Università degli Studi
di Roma "Sapienza"

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Anno accademico 2007/08

RAPPRESENTAZIONE DIAGRAMMI E/R ATTRAVERSO ONTOLOGIE CON QUERY SparSQL

Seminari di Ingegneria del Software
Prof. Giuseppe De Giacomo

a cura di:

ROBERTO PALMIERI

Indice

Introduzione

Il presente lavoro ha come obiettivo la traduzione di diagrammi E/R in ontologie. Ogni diagramma E/R descrive in maniera semanticamente precisa una realtà di interesse. Una ontologia è una concettualizzazione di un dominio d'interesse espressa in logica. Attraverso questo lavoro verranno forniti gli strumenti per una traduzione diretta, dove possibile, di ogni costrutto E/R in costrutti per ontologie. Inoltre nello schema E/R possiamo esprimere vari vincoli, come per esempio le molteplicità delle relazioni, oppure la molteplicità degli attributi, oppure vincoli sulle generalizzazioni, etc… Nell'ontologia alcuni di questi vincoli potranno essere espressi direttamente altri attraverso l'utilizzo di metodologie separate.

Questo lavoro ha come scopo la traduzione di 4 esami del corso di Base di Dati. Per ogni esame si dovrà esprimere il diagramma ER compreso di vincoli nell'ontologia e successivamente valutare le query dell'esame sull'ontologia risultante.

Dalle specifiche del lavoro sono stati scelti:

- DL-Lite come linguaggio per la rappresentazione di ontologie;
- Sintassi funzionale per DL-Lite;
- Linguaggio sparSQL per le query sull'ontologia e per la rappresentazione dei vincoli non esprimibili;

Successivamente è stato utilizzato un tool grafico (QuOnto/Mastrol ToolKit)per scrivere l'ontologia in sintassi funzionale completa di asserzioni e valutare le query SparSQL.

Nel documento verrà prima descritta la metodologia generica di passaggio da schema E/R a ontologia e successivamente si applicherà la metodologia ai 4 casi di studio.

Linguaggi per ontologie

Le ontologie

il termine ontologia è mutuato dalla filosofia e si riferisce alla scienza della descrizione del mondo ed in particolare di tutte le entità che ne fanno parte compresa la maniera con la quale esse sono correlate tra loro. Attraverso le ontologie possiamo rappresentare un mondo più simile alla realtà in cui le informazioni possono essere incomplete o mancanti. Questo aspetto si allontana molto dal concetto di rigidità dei schemi finora utilizzati per la rappresentazione del mondo di interesse(E/R, UML,…)

Il linguaggio OWL

Il linguaggio OWL (Web Ontology Language) è lo standard attualmente proposto dal W3C (www.w3.org) per la definizione di ontologie per il web semantico. Sviluppato a partire da DAML+OIL (un linguaggio con logica SHIQ, a sua volta basato sui linguaggi OIL e DAML-ONT), OWL prevede tre livelli di complessità crescente:

- OWL Lite, semplice da usare e implementare ma scarsamente espressivo. Questo sottoinsieme di OWL è scomparso definitivamente dall'introduzione di OWL 2.0.
- OWL DL, che implementa la logica descrittiva SHOIN(Dn), abbastanza espressivo ma comunque decidibile e dotato di procedure di ragionamento di complessità nota, approfonditamente studiate e ormai ben ottimizzate;
- OWL Full, che consente di definire rappresentazioni che vanno al di là della logica predicativa del primo ordine; OWL Full è molto espressivo ma non decidibile e quindi problematico dal punto di vista della meccanizzazione del ragionamento.

Il linguaggio utilizzato per la rappresentazione delle ontologie in questo lavoro è DL-Lite. Questo ha lo stesso potere espressivo di OWL DL con la differenza che tramite DL-Lite si può esprimere il costrutto di modellazione degli attributi di ruolo che non possono essere espressi in OWL DL.

Un'ontologia può includere le descrizioni delle classi, proprietà e delle loro istanze. Data un'ontologia, la semantica formale di OWL specifica come derivare le sue conseguenze logiche, ovvero i fatti che non sono presenti letteralmente nell'ontologia ma che possono essere derivati logicamente dalla semantica: questa rappresenta la capacità di ragionamento dell'ontologia. Grazie alla logica utilizzata per esprimere il mondo di interesse, si possono fare deduzioni per ovviare ad eventi di incompletezza d'informazione e/o mancanza di dati.

Come si evince dalla figura sottostante, OWL è costruito sopra RDF che a sua volta si appoggia sulla sintassi XML e di conseguenza eredita tutte le caratteristiche di questi ultimi.

Una base di conoscenza di DL è tipicamente composta di due componenti, una "**Tbox**" ed una "**Abox**".

La **Terminological box** rende disponibile ed accessibile la rappresentazione formale del modello concettuale di un frammento di realtà contenendo quegli enunciati terminologici assunti come veri (livello intensionale della conoscenza).

L'**Assertion box** rende invece disponibile ed accessibile la rappresentazione formale del modello concreto di un frammento di realtà: essa contiene conoscenze fattuali sottoforma di asserzioni (livello estensionale della conoscenza). Al contrario delle basi di dati, la semantica dell'Abox è compatibile con una situazione di *conoscenza parziale*, ovvero: tutto ciò che è contenuto nell'Abox è vero mentre tutto ciò che non vi è contenuto non è né vero né falso ma semplicemente non si conosce. Questo e' in netta contrapposizione con la semantica adottata nelle basi di dati, in cui si tiene conto dell' *assunzione del mondo chiuso*, la quale afferma che tutto quello che non e' contenuto nell'ABox e' necessariamente falso.

La sintassi funzionale

Per la sua natura DL-Lite, appoggiandosi a RDF e quindi XML prevede una sintassi molto complicata da scrivere per un utente. Per questo è stato creato un formalismo più funzionale e mnemonico per l'utente,

ovvero la sintassi funzionale. Questa eredita tutti i costrutti già definiti per OWL DL ed aggiunge gli statement per la modellazione degli attributi di ruolo.

In generale le due sintassi riconosciute per DL-Lite sono la sintassi tedesca e quella funzionale.

- La **sintassi tedesca,** prevede la descrizione della TBOX attraverso l'utilizzo di costrutti logici/matematici. Questo tipo di sintassi, nonostante rappresenti la vera sintassi per esprimere ontologie in DL-Lite, viene tradotta in pesanti documenti xml da far parsare ai software di interpretazione di ontologie. Per TBOX molto medie/grandi questo documento XML è praticamente impossibile da scrivere manualmente e quindi c'è bisogno di utilizzare dei software per automatizzare il processo.

- Per far fronte ai problemi sopra elencati della sintassi tedesca, è stata realizzata una sintassi che si avvicinasse molto alla maniera di operare dell'utente. Questa sintassi utilizza costrutti testuali che richiamano all'utilità di ciò che si vuole esprimere. In questa maniera anche TBOX molto grandi possono essere scritte in maniera relativamente semplice dall'utente. La TBOX risultante sarà data in pasto ad un software che all'interno avrà un ragionatore che tradurrà ogni statement di sintassi funzionale in un'espressione in logica descrittiva in maniera del tutto trasparente.

Riporto uno stralcio della dichiarazione di una TBOX in XML:

```
<?xml version="1.0"?>
<!DOCTYPE ontology SYSTEM "../../DTDs/TBox.dtd">
<ontology>
      <alphabet>
            <atomicC>Prenotazione</atomicC>
        <atomicC>Collettiva</atomicC>
            <atomicC>Singola</atomicC>
            <atomicR>Effettua</atomicR>
      </alphabet>
      <tbox>
            <!-- functionality constraints -->
            <funct>
                  <basicR dir="inverse">
                        <atomicR>Effettua</atomicR>
                  </basicR>
            </funct>
                  <funct>
                  <atomicRA>data</atomicRA>
            </funct>
          <!-- inclusion assertions -->
            <inclusionAssertion>
                  <basicC>
                        <atomicC>Collettiva</atomicC>
                  </basicC>
```

Mentre in sintassi funzionale:

Class(Prova): per indicare la presenza nell'ontologia del concetto Prova.

SubClassOf(Studente Persona): per indicare che il concetto Studente è in ISA con il concetto Persona.

ObjectMinCardinality (1 Relation): per indicare la cardinalità minima tra i ruoli, ovvero il quantificatore esistenziale.

Dagli esempi possiamo notare come l'associazione di costrutti mnemonici e rappresentativi applicati alla fintassi funzionale possa migliorare la scrittura dell'ontologia.

Interrogare ontologie

Come abbiamo visto finora, le ontologie sono metodologie per descrivere domini di interesse e in quanto tali deve esistere una maniera per interrogare queste strutture, così come esiste per le basi di dati. Mentre nelle basi di dati si ha l'assunzione implicita di CWA (Closed World Assumpion) ovvero nelle interrogazioni si considera vero quello che è presente sulla base di dati e falso quello che non c'è. Nelle ontologie invece l'assunzione implicita è di OWA (Open World Assumpion) ovvero si considera vero tutto ciò che è asserito e quello che non è asserito si considera ignoto (ovvero non noto).

Esempio:

Person

Name
Paolo
Maria
Luisa

Student

Name
Luca
Maria

Query:

$q(x):- (Person(x) \wedge \neg Student(x))$

Answer in CWA case: {Paolo, Luisa}
Answer in OWA case : { }

Come possiamo osservare, con l'assunzione di CWA noi consideriamo soltanto Luca e Maria appartenenti alla relazione Studente e quindi possiamo rispondere alla domanda che le uniche Persone che non sono Studenti sono effettivamente Paolo e Luisa. Se passo ad un'assunzione di tipo OWA, quindi assumendo il metodo di ragionamento delle ontologie, la risposta alla domanda precedente non diventa più ovvia. Infatti nell'OWA siamo sicuri che Maria è una persona ed è uno studente, come siamo sicuri del fatto che Luca è uno studente ma non possiamo asserire nulla di certo su Luisa e Paolo. Loro sono presenti nella relazione persona ma non in studente quindi sappiamo con certezza che sono persone ma non possiamo dire nulla sulla loro appartenenza alla relazione studente.

Su queste assunzioni il linguaggio SQL, che è di tipo FOL – First Order Logic, se utilizzato per interrogare ontologie è indecidibile: questo perché, nell'assunzione OWA potrà esistere un modello per cui una certa formula è vera ed uno nel quale la stessa formula è giudicata falsa. Chiaramente questa incertezza comporta l'impossibilità di valutare le query SQL su ontologie.

Per far fronte a questo problema si è pensato di considerare un sottoinsieme (in termini di potere espressivo) della FOL: le CQ – Conjuntive Query e le UCQ - Union Conjuntive Query. Questo linguaggio però si è dimostrato troppo limitativo per poter fare interrogazioni complesse sulle ontologie. La mancanza maggiore è l'impossibilità di utilizzare l'operatore NOT nelle query ed il quantificatore universale. Se però quest'ultima è una limitazione fittizia in quanto con il quantificatore esistenziale ed la negazione possiamo ricostruire l'universale ma la mancanza del NOT è insormontabile. Questo si riflette nel non poter fare interrogazioni del tipo "Restituire tutte le persone che non sono Studenti" che sono abbastanza frequenti e di grande utilità.(Come visto nell'esempio)

Tenendo conto di questa inefficienza si è deciso di passare a linguaggi che potessero essere paragonati come potere espressivo alla FOL: i linguaggi epistemici. Questi ultimi si basano su ciò che si conosce e non su ciò che è vero. Una tipologia di linguaggio epistemico è rappresentata da EQL-LITE(UCQ). Questo ha la caratteristica di avere atomi che sono unioni di query congiuntive. Con l'introduzione delle query epistemiche possiamo effettuare query su ontologie decidibili e che hanno un potere espressivo analogo alla FOL.

Come detto anche nell'introduzione, il linguaggio di interrogazione utilizzato in questo lavoro per interrogare le ontologie è SparSQL. Questo rappresenta un'implementazione concreta di EQL-LITE(UCQ) su ontologie di tipo DL-Lite.

SparSQL quindi è decidibile e per far questo utilizza un principio leggermente diverso dall'OWA, effettua una chiusura dinamica sulla conoscenza presente sull'ontologia: questo permette di recuperare la decidibilità. Dal nome SparSQL riusciamo a differenziare Sparql e SQL.

- Sparql è il linguaggio per esprimere le unioni di query congiuntive. Il compito di Sparql in SparSQL è quello di estrarre la conoscenza dall'ontologia (attraverso sparqltables), effettuando la chiusura dinamica sopra descritta. Inoltre permette l'utilizzo di variabili molto utili per le query complesse.
- SQL è il linguaggio utilizzato per manipolare la conoscenza estratta da Sparql. Attraverso un linguaggio familiare possiamo esprimere quindi le nostre query su ontologie come se utilizzassimo una base di dati.

La struttura di query SparSQL è la seguente:

SELECT List Attributes Or Expressions

FROM (sparqltable (<Query Sparql>) alias)+

[where Conditions]

[group by List Attributes Of Grouping]

[having Aggregates Conditions]

[order by List Attributes Of Sorting]

SparSQL ci permette inoltre di effettuare particolari tipi di query, ovvero le query booleane. Questa tipologia di query si è rivelata molto utile nell'ambito di questo lavoro in quanto permetterà di verificare tutti quei vincoli dello schema ER che non possono essere espressi direttamente sulle ontologie. La struttura di questo tipo di query è la seguente:

VERIFY <condizione>

Dove la condizione può essere applicata ad una query SparSQL vista precedentemente ed è tutto ciò che può andare nella clausola where di una query SQL.

Traduzione ER in ontologie

Traduzione Entità -> Concetto

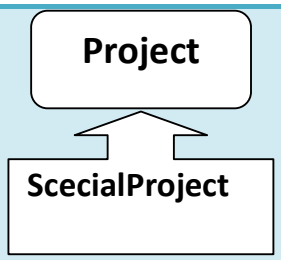
Ogni entità dello schema ER verrà tradotta in un concetto nelle ontologie. In questo esempio la traduzione avverrà così:

ER	Sintassi Tedesca	Sintassi Funzionale
		Class(Project)

In sintassi germanica, ogni concetto fa parte dell'alfabeto dell'ontologia.

Traduzione ISA tra Entità

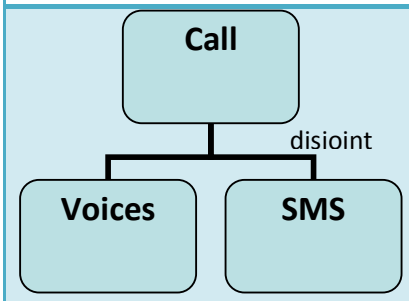
L'ISA tra Entità verrà tradotta con la stessa semantica anche nell'ontologia:

ER	Sintassi Tedesca	Sintassi Funzionale
	$SpecialProject \sqsubseteq Project$	SubClassOf(SpecialProject Project)

Ricollegandomi a quanto detto in fase di traduzione delle entità, traducendo l'ISA, l'ontologia dedurrà anche la presenza dei due concetti (Entità) Project e SpecialProject.

Traduzione del vincolo di disgiuntezza nelle generalizzazioni

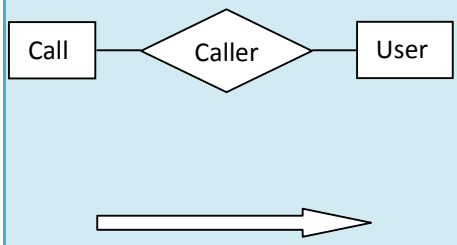
Il vincolo di disgiuntezza tra entità può essere espresso direttamente nell'ontologia attraverso un costrutto specifico.

ER	Sintassi Tedesca	Sintassi Funzionale
	$Voices \sqsubseteq \neg SMS$	DisjointClasses(Voices SMS)

N.B. Non è possibile tradurre la completezza nelle generalizzazioni. Per far rispettare il vincolo all'interno dell'ontologia verrà utilizzata una query booleana.

Traduzione Relazione -> Ruolo

Ogni relazione tra entità espressa nell'ER è considerata come ruolo nelle ontologie. Per poter essere rappresentata, ad ogni relazione deve essere assegnato un verso di percorrenza. Questo permetterà di identificare dominio e range del ruolo all'interno dell'ontologia. Infatti per ogni ruolo (relazione) saranno associati due statement: uno che indica il dominio del ruolo, ovvero il concetto sorgente della relazione(A nell'esempio) ed uno che indica il range del ruolo, ovvero il concetto target della relazione (B nell'esempio)

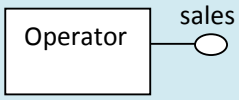
ER	Sintassi Tedesca	Sintassi Funzionale
	$\exists Caller \sqsubseteq Call$ $\exists Caller^{-} \sqsubseteq User$	ObjectPropertyDomain(Caller Call) ObjectPropertyRange(Caller User)

Nell'esempio consideriamo la relazione Caller interpretandola come: di ogni chiamata interessa l'utenza chiamante. Questa traduzione non specifica nessuna limitazione sulle molteplicità, ovvero si assume la molteplicità 0..,n per entrambi i versi del ruolo (relazione), quindi nell'esempio una chiamata può avere 0..n utenze chiamanti ed un'utenza può essere la chiamante di 0..n telefonate.

Traduzione attributi di entità -> attributi di concetto

Ogni attributo associato ad un'istanza nello schema ER viene tradotto come attributo di concetto nelle ontologie. Questo avviene utilizzando due statement ovvero uno per la definizione del range che rappresenta l'insieme dei valori che può assumere l'attributo ed uno per indicare il dominio dell'attributo stesso. Quest'ultimo non è obbligatorio ma se lo indichiamo esplicitamente andiamo ad indicare che l'attributo sales appartiene soltanto al concetto Operator ed a nessun altro nell'ontologia. L'indicazione del dominio deve essere omessa nel caso in cui nel diagramma E/R si riutilizzino i nomi degli attributi per indicare che entità distinte abbiamo lo stesso attirbuto.

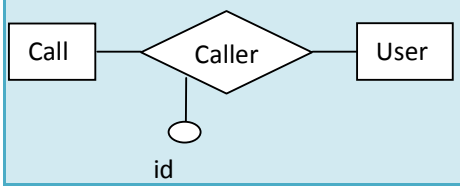
Nell'ontologia, ogni attributo di concetto che appartiene esclusivamente a quel concetto, ha come dominio il concetto stesso.

ER	Sintassi Tedesca	Sintassi Funzionale
	**range** (sales) ⊑ xsd:float **domain** (sales) ⊑ Operator	DataPropertyRange (sales rdf:float) DataPropertyDomain (sales Operator)

Nell'esempio consideriamo l'entità Operatore con l'attributo vendite. Nell'ontologia diventerà un attributo di concetto con range float e dominio Operatore. Questo permette all'ontologia di capire che l'attributo vendite appartiene al concetto Operatore.

Traduzione attributi di relazione -> attributi di ruolo

Ogni attributo associato ad una relazione nello schema ER viene tradotto come attributo di ruolo nell'ontologia. Questo avviene, analogamente per quanto detto per gli attributi di concetto, utilizzando due statement ovvero uno per la definizione del range che indica i valori che può assumere l'attributo ed uno per indicare il dominio dell'attributo stesso. Questo perché nell'ontologia, ogni attributo di ruolo ha come dominio il ruolo stesso.

ER	Sintassi Tedesca	Sintassi Funzionale
	**range** (id) ⊑ xsd:integer **domain** (id) ⊑ Caller	DataProperty2Range (id rdf:integer) DataProperty2Domain (id Caller)

Nell'esempio consideriamo la relazione Chiamante con l'attributo id. Nell'ontologia diventerà un attributo di ruolo con range integer e dominio il ruolo stesso ovvero Caller. Questo permette all'ontologia di capire che l'attributo id appartiene al ruolo Caller.

Traduzione cardinalità delle relazioni

Nello schema ER, ad ogni relazione viene associato un vincolo di cardinalità. Questo può non essere indicato e quindi considerare imprimente 0..n, oppure può essere specificato esplicitamente. Nell'ontologia non si possono rappresentare tutte le cardinalità esprimibili nello schema ER bensì soltanto la cardinalità di tipo 0..n, 1..n ,0..1, 1..1. Per tutti gli altri tipi si devono formulare query booleane che ne verifichino la consistenza.

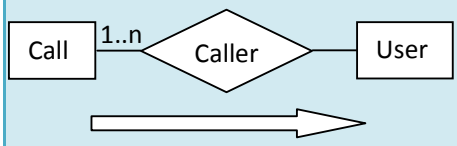
Cardinalità 0..n

Con la dichiarazione dei ruoli nota precedentemente, si definisce implicitamente questo tipo di molteplicità.

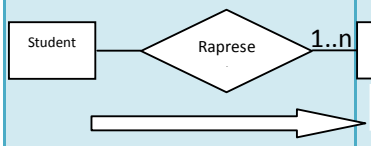
Cardinalità 1..n

Per indicare ruoli con cardinalità minima uguale a 1 si devono usare diversi modi a seconda sella posizione della molteplicità.

Cardinalità espressa nel concetto sorgente del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	$Call \sqsubseteq \exists\ Caller$	SubClassOf(Call ObjectMinCardinality(1 Caller)) Oppure SubClassOf (Call ObjectSomeValueFrom(Caller owl:anyType)

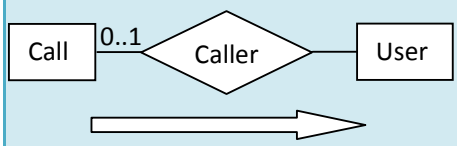
Cardinalità espressa nel concetto terget del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	$\exists\ Rapresents^{-} \sqsubseteq Student$	SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(Rapresents)) Student) Oppure SubClassOf(ObjectSomeValueFrom(InverseObjectPropertyOf(Rapresents) owl:anyType) Student)

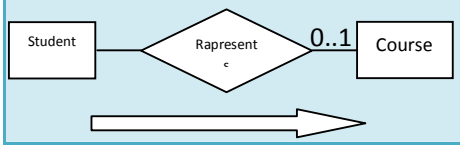
Cardinalità 0..1

Per indicare ruoli con cardinalità massima uguale a 1 si devono usare diversi modi a seconda sella posizione della molteplicità.

Cardinalità espressa nel concetto sorgente del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	(**funct** Caller)	FunctionalObjectProperty(Caller)

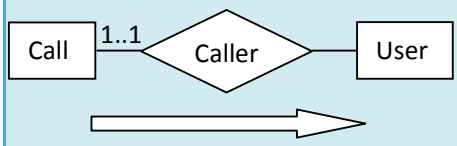
Cardinalità espressa nel concetto terget del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	(**funct** Rapresents $^-$)	FunctionalObjectProperty(InverseObjectPropertyOf(Rapresents))

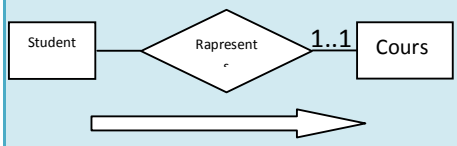
Cardinalità 1..1

Per indicare ruoli con cardinalità uguale a 1 si devono usare diversi modi a seconda sella posizione della molteplicità.

Cardinalità espressa nel concetto sorgente del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	$\text{Call} \sqsubseteq \exists \text{ Caller}$ (**funct** Caller)	SubClassOf(Call ObjectMinCardinality(1 Caller)) FunctionalObjectProperty(Caller) Oppure SubClassOf(ObjectSomeValueFrom(Caller owl:anyType) Student)

Cardinalità espressa nel concetto terget del ruolo:

ER	Sintassi Tedesca	Sintassi Funzionale
	$\exists \text{ Rapresents}^- \sqsubseteq \text{Student}$ (**funct** Rapresents $^-$)	SubClassOf(ObjectMinCardinality (1 InverseObjectPropertyOf(Rapresents)) Student) FunctionalObjectProperty(InverseObjectPropertyOf(Rapresents))

Traduzione cardinalità degli attributi di concetto

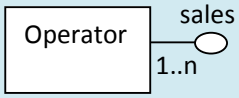
Come indicato in fase di descrizione degli attributi di concetto, l'ontologia li considera con una molteplicità pari a 0..n. In ER l'assunzione implicita che si fa nella dichiarazione degli attributi di Entità è 1..1 ovvero di ogni attributo esiste esattamente un valore. Per riprodurre quindi questa condizione bisogna imporre nell'ontologia la cardinalità massima e minima uguale a uno, come avviene per i ruoli. In generale quindi possiamo indicare attributi con molteplicità: 0..n, 1..n, 0..1, 1..1.

Cardinalità 0..n

Con la semplice dichiarazione degli attributi, si definisce implicitamente questo tipo di molteplicità.

Cardinalità 1..n

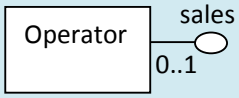
Per indicare attributi con cardinalità minima uguale a 1 si deve usare:

ER	Sintassi Tedesca	Sintassi Funzionale
	Operator ⊑ **domain** (sales)	SubClassOf (Operator DataSomeValueFrom (sales xsd:anyType)) Oppure SubClassOf (Operator dataMinCardinality(1 sales)

In questo caso stiamo traducendo attributi che nell'ER erano considerati multivalore.

Cardinalità 0..1

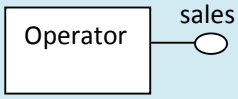
Per indicare attributi con cardinalità massima uguale a 1 si deve usare:

ER	Sintassi Tedesca	Sintassi Funzionale
	(**funct** sales)	FunctionalDataProperty (sales)

In questo caso stiamo traducendo attributi che nell'ER erano considerati opzionali.

Cardinalità 1..1

Per indicare attributi con cardinalità massima e minima uguale a 1 si deve usare:

ER	DLite A	Sintassi Funzionale
	$\text{Operator} \sqsubseteq \textbf{domain}\ (\text{sales})$ (**funct** sales)	SubClassOf(Operator dataSomeValueFrom(sales xsd:anyType)) FunctionalDataProperty (sales)

Questa è la maniera per tradurre generalmente gli attributi di entità nello schema ER.

Traduzione cardinalità degli attributi di ruolo

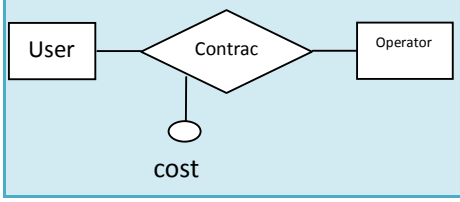
Come indicato in fase di dichiarazione degli attributi di ruolo, l'ontologia li considera con una molteplicità pari a 0..n. In ER l'assunzione implicita che si fa nella dichiarazione degli attributi di Relazione è 1..1 ovvero di ogni attributi esiste esattamente un valore. Per riprodurre quindi questa condizione bisogna imporre nell'ontologia la cardinalità massima e minima uguale a uno, come avviene per i ruoli. In generale quindi possiamo indicare attributi con molteplicità: 0..n, 1..n, 0..1, 1..1.

Cardinalità 0..n

Con la semplice dichiarazione degli attributi, si definisce implicitamente questo tipo di molteplicità.

Cardinalità 1..n

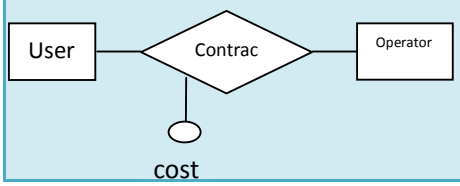
Per indicare attributi con cardinalità minima uguale a 1 si deve usare:

ER	Sintassi Tedesca	Sintassi Funzionale
	$\text{Contract} \sqsubseteq \textbf{domain}\ (\text{cost})$	SubObjectPropertyOf(Contract Data2SomeValueFrom(cost xsd:anyType)) Oppure SubObjectPropertyOf(Contract Data2MinCardinality(1 cost))

In questo caso stiamo traducendo attributi che nell'ER erano considerati multivalore.

Cardinalità 0..1

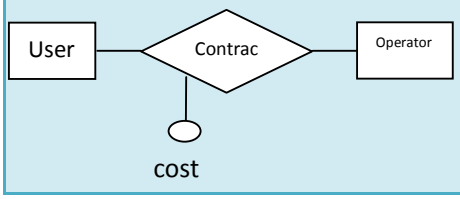
Per indicare attributi con cardinalità massima uguale a 1 si deve usare:

ER	Sintassi Tedesca	Sintassi Funzionale
	(**funct** cost)	FunctionalDataProperty2(cost)

In questo caso stiamo traducendo attributi che nell'ER erano considerati opzionali.

Cardinalità 1..1

Per indicare attributi con cardinalità massima e minima uguale a 1 si deve usare:

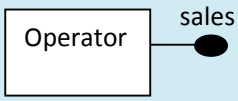
ER	DLite A	Sintassi Funzionale
	$\text{Contract} \sqsubseteq \textbf{domain}\ (\text{cost})$ (**funct** cost)	SubObjectPropertyOf(Contract data2SomeValueFrom(cost xsd:anyType)) FunctionalDataProperty2(cost)

Questa è la maniera per tradurre generalmente gli attributi di relazione nello schema ER.

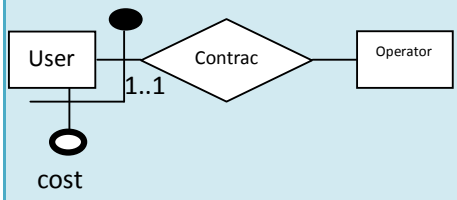
Traduzione vincoli di chiave

Tutti i vincoli di chiave possono essere espressi direttamente in Dl-LiteA . La rappresentazione di questi vincoli ha come scopo imporre la semantica del concetto di chiave nel diagramma ER. Ovvero per ogni chiave non possono esistere due valori uguali. La chiave può essere formata sia da attributi (uno o più di uno) sia da relazioni (una o più di una) sia dall'accoppiata attributi relationi.

Vicoli di chiave su attributi dell'Entità/ Relazione

ER	Sintassi Funzionale
	KeyFor(sales Operator)

Vicoli di chiave esterna.

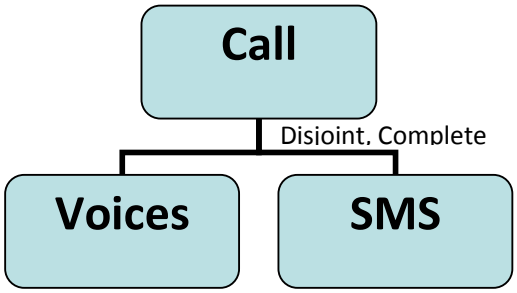
ER	Sintassi Funzionale
	KeyFor(cost, Contract User)

Traduzione Vincoli non esprimibili direttamente

Come accennato anche in precedenza alcuni vincoli che possiamo esprimere direttamente sullo schema E/R non possono essere espressi nelle ontologie in DlLite. In particolare la traduzione del "complete" nelle generalizzazioni e le molteplicità sulle relazioni diverse da quelle illustrate ovver 0..n, 1..n, 0..1, 1..1.

Per tradurre questi vincoli e quindi fornire una maniera per verificare che le asserzioni nell'ontologia siano coerenti con questi vincoli, si devono scrivere delle query booleane a verifica di questi.

Query booleana vincolo di completezza



L'obiettivo è verificare che tutti gli individui di Voices e di SMS formino tutti gli indivisui di Call e viceversa. Quindi una maniera per implementare questo è verificare che la differenza insiemistica tra Call / (Voices U SMS) sia vuota.

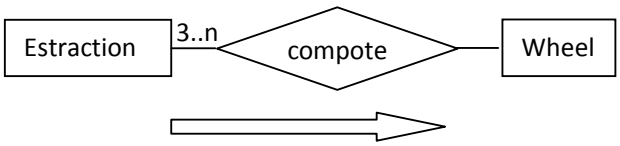
La query sparSQL booleana è la seguente:

```
VERIFY not exists(
SELECT  Uno.x
FROM sparqltable( SELECT ?x
             WHERE{
                  ?x rdf:type 'Calll'.
            } )Uno
WHERE Uno.x not in (

SELECT  Due.x
FROM sparqltable( SELECT ?x
              WHERE{
                  ?x rdf:type 'SMS'.
            })Due
UNION
SELECT  Tre.x
FROM sparqltable( SELECT ?x
              WHERE{
                  ?x rdf:type 'Voices'.
            })Tre
      ))

)
```

Query booleana vincolo di molteplicità diversa da 0 o 1



Per verificare questo vincolo si è deciso di scrivere una query condizionale ovvero che si fa restituire tutti gli individui che partecipano alla relazione di cui si vuole testare la molteplicità raggruppati per il campo chiave, poi conta le occorrenza di ognuno e seleziona quelli che non soddisfano la giusta molteplicità.

Nell'esempio verifichiamo se ogni estrazione è composta da almeno tre ruote. La query booleana è la seguente:

```
VERIFY not exists(
SELECT Uno.x
      FROM sparqltable( SELECT ?x ?w
```

```
      WHERE{
            ?x rdf:type 'Extraction'.
            ?x :Compote ?w.
            ?w rdf:type 'Wheel'.
      })Uno
GROUP BY (Uno.x)
HAVING COUNT(*) < 3

)
```

Caso di studio n.1 Esame Base di Dati 07-04-2005

In questa sezione ci occuperemo di analizzare la traduzione dello schema E/R del compito di base di dati del 07-04-2005. Sulla base dell'ontologia risultante verranno tradotte le query del compito d'esame in query sparSQL.

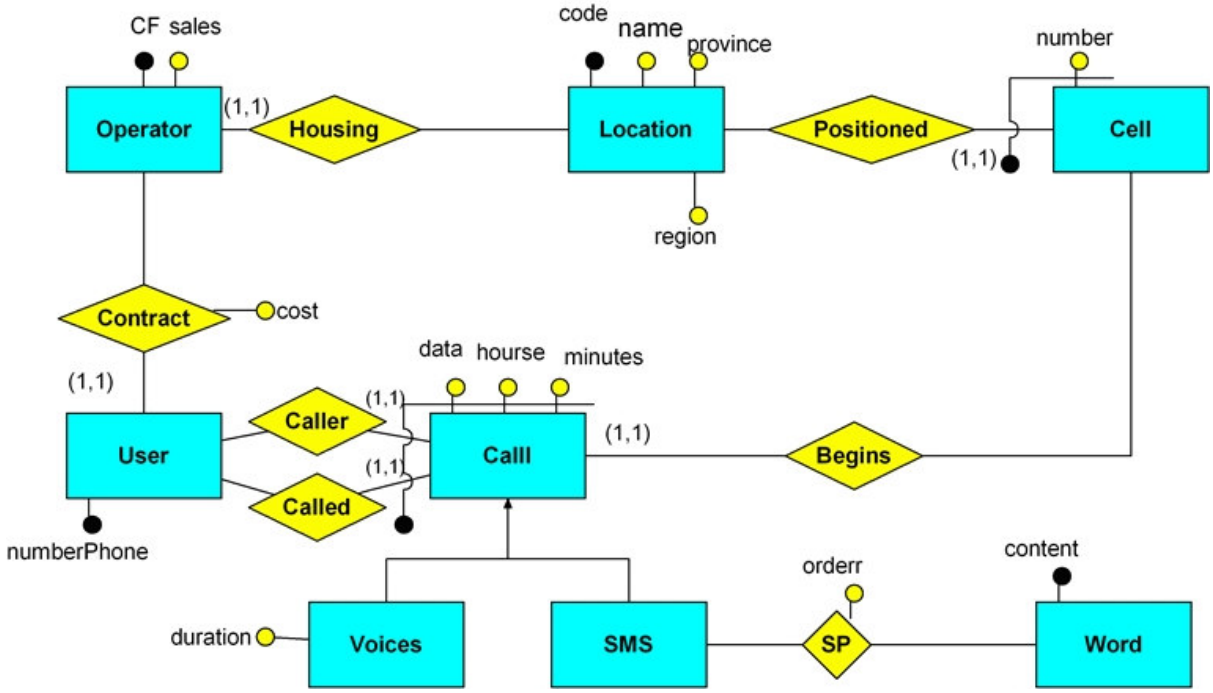
Traduzione schema ER

A seguito di questa traduzione andremo a comporre la TBox dell'ontologia.

Questo è il testo del compito d'esame:

"Si richiede di progettare lo schema concettuale Entità-Relazione di un'applicazione relativa ai dati di interesse per un insieme di operatori nazionali di telefonia mobile. Di ogni operatore telefonico interessa il codice fiscale (identificatore), il fatturato annuale e la località della sede legale. Di ogni località interessa il codice (identificatore), il nome, la provincia e la regione. Di ogni utenza interessa l'operatore telefonico (uno ed uno solo) con cui l'utenza stessa ha stipulato il contratto, il numero telefonico dell'utenza stessa (identificatore), ed il costo per ogni secondo di conversazione previsto dal contratto. Di ogni utenza interessano anche le telefonate fatte dall'utenza stessa, e di ogni telefonata, oltre all'utenza chiamante, interessa l'utenza chiamata, e la data, l'ora ed il minuto in cui è iniziata. Si noti che una stessa utenza non può iniziare più di una chiamata nello stesso minuto della stessa ora della stessa data. Di ogni telefonata interessa anche la cella che ha gestito l'inizio della telefonata, dove ogni cella è identificata da un numero unico nell'ambito della località in cui si trova. Ci sono due e solo due tipi di telefonate: telefonate di tipo "fonia" e telefonate di tipo "sms". Per le telefonate di tipo "fonia" interessa la durata in secondi, mentre per le telefonate di tipo "sms" interessano le parole di cui è formato il messaggio inviato, con l'ordine delle parole nel messaggio. Per tutti gli operatori e per tutti i contratti, il costo di una telefonata di tipo "sms" è calcolato contando ogni parola inviata come un secondo di conversazione."

A seguito dell'analisi delle specifiche, è stato prodotto il seguente schema E/R:



Traduzione ISA, generalizzazioni

Il concetto Voices e SMS sono in ISA con Call quindi verranno tradotti con:

SubClassOf(Voices Calll)

SubClassOf(SMS Calll)

Inoltre per indicare il vincolo di disjoint scriveremo:

DisjointClasses(Voices SMS)

Come detto anche in precedenza, non è possibile esprimere direttamente il vincolo di completezza tra Voices e SMS. Per mantenere la consistenza sui dati, verrà utilizzata una query booleana che indicheremo in seguito.

Quindi ricapitolando:

SubClassOf(Voices Calll)

SubClassOf(SMS Calll)

DisjointClasses(Voices SMS)

Definizione ruoli

Dalla definizione dei ruoli dell'ontologia ed in particolare dalla definizione del loro dominio e range, l'ontologia dedurrà la presenza di tutti i concetti presenti nello schema ER. Non essendoci entità isolate, riusciamo a coprirle tutte.

Per semplicità è stata assegnata la direzione di interpretazione delle relazioni sullo schema ER considerando come sorgente l'entità a cui apparteneva la molteplicità diversa da 0..n. Questo è stato possibile perché non ci sono relazioni con molteplicità diverse da 0..n in entrambi i lati.

La traduzione dei ruoli quindi sarà questa:

ObjectPropertyDomain(SP SMS)

ObjectPropertyRange(SP Word)

ObjectPropertyDomain(Begins Calll)

ObjectPropertyRange(Begins Cell)

ObjectPropertyDomain(Caller Calll)

ObjectPropertyRange(Caller User)

ObjectPropertyDomain(Called Calll)

ObjectPropertyRange(Called User)

ObjectPropertyDomain(Contract User)

ObjectPropertyRange(Contract Operator)

ObjectPropertyDomain(Positioned Cell)

ObjectPropertyRange(Positioned Location)

ObjectPropertyDomain(Housing Operator)

ObjectPropertyRange(Housing Location)

Come detto in fase di descrizione generale, ogni ruolo ha due statement: uno che ne indica il domain mentre un altro che ne indica il range e questi dipendono strettamente dalla direzione di interpretazione scelta sullo schema E/R.

Definite in questa maniera, i ruoli hanno tutti come cardinalità 0..n. Per restringerle bisogna utilizzare altri costrutti.

Cardinalità minima dei ruoli

In questa fase andiamo a restringere la cardinalità minima di tutti quei ruoli che hanno come molteplicità 1..x.

SubClassOf(Calll ObjectMinCardinality(1 Begins))

SubClassOf(Calll ObjectMinCardinality(1 Caller))

SubClassOf(Calll ObjectMinCardinality(1 Called))

SubClassOf(Celll ObjectMinCardinality(1 Positioned))

SubClassOf(Operator ObjectMinCardinality(1 Housing))

SubClassOf(User ObjectMinCardinality(1 Contract))

Cardinalità massima dei ruoli

In questa fase andiamo a restringere la cardinalità massima di tutti quei ruoli che hanno come molteplicità x..1. Unendo i costrutti precedenti con quelli di seguito elencati andremo a realizzare tutte i ruoli con le molteplicità uguali a 1..1

FunctionalObjectProperty(Begins)

FunctionalObjectProperty(Positioned)

FunctionalObjectProperty(Housing)

FunctionalObjectProperty(Caller)

FunctionalObjectProperty(Called)

FunctionalObjectProperty(Contract)

Definizione attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataPropertyRange(data rdf:date)

DataPropertyRange(hours rdf:integer)

DataPropertyRange(minutes rdf:integer)

DataPropertyRange(content rdf:string)

DataPropertyRange(number rdf:string)

DataPropertyRange(code rdf:string)

DataPropertyRange(name rdf:string)

DataPropertyRange(province rdf:string)

DataPropertyRange(region rdf:string)

DataPropertyRange(cf rdf:string)

DataPropertyRange(sales rdf:float)

DataPropertyRange(numberPhone rdf:string)

Definizione dominio attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale concetto l'attributo va a descrivere.

DataPropertyDomain(data Calll)

DataPropertyDomain(hours Calll)

DataPropertyDomain(minutes Calll)

DataPropertyDomain(duration Voices)

DataPropertyDomain(content Word)

DataPropertyDomain(number Cell)

DataPropertyDomain(code Location)

DataPropertyDomain(name Location)

DataPropertyDomain(province Location)

DataPropertyDomain(region Location)

DataPropertyDomain(sales Operator)

DataPropertyDomain(cf Operator)

DataPropertyDomain(numberPhone User)

Come detto anche per i ruoli, gli attributi così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi.

SubClassOf(Calll dataSomeValueFrom(data xsd:anyType))

SubClassOf(Calll dataSomeValueFrom(hours xsd:anyType))

SubClassOf(Calll dataSomeValueFrom(minutes xsd:anyType))

SubClassOf(Voices dataSomeValueFrom(duration xsd:anyType))

SubClassOf(Word dataSomeValueFrom(content xsd:anyType))

SubClassOf(Cell dataSomeValueFrom(number xsd:anyType))

SubClassOf(Location dataSomeValueFrom(code xsd:anyType))

SubClassOf(Location dataSomeValueFrom(name xsd:anyType))

SubClassOf(Location dataSomeValueFrom(province xsd:anyType))

SubClassOf(Location dataSomeValueFrom(region xsd:anyType))

SubClassOf(Operator dataSomeValueFrom(cf xsd:anyType))

SubClassOf(Operator dataSomeValueFrom(sales xsd:anyType))

SubClassOf(User dataSomeValueFrom(numberPhone xsd:anyType))

Cardinalità massima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty(data)

FunctionalDataProperty(hours)

FunctionalDataProperty(minutes)

FunctionalDataProperty(duration)

FunctionalDataProperty(content)

FunctionalDataProperty(number)

FunctionalDataProperty(code)

FunctionalDataProperty(name)

FunctionalDataProperty(province)

FunctionalDataProperty(region)

FunctionalDataProperty(cf)

FunctionalDataProperty(sales)

FunctionalDataProperty(numberPhone)

Definizione attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataProperty2Range(cost rdf:float)

DataProperty2Range(orderr rdf:integer)

Definizione dominio attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale ruolo l'attributo va a descrivere.

DataProperty2Domain(orderr SP)

DataProperty2Domain(cost Contract)

Come detto anche per gli attributi di concetto, anche gli attributi di ruolo così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi.

SubObjectPropertyOf(SP data2SomeValueFrom(orderr xsd:anyType))

SubObjectPropertyOf(Contract data2SomeValueFrom(cost xsd:anyType))

Cardinalità massima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty2(orderr)

FunctionalDataProperty2(cost)

Vincoli di chiave

Per ogni vincolo di chiave definita all'interno dello schema E/R esprimiamo il vincolo nell'ontologia.

KeyFor(data,hours,minutes,Caller,Called Calll)

KeyFor(content Word)

KeyFor(number,Positioned Cell)

KeyFor(code Location)

KeyFor(cf Operator)

KeyFor(numberPhone User)

Le chiavi esterne si differenziano da quelle primarie perché hanno, separato da virgola, anche il ruolo che contribuisce all'identificazione oltre eventualmente all'attributo.

Query in sparSQL

Andiamo adesso ad esprimere le query sul indicate nel testo sull'ontologia risultante.

- Query 1:

  Dato un operatore telefonico P, calcolare la data, l'ora ed il minuto di inizio di tutte le telefonate che hanno chiamato utenze che hanno il contratto con P.

  In sparSQL:

```
SELECT Calll.d, Calll.h, Calll.m
FROM sparqltable( SELECT ?d ?h ?m
        WHERE{
            ?x rdf:type 'Calll'.
            ?x :data ?d.
            ?x :hours ?h.
            ?x :minutes ?m.
            ?x :Called ?z.
            ?z rdf:type 'User'.
            ?z :Contract ?o.
            ?o rdf:type 'Operator'.
            ?o :cf 'wind'.
        }
 )Calll
```

- Query 2:

  Restituire il codice di ogni utenza U che ha fatto telefonate solo ad utenze che hanno il contratto con l'operatore con cui U ha il contratto.

  In sparSQL:

```
SELECT Calll.nu
FROM sparqltable( SELECT ?nu ?codice ? codicedue
        WHERE{
            ?x rdf:type 'Calll'.
            ?x :Caller ?z.
            ?z rdf:type 'User'.
            ?z :numberPhone ?nu.
            ?z :Contract ?o.
            ?o rdf:type 'Operator'.
            ?o :cf ?codice.
            ?x :Called ?w.
            ?w rdf:type 'User'.
            ?w :Contract ?p.
```

```
            ?p rdf:type 'Operator'.
            ?p :cf ?codicedue.
                              }
   )Calll

WHERE Calll.codice = Calll.codicedue
```

- Query 3:

  Data un'utenza U, calcolare la spesa complessiva addebitata ad U per tutte le telefonate di tipo "sms" fatte nel 2004.

  In sparSQL:

```
SELECT  Calll.x , Calll.c*COUNT(*)
FROM sparqltable( SELECT ?nu ?str ?d ?x ?c
        WHERE{
            ?x rdf:type 'SMS'.
            ?x :data ?d.
            ?x :Caller ?z.
            ?z rdf:type 'User'.
            ?z :numberPhone ?nu.
            ?x :SP ?w.
            ?w rdf:type 'Word'.
            ?w :content ?str.
            (?z :Contract ?p) :cost ?c.
            ?p rdf:type 'Operator'.
      }
   )Calll
WHERE Calll.nu = 5556 AND Calll.d BETWEEN '2004-01-01' AND '2004-12-31'
GROUP BY Calll.x
```

Vincoli non esprimibili direttamente nell'ontologia

Per tutti quei vincoli che non possono essere espressi nell'ontologia si utilizzano delle query booleane che ne verificano le condizioni. Queste query saranno utilizzate per verificare i vincoli direttamente sulle istanze delle ontologie.

In questo caso di studio l'unico vincolo non esprimibile è rappresentato dalla completezza della generalizzazione. La query relativa è la seguente:

```
VERIFY not exists(SELECT  Uno.x
                      FROM sparqltable( SELECT ?x
                      WHERE{
                      ?x rdf:type 'Calll'.
                      } )Uno
                WHERE Uno.x not in (

                      SELECT  Due.x
                      FROM sparqltable( SELECT ?x
                            WHERE{
```

```
                ?x rdf:type 'SMS'.
                })Due
        UNION
        SELECT  Tre.x
        FROM sparqltable( SELECT ?x
                        WHERE{
                        ?x rdf:type 'Voices'.
        })Tre
)))
```

Esempio di asserzioni che permettano di testare le query:

classassertion(t01 Calll)

classassertion(t02 Calll)

classassertion(u01 User)

classassertion(u02 User)

classassertion(o01 Operator)

classassertion(s01 SMS)

classassertion(w01 Word)

classassertion(w02 Word)

dataPropertyAssertion(data t01 2004-06-29)

dataPropertyAssertion(hours t01 15)

dataPropertyAssertion(minutes t01 31)

dataPropertyAssertion(data t02 2004-07-31)

dataPropertyAssertion(hours t02 12)

dataPropertyAssertion(minutes t02 30)

dataPropertyAssertion(data s01 2004-05-11)

dataPropertyAssertion(hours s01 12)

dataPropertyAssertion(minutes s01 00)

dataPropertyAssertion(cf o01 wind)

dataPropertyAssertion(numberPhone u01 5556)

dataPropertyAssertion(numberPhone u02 5557)

objectPropertyAssertion(Called t01 u01)

objectPropertyAssertion(Called t02 u01)

objectPropertyAssertion(Caller t01 u02)

objectPropertyAssertion(Caller t02 u02)

objectPropertyAssertion(Caller s01 u01)

objectPropertyAssertion(Contract u01 o01)

objectPropertyAssertion(Contract u02 o01)

objectPropertyAssertion(SP s01 w01)

objectPropertyAssertion(SP s01 w02)

dataPropertyAssertion(content w01 ciao)

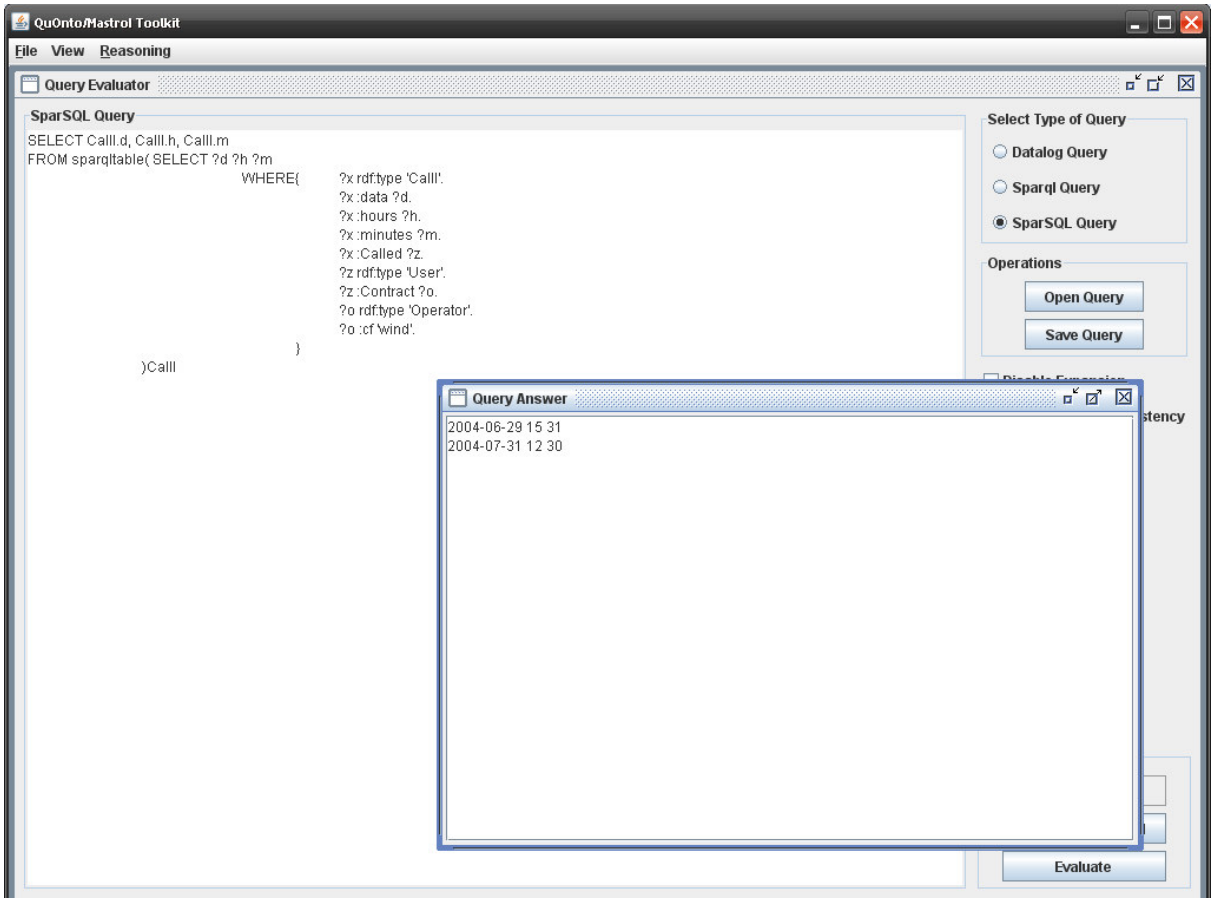
dataPropertyAssertion(content w02 emma)

dataProperty2Assertion(cost u01 o01 120)

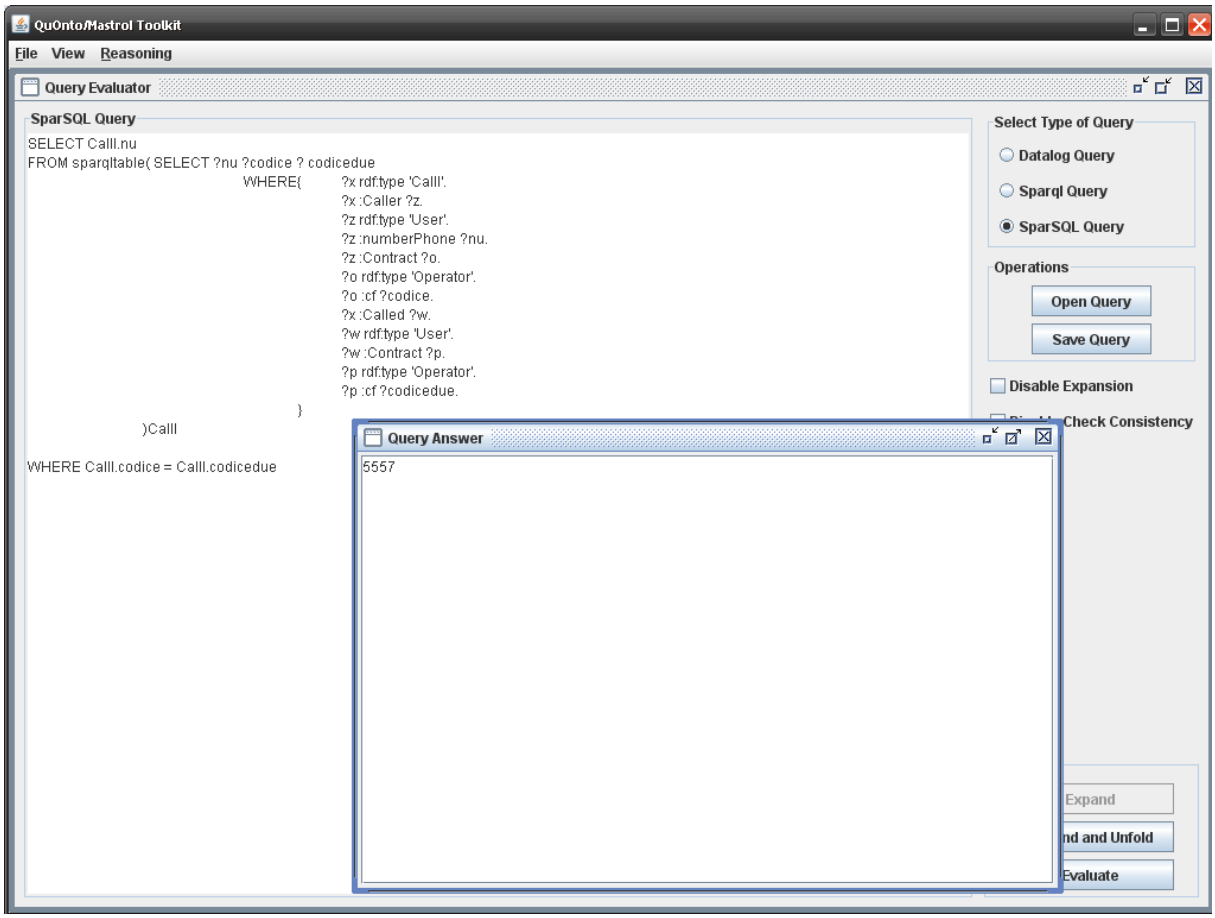
Query su QuOnto/Mastro Toolkit:

Presento una serie di screen del software utilizzato per la scrittura e l'interrogazione dell'ontologia.

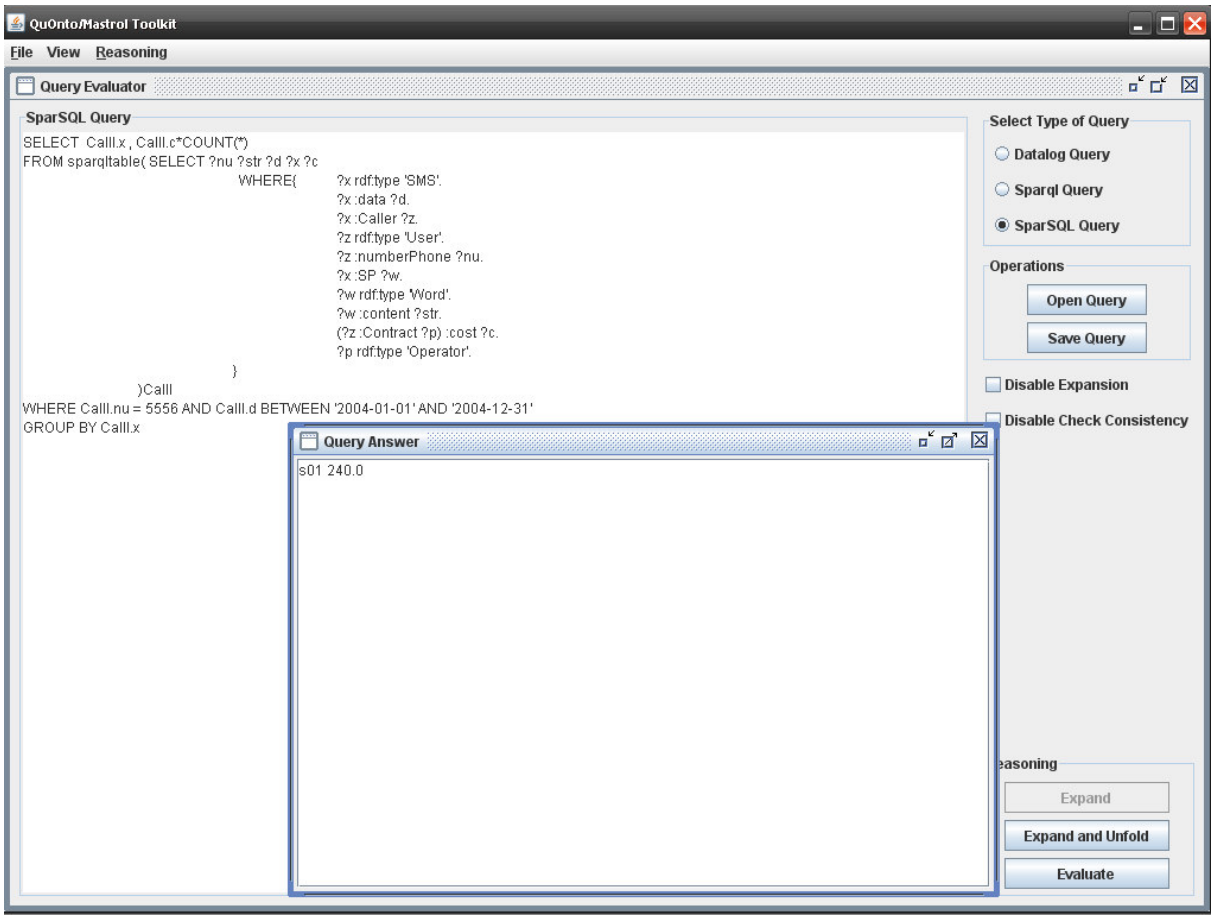
Risultato Query 1:



Risultato Query 2:



Risultato Query 3:



Caso di studio n.2 Esame Base di Dati 15-09-2005

In questa sezione ci occuperemo di analizzare la traduzione dello schema E/R del compito di base di dati del 15-09-2005. Sulla base dell'ontologia risultante verranno tradotte le query del compito d'esame in query sparSQL.

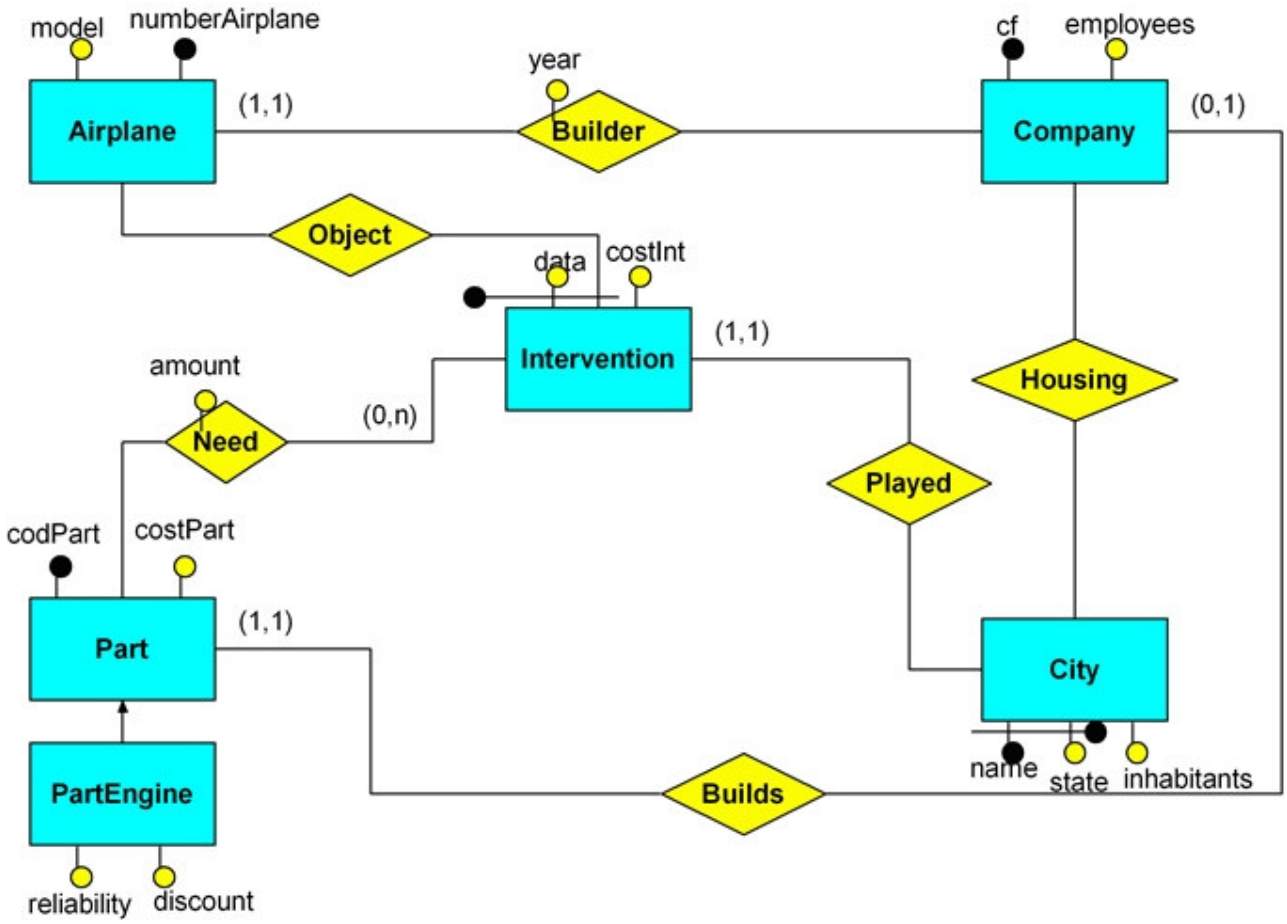
Traduzione schema ER

A seguito di questa traduzione andremo a comporre la TBox dell'ontologia.

Questo è il testo del compito d'esame:

"Si richiede di progettare lo schema concettuale Entità-Relazione di un'applicazione relativa agli interventi di manutenzione su aerei. Di ogni aereo interessa il codice (identificatore), il modello, la ditta costruttrice, e l'anno di costruzione. Di ogni ditta costruttrice interessa il codice fiscale (identificatore), la città in cui ha sede, ed il numero di dipendenti (quest'ultima informazione può però
non essere disponibile). Di ogni città interessa il nome (unico nell'ambito dello stato), lo stato ed il numero di abitanti. Gli aerei sono soggetti ad interventi di manutenzione: di ogni intervento interessa
l'aereo oggetto dell'intervento, la data, il costo di manodopera dell'intervento e la città in cui si [è svolto. Un aereo non può essere oggetto di più di un intervento di manutenzione al giorno. Un intervento di manutenzione può richiedere la sostituzione di pezzi. Di ogni pezzo interessa il codice (identificatore), il costo unitario, e la ditta costruttrice. Se il pezzo è un pezzo del motore, interessa sapere anche l'indice di affidabilità (intero positivo). Infine, per ogni pezzo oggetto di sostituzione nell'ambito di un intervento di manutenzione interessa sapere la quantità sostituita (ad esempio, in un
intervento di manutenzione può essere necessario sostituire 3 poggiatesta dell'aereo), e nel caso di pezzo di motore, lo sconto praticato sul prezzo unitario."

A seguito dell'analisi delle specifiche, è stato prodotto il seguente schema E/R:



Traduzione ISA, generalizzazioni

Il concetto PartEngine èin ISA con Part quindi verrà tradotto con:

SubClassOf(PartEngine Part)

Definizione ruoli

Dalla definizione dei ruoli dell'ontologia ed in particolare dalla definizione del loro dominio e range, l'ontologia dedurrà la presenza di tutti i concetti presenti nello schema ER. Non essendoci entità isolate, riusciamo a coprirle tutte.

Per semplicità è stata assegnata la direzione di interpretazione delle relazioni sullo schema ER considerando come sorgente l'entità a cui apparteneva la molteplicità diversa da 0..n. Questo è stato possibile perché non ci sono relazioni con molteplicità diverse da 0..n in entrambi i lati.

La traduzione dei ruoli quindi sarà questa:

ObjectPropertyDomain(Object Intervention)

ObjectPropertyRange(Object Airplane)

ObjectPropertyDomain(Need Intervention)

ObjectPropertyRange(Need Part)

ObjectPropertyDomain(Played Intervention)

ObjectPropertyRange(Played City)

ObjectPropertyDomain(Builder Airplane)

ObjectPropertyRange(Builder Company)

ObjectPropertyDomain(Housing Company)

ObjectPropertyRange(Housing City)

ObjectPropertyDomain(Builds Part)

ObjectPropertyRange(Builds Company)

Come detto in fase di descrizione generale, ogni ruolo ha due statement: uno che ne indica il domain mentre un altro che ne indica il range e questi dipendono strettamente dalla direzione di interpretazione scelta sullo schema E/R.

Definite in questa maniera, i ruoli hanno tutti come cardinalità 0..n. Per restringerle bisogna utilizzare altri costrutti.

Cardinalità minima dei ruoli

In questa fase andiamo a restringere la cardinalità minima di tutti quei ruoli che hanno come molteplicità 1..x.

SubClassOf(Airplane ObjectMinCardinality(1 Builder))

SubClassOf(Intervention ObjectMinCardinality(1 Object))

SubClassOf(Intervention ObjectMinCardinality(1 Played))

SubClassOf(Company ObjectMinCardinality(1 Housing))

SubClassOf(Part ObjectMinCardinality(1 Builds))

Cardinalità massima dei ruoli

In questa fase andiamo a restringere la cardinalità massima di tutti quei ruoli che hanno come molteplicità x..1. Unendo i costrutti precedenti con quelli di seguito elencati andremo a realizzare tutte i ruoli con le molteplicità uguali a 1..1

FunctionalObjectProperty(Builder)

FunctionalObjectProperty(Object)

FunctionalObjectProperty(Played)

FunctionalObjectProperty(Housing)

FunctionalObjectProperty(Builds)

Definizione attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataPropertyRange(numberAirplane rdf:string)

DataPropertyRange(model rdf:string)

DataPropertyRange(data rdf:date)

DataPropertyRange(costInt rdf:float)

DataPropertyRange(cf rdf:string)

DataPropertyRange(employees rdf:string)

DataPropertyRange(codPart rdf:string)

DataPropertyRange(costPart rdf:float)

DataPropertyRange(name rdf:string)

DataPropertyRange(state rdf:string)

DataPropertyRange(inhabitants rdf:integer)

DataPropertyRange(reliability rdf:string)

DataPropertyRange(discount rdf:float)

Definizione dominio attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale concetto l'attributo va a descrivere.

DataPropertyDomain(numberAirplane Airplane)

DataPropertyDomain(model Airplane)

DataPropertyDomain(data Intervention)

DataPropertyDomain(costInt Intervention)

DataPropertyDomain(cf Company)

DataPropertyDomain(employees Company)

DataPropertyDomain(codPart Part)

DataPropertyDomain(costPart Part)

DataPropertyDomain(name City)

DataPropertyDomain(state City)

DataPropertyDomain(inhabitants City)

DataPropertyDomain(reliability PartEngine)

DataPropertyDomain(discount PartEngine)

Come detto anche per i ruoli, gli attributi così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi tranne di *employees* che da specifica era considerato opzionale .

SubClassOf(Airplane dataSomeValueFrom(numberAirplane xsd:anyType))

SubClassOf(Airplane dataSomeValueFrom(model xsd:anyType))

SubClassOf(Intervention dataSomeValueFrom(data xsd:anyType))

SubClassOf(Intervention dataSomeValueFrom(costInt xsd:anyType))

SubClassOf(Company dataSomeValueFrom(cf xsd:anyType))

SubClassOf(Company dataSomeValueFrom(employees xsd:anyType))

SubClassOf(Part dataSomeValueFrom(codPart xsd:anyType))

SubClassOf(Part dataSomeValueFrom(costPart xsd:anyType))

SubClassOf(City dataSomeValueFrom(name xsd:anyType))

SubClassOf(City dataSomeValueFrom(state xsd:anyType))

SubClassOf(City dataSomeValueFrom(inhabitants xsd:anyType))

SubClassOf(PartEngine dataSomeValueFrom(reliability xsd:anyType))

SubClassOf(PartEngine dataSomeValueFrom(discount xsd:anyType))

Cardinalità massima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty(numberAirplane)

FunctionalDataProperty(model)

FunctionalDataProperty(data)

FunctionalDataProperty(costInt)

FunctionalDataProperty(cf)

FunctionalDataProperty(employees)

FunctionalDataProperty(codPart)

FunctionalDataProperty(costPart)

FunctionalDataProperty(name)

FunctionalDataProperty(state)

FunctionalDataProperty(inhabitants)

FunctionalDataProperty(reliability)

FunctionalDataProperty(discount)

Definizione attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataProperty2Range(year rdf:integer)

DataProperty2Range(amount rdf:integer)

Definizione dominio attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale ruolo l'attributo va a descrivere.

DataProperty2Domain(year Builder)

DataProperty2Domain(amount Need)

Come detto anche per gli attributi di concetto, anche gli attributi di ruolo così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi.

SubObjectPropertyOf(Builder data2SomeValueFrom(year xsd:anyType))

SubObjectPropertyOf(Need data2SomeValueFrom(amount xsd:anyType))

Cardinalità massima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty2(year)

FunctionalDataProperty2(amount)

Vincoli di chiave

Per ogni chiave definita all'interno dello schema E/R esprimiamo il vincolo nell'ontologia.

KeyFor(numberAirplane Airplane)

KeyFor(data,Object Intervention)

KeyFor(cf Company)

KeyFor(codPart Part)

KeyFor(name,state City)

Le chiavi esterne si differenziano da quelle primarie perché hanno, separato da virgola, anche il ruolo che contribuisce all'identificazione oltre eventualmente all'attributo.

Query in sparSQL

Andiamo adesso ad esprimere le query sul indicate nel testo sull'ontologia risultante.

- Query 1:

  Per ogni aereo, restituire il modello, l'anno di costruzione, e la data di tutti gli interventi di manutenzione di cui è stato oggetto.

  In sparSQL:

```
SELECT Q.mo, Q.an, Q.d
FROM sparqltable( SELECT ?mo ?d ?an
        WHERE{
            ?x rdf:type 'Airplane'.
            ?x :model ?mo.
            (?x :Builder ?z) :year ?an.
```

```
            ?z rdf:type 'Company'.
            ?i rdf:type 'Intervention'.
            ?i :Object ?x.
            ?i :data ?d.
                              }
   )Q
```

- Query 2:

  Dato un modello M di aereo, calcolare il codice di tutti i pezzi che sono stati oggetto di sostituzione in interventi di manutenzione di aerei di modello M.

  In sparSQL:

```
SELECT Q.co
FROM sparqltable( SELECT ?mo ?co
        WHERE{
            ?x rdf:type 'Airplane'.
            ?x :model ?mo.
            ?y rdf:type 'Intervention'.
            ?y :Object ?x.
            ?y :Need ?z.
            ?z rdf:type 'Part'.
            ?z :codPart ?co.
      })Q

WHERE Q.mo  = 'jet'
```

- Query 3:

  Per ogni intervento di manutenzione, calcolarne il costo nominale, ossia il costo che si ottiene tenendo conto del costo della manodopera e del costo complessivo dei pezzi sostituiti, ed ignorando invece gli eventuali sconti sui pezzi di motore.

  In sparSQL:

```
SELECT  Q.x, Q.ci+SUM(Q.cp*Q.am)
FROM  sparqltable( SELECT ?x ?ci ?cp ?am
        WHERE{
            ?x rdf:type 'Intervention'.
            ?x :costInt ?ci.
            (?x :Need ?y) :amount ?am.
            ?y rdf:type 'Part'.
            ?y :costPart ?cp.
      }
   )Q

GROUP BY(Q.x)
```

Vincoli non esprimibili direttamente nell'ontologia

Non ci sono vincoli non esprimibili nell'ontologia

Esempio di asserzioni che permettano di testare le query:

classassertion(a01 Airplane)

classassertion(a02 Airplane)

classassertion(c01 Company)

classassertion(i01 Intervention)

classassertion(i02 Intervention)

classassertion(i03 Intervention)

classassertion(p01 Part)

classassertion(p02 Part)

objectPropertyAssertion(Builder a01 c01)

objectPropertyAssertion(Builder a02 c01)

objectPropertyAssertion(Object i01 a01)

objectPropertyAssertion(Object i02 a01)

objectPropertyAssertion(Object i03 a02)

objectPropertyAssertion(Need i01 p01)

objectPropertyAssertion(Need i02 p02)

objectPropertyAssertion(Need i02 p01)

objectPropertyAssertion(Need i03 p02)

dataPropertyAssertion(model a01 jet)

dataPropertyAssertion(model a02 charter)

dataProperty2Assertion(year a01 c01 2005)

dataProperty2Assertion(year a02 c01 2005)

dataPropertyAssertion(data i01 2004-07-15)

dataPropertyAssertion(data i02 2008-05-01)

dataPropertyAssertion(data i03 2005-12-31)

dataPropertyAssertion(costInt i01 2)

dataPropertyAssertion(costInt i02 3)

dataPropertyAssertion(costInt i03 4)

dataPropertyAssertion(codPart p01 ruota)

dataPropertyAssertion(costPart p01 10)

dataPropertyAssertion(codPart p02 frizione)

dataPropertyAssertion(costPart p02 15)

dataProperty2Assertion(amount i01 p01 2)

dataProperty2Assertion(amount i02 p01 1)

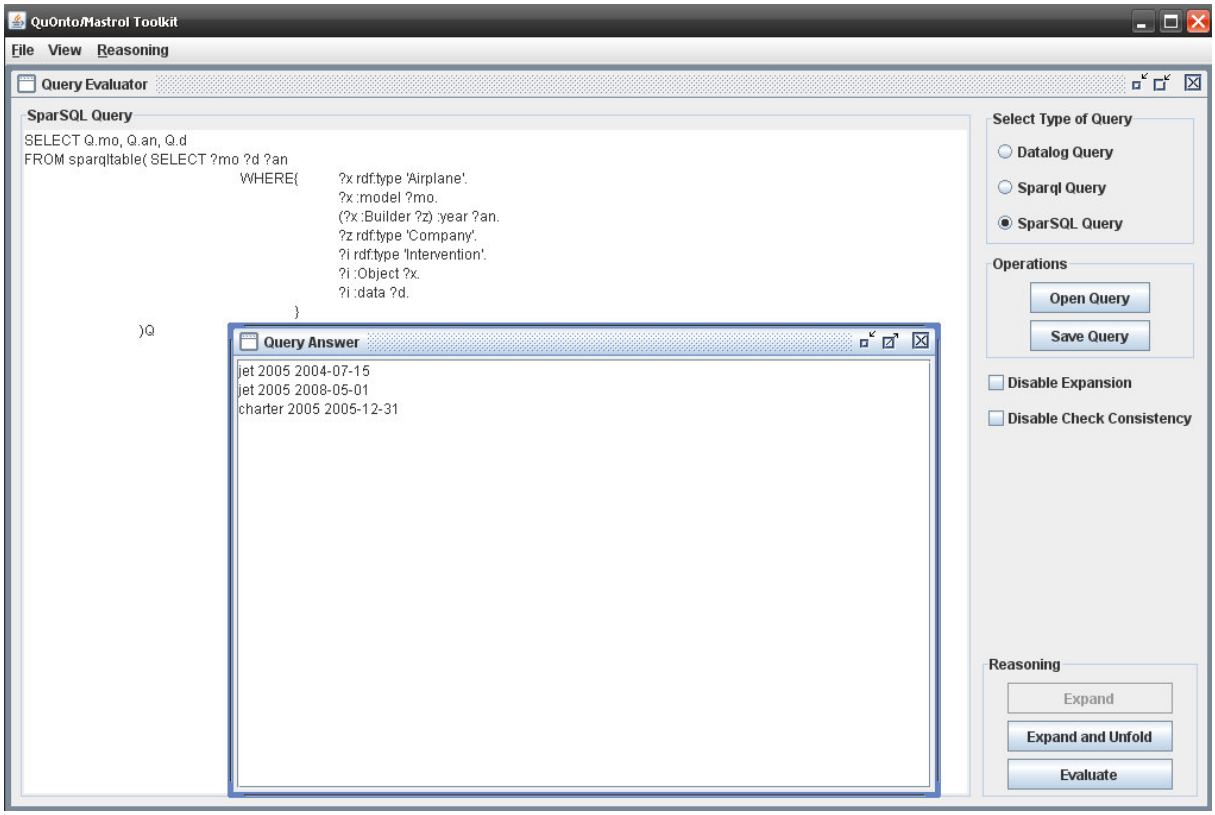
dataProperty2Assertion(amount i02 p02 2)

dataProperty2Assertion(amount i03 p02 5)

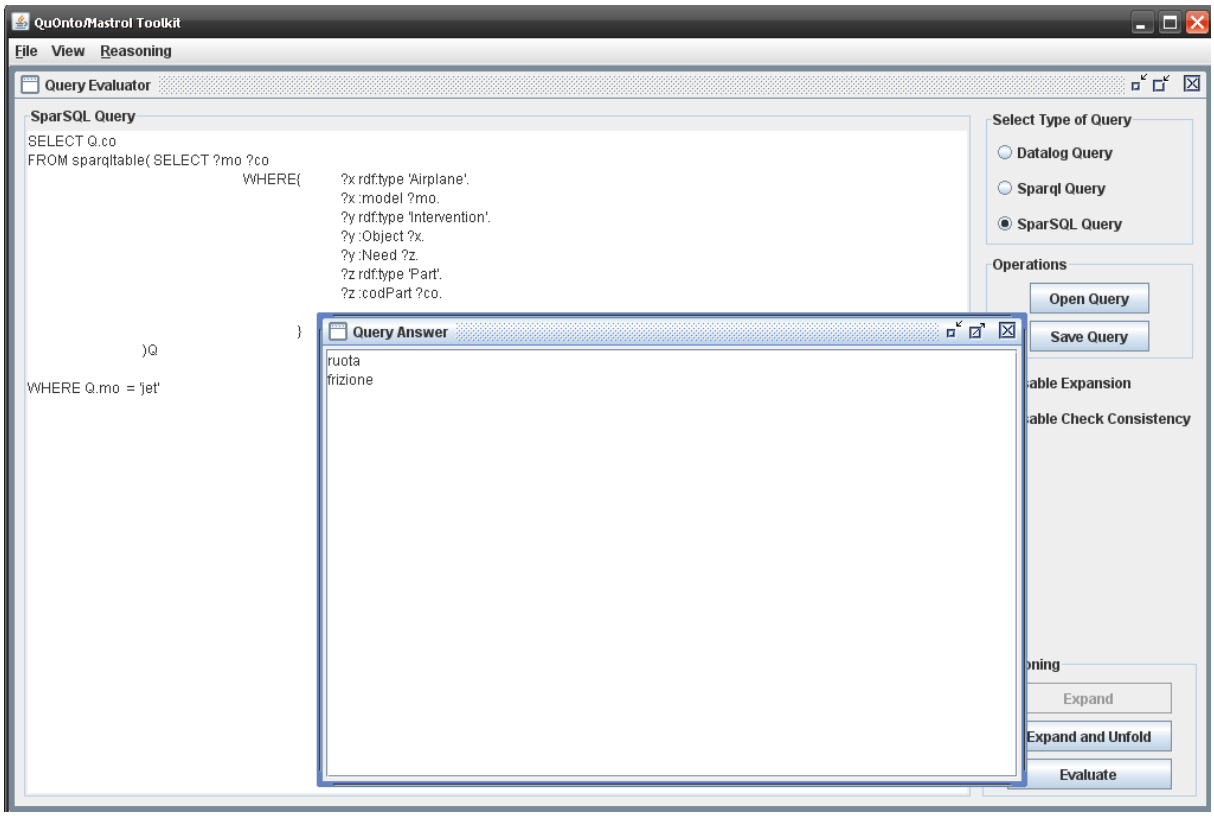
Query su QuOnto/Mastro Toolkit:

Presento una serie di screen del software utilizzato per la scrittura e l'interrogazione dell'ontologia.

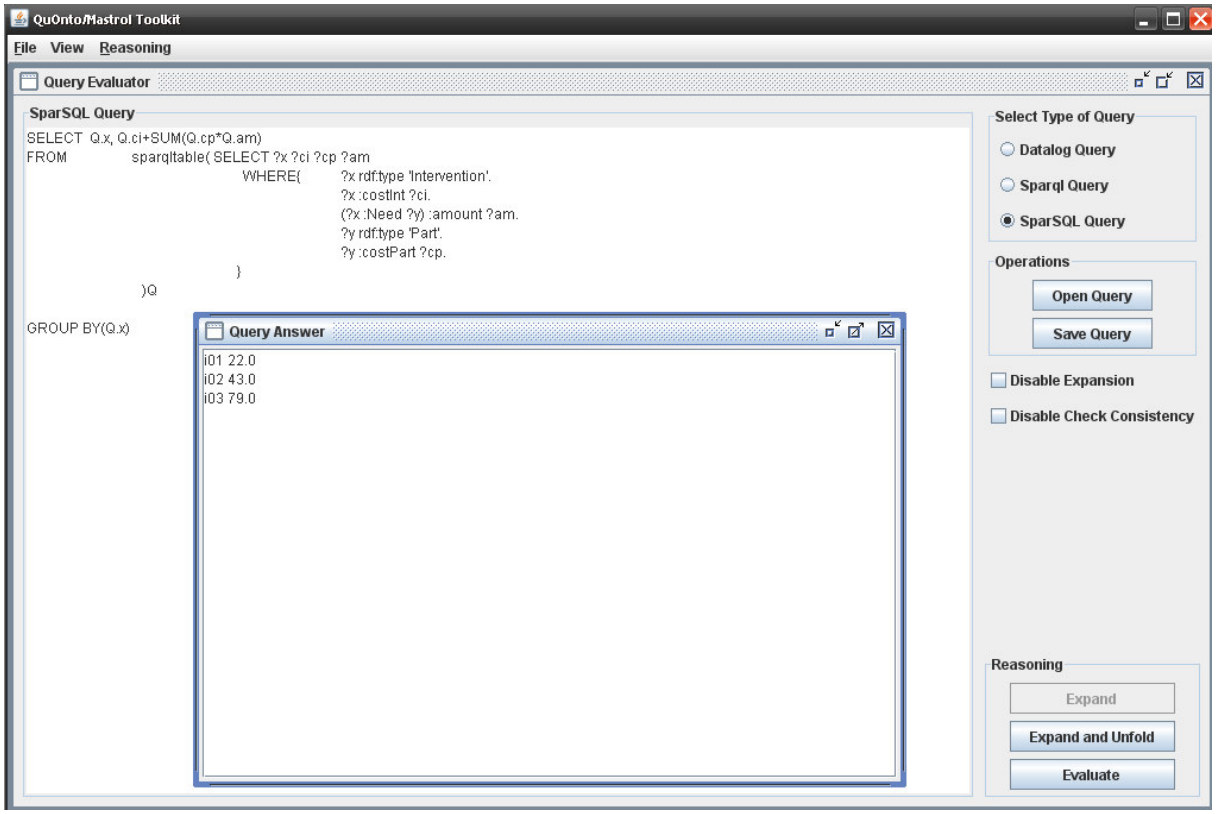
Risultato Query 1:



Risultato Query 2:



Risultato Query 3:



Caso di studio n.3 Esame Base di Dati 16-12-2005

In questa sezione ci occuperemo di analizzare la traduzione dello schema E/R del compito di base di dati del 16-12-2005 File B. Sulla base dell'ontologia risultante verranno tradotte le query del compito d'esame in query sparSQL.

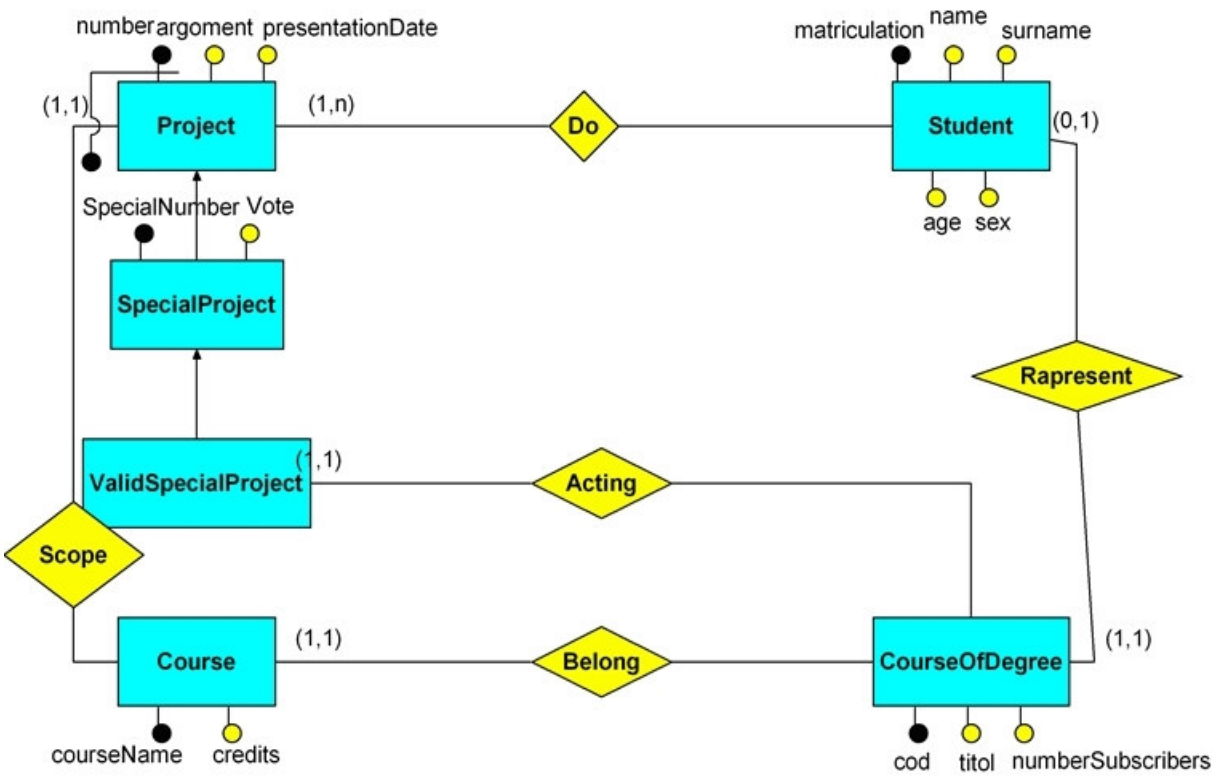
Traduzione schema ER

A seguito di questa traduzione andremo a comporre la TBox dell'ontologia.

Questo è il testo del compito d'esame:

"Si richiede di progettare lo schema concettuale Entità-Relazione di un'applicazione relativa ai progetti svolti dagli studenti di una università. Ogni progetto viene svolto da uno o più studenti nell'ambito di un corso, e di ogni progetto interessa sapere: gli studenti che l'hanno svolto, il corso nell'ambito del quale l'hanno svolto, la data di presentazione, il tipo (software, hardware, ecc.), ed il numero ad esso assegnato (unico nell'ambito del corso). Di ogni corso interessa il nome (identificativo), il numero di crediti ed il corso di laurea al quale afferisce (uno ed uno solo). Di ogni studente interessa il numero di matricola (identificativo), il nome, il cognome, l'età, il sesso ed il corso di laurea in cui fa eventualmente il rappresentante (ogni studente fa il rappresentante in al massimo un corso di laurea). Di ogni corso di laurea interessa il codice (identificativo), il nome, il numero di studenti iscritti, ed il rappresentante degli studenti. Alcuni progetti, una volta svolti, vengono classificati come "sperimentali". Ad ogni progetto classificato come sperimentale viene assegnato un codice speciale (unico nell'ambito dei progetti sperimentali), ed un voto da 1 a 10. A seguito di una delibera di un corso di laurea, un progetto sperimentale può essere considerato valido come esame finale. Pertanto, di ogni progetto sperimentale considerato valido come esame finale interessa sapere il corso di laurea (uno ed uno solo) che ne ha deliberato la validità."

A seguito dell'analisi delle specifiche, è stato prodotto il seguente schema E/R:



Traduzione ISA, generalizzazioni

Il concetto SpecialProject è in ISA con Project quindi verrà tradotto con:

SubClassOf(SpecialProject Project)

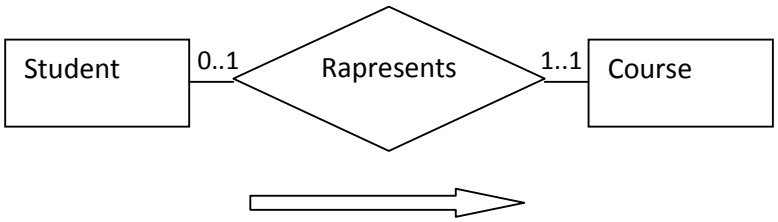
Ed inoltre ValidSpecialProject è in Isa con SpecialProject quindi si tradurrà come:

SubClassOf(ValidSpecialProject SpecialProject)

Definizione ruoli

Dalla definizione dei ruoli dell'ontologia ed in particolare dalla definizione del loro dominio e range, l'ontologia dedurrà la presenza di tutti i concetti presenti nello schema ER. Non essendoci entità isolate, riusciamo a coprirle tutte.

Per semplicità è stata assegnata la direzione di interpretazione delle relazioni sullo schema ER considerando come sorgente l'entità a cui apparteneva la molteplicità diversa da 0..n. Esiste un'unica variante a questa regola ed è rappresentata dalla relazione Rapresents:



Che verrà interpretata con il verso della freccia.

ObjectPropertyDomain(Do Project)

ObjectPropertyRange(Do Student)

ObjectPropertyDomain(Scope Project)

ObjectPropertyRange(Scope Course)

ObjectPropertyDomain(Acting ValidSpecialProject)

ObjectPropertyRange(Acting CourseOfDegree)

ObjectPropertyDomain(Belong Course)

ObjectPropertyRange(Belong CourseOfDegree)

ObjectPropertyDomain(Rapresents Student)

ObjectPropertyRange(Rapresents CourseOfDegree)

Come detto in fase di descrizione generale, ogni ruolo ha due statement: uno che ne indica il domain mentre un altro che ne indica il range e questi dipendono strettamente dalla direzione di interpretazione scelta sullo schema E/R.

Definite in questa maniera, i ruoli hanno tutti come cardinalità 0..n. Per restringerle bisogna utilizzare altri costrutti.

Cardinalità minima dei ruoli

In questa fase andiamo a restringere la cardinalità minima di tutti quei ruoli che hanno come molteplicità 1..x.

SubClassOf(Project ObjectMinCardinality(1 Do))

SubClassOf(Project ObjectMinCardinality(1 Scope))

SubClassOf(ValidSpecialProject ObjectMinCardinality(1 Acting))

SubClassOf(Course ObjectMinCardinality(1 Belong))

SubClassOf(ObjectMinCardinality(1 InverseObjectPropertyOf(Rapresents)) Student)

Cardinalità massima dei ruoli

In questa fase andiamo a restringere la cardinalità massima di tutti quei ruoli che hanno come molteplicità x..1. Unendo i costrutti precedenti con quelli di seguito elencati andremo a realizzare tutte i ruoli con le molteplicità uguali a 1..1

FunctionalObjectProperty(Scope)

FunctionalObjectProperty(Acting)

FunctionalObjectProperty(Belong)

FunctionalObjectProperty(Rapresents)

FunctionalObjectProperty(InverseObjectPropertyOf(Rapresents))

Definizione attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataPropertyRange(presentationDate rdf:date)

DataPropertyRange(argoment rdf:string)

DataPropertyRange(number rdf:string)

DataPropertyRange(specialNumber rdf:string)

DataPropertyRange(vote rdf:integer)

DataPropertyRange(matriculation rdf:string)

DataPropertyRange(name rdf:string)

DataPropertyRange(surname rdf:string)

DataPropertyRange(age rdf:integer)

DataPropertyRange(sex rdf:string)

DataPropertyRange(cod rdf:string)

DataPropertyRange(titol rdf:string)

DataPropertyRange(numberSubscribers rdf:integer)

DataPropertyRange(courseName rdf:string)

DataPropertyRange(credits rdf:integer)

Definizione dominio attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale concetto l'attributo va a descrivere.

DataPropertyDomain(presentationDate Project)

DataPropertyDomain(argoment Project)

DataPropertyDomain(number Project)

DataPropertyDomain(specialNumber SpecialProject)

DataPropertyDomain(vote SpecialProject)

DataPropertyDomain(matriculation Student)

DataPropertyDomain(name Student)

DataPropertyDomain(surname Student)

DataPropertyDomain(age Student)

DataPropertyDomain(sex Student)

DataPropertyDomain(cod CourseOfDegree)

DataPropertyDomain(titol CourseOfDegree)

DataPropertyDomain(numberSubscribers CourseOfDegree)

DataPropertyDomain(courseName Course)

DataPropertyDomain(credits Course)

Come detto anche per i ruoli, gli attributi così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi.

SubClassOf(Project dataSomeValueFrom(presentationDate xsd:anyType))

SubClassOf(Project dataSomeValueFrom(argoment xsd:anyType))

SubClassOf(Project dataSomeValueFrom(number xsd:anyType))

SubClassOf(SpecialProject dataSomeValueFrom(specialNumber xsd:anyType))

SubClassOf(SpecialProject dataSomeValueFrom(vote xsd:anyType))

SubClassOf(Student dataSomeValueFrom(matriculation xsd:anyType))

SubClassOf(Student dataSomeValueFrom(name xsd:anyType))

SubClassOf(Student dataSomeValueFrom(surname xsd:anyType))

SubClassOf(Student dataSomeValueFrom(age xsd:anyType))

SubClassOf(Student dataSomeValueFrom(sex xsd:anyType))

SubClassOf(CourseOfDegree dataSomeValueFrom(cod xsd:anyType))

SubClassOf(CourseOfDegree dataSomeValueFrom(titol xsd:anyType))

SubClassOf(CourseOfDegree dataSomeValueFrom(numberSubscribers xsd:anyType))

SubClassOf(Course dataSomeValueFrom(courseName xsd:anyType))

SubClassOf(Course dataSomeValueFrom(credits xsd:anyType))

Cardinalità massima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty(presentationDate)

FunctionalDataProperty(argoment)

FunctionalDataProperty(number)

FunctionalDataProperty(specialNumber)

FunctionalDataProperty(vote)

FunctionalDataProperty(matriculation)

FunctionalDataProperty(name)

FunctionalDataProperty(surname)

FunctionalDataProperty(age)

FunctionalDataProperty(sex)

FunctionalDataProperty(cod)

FunctionalDataProperty(titol)

FunctionalDataProperty(numberSubscribers)

FunctionalDataProperty(courseName)

FunctionalDataProperty(credits)

Definizione attributi di ruolo

Non sono presenti attributi di ruolo.

Definizione dominio attributi di ruolo

Non sono presenti attributi di ruolo.

Cardinalità minima degli attributi di ruolo

Non sono presenti attributi di ruolo.

Cardinalità massima degli attributi di ruolo

Non sono presenti attributi di ruolo.

Vincoli di chiave

Per ogni chiave definita all'interno dello schema E/R esprimiamo il vincolo nell'ontologia.

KeyFor(number,Scope Project)

KeyFor(matriculation Student)

KeyFor(cod CourseOfDegree)

KeyFor(courseName Course)

Le chiavi esterne si differenziano da quelle primarie perché hanno, separato da virgola, anche il ruolo che contribuisce all'identificazione oltre eventualmente all'attributo.

Query in sparSQL

Andiamo adesso ad esprimere le query sul indicate nel testo sull'ontologia risultante.

- Query 1:

  Calcolare il numero di matricola ed il sesso degli studenti che hanno svolto almeno un progetto per il corso di "Basi di dati".

  In sparSQL:

```
SELECT DISTINCT Q.ma, Q.se
FROM sparqltable( SELECT ?ma ?se ?cou
        WHERE{
```

```
            ?x rdf:type 'Project'.
            ?x :Do ?y.
            ?y rdf:type 'Student'.
            ?y :matriculation ?ma.
            ?y :sex ?se.
            ?x :Scope ?z.
            ?z rdf:type 'Course'.
            ?z :courseName ?cou.
                              }
   )Q
WHERE Q.cou = 'BD'
```

- Query 2:

Calcolare il nome ed il numero di crediti dei corsi nell'ambito dei quali non è stato svolto alcun progetto presentato nel 2004.

In sparSQL:

```
SELECT  Uno.na, Uno.c
FROM sparqltable( SELECT ?na ?c
             WHERE{
                  ?x rdf:type 'Course'.
                  ?x :courseName ?na.
                  ?x :credits ?c.
            } )Uno
WHERE Uno.na not in (

      SELECT  Due.nam
      FROM sparqltable( SELECT ?nam ?d
              WHERE{
                  ?x rdf:type 'Project'.
                  ?x :Scope ?y.
                  ?y rdf:type 'Course'.
                  ?y :courseName ?nam.
                  ?x :presentationDate ?d.
            })Due
      WHERE Due.d  BETWEEN '2004-01-01' and '2004-12-31'
)
```

- Query 3:

Per ogni studente, calcolare la media dei voti ottenuti per tutti i progetti sperimentali da esso svolto.

In sparSQL:

```
SELECT  Q.ma, AVG(Q.v)
FROM  sparqltable( SELECT ?ma ?v
              WHERE{
                  ?x rdf:type 'SpecialProject'.
                  ?x :Do ?y.
                  ?y rdf:type 'Student'.
                  ?y :matriculation ?ma.
```

```
                    ?x :vote ?v.
                })Q

    GROUP BY(Q.ma)
```

Vincoli non esprimibili direttamente nell'ontologia

Non ci sono vincoli non esprimibili nell'ontologia

Esempio di asserzioni che permettano di testare le query:

classassertion(s01 Student)

classassertion(s02 Student)

classassertion(s03 Student)

classassertion(p01 Project)

classassertion(p02 SpecialProject)

classassertion(p03 SpecialProject)

classassertion(c01 Course)

classassertion(c02 Course)

classassertion(cd1 CourseOfDegree)

classassertion(cd2 CourseOfDegree)

objectPropertyAssertion(Do p01 s01)

objectPropertyAssertion(Do p01 s02)

objectPropertyAssertion(Do p02 s03)

objectPropertyAssertion(Do p03 s03)

objectPropertyAssertion(Do p03 s02)

objectPropertyAssertion(Scope p01 c01)

objectPropertyAssertion(Scope p02 c02)

objectPropertyAssertion(Rapresents s01 cd1)

objectPropertyAssertion(Rapresents cd1 s01)

dataPropertyAssertion(matriculation s01 801977)

dataPropertyAssertion(matriculation s02 801014)

dataPropertyAssertion(matriculation s03 780156)

dataPropertyAssertion(vote p02 28)

dataPropertyAssertion(vote p03 30)

dataPropertyAssertion(sex s01 m)

dataPropertyAssertion(sex s02 m)

dataPropertyAssertion(sex s03 f)

dataPropertyAssertion(courseName c01 BD)

dataPropertyAssertion(courseName c02 SGBD)

dataPropertyAssertion(credits c01 5)

dataPropertyAssertion(credits c02 6)

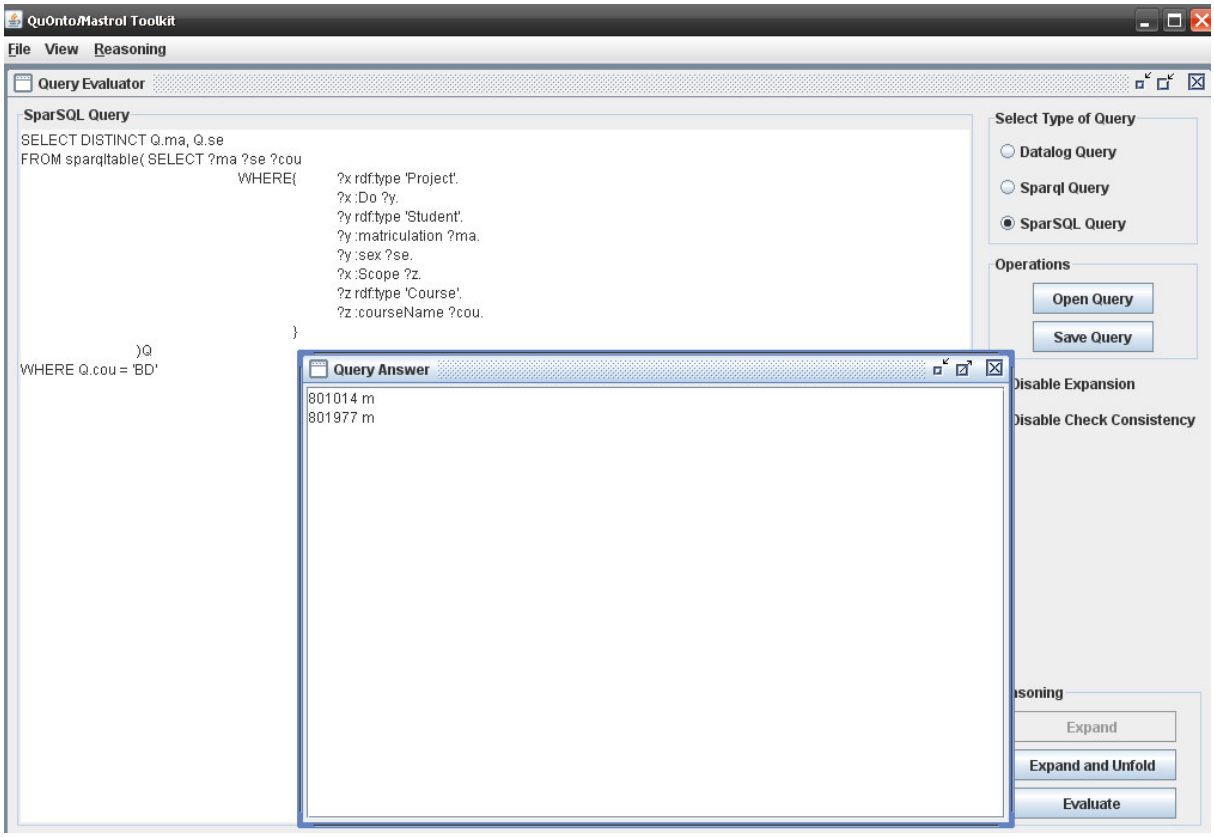
dataPropertyAssertion(presentationDate p01 20040511)

dataPropertyAssertion(presentationDate p02 20050511)

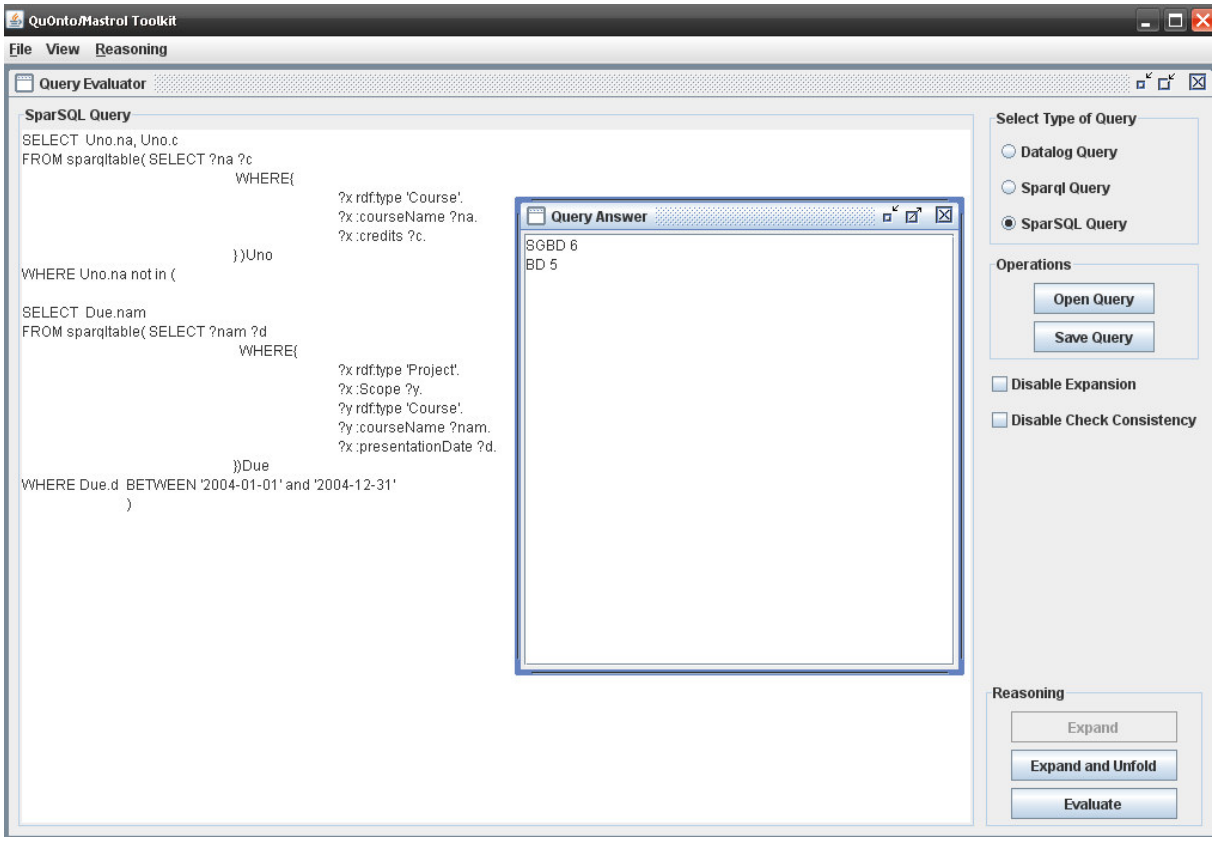
Query su QuOnto/Mastro Toolkit:

Presento una serie di screen del software utilizzato per la scrittura e l'interrogazione dell'ontologia.

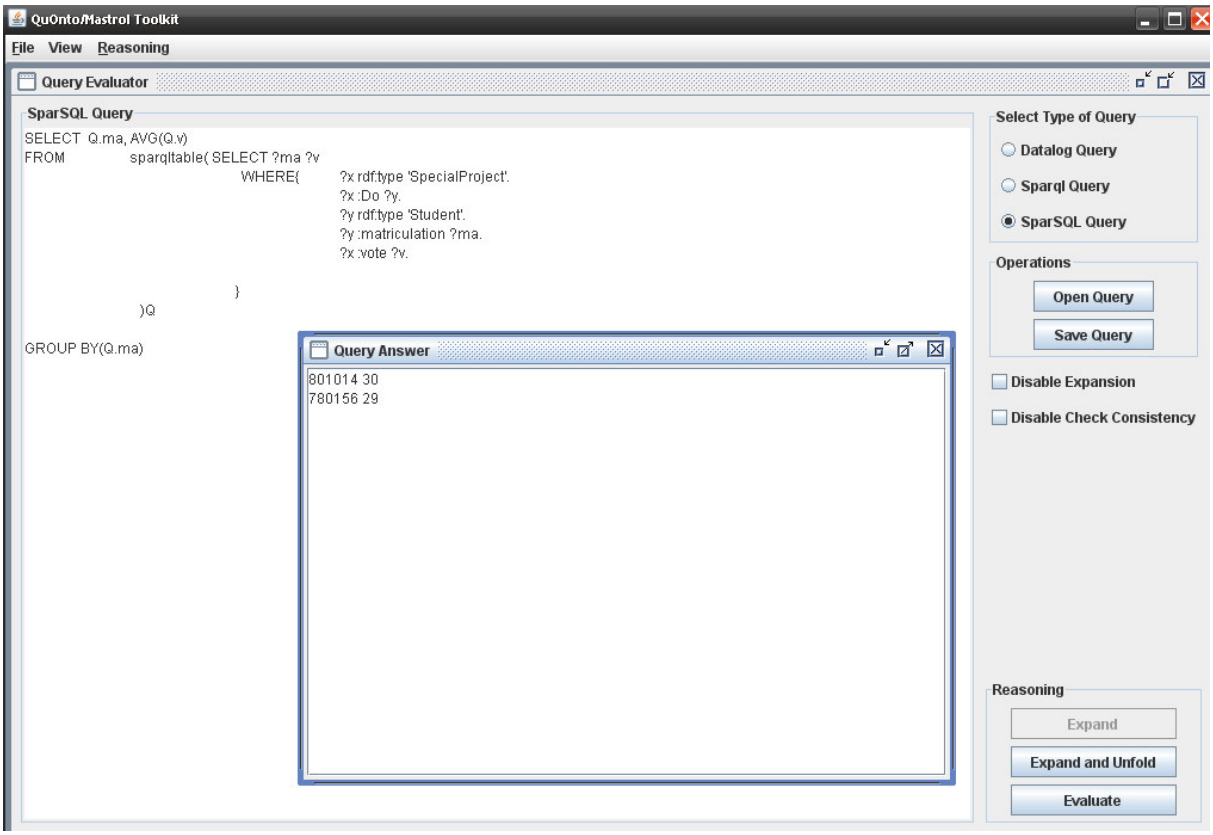
Risultato Query 1:



Risultato Query 2:



Risultato Query 3:



Caso di studio n.4 Esame Base di Dati 08-01-2004

In questa sezione ci occuperemo di analizzare la traduzione dello schema E/R del compito di base di dati del 08-01-2004 Fila A. Sulla base dell'ontologia risultante verranno tradotte le query del compito d'esame in query sparSQL.

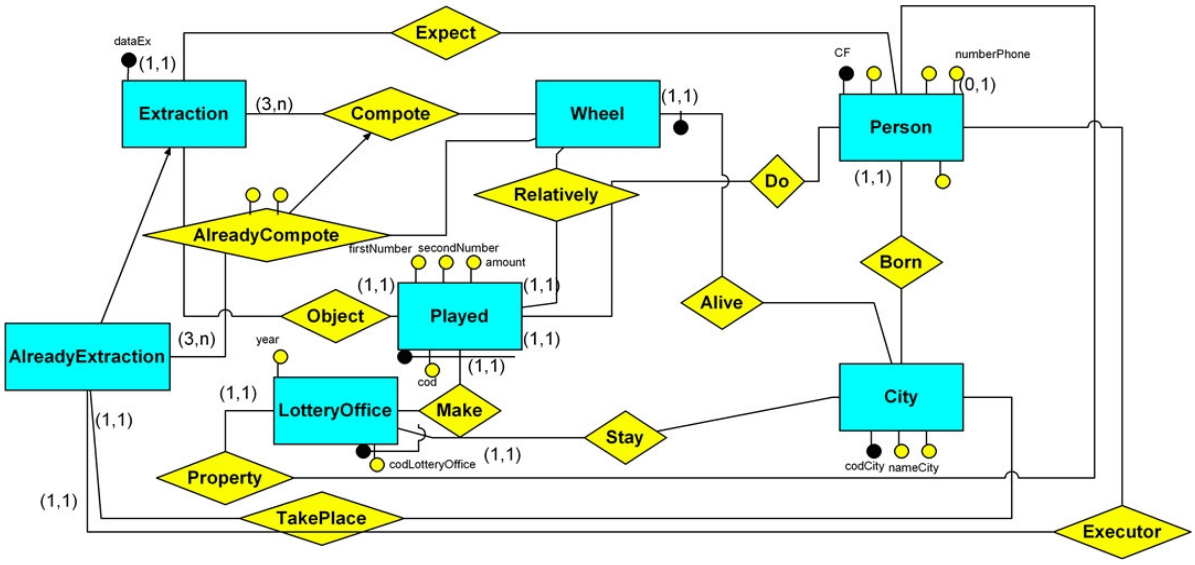
Traduzione schema ER

A seguito di questa traduzione andremo a comporre la TBox dell'ontologia.

Questo è il testo del compito d'esame:

" Si richiede di progettare lo schema concettuale Entità-Relazione di un'applicazione per la gestione delle estrazioni di un gioco simile al Lotto. Di ogni estrazione interessa la data, la persona prevista per l'effettiva estrazione dei numeri dall'urna, e le varie ruote (almeno tre) considerate in quella estrazione. Si noti che non si può svolgere pi`u di una estrazione al giorno. Ogni ruota `e associata ad una città (ad esempio, c'`e la ruota di Milano, di Roma, di Napoli, ecc.), che non ha nulla a che vedere con la città in cui ha luogo l'estrazione. Si noti che le ruote considerate nelle varie estrazioni variano da estrazione a estrazione. Di ogni estrazione già svolta interessa la città in cui si `e tenuta, la persona che ha effettuato l'estrazione dei numeri dall'urna (che può anche essere diversa da quella originariamente prevista), ed i due numeri usciti nelle varie ruote: ogni estrazione prevede infatti che vengano estratti due numeri per ognuna delle ruote considerate in quella estrazione. Di ogni persona interessa il codice fiscale (identificativo), il nome, il cognome, la data di nascita, la città di nascita, ed il numero di telefono (se noto). Di ogni città interessa il codice (identificativo), il nome, ed il numero di abitanti. Una giocata riguarda due numeri, ed è effettuata da una persona, in una certa ricevitoria, per una certa estrazione, per una certa ruota, per un certo ammontare di soldi puntati. Di ogni giocata interessano tutte le suddette proprietà (i due numeri, la persona, la ricevitoria, l'estrazione, la ruota, ed i soldi), più il suo codice, che è unico nell'ambito della ricevitoria. Infine, di ogni ricevitoria interessa la città in cui si trova, il codice (unico nell'ambito della città), l'anno di inaugurazione, e la persona che ne è proprietaria (una ed una sola). "

A seguito dell'analisi delle specifiche, è stato prodotto il seguente schema E/R:



Traduzione ISA, generalizzazioni

Il concetto AlreadyExtractiont è in ISA con Extraction quindi verrà tradotto con:

SubClassOf(AlreadyExtraction Extraction)

Inoltre il ruolo AlreadyCompote è in ISA con Compote quindi verrà tradotto con:

SubObjectPropertyOf(AlreadyCompote Compote)

Definizione ruoli

Dalla definizione dei ruoli dell'ontologia ed in particolare dalla definizione del loro dominio e range, l'ontologia dedurrà la presenza di tutti i concetti presenti nello schema ER. Non essendoci entità isolate, riusciamo a coprirle tutte.

Per semplicità è stata assegnata la direzione di interpretazione delle relazioni sullo schema ER considerando come sorgente l'entità a cui apparteneva la molteplicità diversa da 0..n. Questo è stato possibile perché non ci sono relazioni con molteplicità diverse da 0..n in entrambi i lati.

La traduzione dei ruoli quindi sarà questa:

ObjectPropertyDomain(Expect Extraction)

ObjectPropertyRange(Expect Person)

ObjectPropertyDomain(Compote Extraction)

ObjectPropertyRange(Compote Wheel)

ObjectPropertyDomain(AlreadyCompote AlreadyExtraction)

ObjectPropertyRange(AlreadyCompote Wheel)

ObjectPropertyDomain(TakePlace AlreadyExtraction)

ObjectPropertyRange(TakePlace City)

ObjectPropertyDomain(Executor AlreadyExtraction)

ObjectPropertyRange(Executor Person)

ObjectPropertyDomain(Alive Wheel)

ObjectPropertyRange(Alive City)

ObjectPropertyDomain(Relatively Played)

ObjectPropertyRange(Relatively Wheel)

ObjectPropertyDomain(Do Played)

ObjectPropertyRange(Do Person)

ObjectPropertyDomain(Object Played)

ObjectPropertyRange(Object Extraction)

ObjectPropertyDomain(Make Played)

ObjectPropertyRange(Make LotteryOffice)

ObjectPropertyDomain(Born Person)

ObjectPropertyRange(Born City)

ObjectPropertyDomain(Property LotteryOffice)

ObjectPropertyRange(Property Person)

ObjectPropertyDomain(Stay LotteryOffice)

ObjectPropertyRange(Stay City)

Come detto in fase di descrizione generale, ogni ruolo ha due statement: uno che ne indica il domain mentre un altro che ne indica il range e questi dipendono strettamente dalla direzione di interpretazione scelta sullo schema E/R.

Definiti in questa maniera, i ruoli hanno tutti come cardinalità 0..n. Per restringerle bisogna utilizzare altri costrutti.

Cardinalità minima dei ruoli

In questa fase andiamo a restringere la cardinalità minima di tutti quei ruoli che hanno come molteplicità 1..x.

SubClassOf(Extraction ObjectMinCardinality(1 Expect))

SubClassOf(AlreadyExtraction ObjectMinCardinality(1 TakePlace))

SubClassOf(AlreadyExtraction ObjectMinCardinality(1 Executor))

SubClassOf(Wheel ObjectMinCardinality(1 Alive))

SubClassOf(Played ObjectMinCardinality(1 Do))

SubClassOf(Played ObjectMinCardinality(1 Object))

SubClassOf(Played ObjectMinCardinality(1 Make))

SubClassOf(Person ObjectMinCardinality(1 Born))

SubClassOf(LotteryOffice ObjectMinCardinality(1 Property))

SubClassOf(LotteryOffice ObjectMinCardinality(1 Stay))

SubClassOf(Played ObjectMinCardinality(1 Relatively))

Cardinalità massima dei ruoli

In questa fase andiamo a restringere la cardinalità massima di tutti quei ruoli che hanno come molteplicità x..1. Unendo i costrutti precedenti con quelli di seguito elencati andremo a realizzare tutte i ruoli con le molteplicità uguali a 1..1

FunctionalObjectProperty(Expect)

FunctionalObjectProperty(TakePlace)

FunctionalObjectProperty(Executor)

FunctionalObjectProperty(Alive)

FunctionalObjectProperty(Do)

FunctionalObjectProperty(Object)

FunctionalObjectProperty(Make)

FunctionalObjectProperty(Born)

FunctionalObjectProperty(Property)

FunctionalObjectProperty(Stay)

FunctionalObjectProperty(Relatively)

Definizione attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataPropertyRange(dataEx rdf:date)

DataPropertyRange(firstNumber rdf:integer)

DataPropertyRange(secondNumber rdf:integer)

DataPropertyRange(amount rdf:float)

DataPropertyRange(codPlayed rdf:string)

DataPropertyRange(year rdf:integer)

DataPropertyRange(codLotteryOffice rdf:string)

DataPropertyRange(codCity rdf:string)

DataPropertyRange(nameCity rdf:string)

DataPropertyRange(inhabitants rdf:integer)

DataPropertyRange(cf rdf:string)

DataPropertyRange(name rdf:string)

DataPropertyRange(surname rdf:string)

DataPropertyRange(birthday rdf:date)

DataPropertyRange(telNumber rdf:string)

Definizione dominio attributi di concetto

Per ogni attributo di concetto andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale concetto l'attributo va a descrivere.

DataPropertyDomain(dataEx Extraction)

DataPropertyDomain(firstNumber Played)

DataPropertyDomain(secondNumber Played)

DataPropertyDomain(amount Played)

DataPropertyDomain(codPlayed Played)

DataPropertyDomain(year LotteryOffice)

DataPropertyDomain(codLotteryOffice LotteryOffice)

DataPropertyDomain(codCity City)

DataPropertyDomain(nameCity City)

DataPropertyDomain(inhabitants City)

DataPropertyDomain(cf Person)

DataPropertyDomain(name Person)

DataPropertyDomain(surname Person)

DataPropertyDomain(birthday Person)

DataPropertyDomain(telNumber Person)

Come detto anche per i ruoli, gli attributi così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi tranne di *employees* che da specifica era considerato opzionale .

SubClassOf(Extraction dataSomeValueFrom(dataEx xsd:anyType))

SubClassOf(Played dataSomeValueFrom(firtNumber xsd:anyType))

SubClassOf(Played dataSomeValueFrom(secondNumber xsd:anyType))

SubClassOf(Played dataSomeValueFrom(amount xsd:anyType))

SubClassOf(Played dataSomeValueFrom(codPlayed xsd:anyType))

SubClassOf(LotteryOffice dataSomeValueFrom(year xsd:anyType))

SubClassOf(LotteryOffice dataSomeValueFrom(codLotteryOffice xsd:anyType))

SubClassOf(City dataSomeValueFrom(codCity xsd:anyType))

SubClassOf(City dataSomeValueFrom(nameCity xsd:anyType))

SubClassOf(City dataSomeValueFrom(inhabitants xsd:anyType))

SubClassOf(Person dataSomeValueFrom(cf xsd:anyType))

SubClassOf(Person dataSomeValueFrom(name xsd:anyType))

SubClassOf(Person dataSomeValueFrom(surname xsd:anyType))

SubClassOf(Person dataSomeValueFrom(birthday xsd:anyType))

Cardinalità massima degli attributi di concetto

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty(dataEx)

FunctionalDataProperty(firtNumber)

FunctionalDataProperty(secondNumber)

FunctionalDataProperty(amount)

FunctionalDataProperty(codPlayed)

FunctionalDataProperty(year)

FunctionalDataProperty(codLotteryOffice)

FunctionalDataProperty(codCity)

FunctionalDataProperty(nameCity)

FunctionalDataProperty(inhabitants)

FunctionalDataProperty(cf)

FunctionalDataProperty(name)

FunctionalDataProperty(surname)

FunctionalDataProperty(birthday)

FunctionalDataProperty(telNumber)

Definizione attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il range, ovvero lo spazio all'interno del quale l'attributo può assumere valori.

DataProperty2Range(num1 rdf:integer)

DataProperty2Range(num2 rdf:integer)

Definizione dominio attributi di ruolo

Per ogni attributo di ruolo andiamo ad indicarne il dominio, ovvero indichiamo all'ontologia quale ruolo l'attributo va a descrivere.

DataProperty2Domain(num1 AlreadyCompote)

DataProperty2Domain(num2 AlreadyCompote)

Come detto anche per gli attributi di concetto, anche gli attributi di ruolo così definiti vengono descritti con una molteplicità di tipo 0..n. Per restringere tale molteplicità bisogna utilizzare i prossimi costrutti.

Cardinalità minima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità minima di tutti gli attributi.

SubObjectPropertyOf(AlreadyCompote data2SomeValueFrom(num1 xsd:anyType))

SubObjectPropertyOf(AlreadyCompote data2SomeValueFrom(num2 xsd:anyType))

Cardinalità massima degli attributi di ruolo

In questa fase andiamo a restringere la cardinalità massima di tutti gli attributi.

FunctionalDataProperty2(num1)

FunctionalDataProperty2(num2))

Vincoli di chiave

Per ogni chiave definita all'interno dello schema E/R esprimiamo il vincolo nell'ontologia.

KeyFor(dataEx Extraction)

KeyFor(codPlayed,Make Played)

KeyFor(codLotteryOffice,Stay LotteryOffice)

KeyFor(codCity City)

KeyFor(cf Person)

KeyFor(City Wheel)

Le chiavi esterne si differenziano da quelle primarie perché hanno, separato da virgola, anche il ruolo che contribuisce all'identificazione oltre eventualmente all'attributo.

Query in sparSQL

Andiamo adesso ad esprimere le query sul indicate nel testo sull'ontologia risultante.

- Query 1:

  Calcolare nome e cognome di tutte le persone che hanno effettuato l'estrazione dei numeri dall'urna per almeno una estrazione del 2003.

  In sparSQL:

```
SELECT DISTINCT Q.n, Q.c
```

```
FROM sparqltable( SELECT ?d ?n ?c
              WHERE{    ?x rdf:type 'AlreadyExtraction'.
                  ?x :Executor ?y.
                  ?y rdf:type 'Person'.
                  ?x :dataEx ?d.
                  ?y :name ?n.
                  ?y :surname ?c.
                                  }
         )Q
WHERE Q.d BETWEEN '2003-01-01' AND '2003-12-31'
```

- Query 2:

Calcolare nome e cognome di ogni persona che ha effettuato l'estrazione dei numeri dall'urna per almeno una estrazione tenuta nella sua citt`a di nascita.

In sparSQL:

```
SELECT DISTINCT Q.n, Q.s
FROM sparqltable( SELECT ?z ?w ?n ?s
              WHERE{    ?x rdf:type 'AlreadyExtraction'.
                  ?x :Executor ?y.
                  ?y rdf:type 'Person'.
                  ?y :name ?n.
                  ?y :surname ?s.
                  ?y :Born ?z.
                  ?z rdf:type 'City'.
                  ?x :TakePlace ?w.
                  ?w rdf:type 'City'.
                                  }
         )Q
WHERE Q.z = Q.w
```

- Query 3:

Calcolare nome e cognome delle persone che, per almeno una estrazione, hanno effettuato almeno una giocata per tutte le ruote previste in quella estrazione.

In sparSQL:

```
SELECT Uno.n
      FROM sparqltable( SELECT ?n
      WHERE{
            ?x rdf:type 'Played'.
            ?x :Do?p.
            ?p rdf:type 'Person'.
            ?p :name ?n.
      })Uno
WHERE Uno.n NOT IN(
SELECT Due.n
      FROM sparqltable( SELECT ?w ?n
      WHERE{
            ?z rdf:type 'Extraction'.
            ?z :Compote ?w.
            ?w rdf:type 'Wheel'.
```

```
            ?x rdf:type 'Played'.
            ?x :Do?p.
            ?p rdf:type 'Person'.
            ?p :name ?n.
            ?x :Object ?z.
      })Due
WHERE Due.w NOT IN(
SELECT  Tre.r
      FROM sparqltable( SELECT ?r ?x ?p ?o
             WHERE{
                  ?x rdf:type 'Played'.
                  ?x :Relatively ?r.
                  ?r rdf:type 'Wheel'.
                  ?x :Do ?y.
                  ?y rdf:type 'Person'.
                  ?y :name ?p.
                  ?x :Object ?o.
      })Tre
))
```

- Query 4:

Per ogni persona e per ogni estrazione, calcolare i soldi puntati complessivamente da quella persona per quella estrazione.

In sparSQL:

```
SELECT  Q.n, Q.d, SUM(Q.am)
FROM  sparqltable( SELECT ?n ?am ?d
              WHERE{
                  ?x rdf:type 'Played'.
                  ?x :amount ?am.
                  ?x :Do ?y.
                  ?y rdf:type 'Person'.
                  ?y :name ?n.
                  ?x :Object ?z.
                  ?z rdf:type 'Extraction'.
                  ?z :dataEx ?d.
             })Q

GROUP BY Q.n, Q.d
```

Vincoli non esprimibili direttamente nell'ontologia

Analizzando le specifiche le molteplicità delle relazioni sull'E/R si può notare che Extractoin partecipa alla relazione Compote con una cardinalità minima di 3. Questo non può essere espresso direttamente nell'ontologia quindi bisogna ricorrere ad una query sparSQL booleana che verifica la consistenza delle asserzioni sull'ontologia(ovvero la consistenza sui dati).

In questa query si contano le tuple risultanti percorrendo dal concetto Extraction al concetto Wheel tramite il ruolo Compote e raggruppando per Extraction. Selezionando successivamente quelle tuple che hanno meno di 3 partecipazioni alla relazione.

```
VERIFY EXISTS(SELECT Uno.x
        FROM sparqltable( SELECT ?x ?w
        WHERE{
                ?x rdf:type 'Extraction'.
                ?x :Compote ?w.
                ?w rdf:type 'Wheel'.
        })Uno
GROUP BY (Uno.x)
HAVING COUNT(*) < 3
)
```

Esempio di asserzioni che permettano di testare le query:

classassertion(p01 Person)

classassertion(p02 Person)

classassertion(p03 Person)

classassertion(c01 City)

classassertion(g01 Played)

classassertion(g02 Played)

classassertion(g03 Played)

classassertion(g04 Played)

classassertion(g05 Played)

classassertion(w01 Wheel)

classassertion(w02 Wheel)

classassertion(w03 Wheel)

classassertion(es1 AlreadyExtraction)

classassertion(es2 AlreadyExtraction)

classassertion(e01 Extraction)

classassertion(e02 Extraction)

objectPropertyAssertion(Executor es1 p01)

objectPropertyAssertion(Executor es2 p02)

objectPropertyAssertion(Born p01 c01)

objectPropertyAssertion(TakePlace es1 c01)

objectPropertyAssertion(Do g01 p02)

objectPropertyAssertion(Do g02 p02)

objectPropertyAssertion(Do g03 p01)

objectPropertyAssertion(Do g04 p01)

objectPropertyAssertion(Do g05 p01)

objectPropertyAssertion(Object g01 e02)

objectPropertyAssertion(Object g02 e02)

objectPropertyAssertion(Object g03 e01)

objectPropertyAssertion(Object g04 e01)

objectPropertyAssertion(Object g05 e01)

objectPropertyAssertion(Compote e01 w01)

objectPropertyAssertion(Compote e01 w02)

objectPropertyAssertion(Compote e01 w03)

objectPropertyAssertion(Compote e02 w01)

objectPropertyAssertion(Compote e02 w02)

objectPropertyAssertion(Relatively g03 w01)

objectPropertyAssertion(Relatively g04 w02)

objectPropertyAssertion(Relatively g05 w03)

objectPropertyAssertion(Relatively g01 w01)

objectPropertyAssertion(Relatively g02 w02)

dataPropertyAssertion(dataEx e01 2003-01-01)

dataPropertyAssertion(dataEx e02 2008-01-01)

dataPropertyAssertion(dataEx es1 2003-01-02)

dataPropertyAssertion(dataEx es2 2004-01-01)

dataPropertyAssertion(name p01 Emma)

dataPropertyAssertion(surname p01 CognomeDiEmma)

dataPropertyAssertion(name p02 Roberto)

dataPropertyAssertion(surname p02 CognomeDiRoberto)

dataPropertyAssertion(name p03 Flavio)

dataPropertyAssertion(surname p03 CognomeDiFlavio)

dataPropertyAssertion(amount g01 11)

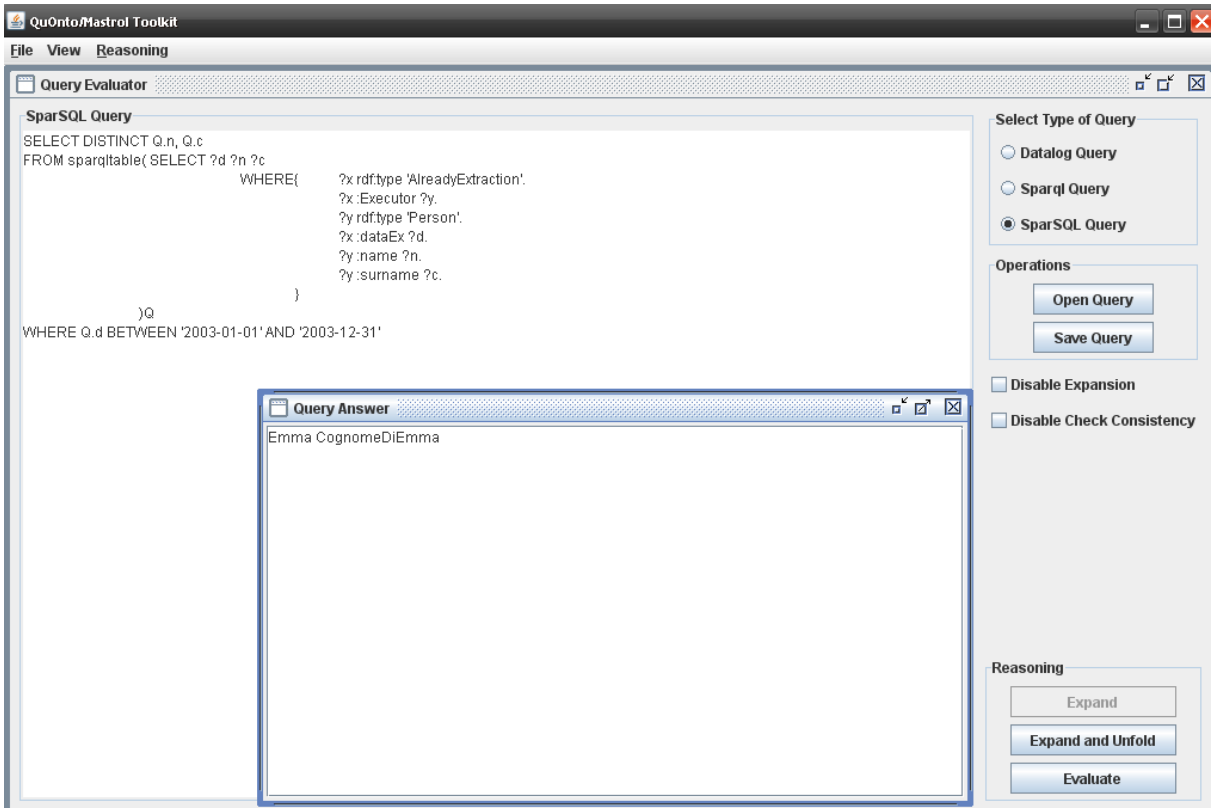
dataPropertyAssertion(amount g02 14)

dataPropertyAssertion(amount g03 2)

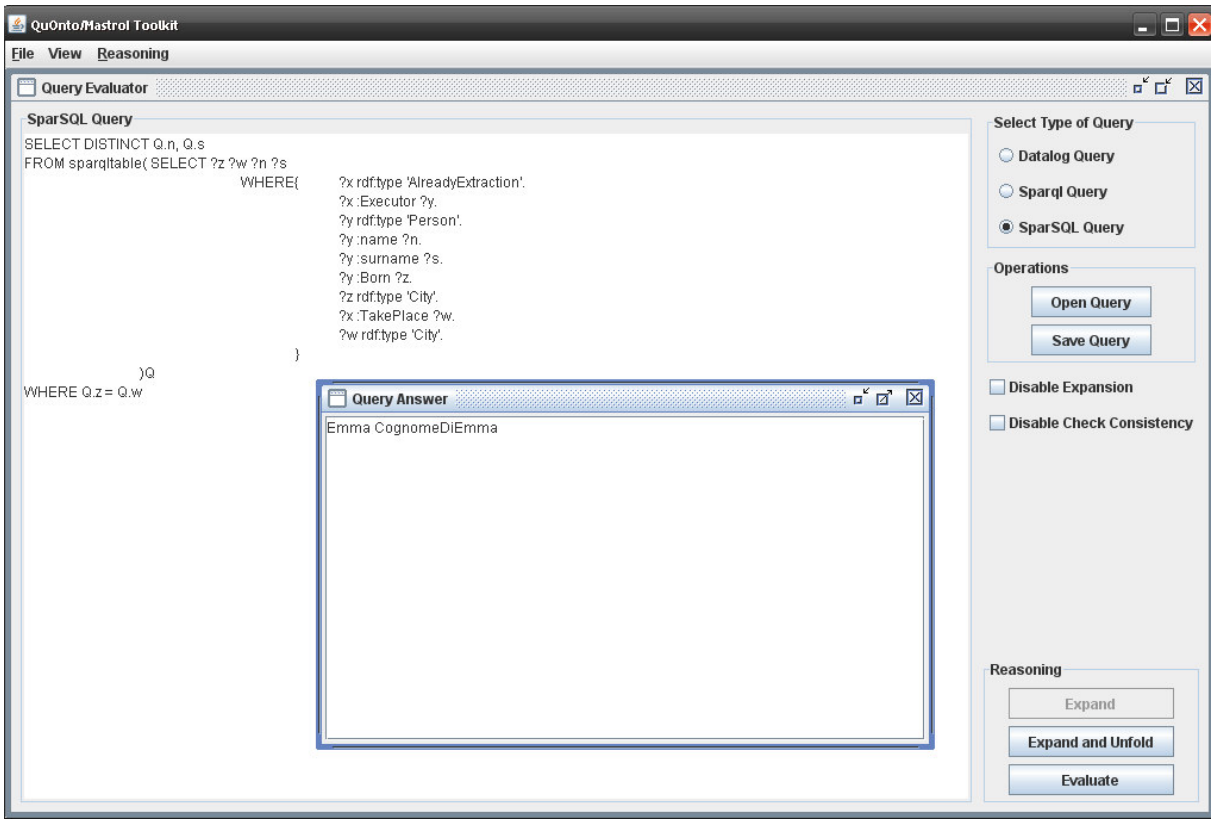
Query su QuOnto/Mastro Toolkit:

Presento una serie di screen del software utilizzato per la scrittura e l'interrogazione dell'ontologia.

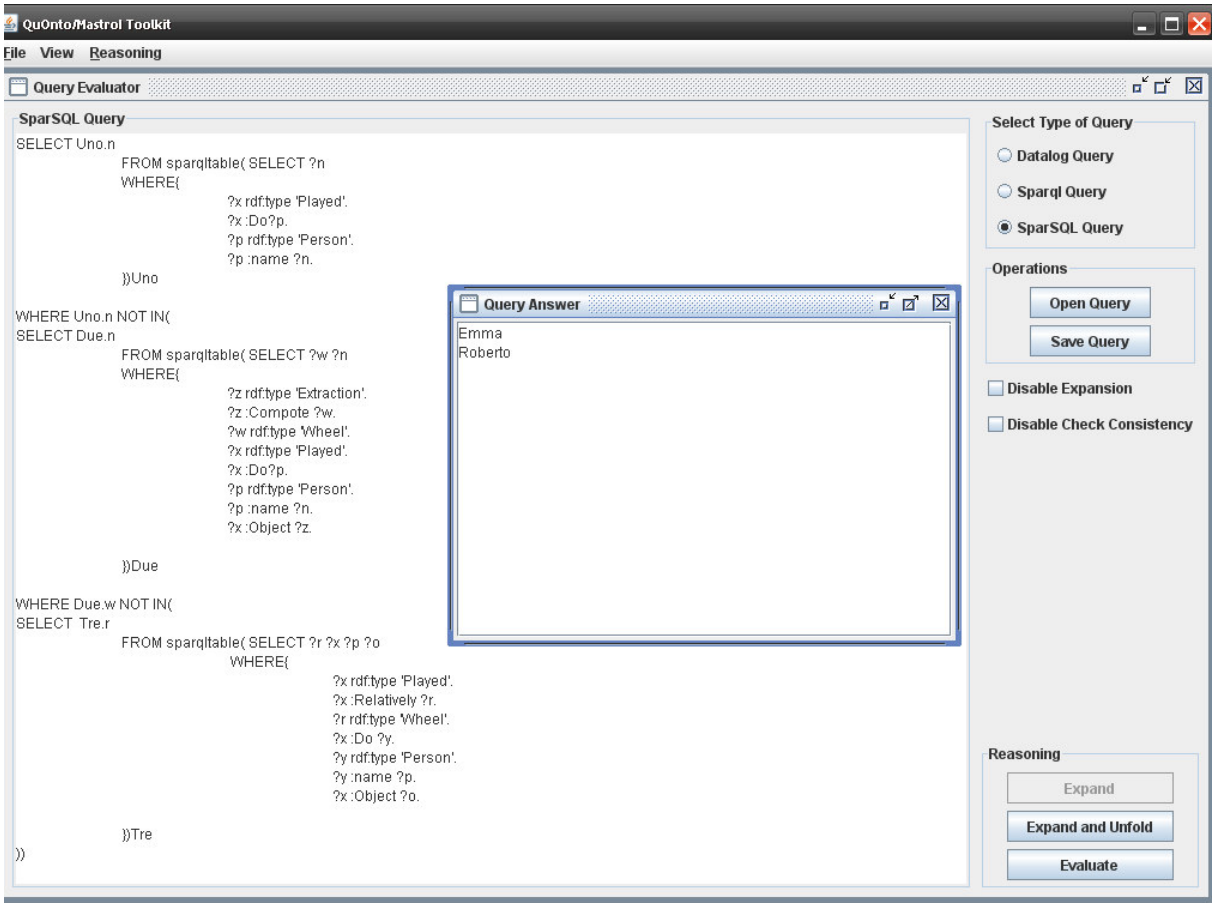
Risultato Query 1:



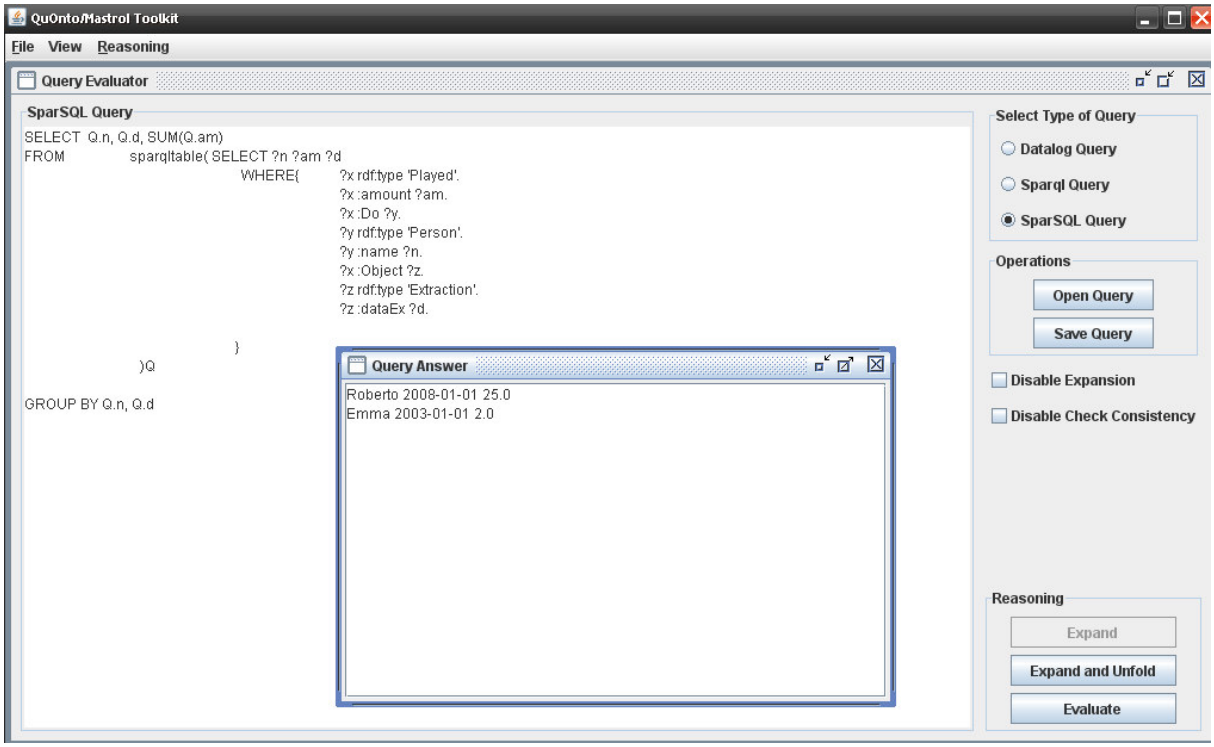
Risultato Query 2:



Risultato Query 3:



Risultato Query 4:



Riferimenti bibliografici

- G. De Giacomo, dispense del corso di “Seminario di Ingegneria del Software” anno accademico 2007/2008
- Emma Di Pasquale, slide presentazione SparSql anno accademico 2007/2008
- G. De Giacomo, D. Calvanese, D. Lembo, M. Lenzerini, R. Rosati, “Epistemic First-Order Queries over Description Logic Knowledge Bases” 2006 International Workshop on Description Logics Lake District, U.K., May 30 -- June 1, 2006
- G. De Giacomo, D. Calvanese, D. Lembo, M. Lenzerini, R. Rosati, “Path-based Identification Constraints in Description Logics”
- Emma Di Pasquale, Domenico Fabio Savo, Functional-Style Syntax for DL-Lite_A and Mappings
- Marco Colombetti, “Appunti delle lezioni di “Ingegneria della conoscenza” Politecnico di Milano anno accademico 2004/2005