropertyDomain(nomeComune Comune)

DataPropertyDomain(regione Comune)

DataPropertyDomain(salarioMedio Comune)

DataPropertyDomain(nomeTeatro Teatro)

DataPropertyDomain(annoFond Teatro)

DataPropertyDomain(fila Posto)

DataPropertyDomain(num Posto)

DataPropertyDomain(categoria Posto)

Definizione del range degli attributi di concetto

DataPropertyRange(nomeComune rdf:string)

DataPropertyRange(salarioMedio rdf:integer)

DataPropertyRange(regione rdf:string)

DataPropertyRange(dataCollettiva rdf:date)

DataPropertyRange(sconto rdf:integer)

DataPropertyRange(sommaPagata rdf:integer)

DataPropertyRange(codicePrenotazione rdf:string)

DataPropertyRange(numCollettiva rdf:integer)

DataPropertyRange(codFis rdf:string)

DataPropertyRange(annoFond rdf:date)

DataPropertyRange(nomeTeatro rdf:string)

DataPropertyRange(categoria rdf:string)

DataPropertyRange(num rdf:integer)

DataPropertyRange(fila rdf:string)

DataPropertyRange(dataPosto rdf:date)

Definizione di cardinalità minima e massima (1,1) degli attributi di concetto

SubClassOf(Comune DataSomeValueFrom(salarioMedio xsd:anyType))

SubClassOf(Posto DataSomeValueFrom(categoria xsd:anyType))

SubClassOf(Posto DataSomeValueFrom(num xsd:anyType))

SubClassOf(Posto DataSomeValueFrom(fila xsd:anyType))

SubClassOf(PostoAssegnato DataSomeValueFrom(dataPosto xsd:anyType))

SubClassOf(Collettiva DataSomeValueFrom(dataCollettiva xsd:anyType))

SubClassOf(Collettiva DataSomeValueFrom(sconto xsd:anyType))

SubClassOf(Collettiva DataSomeValueFrom(numCollettiva xsd:anyType))

SubClassOf(Prenotazione DataSomeValueFrom(sommaPagata xsd:anyType))

SubClassOf(Prenotazione DataSomeValueFrom(codicePrenotazione xsd:anyType))

SubClassOf(Persona DataSomeValueFrom(sesso xsd:anyType))

SubClassOf(Persona DataSomeValueFrom(codFis xsd:anyType))

SubClassOf(Persona DataSomeValueFrom(dataN xsd:anyType))

SubClassOf(Comune DataSomeValueFrom(nomeComune xsd:anyType))

SubClassOf(Comune DataSomeValueFrom(regione xsd:anyType))

SubClassOf(Teatro DataSomeValueFrom(annoFond xsd:anyType))

SubClassOf(Teatro DataSomeValueFrom(nomeTeatro xsd:anyType))

FunctionalDataProperty(codFis)

FunctionalDataProperty(sesso)

FunctionalDataProperty(dataN)

FunctionalDataProperty(nomeComune)

FunctionalDataProperty(regione)

FunctionalDataProperty(salarioMedio)

FunctionalDataProperty(nomeTeatro)

FunctionalDataProperty(annoFond)

FunctionalDataProperty(fila)

FunctionalDataProperty(num)

FunctionalDataProperty(categoria)

FunctionalDataProperty(dataPosto)

FunctionalDataProperty(codicePrenotazione)

FunctionalDataProperty(sommaPagata)

FunctionalDataProperty(sconto)

FunctionalDataProperty(dataCollettiva)

FunctionalDataProperty(numCollettiva)

Definizione del dominio degli attributi di ruolo

ObjectPropertyDataDomain(dataAnno effettua)

ObjectPropertyDataDomain(dataMese effettua)

ObjectPropertyDataDomain(dataGiorno effettua)

ObjectPropertyDataDomain(da risiede)

Definizione del range degli attributi di ruolo

ObjectPropertyDataRange(dataAnno rdf:integer)

ObjectPropertyDataRange(dataMese rdf:integer)

ObjectPropertyDataRange(dataGiorno rdf:integer)

ObjectPropertyDataRange(da rdf:string)

Definizione di cardinalità minima degli attributi di ruolo

SubObjectPropertyOf(effettua ObjectPropertyDataSomeValueFrom(dataAnno xsd:anyType))

SubObjectPropertyOf(effettua ObjectPropertyDataSomeValueFrom(dataMese xsd:anyType))

SubObjectPropertyOf(effettua ObjectPropertyDataSomeValueFrom(dataGiorno xsd:anyType))

SubObjectPropertyOf(risiede ObjectPropertyDataSomeValueFrom(da xsd:anyType))

Definizione di cardinalità massima degli attributi di ruolo

FunctionalObjectPropertyData(dataGiorno)

FunctionalObjectPropertyData(dataMese)

FunctionalObjectPropertyData(dataAnno)

FunctionalObjectPropertyData(da)

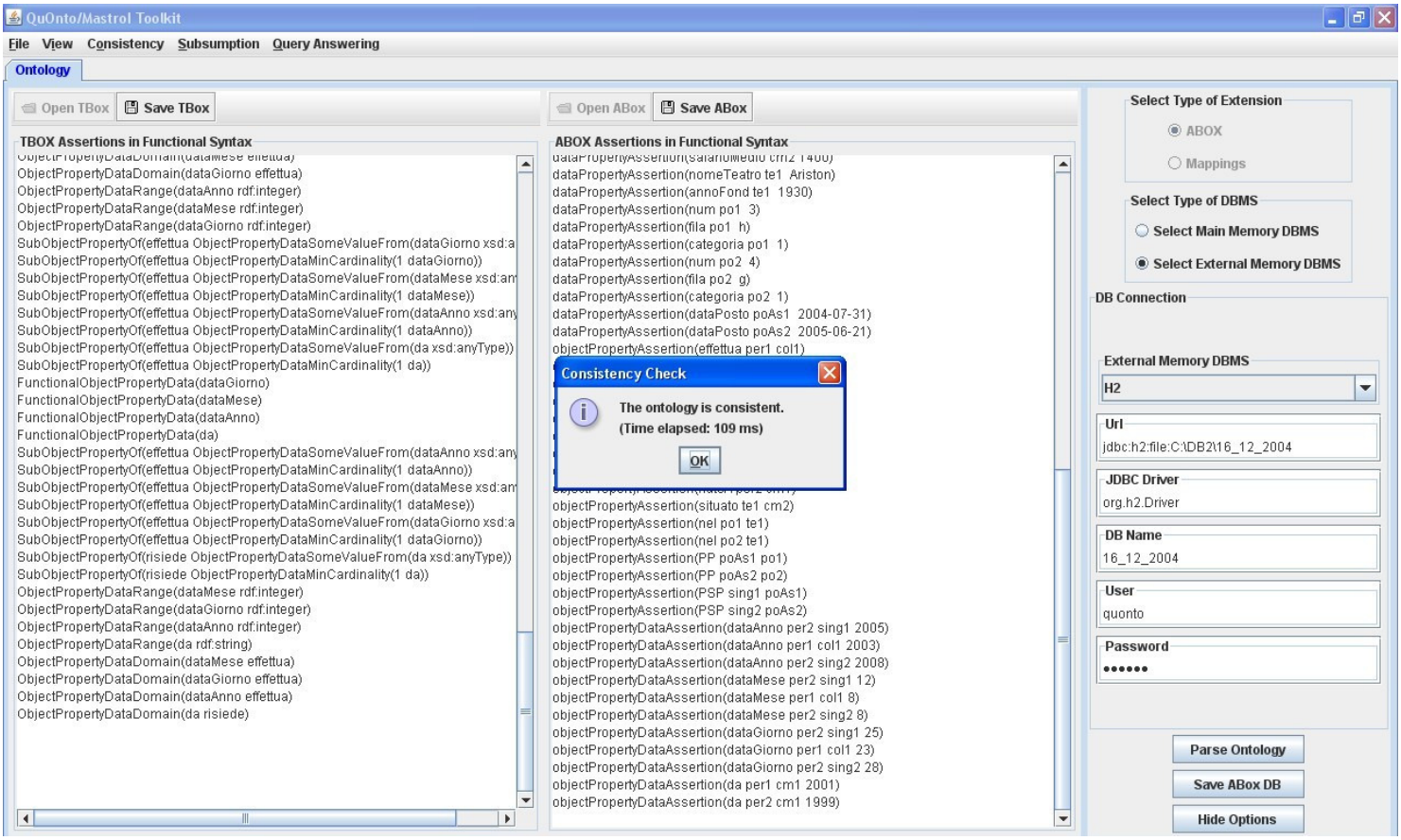
Il tool QuOnto per interrogare l'ontología

Mediante il tool QuOnto/ Mastro al quale sono state fornite in input la TBox e l'ABox espresse in sintassi funzionale OWL è stato possibile

- Effettuare il controllo di consistenza dell'ontología.
- Effettuare il query answering sull'ontología
- Valutare utilizzando query booleane espresse in SparSQL i vincoli di integrità non esprimibili in sintassi funzionale.
- Generare un database H2, che sarà utilizzato successivamente dal tool Protege 3.3.1 per effettuare i mapping.

Controllo di consistenza sull'ontología

Una volta importata l'ABox e la TBox con il tool è possibile effettuare il controllo di consistenza sull'ontología:



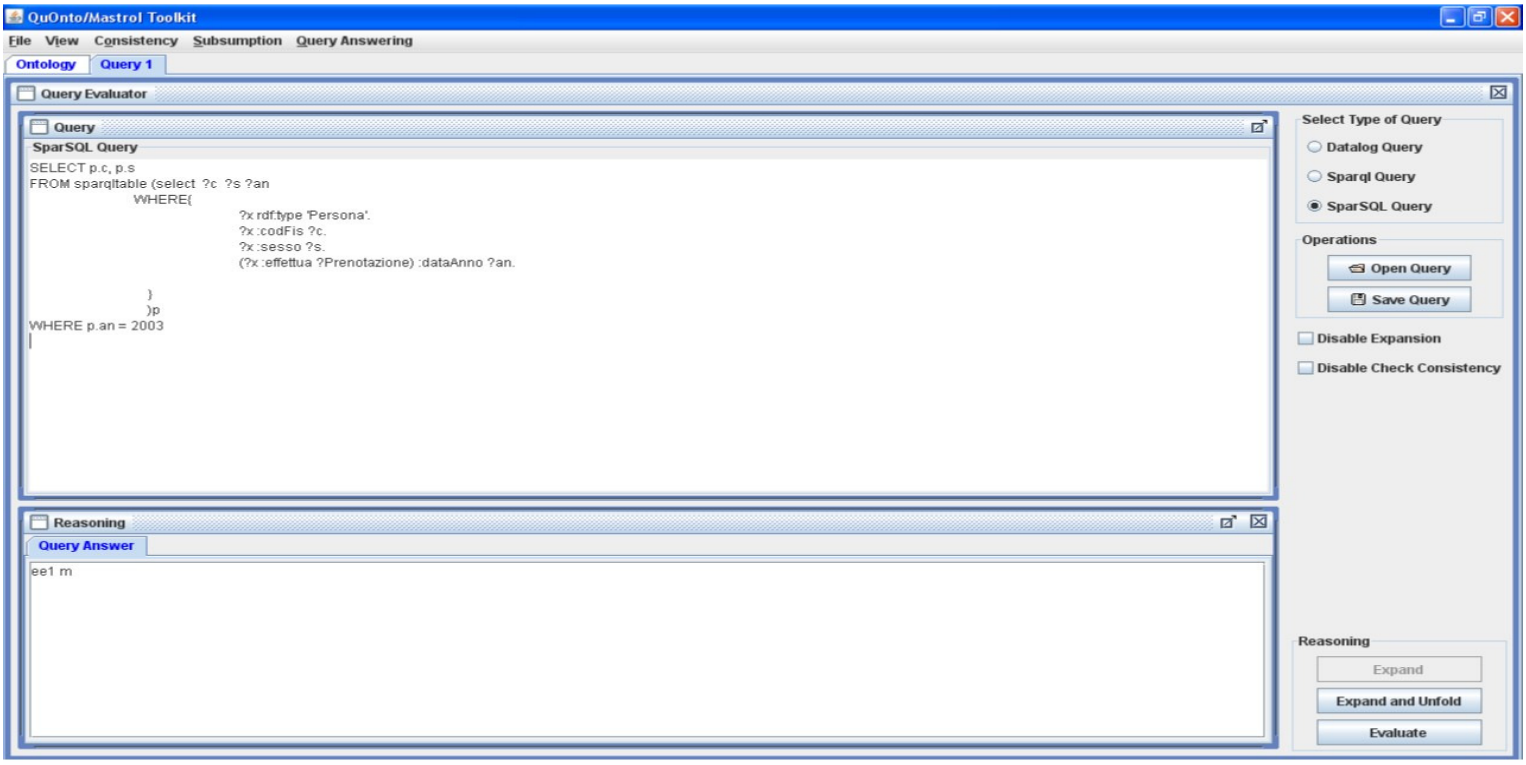
Query answering sull'ontología

Query n°1:

Calcolare il codice fiscale ed il sesso delle persone che hanno effettuato almeno una prenotazione nel 2003.

```
SELECT p.c, p.s
FROM sparqltable (select  ?c  ?s ?an
        WHERE{
                ?x rdf:type 'Persona'.
                ?x :codFis ?c.
                ?x :sesso ?s.
                (?x :effettua ?Prenotazione) :dataAnno ?an.

                    }
                    )p
WHERE p.an = 2003
```

Screenshot Query 1:



Query n° 2:

Calcolare il codice fiscale e la data di nascita delle persone che hanno effettuato almeno una prenotazione collettiva per una data di agosto e per la quale hanno pagato una somma di almeno 200 Euro.

SELECT p.c, p.d

FROM sparqltable (select ?mese ?c ?d ?sp

WHERE{

?x rdf:type 'Persona'.

?x :codFis ?c.

?x :dataN ?d.

?z rdf:type 'Collettiva'.

(?x :effettua ?z) :dataMese ?mese.

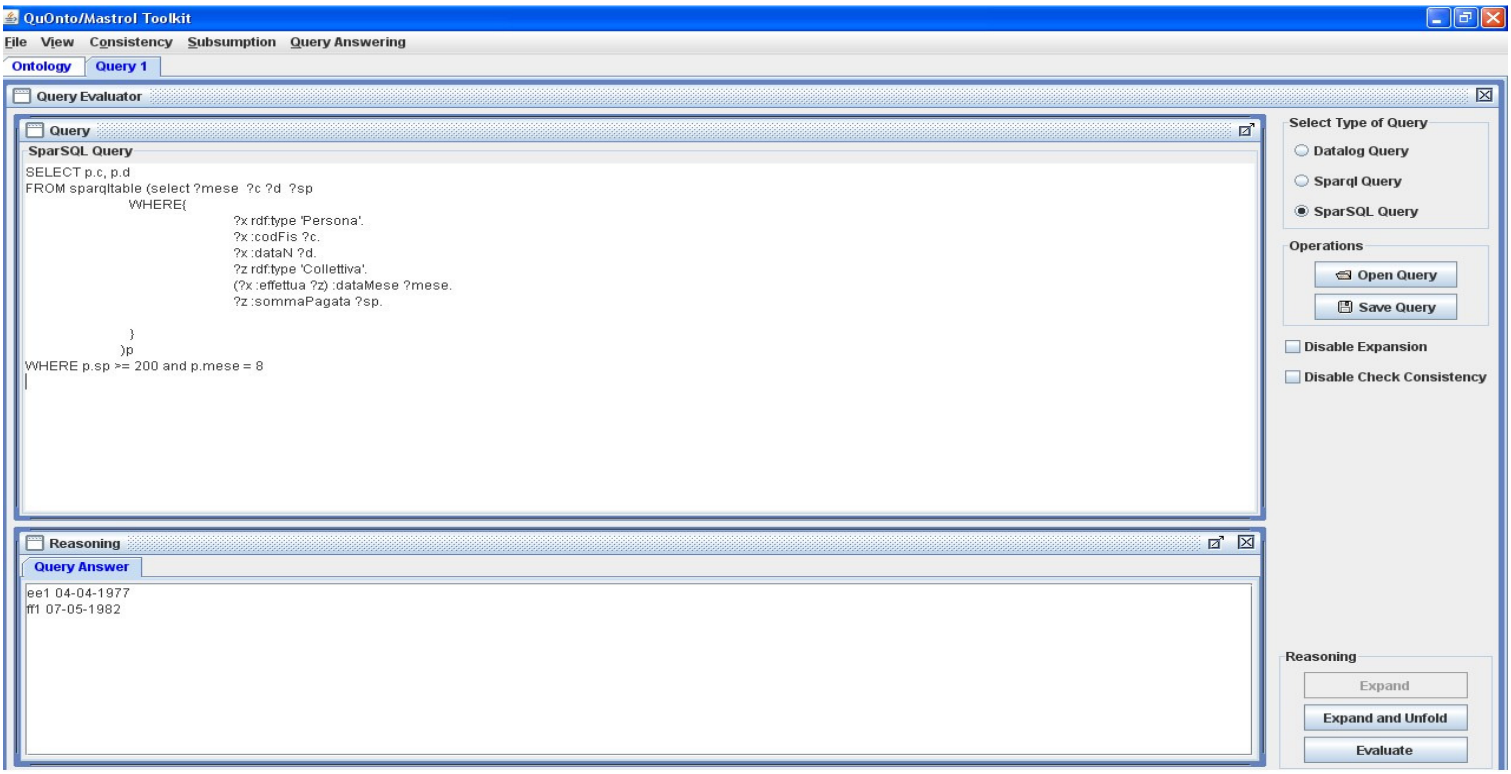
?z :sommaPagata ?sp.

}

)p

WHERE p.sp >= 200 and p.mese = 8

Screenshot Query 2:



Query n° 3:

Calcolare il comune di residenza e la data di nascita delle donne che hanno effettuato almeno una prenotazione singola per un teatro situato in un comune diverso da quello in cui risiedono.

SELECT p.s,p.comRe,p.regRe

FROM sparqltable (select ?s ?c ?noTe ?comTe ?regTe ?comRe ?regRe

WHERE{

?x rdf:type 'Persona'.

?x :sesso?s.

?x :risiede?c.

?c :nomeComune?comRe.

?c :regione?regRe.

?x :effettua?y.

?y rdf:type 'Singola'.

?y :PSP?z.

?z rdf:type 'PostoAssegnato'.

?z :PP?a.

?a rdf:type 'Posto'.

?a :nel?t.

?t rdf:type 'Teatro'.

?t :nomeTeatro?noTe.

?t :situato?com.

?com rdf:type 'Comune'.

?com :nomeComune?comTe.

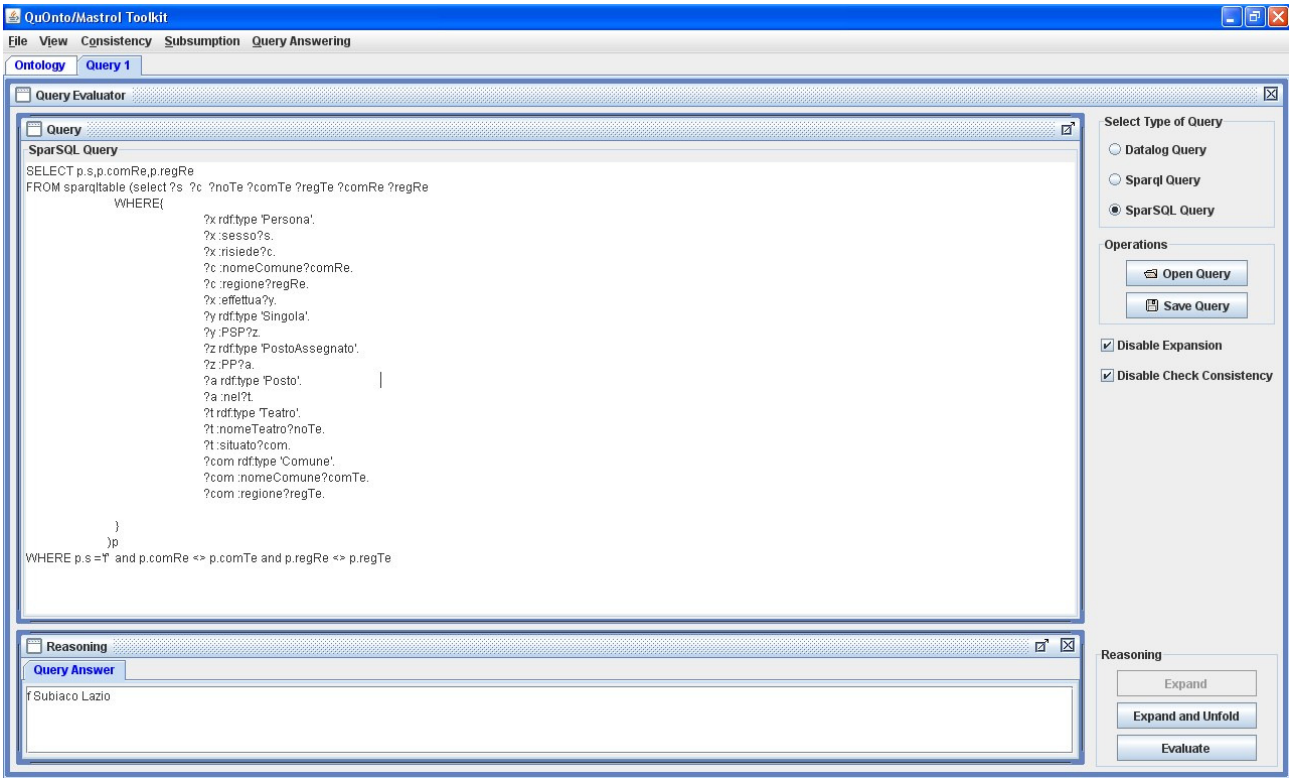
?com :regione?regTe.

}

)p

WHERE p.s ='f' and p.comRe <> p.comTe and p.regRe <> p.regTe

Screenshot Query 3



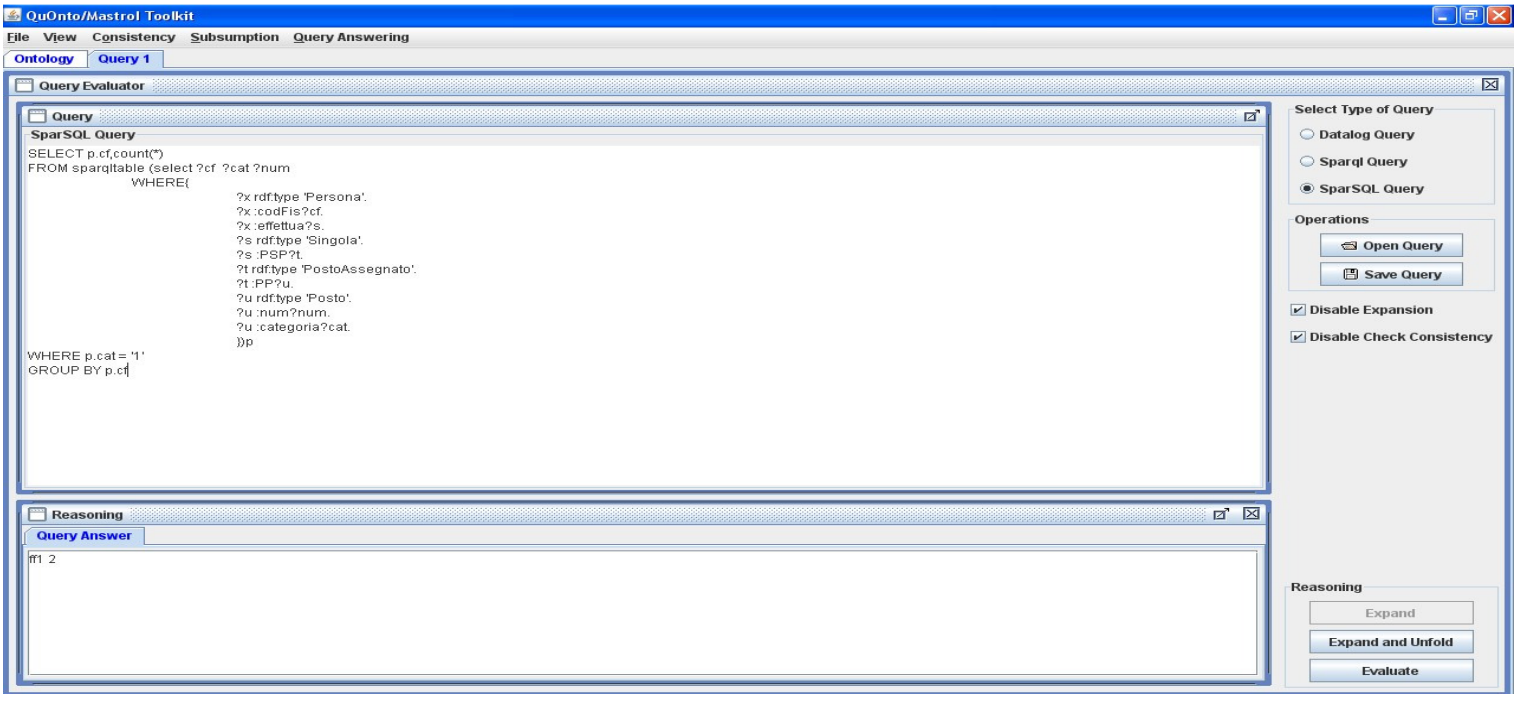
:

Query n° 4:

Per ogni persona che ha effettuato almeno una prenotazione singola per la quale è stato assegnato un posto di categoria1, contare il numero di prenotazione(singole o collettive) effettuate.

```
SELECT p.cf,count(*)
FROM sparqltable (select ?cf  ?cat ?num
        WHERE{
                ?x rdf:type 'Persona'.
                ?x :codFis?cf.
                ?x :effettua?s.
                ?s rdf:type 'Singola'.
                ?s :PSP?t.
                ?t rdf:type 'PostoAssegnato'.
                ?t :PP?u.
                ?u rdf:type 'Posto'.
                ?u :num?num.
                ?u :categoria?cat.
                })p
    WHERE p.cat = '1'
GROUP BY p.c
```

Screenshot Query 4



Query booleana per verificare il vincolo di completezza della generalizzazione

Per verificare il vincolo di completezza della generalizzazione presente nel diagramma ER, dato che la sintassi funzionale non ne permette la definizione, si utilizza una query booleana SparSQL che permette di verificare la consistenza dell'ontología rispetto a tale vincolo.

```
VERIFY not exists(
        SELECT prenotazione.x
        FROM sparqltable( SELECT ?x
                          WHERE
                         {
                          ?x rdf:type 'Prenotazione'.
                         }
                         )prenotazione
WHERE prenotazione.x not in (
                         SELECT collettiva.x
                         FROM sparqltable( SELECT ?x
                                           WHERE{
                                                 ?x rdf:type 'Collettiva'.
                                                 }
                                          )collettiva
                         UNION
                         SELECT singola.x
                         FROM sparqltable( SELECT ?x
                                           WHERE{
                                                 ?x rdf:type 'Singola'.
                                                }
                                          )singola
                    ))
)
```

Generazione dell'ontología owl

Per la generazione del file .owl si è utilizzato Protege 4.0 che permette di importare un' ontología scritta in sintassi funzionale restituendo in output l'ontología in formato RDF/XML. A questo punto si è utilizzato Protege 3.3.1 ed i plugin OBDA per generare i mapping (per collegare le Casses, gli Object Properties e le Data Properties ai dati).

Generazione dei mapping:

Mapping di concetti (es. concetto *Persona*) :

MappingPersona
Persona(funct($term))
SELECT term FROM Persona

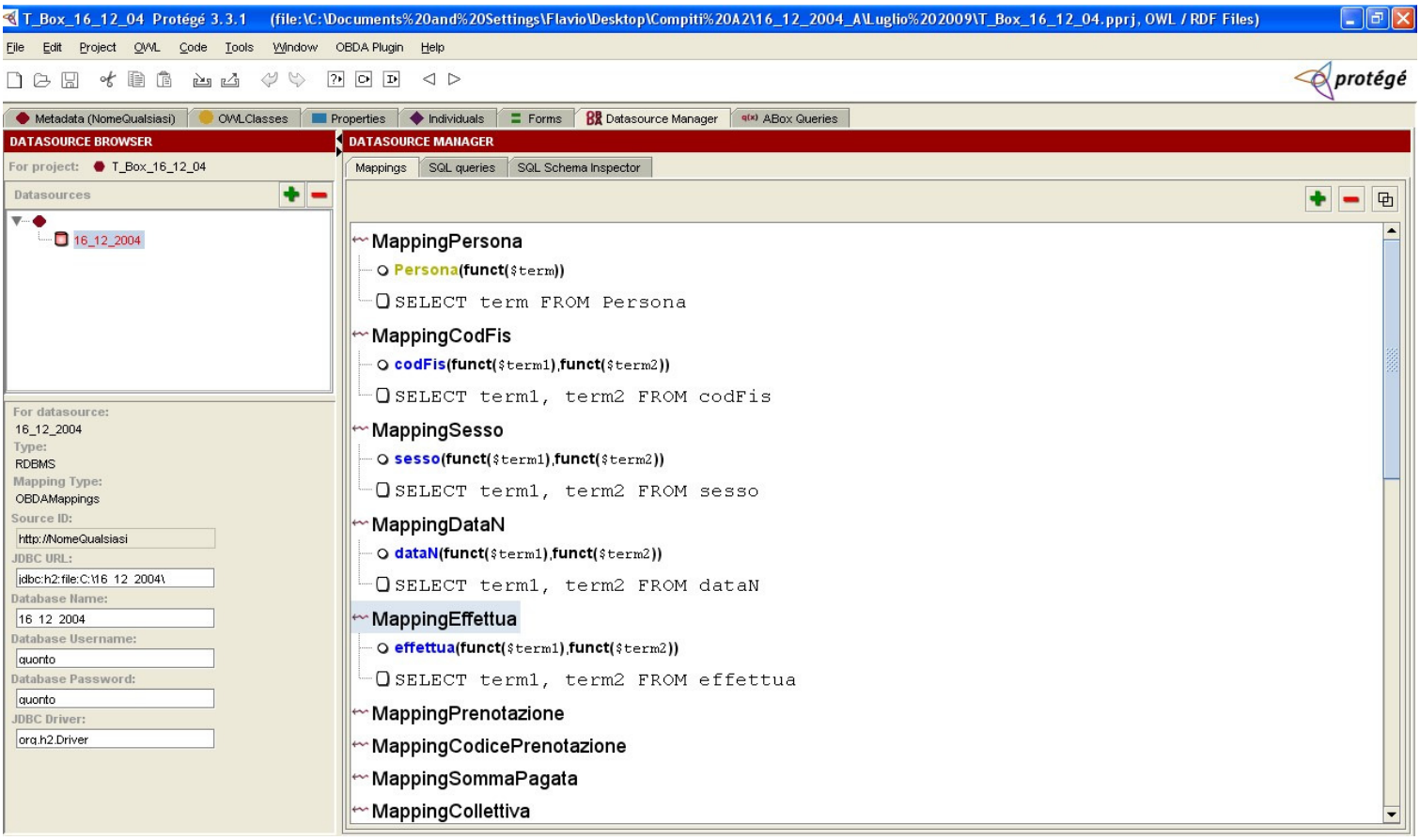
Mapping di attributi concetto (es. attributo di concetto *codFis*) :

MappingCodFis
codFis(funct($term1),funct($term2))
SELECT term1, term2 FROM codFis

Mapping di ruolo (es. ruolo *Effettua*) :

MappingEffettua
effettua(funct($term1),funct($term2))
SELECT term1, term2 FROM effettua

Screenshot Mapping



Query answering mediante Protégé

Query n°1:

Calcolare il codice fiscale ed il sesso delle persone che hanno effettuato almeno una prenotazione nel 2003.

Non è stato possibile esprimere il vincolo "almeno una prenotazione" in quanto non ci sono operatori che lo permettono, inoltre non è possibile estrarre gli attributi "codice Fiscale" e "sesso" dalla classe Persona.

Query n° 2:

Calcolare il codice fiscale e la data di nascita delle persone che hanno effettuato almeno una prenotazione collettiva per una data di agosto e per la quale hanno pagato una somma di almeno 200 Euro.

Non è stato possibile esprimere il vincolo "almeno una prenotazione" in quanto non ci sono operatori che lo permettono, inoltre non è possibile estrarre gli attributi "codice Fiscale" e "data di nascita" dalla classe Persona e "una somma di almeno 200 euro".

Query n° 3:

Calcolare il comune di residenza e la data di nascita delle donne che hanno effettuato almeno una prenotazione singola per un teatro situato in un comune diverso da quello in cui risiedono.

Per esprimere la seguente query sarebbe stato necessario inserire un ciclo, dato che in OWL non ci sono variabili, si possono solo esprimere queries ad albero.

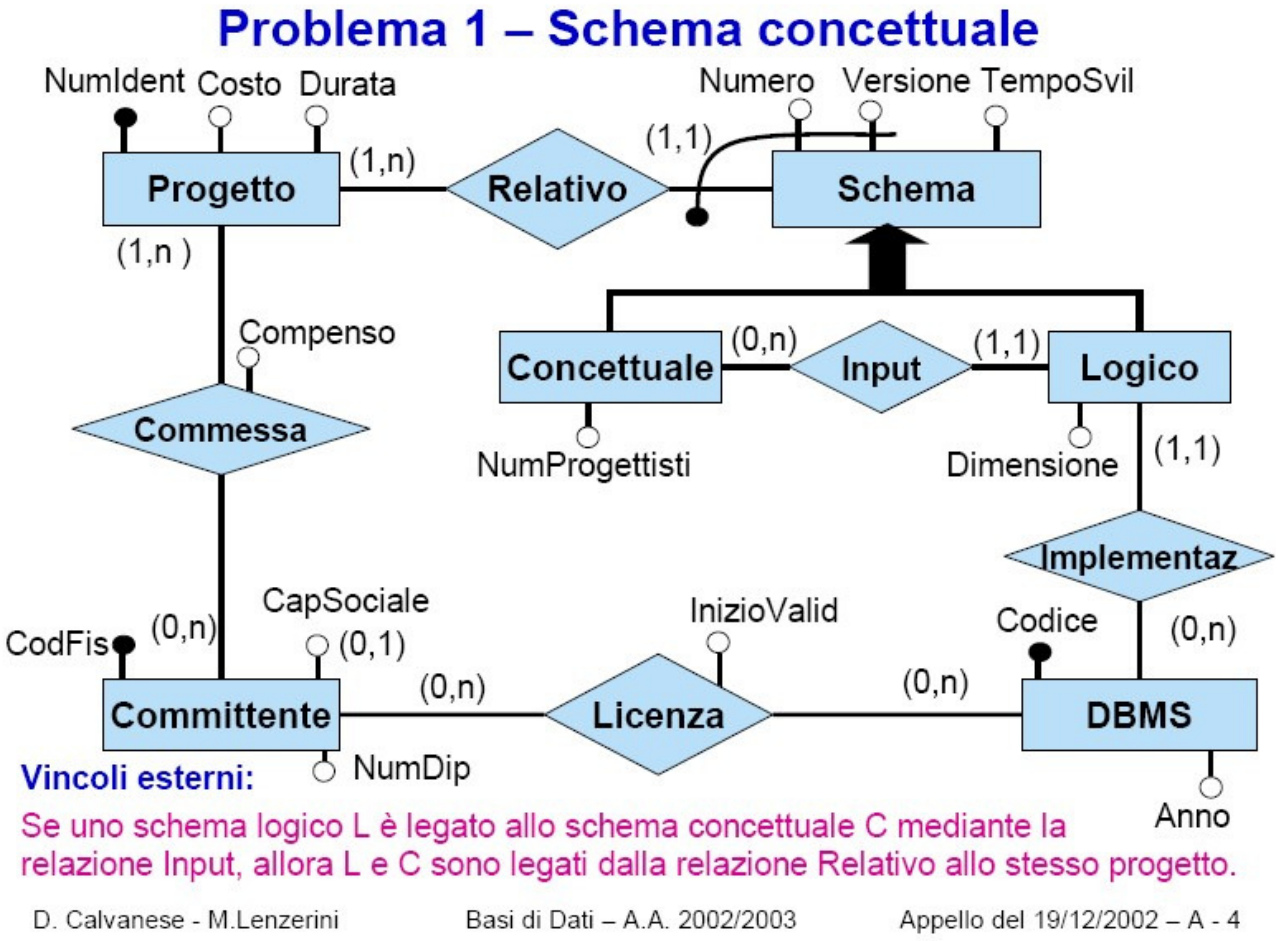
Query n° 4:

Per ogni persona che ha effettuato almeno una prenotazione singola per la quale è stato assegnato un posto di categoria1, contare il numero di prenotazioni(singole o collettive) effettuate.

Non è possibile esprimere il vincolo "almeno una prenotazione singola" e non è possibile "contare il numero di prenotazioni) in quanto non ci sono operatori che lo permettono

Traduzione compito A del 19/12/2002

Diagramma Er:



Traduzione dello schema in sintassi tedesca:

Generalizzazione e vincolo di disgiunzione:

Concettuale $\sqsubseteq$ Schema

Logico $\sqsubseteq$ Schema

Logico $\sqsubseteq \neg$ Concettuale

Definizione attributi di concetto:

Progetto $\sqsubseteq$ δ(numIdent)

δ(numIdent) $\sqsubseteq$ Progetto

ρ(numIdent) $\sqsubseteq$ xsd: String

(funct numIdent)

Progetto $\sqsubseteq$ δ(costo)

δ(costo) $\sqsubseteq$ Progetto

ρ(costo) $\sqsubseteq$ xsd: integer

(funct costo)

Progetto $\sqsubseteq$ δ(durata)

δ(durata) $\sqsubseteq$ Progetto

ρ(durata) $\sqsubseteq$ xsd: integer

(funct durata)

Schema $\sqsubseteq$ δ(numero)

δ(numero) $\sqsubseteq$ Schema

ρ(numero) $\sqsubseteq$ xsd: integer

(funct numero)

Schema $\sqsubseteq$ δ(versione)

δ(versione) $\sqsubseteq$ Schema

ρ(versione) $\sqsubseteq$ xsd: String

(funct versione)

Schema $\sqsubseteq$ δ(tempoSvil)

δ(tempoSvil) $\sqsubseteq$ Schema

ρ(tempoSvil) $\sqsubseteq$ xsd: integer

(funct tempoSvil)

Concettuale $\sqsubseteq$ δ(numProggettisti)

δ(numProggettisti) $\sqsubseteq$ Concettuale

ρ(numProggettisti) $\sqsubseteq$ xsd: integer

funct(numProggettisti)

Logico$\sqsubseteq$ δ(dimensione)

δ(dimensione) $\sqsubseteq$ Logico

ρ(dimensione) $\sqsubseteq$ xsd: integer

(funct dimensione)

DBMS $\sqsubseteq$ δ(codice)

δ(codice) $\sqsubseteq$ DBMS

ρ(codice) $\sqsubseteq$ xsd:String

(funct codice)

DBMS $\sqsubseteq$ δ(anno)

δ(anno) $\sqsubseteq$ DBMS

ρ(anno) $\sqsubseteq$ xsd:date

(funct anno)

Committente$\sqsubseteq$ δ(codFis)

δ(codFis) $\sqsubseteq$ Committente

$\rho(\text{codFis}) \sqsubseteq$ xsd: String

(funct codFis)

$\delta(\text{capSociale}) \sqsubseteq$ Committente

$\rho(\text{capSociale}) \sqsubseteq$ xsd: integer

(funct capSociale)

Committente$\sqsubseteq \delta(\text{numDip})$

$\delta(\text{numDip}) \sqsubseteq$ Committente

$\rho(\text{numDip}) \sqsubseteq$ xsd: String

(funct numDip)

Definizione dei ruoli e cardinalità

$\exists$ relativo $\sqsubseteq$ Schema

$\exists$ relativo $^{-} \sqsubseteq$ Progetto

Schema $\sqsubseteq \exists$ relativo

(funct relativo)

$\exists$ input $\sqsubseteq$ Concettuale

$\exists$ input $^{-} \sqsubseteq$ Logico

Logico $\sqsubseteq \exists$ input $^{-}$

(funct input^{-})

$\exists$ implementaz $\sqsubseteq$ Logico

$\exists$implementaz $^{-} \sqsubseteq$ DBMS

Logico $\sqsubseteq \exists$ implementaz

(funct implementaz)

$\exists$ licenza $\sqsubseteq$ DBMS

$\exists$licenza $^{-} \sqsubseteq$ Committente

$\exists$ commessa $\sqsubseteq$ Committente

$\exists$commessa $^{-} \sqsubseteq$ Progetto

$\exists$commessa$^{-} \sqsubseteq$ Committente

Definizione attributi di ruolo:

commessa $\sqsubseteq \delta(\text{compenso})$

$\delta(\text{compenso}) \sqsubseteq$ commessa

$\rho(\text{compenso}) \sqsubseteq$ xsd: integer

(funct compenso)

Licenza $\sqsubseteq \delta(\text{inizioValid})$

$\delta(\text{inizioValid}) \sqsubseteq$ Licenza

$\rho(\text{inizioValid}) \sqsubseteq$ xsd: date

(funct inizioValid)

Traduzione dello schema in sintassi funzionale

Generalizzazione e vincolo di disgiunzione:

SubClassOf(Concettuale Schema)

SubClassOf(Logico Schema)

DisjointClasses(Logico Concettuale)

Definizione dei ruoli

ObjectPropertyDomain(relativo Schema)

ObjectPropertyRange(relativo Progetto)

ObjectPropertyDomain(input Concettuale)

ObjectPropertyRange(input Logico)

ObjectPropertyDomain(implementaz Logico)

ObjectPropertyRange(implementaz DBMS)

ObjectPropertyDomain(licenza DBMS)

ObjectPropertyRange(licenza Committente)

ObjectPropertyDomain(commessa Committente)

ObjectPropertyRange(commessa Progetto)

Definizione delle cardinalità delle relazioni

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(relativo)) Progetto)

SubClassOf(Schema ObjectMinCardinality(1 relativo))

SubClassOf(Logico ObjectMinCardinality(1 implementaz))

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(input)) Logico)

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(commessa)) Progetto)

FunctionalObjectProperty(relativo)

FunctionalObjectProperty(implementaz)

FunctionalObjectProperty(InverseObjectPropertyOf(input))

Definizione del dominio degli attributi di concetto

DataPropertyDomain(numDip Committente)

DataPropertyDomain(capSociale Committente)

DataPropertyDomain(codFis Committente)

DataPropertyDomain(anno DBMS)

DataPropertyDomain(codice DBMS)

DataPropertyDomain(dimensione Logico)

DataPropertyDomain(numProgettisti Concettuale)

DataPropertyDomain(tempoSvil Schema)

DataPropertyDomain(numero Schema)

DataPropertyDomain(versione Schema)

DataPropertyDomain(durata Progetto)

DataPropertyDomain(costo Progetto)

DataPropertyDomain(numIdent Progetto)

Definizione del range degli attributi di concetto

DataPropertyRange(numIdent rdf:string)

DataPropertyRange(costo rdf:string)

DataPropertyRange(durata rdf:integer)

DataPropertyRange(versione rdf:string)

DataPropertyRange(tempoSvil rdf:integer)

DataPropertyRange(numero rdf:integer)

DataPropertyRange(numProgettisti rdf:integer)

DataPropertyRange(dimensione rdf:integer)

DataPropertyRange(codice rdf:string)

DataPropertyRange(anno rdf:integer)

DataPropertyRange(codFis rdf:string)

DataPropertyRange(numDip rdf:integer)

DataPropertyRange(capSociale rdf:integer)

Definizione di cardinalità minima e massima (1,1) degli attributi di concetto

SubClassOf(Progetto DataSomeValueFrom(numIdent xsd:anyType))

SubClassOf(Progetto DataSomeValueFrom(costo xsd:anyType))

SubClassOf(Progetto DataSomeValueFrom(durata xsd:anyType))

SubClassOf(Schema DataSomeValueFrom(versione xsd:anyType))

SubClassOf(Schema DataSomeValueFrom(numero xsd:anyType))

SubClassOf(Schema DataSomeValueFrom(tempoSvil xsd:anyType))

SubClassOf(Concettuale DataSomeValueFrom(numProgettisti xsd:anyType))

SubClassOf(Logico DataSomeValueFrom(dimensione xsd:anyType))

SubClassOf(DBMS DataSomeValueFrom(codice xsd:anyType))

SubClassOf(DBMS DataSomeValueFrom(anno xsd:anyType))

SubClassOf(Committente DataSomeValueFrom(codFis xsd:anyType))

SubClassOf(Committente DataSomeValueFrom(capSociale xsd:anyType))

SubClassOf(Committente DataSomeValueFrom(numDip xsd:anyType))

FunctionalDataProperty(numDip)

FunctionalDataProperty(capSociale)

FunctionalDataProperty(codFis)

FunctionalDataProperty(anno)

FunctionalDataProperty(codice)

FunctionalDataProperty(dimensione)

FunctionalDataProperty(numProgettisti)

FunctionalDataProperty(tempoSvil)

FunctionalDataProperty(numero)

FunctionalDataProperty(versione)

FunctionalDataProperty(durata)

FunctionalDataProperty(costo)

FunctionalDataProperty(numIdent)

Definizione del dominio degli attributi di ruolo

ObjectPropertyDataDomain(compenso commessa)

ObjectPropertyDataDomain(inizioValid licenza)

Definizione del range degli attributi di ruolo

ObjectPropertyDataRange(compenso rdf:integer)

ObjectPropertyDataRange(inizioValid rdf:string)

Definizione di cardinalità minima degli attributi di ruolo

SubObjectPropertyOf(commessa ObjectPropertyDataSomeValueFrom(compenso xsd:anyType))

SubObjectPropertyOf(licenza ObjectPropertyDataSomeValueFrom(inizioValid xsd:anyType))

Definizione di cardinalità massima degli attributi di ruolo

FunctionalObjectPropertyData(compenso)

FunctionalObjectPropertyData(inizioValid)

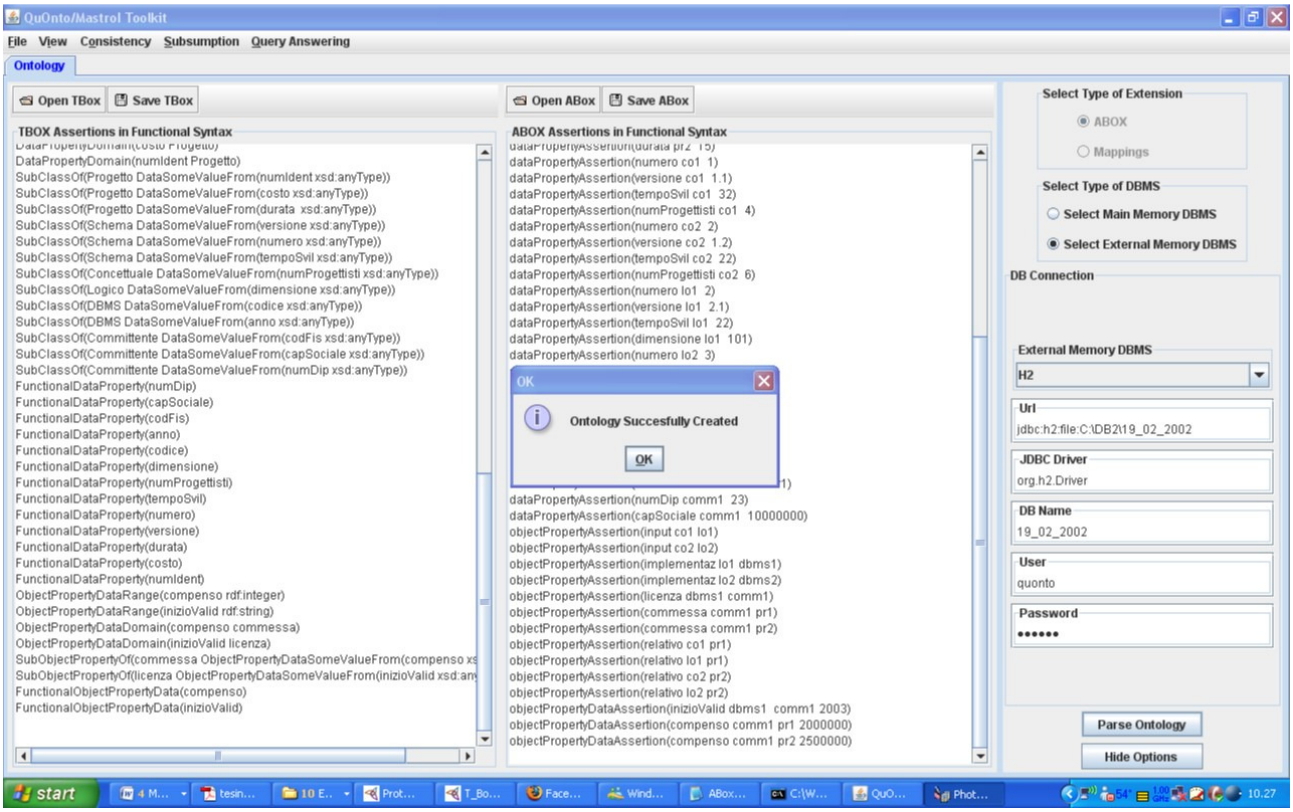
Il tool QuOnto per interrogare l'ontología

Mediante il tool QuOnto/ Mastro al quale sono state fornite in input la TBox e l'ABox espresse in sintassi funzionale OWL è stato possibile

- Effettuare il controllo di consistenza dell'ontología.
- Effettuare il query answering sull'ontología
- Valutare utilizzando query booleane espresse in SparSQL i vincoli di integrità non esprimibili in sintassi funzionale.
- Generare un database H2, che sarà utilizzato successivamente dal tool Protege 3.3.1 per effettuare i mapping.

Controllo di consistenza sull'ontología

Una volta importata l'ABox e la TBox con il tool è possibile effettuare il controllo di consistenza sull'ontología:



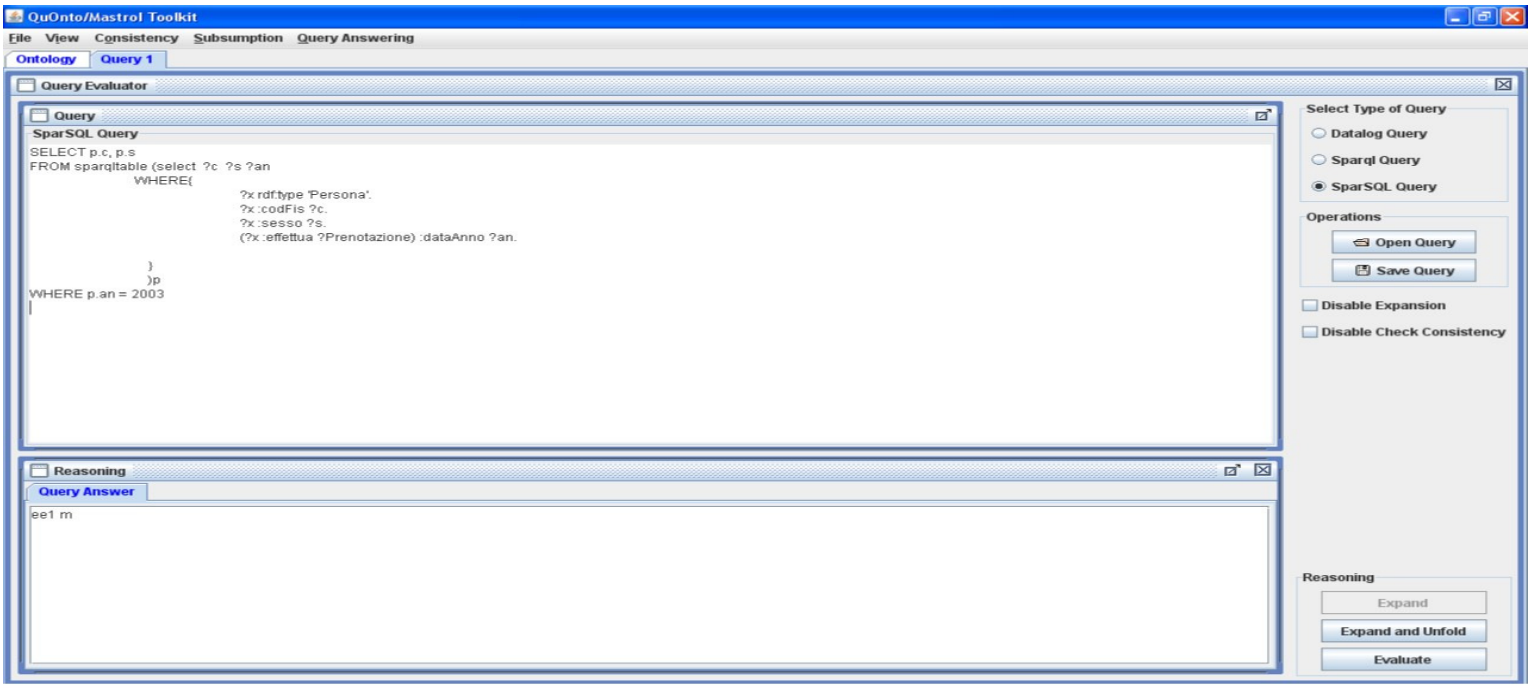
Query answering sull'ontología

Query n°1:

Per ogni schema concettuale che ha richiesto piu' di 30 giorni, si vogliono conoscere i dati relativi al progetto, al numero e alla versione.

```
SELECT p.numPro,p.numSch,p.ver
FROM sparqltable (select ?numPro ?numSch ?ver ?temp
        WHERE{
                ?x rdf:type 'Concettuale'.
                ?pr rdf:type 'Progetto'.
                ?x :relativo?pr.
                ?x :numProgettisti?num.
                ?x :tempoSvil?temp.
                ?x :versione?ver.
                ?pr :numIdent?numPro.
                ?x :numero?numSch.
})p
WHERE p.temp > 30
```

Screenshot Query 1:



Query n° 2:

Per ogni schema logico di dimensione maggiore di 100, si vogliono conoscere i dati relativi al progetto, al numero e alla versione.

SELECT p.costo,p.tempo

FROM sparqltable (select ?dim ?costo ?tempo

WHERE{

?x rdf:type 'Logico'.

?x :relativo?pr.

?x :dimensione?dim.

?x :tempoSvil?tempo.

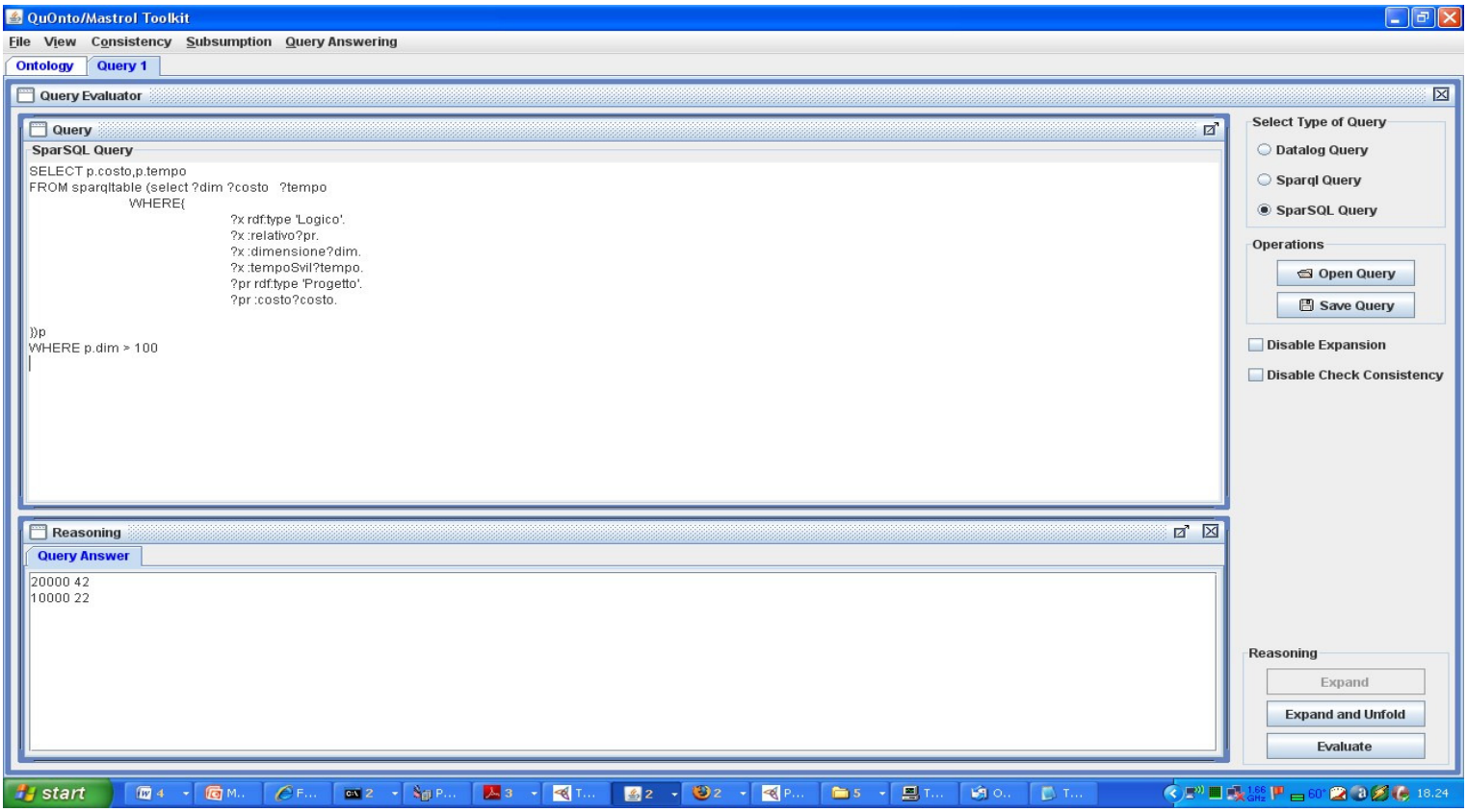
?pr rdf:type 'Progetto'.

?pr :costo?costo.

})p

WHERE p.dim > 100

Screenshot Query 2:



Query n° 3:

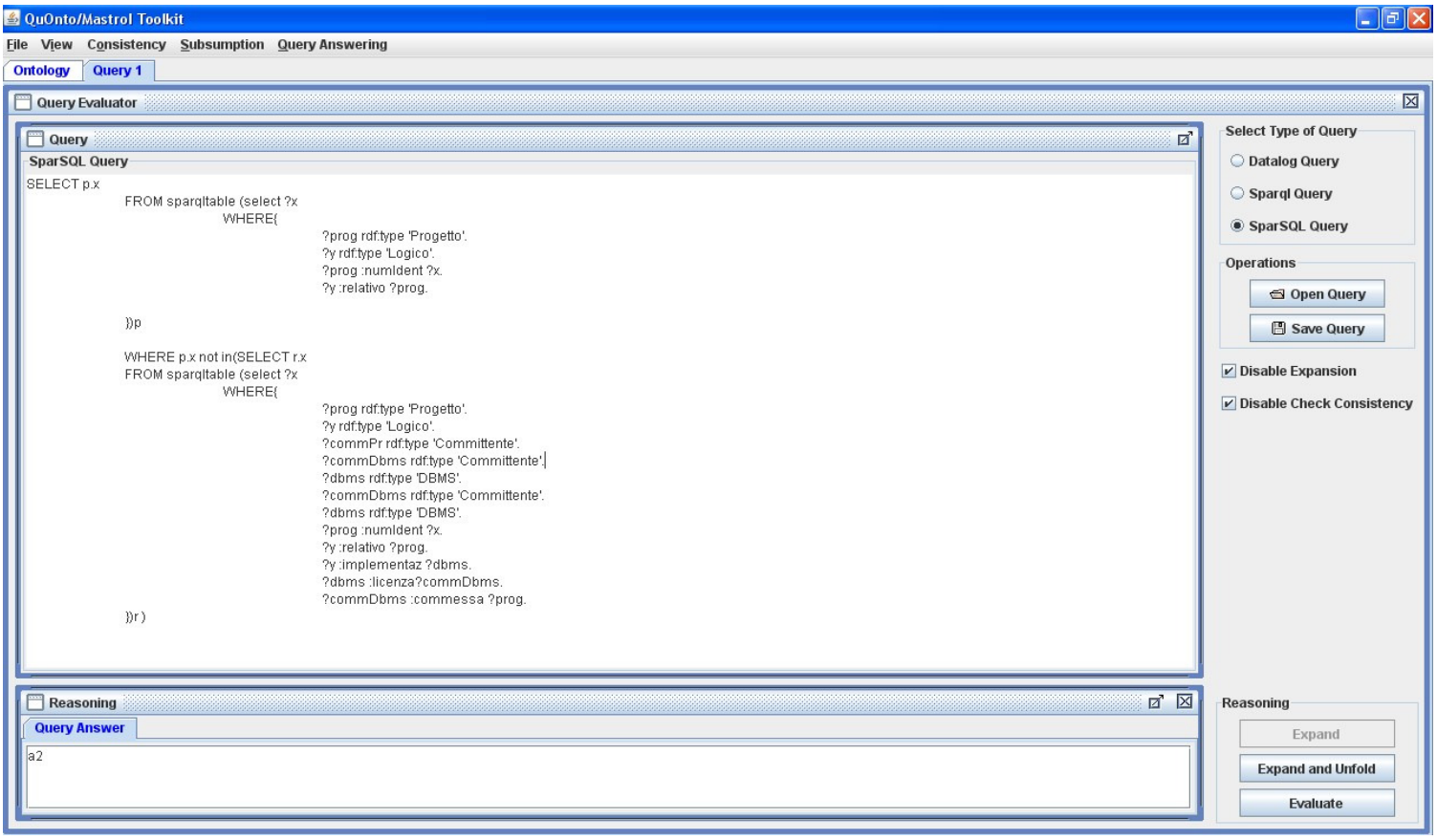
Fornire la lista dei progetti per i quali è stato prodotto almeno uno schema logico implementato in un DBMS per il quale almeno un committente del relativo progetto non ha licenza.

```
SELECT p.x
        FROM sparqltable (select ?x
                WHERE{
                        ?prog rdf:type 'Progetto'.
                        ?y rdf:type 'Logico'.
                        ?prog :numIdent ?x.
                        ?y :relativo ?prog.

        })p

        WHERE p.x not in(SELECT r.x
        FROM sparqltable (select ?x
                WHERE{
                        ?prog rdf:type 'Progetto'.
                        ?y rdf:type 'Logico'.
                        ?commPr rdf:type 'Committente'.
```

?commDbms rdf:type 'Committente'.
?dbms rdf:type 'DBMS'.
?commDbms rdf:type 'Committente'.
?dbms rdf:type 'DBMS'.
?prog :numIdent ?x.
?y :relativo ?prog.
?y :implementaz ?dbms.
?dbms :licenza?commDbms.
?commDbms :commessa ?prog.

})r)

Screenshot Query 3



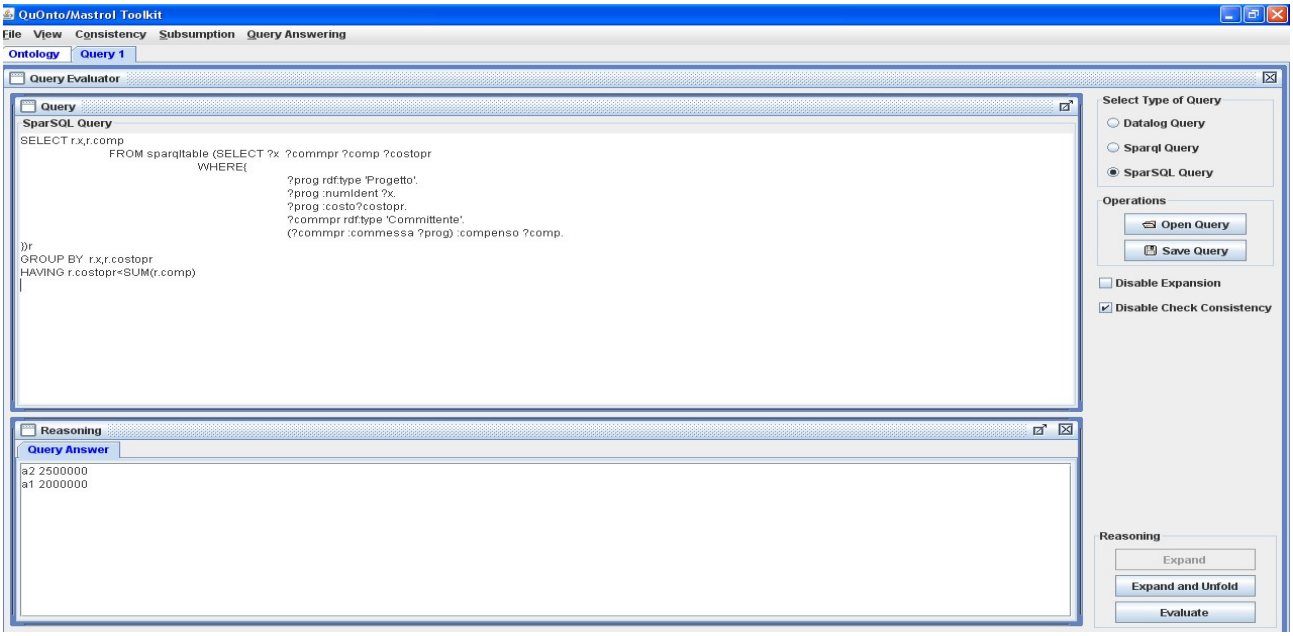
:

Query n° 4:

Produrre la lista di tutti i Progetti il cui costo è inferiore al compenso totale che hanno determinato per l'azienda, dove il compenso totale che un progetto determina per l'azienda è semplicemente la somma dei compensi erogati dai relativi committenti per quel progetto.

```
SELECT r.x,r.comp
        FROM sparqltable (SELECT ?x  ?commpr ?comp ?costopr
                WHERE{
                        ?prog rdf:type 'Progetto'.
                        ?prog :numIdent ?x.
                        ?prog :costo?costopr.
                        ?commpr rdf:type 'Committente'.
                                                (?commpr :commessa ?prog) :compenso ?comp.
})r
GROUP BY  r.x,r.costopr
HAVING r.costopr<SUM(r.comp)
```

Screenshot Query 4



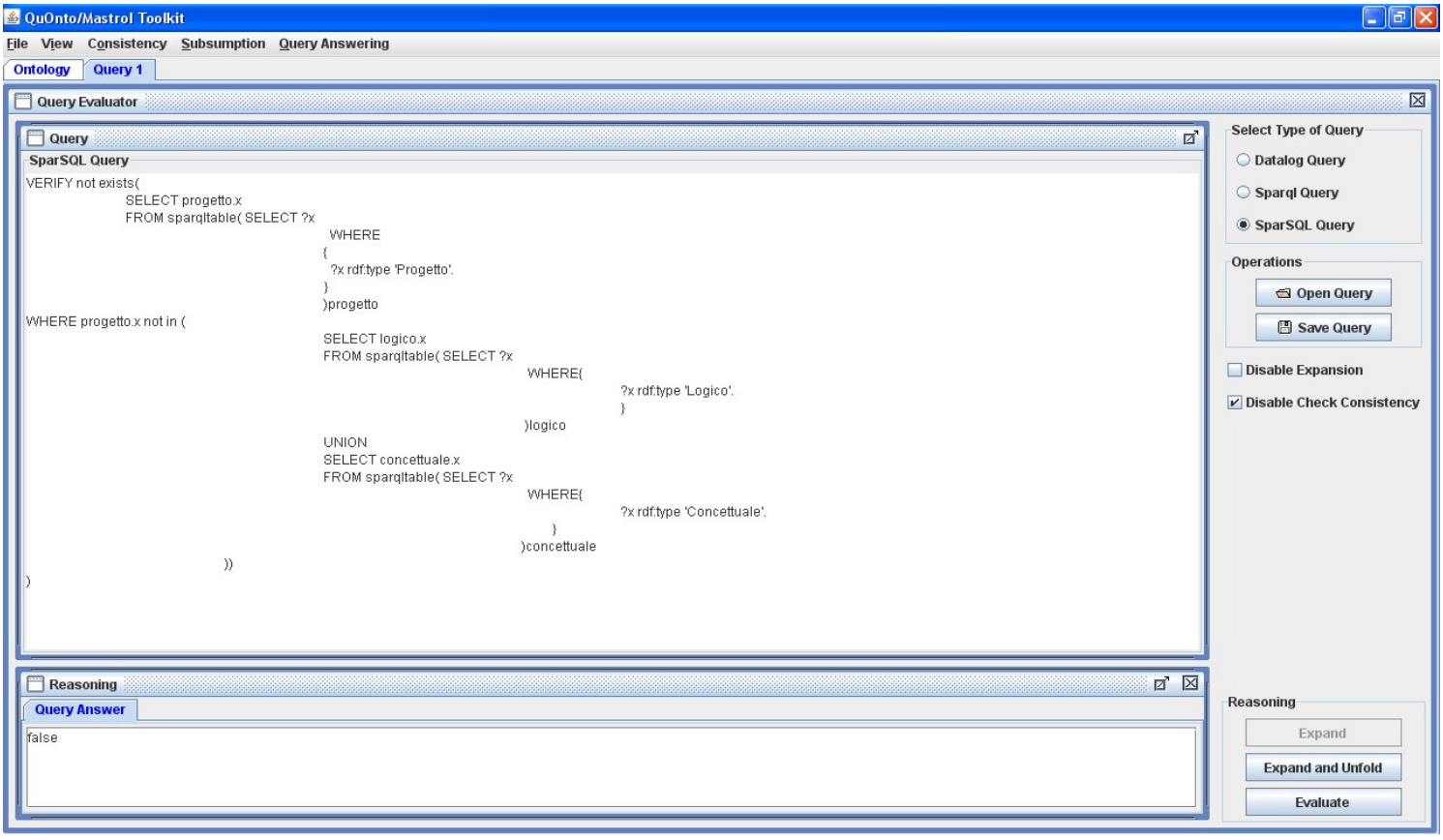
Query booleana per verificare il vincolo di completezza della generalizzazione

Per verificare il vincolo di completezza della generalizzazione presente nel diagramma ER, dato che la sintassi funzionale non ne permette la definizione, si utilizza una query booleana SparSQL che permette di verificare la consistenza dell'ontología rispetto a tale vincolo.

```
VERIFY not exists(
        SELECT progetto.x
        FROM sparqltable( SELECT ?x
                          WHERE
                         {
                          ?x rdf:type 'Progetto'.
                         }
```

```
                    )progetto
WHERE progetto.x not in (
                    SELECT logico.x
                    FROM sparqltable( SELECT ?x
                                WHERE{
                                    ?x rdf:type 'Logico'.
                                    }
                                )logico
                    UNION
                    SELECT concettuale.x
                    FROM sparqltable( SELECT ?x
                                WHERE{
                                    ?x rdf:type 'Concettuale'.
                                }
                                )concettuale
                ))
)
```

Screenshot QueryBooleana



Generazione dell'ontología owl

Per la generazione del file .owl si è utilizzato Protege 4.0 che permette di importare un' ontología scritta in sintassi funzionale restituendo in output l'ontologia in formato RDF/XML. A questo punto si è utilizzato Protege 3.3.1 ed i plugin OBDA per generare i mapping (per collegare le Casses, gli Object Properties e le Data Properties ai dati).

Generazione dei mapping:

Mapping di concetti (es. concetto *Persona*) :

MappingPersona
Persona(funct($term))
SELECT term FROM Persona

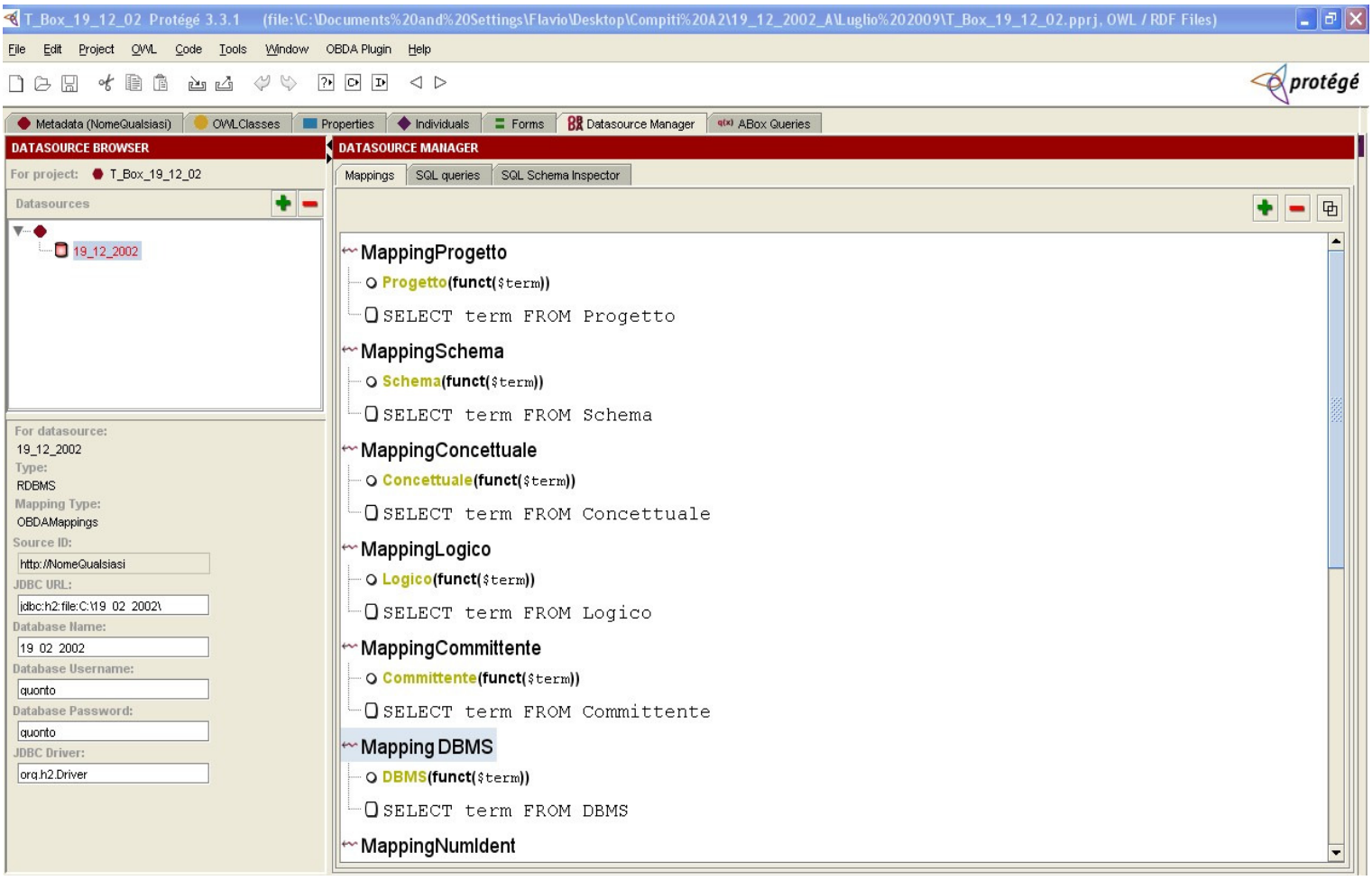
Mapping di attributi concetto (es. attributo di concetto *codFis*) :

MappingCodFis
codFis(funct($term1),funct($term2))
SELECT term1, term2 FROM codFis

Mapping di ruolo (es. ruolo *Effettua*) :

MappingEffettua
effettua(funct($term1),funct($term2))
SELECT term1, term2 FROM effettua

Screenshot Mapping



Query answering mediante Protégé

Query n°1:

Per ogni schema concettuale che ha richiesto piu' di 30 giorni, si vogliono conoscere i dati relativi al progetto, al numero e alla versione.

Non è stato possibile esprimere il vincolo "pie di" in quanto non ci sono operatori che lo permettono, inoltre non è possibile estrarre gli attributi "numero" e "versione" dalla classe Progetto.

Query n° 2:

Per ogni schema logico di dimensione maggiore di 100, si vogliono conoscere i dati relativi al progetto, al numero e alla versione.

Non è stato possibile esprimere il vincolo "maggiore di" in quanto non ci sono operatori che lo permettono, inoltre non è possibile estrarre gli attributi "sviluppo" e "costo" dalla classe .

Query n° 3:

Fornire la lista dei progetti per i quali è stato prodotto almeno uno schema logico implementato in un DBMS per il quale almeno un committente del relativo progetto non ha licenza.

Per esprimere la seguente query sarebbe stato necessario inserire un ciclo, dato che in OWL non ci sono variabili, si possono solo esprimere queries ad albero.

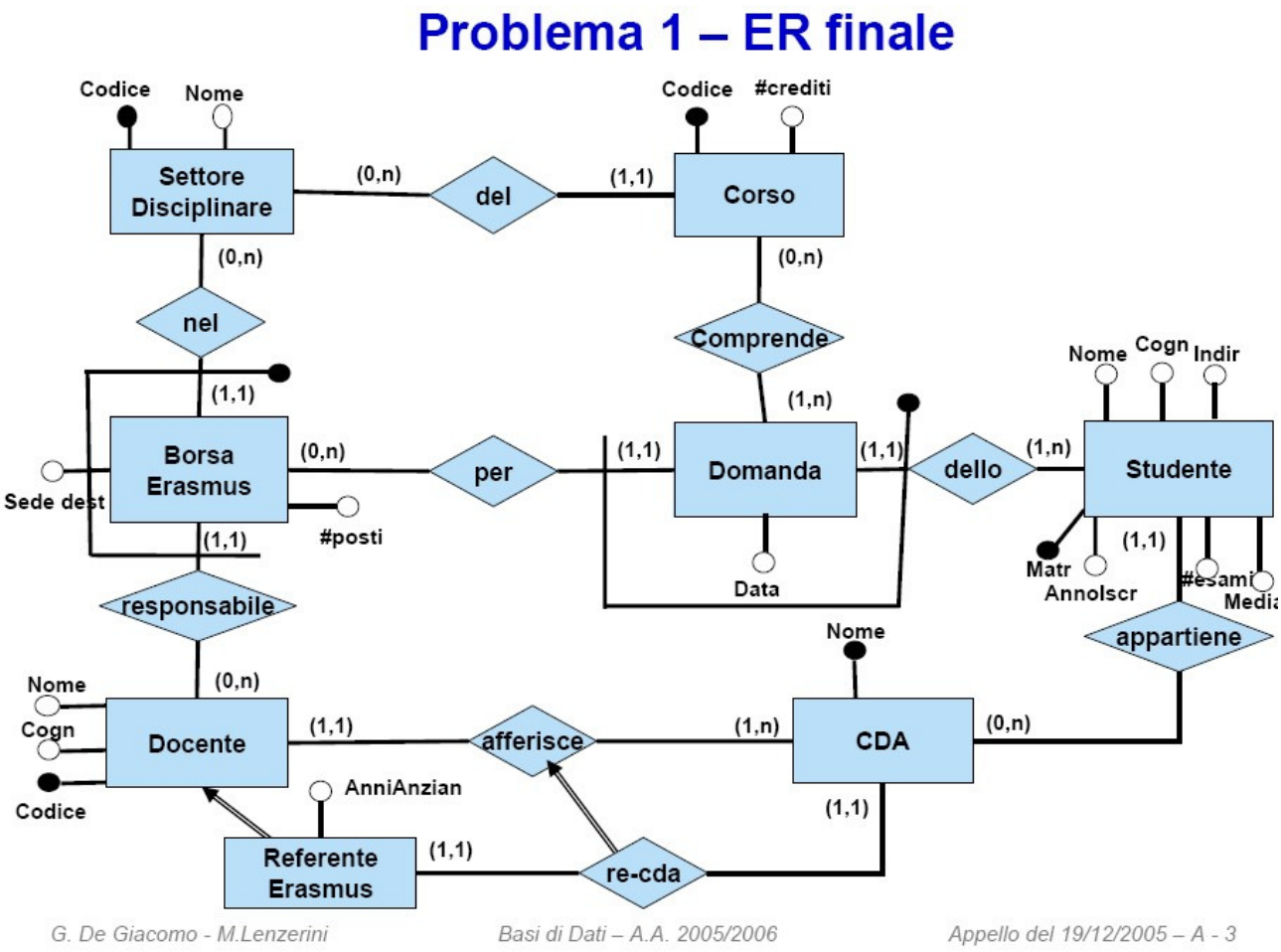
Query n° 4:

Produrre la lista di tutti i Progetti il cui costo è inferiore al compenso totale che hanno determinato per l'azienda, dove il compenso totale che un progetto determina per l'azienda è semplicemente la somma dei compensi erogati dai relativi committenti per quel progetto.

Non è possibile esprimere il vincolo "inferiore a " e la "somma dei compensi erogati" in quanto non ci sono operatori che lo permettono

Traduzione compito A del 19/12/2005

Diagramma Er:



Traduzione dello schema in sintassi tedesca:

Generalizzazione e vincolo di disgiunzione:

ReferenteErasmus $\sqsubseteq$ Docente

re-cda $\sqsubseteq$ afferisce

Definizione attributi di concetto:

SettoreDisciplinare $\sqsubseteq$ δ(codiceSettore)

δ(codiceSettore) $\sqsubseteq$ SettoreDisciplinare

ρ (codiceSettore) $\sqsubseteq$ xsd: int

(funct codiceSettore)

SettoreDisciplinare $\sqsubseteq$ δ(nome)

δ(nome) $\sqsubseteq$ SettoreDisciplinare

ρ(nome) $\sqsubseteq$ xsd: String

(funct nome)

Corso $\sqsubseteq$ δ(codiceCorso)

δ(codiceCorso) $\sqsubseteq$ Corso

ρ (codiceCorso) $\sqsubseteq$ xsd:int

(funct codiceCorso)

Corso $\sqsubseteq$ δ(#crediti)

δ(#crediti) $\sqsubseteq$ Corso

ρ(#crediti) $\sqsubseteq$ xsd: int

(funct #crediti)

Domande $\sqsubseteq$ δ(data)

δ(data) $\sqsubseteq$ Domande

ρ(data) $\sqsubseteq$ xsd: int

(funct data)

Domande $\sqsubseteq$ δ(codiceSettore)

δ(codiceSettore) $\sqsubseteq$ Domande

ρ(codiceSettore) $\sqsubseteq$ xsd: int

(funct codiceSettore)

Domande $\sqsubseteq$ δ(sedeDest)

δ(sedeDest) $\sqsubseteq$ Domande

ρ(sedeDest) $\sqsubseteq$ xsd: int

(funct sedeDest)

Domande $\sqsubseteq$ δ(codiceDocente)

δ((codiceDocente) $\sqsubseteq$ Domande

ρ((codiceDocente) $\sqsubseteq$ xsd: String

(funct codiceDocente)

Domande $\sqsubseteq$ δ(matr)

δ(matr) $\sqsubseteq$ Domande

ρ(matr) $\sqsubseteq$ xsd: int

(funct matr)

Studente $\sqsubseteq$ δ(matr)

δ(matr) $\sqsubseteq$ Studente

ρ(matr) $\sqsubseteq$ xsd: int

(funct matr)

Studente $\sqsubseteq$ δ(annoIscr)

δ(annoIscr) $\sqsubseteq$ Studente

ρ(annoIscr) $\sqsubseteq$ xsd: int

(funct annoIscr)

Studente $\sqsubseteq$ δ(#esami)

δ((#esami) $\sqsubseteq$ Studente

ρ((#esami) $\sqsubseteq$ xsd: int

(funct #esami)

Studente $\sqsubseteq$ δ(media)

δ(media) $\sqsubseteq$ Studente

ρ(media) $\sqsubseteq$ xsd: int

(funct media)

Studente $\sqsubseteq$ δ(nomeStudente)

δ(nomeStudente) $\sqsubseteq$ Studente

ρ(nomeStudente) $\sqsubseteq$ xsd: String

(funct nomeStudente)

Studente $\sqsubseteq$ δ(cogn)

δ(cogn) $\sqsubseteq$ Studente

ρ(cogn) $\sqsubseteq$ xsd: String

(funct cogn)

Studente $\sqsubseteq$ δ(indir)

δ(indir) $\sqsubseteq$ Studente

ρ(indir) $\sqsubseteq$ xsd: String

(funct indir)

CDA $\sqsubseteq$ δ(nomeCda)

δ(nomeCda) $\sqsubseteq$ CDA

ρ(nomeCda) $\sqsubseteq$ xsd: String

(funct nomeCda)

Docente$\sqsubseteq$ δ(codiceDocente)

δ(codiceDocente) $\sqsubseteq$ Docente

ρ(codiceDocente) $\sqsubseteq$ xsd: int

(funct codiceDocente)

Docente$\sqsubseteq$ δ(cogn)

δ(cogn) $\sqsubseteq$ Docente

ρ(cogn) $\sqsubseteq$ xsd: String

(funct cogn)

Docente$\sqsubseteq$ δ(nome)

δ(nome) $\sqsubseteq$ Docente

ρ(nome) $\sqsubseteq$ xsd: String

(funct nome)

Definizione dei ruoli e cardinalità

$\exists$ del $\sqsubseteq$ SettoreDisciplinare

$\exists$ del^{-} $\sqsubseteq$ Corso

$\exists$ del^{-} $\sqsubseteq$ SettoreDisciplinare

(funct del^{-})

$\exists$ comprende $\sqsubseteq$ Domande

$\exists$ comprende^{-} $\sqsubseteq$ Corso

Domande $\sqsubseteq$ $\exists$ comprende

$\exists$ dello $\sqsubseteq$ Domande

$\exists$dello^{-} $\sqsubseteq$ Studente

Domande $\sqsubseteq$ $\exists$dello

(funct dello)

$\exists$ appartiene $\sqsubseteq$ Studente

$\exists$appartiene^{-} $\sqsubseteq$ CDA

Studente ⊑ ∃ appartiene

(funct appartiene)

∃ appartiene ⊑ Studente

∃appartiene $^{-}$ ⊑ CDA

Studente ⊑ ∃ appartiene

(funct appartiene)

(funct afferisce)

Non Esprimibile in quanto *afferisce* compare alla destra di una relazione di inclusione tra ruoli.

Docente ⊑ ∃afferisce

∃afferisce $^{-}$ ⊑ Docente

∃ responsabile ⊑ Docente

∃ responsabile^{-} ⊑ BorsaErasmus

∃ responsabile^{-} ⊑ Docente

(funct responsabile^{-})

∃ per ⊑ Domanda

∃ per^{-} ⊑ BorsaErasmus

Domanda ⊑ ∃ per

(funct per)

∃ nel ⊑ BorsaErasmus

∃ nel^{-} ⊑ SettoreDisciplinare

BorsaErasmus $\sqsubseteq \exists$ nel

(funct nel)

Traduzione dello schema in sintassi funzionale

Generalizzazione e vincolo di disgiunzione:

SubClassOf(ReferenteErasmus Docente)

SubObjectPropertyOf (re-cda afferisce)

Definizione dei ruoli

ObjectPropertyDomain(del SettoreDisciplinare)

ObjectPropertyRange(del Corso)

ObjectPropertyDomain(comprende Domanda)

ObjectPropertyRange(comprende Corso)

ObjectPropertyDomain(appartiene Studente)

ObjectPropertyRange(appartiene CDA)

ObjectPropertyDomain(re-cda CDA)

ObjectPropertyRange(re-cda ReferenteErasmus)

ObjectPropertyDomain(afferisce CDA)

ObjectPropertyRange(afferisce Docente)

ObjectPropertyDomain(nel BorsaErasmus)

ObjectPropertyRange(nel SettoreDisciplinare)

ObjectPropertyDomain(per Domanda)

ObjectPropertyRange(per BorsaErasmus)

ObjectPropertyDomain(responsabile Docente)

ObjectPropertyRange(responsabile BorsaErasmus)

ObjectPropertyDomain(dello Domanda)

ObjectPropertyRange(dello Studente)

Definizione delle cardinalità delle relazioni

SubClassOf(ObjectMinCardinality(1 InverseObjectProperty(del)) Corso)

SubClassOf(Domanda ObjectMinCardinality(1 comprende))

SubClassOf(Studente ObjectMinCardinality(1 appartiene))

SubClassOf(CDA ObjectMinCardinality(1 re-cda))

SubClassOf(CDA ObjectMinCardinality(1 afferisce))

SubClassOf(Domanda ObjectMinCardinality(1 dello))

SubClassOf(ObjectMinCardinality(1 InverseObjectProperty(afferisce)) Docente)

SubClassOf(ObjectMinCardinality(1 InverseObjectProperty(re-cda)) ReferenteErasmus)

SubClassOf(ObjectMinCardinality(1 InverseObjectProperty(dello)) Studente)

SubClassOf(BorsaErasmus ObjectMinCardinality(1 nel))

SubClassOf(Domanda ObjectMinCardinality(1 per))

SubClassOf(ObjectMinCardinality(1 InverseObjectProperty(responsabile)) BorsaErasmus)

FunctionalObjectProperty(appartiene)

FunctionalObjectProperty(recda)

FunctionalObjectProperty(dello)

FunctionalObjectProperty(nel)

FunctionalObjectProperty(per)

FunctionalObjectProperty(InverseObjectPropertyOf(responsabile))

FunctionalObjectProperty(InverseObjectPropertyOf(del))

FunctionalObjectProperty(InverseObjectPropertyOf(recda))

Definizione del dominio degli attributi di concetto

DataPropertyDomain(codiceSettore SettoreDisciplinare)

DataPropertyDomain(nomeSettore SettoreDisciplinare)

DataPropertyDomain(codiceCorso Corso)

DataPropertyDomain(numCrediti Corso)

DataPropertyDomain(data Domanda)

DataPropertyDomain(matr Studente)

DataPropertyDomain(annoIscr Studente)

DataPropertyDomain(numEsami Studente)

DataPropertyDomain(nome Docente)

DataPropertyDomain(codiceDocente Docente)

DataPropertyDomain(cogn Docente)

DataPropertyDomain(anniAnzian ReferenteErasmus)

DataPropertyDomain(nomeCda CDA)

DataPropertyDomain(indir Studente)

DataPropertyDomain(cognStudente Studente)

DataPropertyDomain(nomeStudente Studente)

DataPropertyDomain(media Studente)

DataPropertyDomain(sedeDest BorsaErasmus)

DataPropertyDomain(numPosti BorsaErasmus)

Definizione del range degli attributi di concetto

DataPropertyRange(codiceSettore xsd:string)

DataPropertyRange(nomeSettore xsd:string)

DataPropertyRange(codiceCorso xsd:string)

DataPropertyRange(numCrediti xsd:int)

DataPropertyRange(data xsd:date)

DataPropertyRange(matr xsd:string)

DataPropertyRange(annoIscr xsd:date)

DataPropertyRange(numEsami xsd:int)

DataPropertyRange(media xsd:float)

DataPropertyRange(nomeStudente xsd:string)

DataPropertyRange(cognStudente xsd:string)

DataPropertyRange(indir xsd:string)

DataPropertyRange(nomeCda xsd:string)

DataPropertyRange(anniAnzian xsd:int)

DataPropertyRange(codiceDocente xsd:string)

DataPropertyRange(cogn xsd:string)

DataPropertyRange(nome xsd:string)

DataPropertyRange(sedeDest xsd:string)

DataPropertyRange(numPosti xsd:int)

Definizione di cardinalità minima e massima (1,1) degli attributi di concetto

SubClassOf(SettoreDisciplinare DataSomeValueFrom(codiceSettore xsd:anyType))

SubClassOf(SettoreDisciplinare DataSomeValueFrom(nomeSettore xsd:anyType))

SubClassOf(Corso DataSomeValueFrom(codiceCorso xsd:anyType))

SubClassOf(Corso DataSomeValueFrom(numCrediti xsd:anyType))

SubClassOf(Domanda DataSomeValueFrom(data xsd:anyType))

SubClassOf(CDA DataSomeValueFrom(nomeCda xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(indir xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(cognStudente xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(nomeStudente xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(media xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(numEsami xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(annoIscr xsd:anyType))

SubClassOf(Studente DataSomeValueFrom(matr xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(codiceDocente xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(cogn xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(nome xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(anniAnzian xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(sedeDest xsd:anyType))

SubClassOf(Docente DataSomeValueFrom(numPosti xsd:anyType))

FunctionalDataProperty(codiceSettore)

FunctionalDataProperty(nomeSettore)

FunctionalDataProperty(codiceCorso)

FunctionalDataProperty(numCrediti)

FunctionalDataProperty(data)

FunctionalDataProperty(nomeCda)

FunctionalDataProperty(indir)

FunctionalDataProperty(nomeStudente)

FunctionalDataProperty(cognStudente)

FunctionalDataProperty(media)

FunctionalDataProperty(numEsami)

FunctionalDataProperty(annoIscr)

FunctionalDataProperty(matr)

FunctionalDataProperty(codiceDocente)

FunctionalDataProperty(cogn)

FunctionalDataProperty(nome)

FunctionalDataProperty(anniAnzian)

FunctionalDataProperty(sedeDest)

FunctionalDataProperty(numPosti)

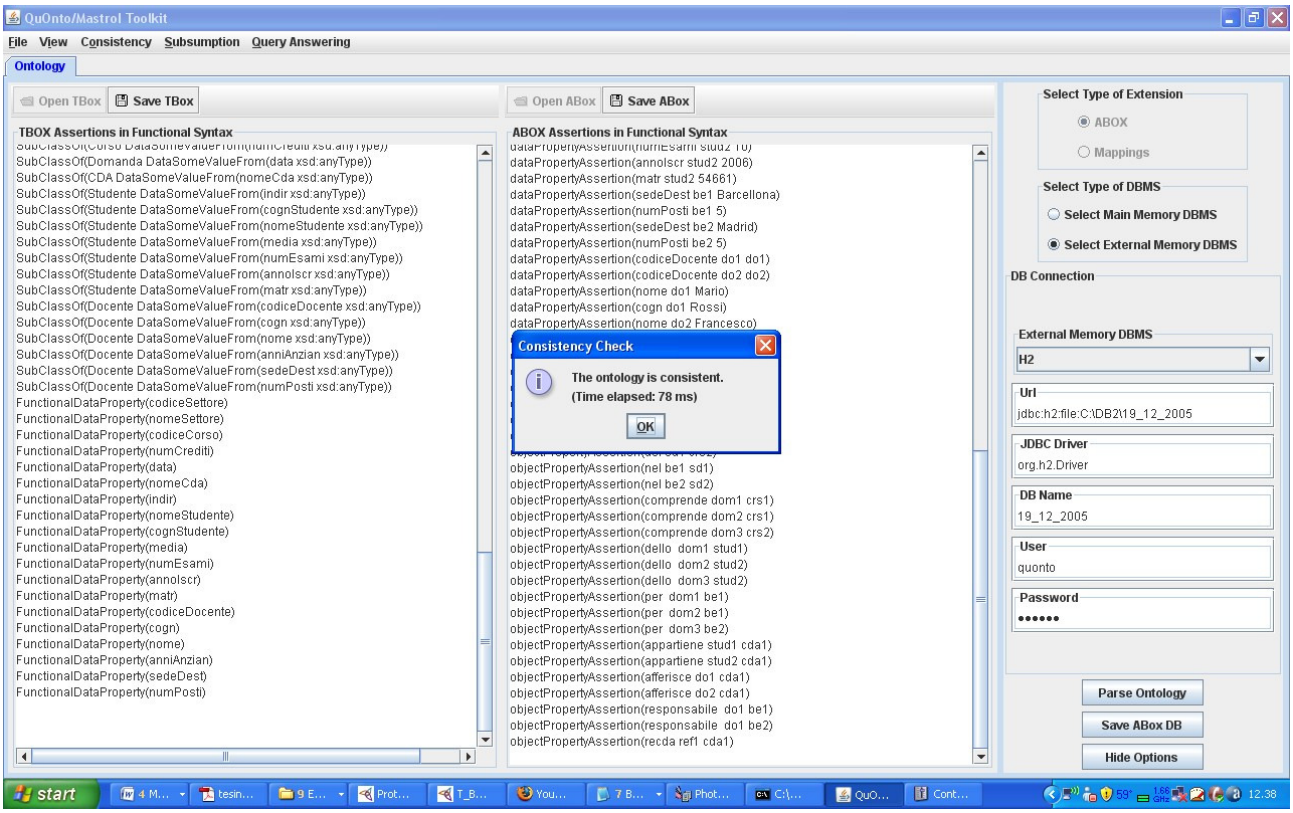
Il tool QuOnto per interrogare l'ontología

Mediante il tool QuOnto/ Mastro al quale sono state fornite in input la TBox e l'ABox espresse in sintassi funzionale OWL è stato possibile

- Effettuare il controllo di consistenza dell'ontología.
- Effettuare il query answering sull'ontología
- Valutare utilizzando query booleane espresse in SparSQL i vincoli di integrità non esprimibili in sintassi funzionale.
- Generare un database H2, che sarà utilizzato successivamente dal tool Protege 3.3.1 per effettuare i mapping.

Controllo di consistenza sull'ontología

Una volta importata l'ABox e la TBox con il tool è possibile effettuare il controllo di consistenza sull'ontología:



Query answering sull'ontología

Query n°1:

Per ogni domanda restituire la matricola, la media, il numero di esami superati, l'anno di iscrizione ed il consiglio d'area dello sudiente che l'ha presentata

SELECT p.sd, p.cs, p.cd ,p.mat,p.med,p.nes,p.ais,p.ca

FROM sparqltable (select ?sd ?cs ?cd ?mat ?med ?nes ?ais ?ca

WHERE{

?a rdf:type 'Domanda'.

?b rdf:type 'Studente'.

?c rdf:type 'CDA'.

?d rdf:type 'SettoreDisciplinare'.

?e rdf:type 'BorsaErasmus'.

?f rdf:type 'Docente'.

?a :dello?b.

?a :per?e.

?b :appartiene?c.

?e :nel?d.

?f :responsabile?e.

?b :matr?mat.

?b :media?med.

?b :numEsami ?nes.

?b :annoIscr?ais.

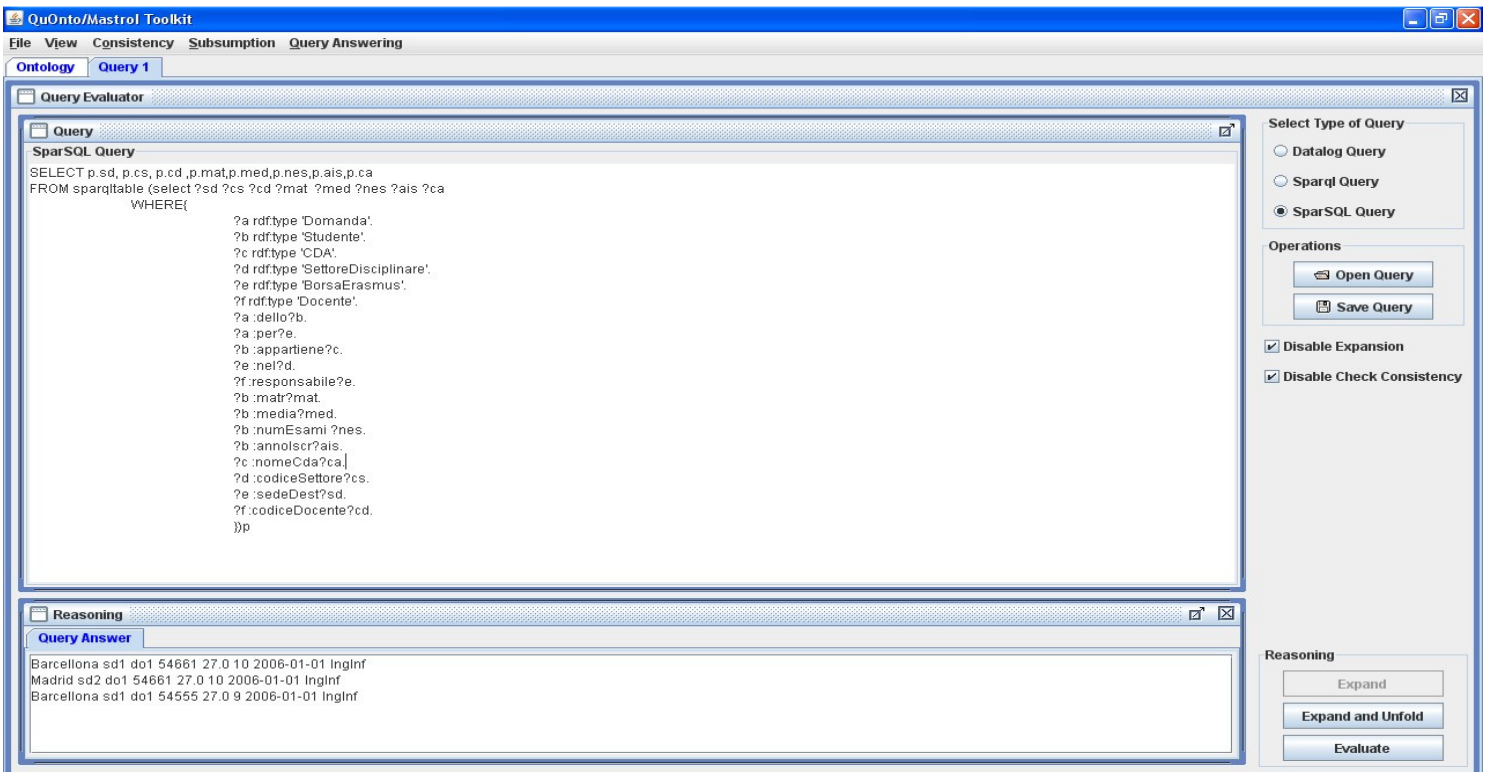
?c :nomeCda?ca.

?d :codiceSettore?cs.

?e :sedeDest?sd.

?f :codiceDocente?cd.

})pWHERE p.temp > 30



Query n° 2:

Restituire le domande per le quali tutti i corsi che lo studente intende frequentare sono nel settore disciplinare della borsa alla quale si riferisce la domanda.

```
SELECT p.sede,p.setdis
FROM sparqltable (select ?sede ?setdis ?bres ?stud
        WHERE{
                ?a rdf:type 'Domanda'.
                ?b rdf:type 'BorsaErasmus'.
                ?c rdf:type 'SettoreDisciplinare'.
                ?d rdf:type 'Studente'.
                ?e rdf:type 'Docente'.
                ?a :per?b.
                ?a:dello?d.
                ?b :nel?c.
                ?e :responsabile?b.
                ?b :sedeDest?sede.
                ?c :codiceSettore?setdis.
                ?e :codiceDocente?bres.
                ?d :matr?stud.
                })p
MINUS
SELECT r.sede,r.setdis
FROM sparqltable (select ?sede ?csetDis ?docRes ?stud ?setdis
        WHERE{
                ?a rdf:type 'Corso'.
                ?b rdf:type 'SettoreDisciplinare'.
                ?h rdf:type 'SettoreDisciplinare'.
                ?d rdf:type 'Domanda'.
                ?e rdf:type 'BorsaErasmus'.
                ?b :del?a.
                ?e :nel?h.
                ?d :comprende?a.
```

?d :dello?g.

?d :per?e.

?f rdf:type 'Docente'.

?f :responsabile?e.

?g rdf:type 'Studente'.

?e :sedeDest?sede.

?f :codiceDocente?docRes.

?b :codiceSettore?csetDis.

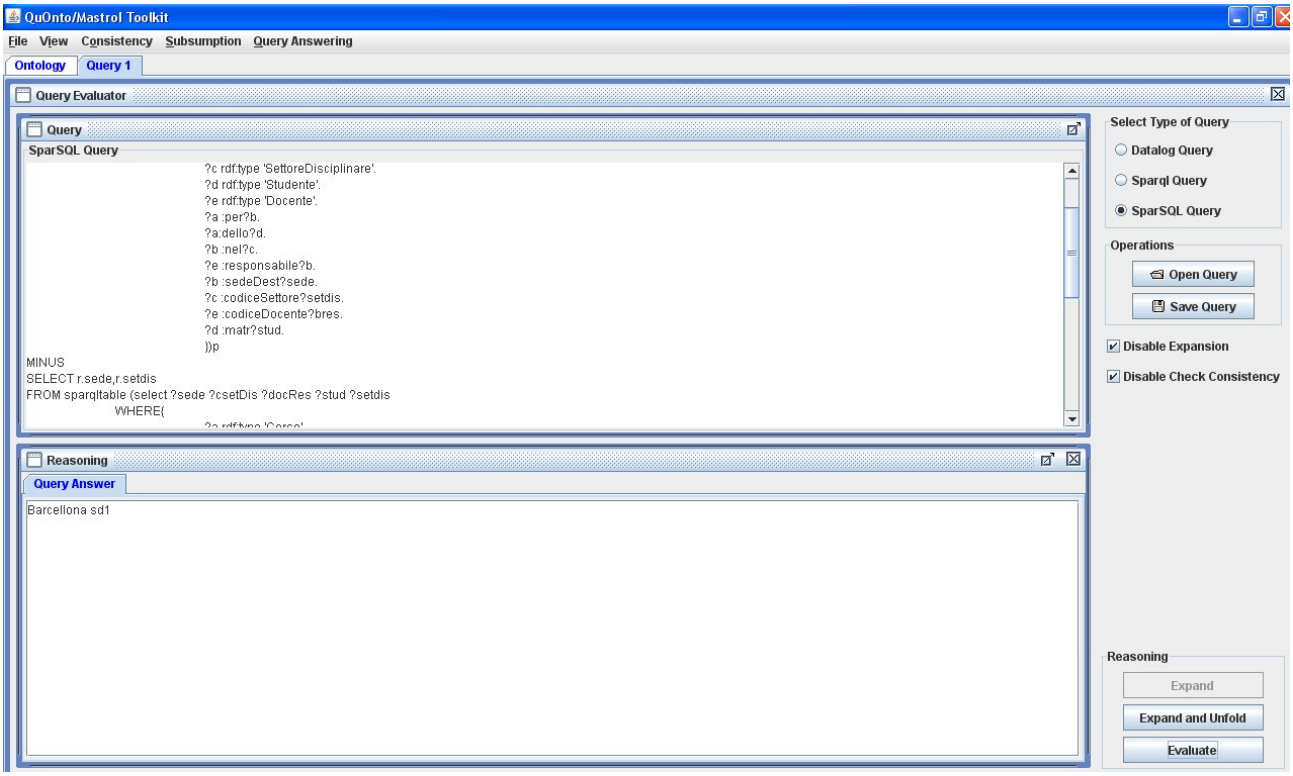
?h :codiceSettore?setdis.

?g :matr?stud.

})r

WHERE r.csetDis <> r.setdis

Screenshot Query 2:

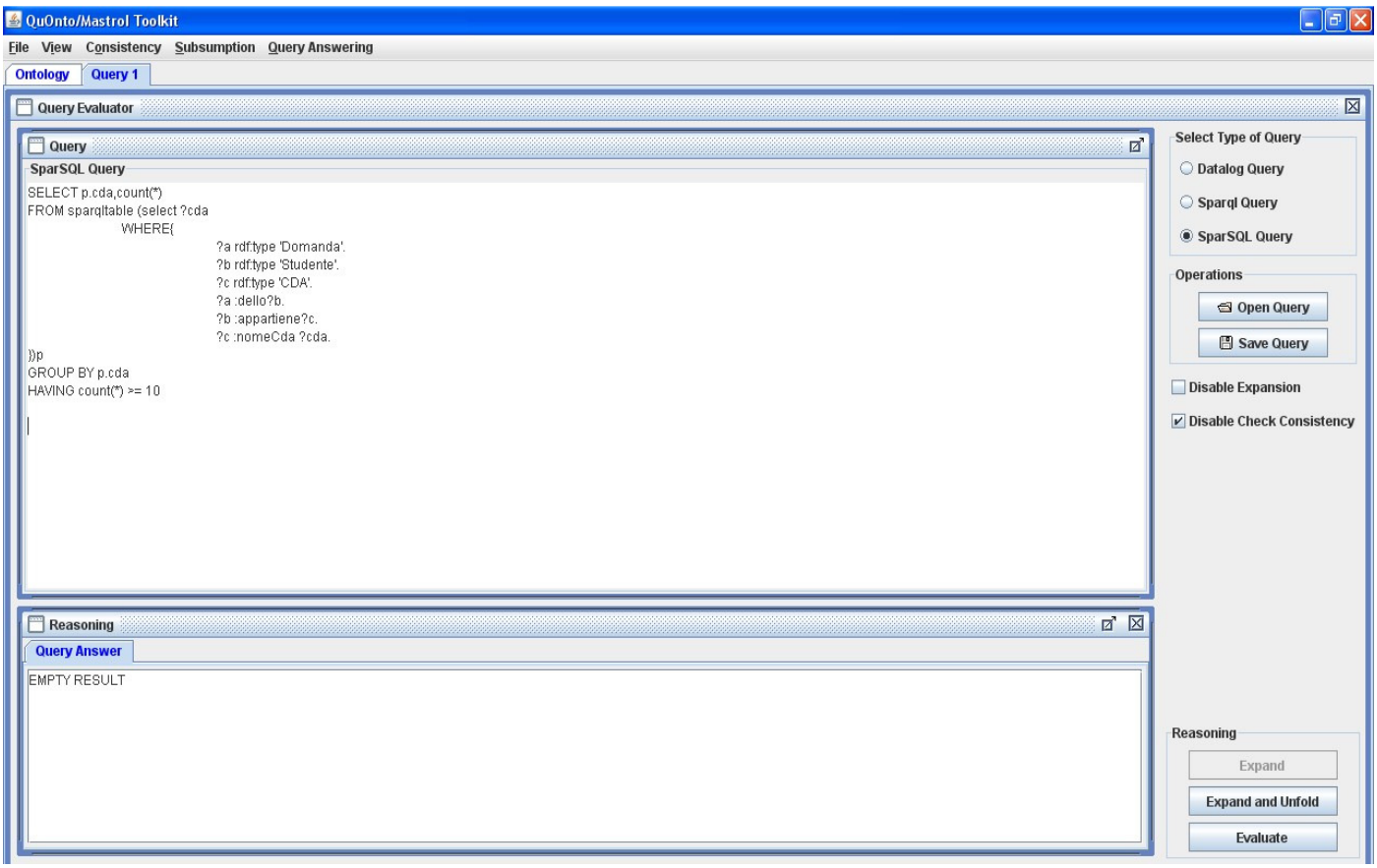


Query n° 3:

Un cosiglio d'area viene detto "rilevante per Erasmus" se sono state presentate almeno 10 domande di borse erasmus da studenti appartenenti al Consiglio d'Area stesso.Per ciascun Consiglio d'Area rilevante per l'erasmus calcolare il numero di domande presentate da studenti ad esso appartenenti.

```
SELECT p.cda,count(*)
FROM sparqltable (select ?cda
        WHERE{
                ?a rdf:type 'Domanda'.
                ?b rdf:type 'Studente'.
                ?c rdf:type 'CDA'.
                ?a :dello?b.
                ?b :appartiene?c.
                ?c :nomeCda ?cda.
})p
GROUP BY p.cda
HAVING count(*) >= 10
```

Screenshot Query 3:



Generazione dell'ontología owl

Per la generazione del file .owl si è utilizzato Protege 4.0 che permette di importare un' ontología scritta in sintassi funzionale restituendo in output l'ontologia scritta nel formato RDF/XML. A questo punto si è utilizzato Protege 3.3.1 ed i plugin OBDA per generare i mapping (per collegare le Casses, gli Object Properties e le Data Properties ai dati).

Generazione dei mapping:

Mapping di concetti (es. concetto *Persona*) :

MappingPersona
Persona(funct($term))
SELECT term FROM Persona

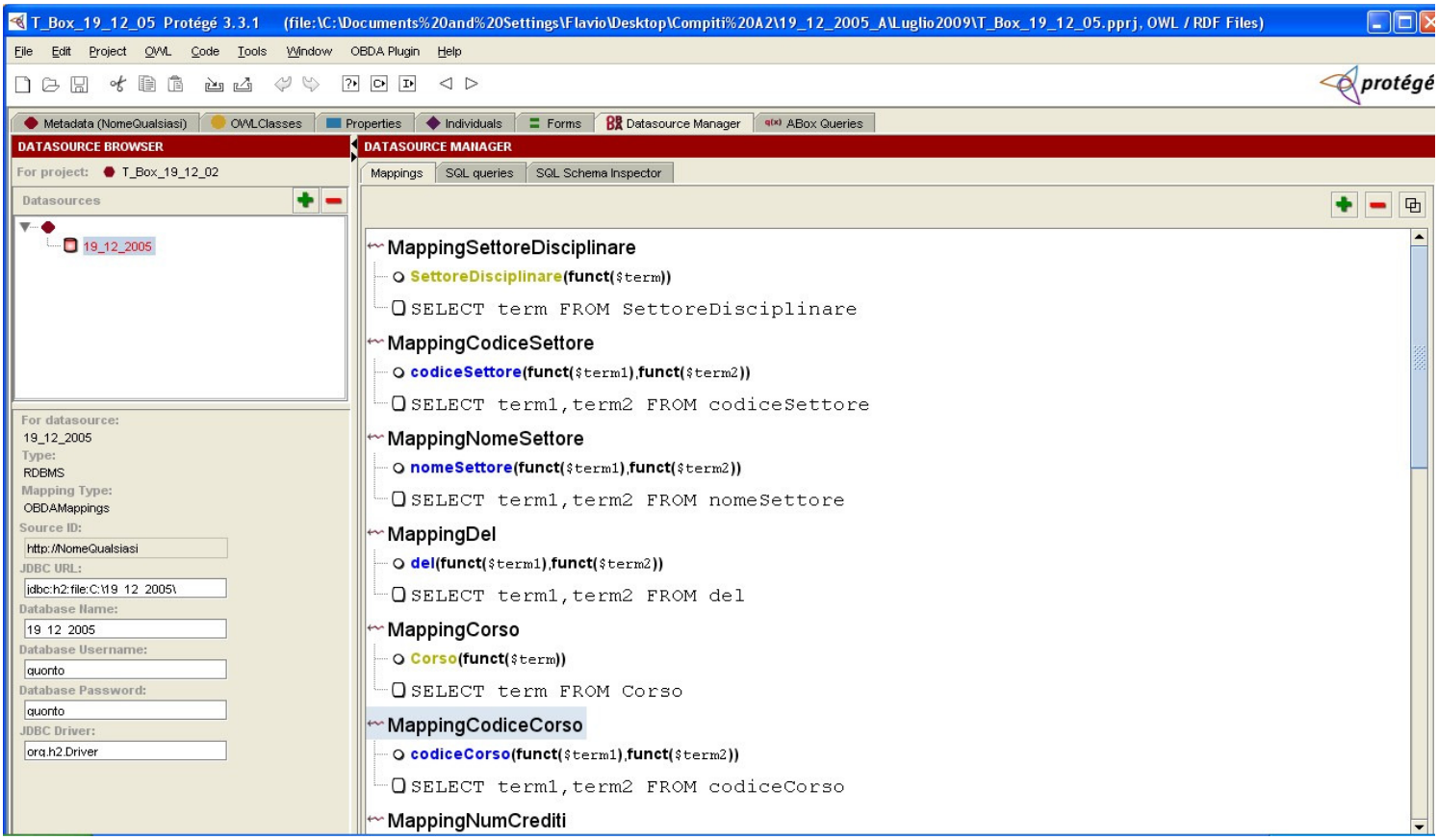
Mapping di attributi concetto (es. attributo di concetto *codFis*) :

MappingCodFis
codFis(funct($term1),funct($term2))
SELECT term1, term2 FROM codFis

Mapping di ruolo (es. ruolo *Effettua*) :

MappingEffettua
effettua(funct($term1),funct($term2))
SELECT term1, term2 FROM effettua

Screenshot Mapping



Query answering mediante Protégé

Query n°1:

Per ogni domanda restituire la matricola, la media, il numero di esami superati, l'anno di iscrizione ed il consiglio d'area dello studente che l'ha presentata.

Non è stato possibile estrarre i valori de gli attributi "matricola" e "media", "anno iscrizione" e "numero di esami superati" e "consiglio d'area" dello studente che l'ha presentata.

Query n° 2:

Restituire le domande per le quali tutti i corsi che lo studente intende frequentare sono nel settore disciplinare della borsa alla quale si riferisce la domanda.

La query non è ad albero, pertanto non è esprimibile in quanto non ci sono variabili in OWL.

Query n° 3:

Un cosiglio d'area viene detto "rilevante per Erasmus" se sono state presentate almeno 10 domande di borse erasmus da studenti appartenenti al Consiglio d'Area stesso.Per ciascun Consiglio d'Area rilevante per l'erasmus calcolare il numero di domande presentate da studenti ad esso appartenenti.

Non è possibile contare il numero di Domande, in quanto manca un operatore che lo permette.

7 Conclusioni

Osservando i risultati degli studi condotti in questo elaborato, possiamo evidenziare come le DL abbiano un potere espressivo diverso rispetto alla DL-LiteA,in seguito ne elencheremo le differenze:

Description logic utilizzando Protègè

- Non è possibile dichiarare attributi di relazioni(object property) in quanto OWL non permette di inserire una *datatype property* come *subproperty* di una *objectproperty.*
- Non è possibile specificare che gli attributi di entità abbiano cardinalità pari a (1,1) in quanto il linguaggio usato per comunicare con il ragionatore (DIG) non consente di definire restrizioni di cardinalità sui *datatypes,* per questo sono stati osti solo con cardinalità *functional,* che specifica che ciascuna entità può avere al massimo un valore per tale attributo.
- Non è possibile esplicitare le chiavi primarie in quanto non è possibile definire *inverseFunctional* una *datatypeProperty*.

Dl-LiteA, utilizzando il tool Mastro/QuOnto

- La DL-LiteA non permette la definizione di completezza di una generalizzazione, mediante pero le query booleane espresse con SparSQL è possibile verificare che l'ontologia sia consistente con il vincolo di completezza.
- Non è possibile esprimere per i ruoli una cardinalità differente da 1.
- Per ogni ruolo atomico o inverso di un ruolo atomico Q che appare in un concetto della forma $\exists Q.C$, l' asserzione (funct Q) e (funct Q^-) non può essere espressa in T;
- Per ogni asserzione di inclusione tra ruoli $Q \sqsubseteq R$ in T. dove R è un ruolo atomico o l' inverso di un ruolo atomico, l' asserzione (funct R) e (funct R^-) non appartiene a T;
- Per ogni asserzione di inclusione di attributivi concetto $U_c \sqsubseteq V_c$ in T, dove V_c è un attributo di concetto atomico, l' asserzione (funct V_c) non appartiene a T;.
- Per ogni asserzione di inclusione che riguarda attributi di ruolo, $U_R \sqsubseteq V_R$ in T dove V_R è un attributo di un ruolo. l'asserzione (funct V_R) non può appartenere a T.

8 Bibliografia

- G. De Giacomo, materiale didattico del corso doi “Seminari di Ingegneria del software” http://www.dis.uniroma1.it/~degiacom/didattica/semingsoft/
- Giuseppe De Giacomo, Diego Calvanese, Domenico Lembo, Maurizio Lenzerini,Antonella Poggi, Riccardo Rosati, *“Linking Data to Ontologies: The Description Logic DL-Lite $_A$”*. http://www.inf.unibz.it/~calvanese/papers-html/OWLED-2006.html
- Emma Di Pasquale, Domenico Fabio Savo, “Functional-Style Syntax for DL-Lite$_A$ and Mappings
- Giuseppe De Giacomo “*Epistemic First_Order Queries over Description Logic Knowledge Bases”*.
- OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/
- Giuseppe De Giacomo, Diego Calvanese, Domenico Lembo, Maurizio Lenzerini,Antonella Poggi, Riccardo Rosati, “Mastro: Efficient integration of relational data through DL ontologies”.
- Emma Di Pasquale, slide presentazione SparSQL a.a . 2007/2008.