Alma Mater Studiorum - Università di Bologna

Corso di laurea magistrale in Ingegneria e Scienze Informatiche - Cesena

Ontologia sui Track Day Event

Progetto per il corso di:
Web Semantico

Autore:

Luca Pascucci
luca.pascucci@studio.unibo.it
763205

Anno Accademico 2016 - 2017

Indice

Elenco delle figure

Capitolo 1

Track Day Event Ontology

Lo scopo di questo progetto è di approfondire ed utilizzare le tecnologie del Web Semantico per creare un'ontologia. Verrà descritto passo per passo come è stata realizzata, quali strumenti sono stati utilizzati e quali tecniche impiegate.

L'ontologia che si intende modellare riguarda gli eventi di Track Day che rappresenta per molti appassionati di motori la possibilità di girare in pista con la propria auto a prezzi solitamente contenuti. In particolare si vuole avere la possibilità di definire una pista con le sue caratteristiche principali (come ad esempio luogo, lunghezza, curve a destra o sinistra) dove si terrà l'evento, l'organizzatore responsabile dell'evento che può essere una persona come piuttosto un'organizzazione. Nell'evento vengono organizzati differenti turni per dare a tutti i partecipanti la possibilità di girare in pista. Esempi di triple di informazioni possono essere "L'evento X ha come partecipante la persona Y", oppure "La persona Y ha come veicolo il mezzo Z" e così via. Una prima modellazione è stata effettuata con RDF/RDF Schema, che ci permette di esporre la sintassi per definire schemi e vocabolari per i metadati, oltre a poter definire una serie di triple come quelle mostrate precedentemente, tramite espressioni nella forma soggetto-predicato-oggetto.

Tramite RDF/RDFS è possibile definire per esempio:

- rdfs:class: Definisce gruppi di individui che condividono alcune proprietà comuni. Ad esempio Luca e Filippo possono essere entrambi istanze della classe *Participant*, come si vedrà in seguito;
- rdfs:subclassOf: Fornisce la possibilità di organizzare le classi in gerarchie di specializzazione. Per esempio, la classe *SemiSlickTyres* è sottoclasse della classe *Tyres*;

- rdf:property: Rappresenta la parte di predicato della tripla soggetto-predicato-oggetto. Per esempio, le classi *Participant* e *Vehicle* sono collegate dalla proprietà *hasVehicle* (Participant hasVehicle Vehicle);
- rdf:domain: Definisce la classe che fa da soggetto per la proprietà;
- rdf:range: Definisce la classe che fa da oggetto per la proprietà.

Capitolo 2

Scenario Iniziale

Inizialmente è stato realizzato un grafo tramite la web app draw.io dove vengono riportate tutte le risorse ed entità che si intende modellare e le proprietà che le legano.

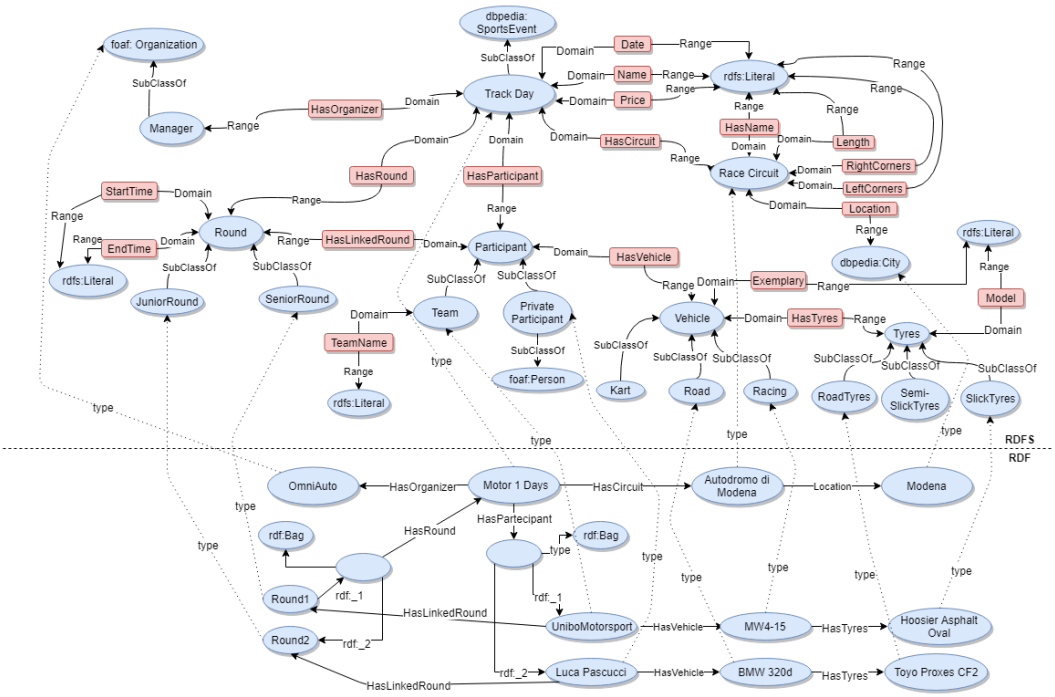

Figura 2.1: Grafo dell'ontologia

Come si può notare dal grafo, il punto centrale dello schema è la classe *TrackDay*. Corrisponde ad un evento sportivo organizzato da un *Manager*

che permette di far girare in pista i partecipanti, essa è associata anche ad un nome, una data ed un prezzo che rappresenta il costo per partecipare all'evento e si collega all'ontologia **DBpedia** essendo sottoclasse della classe *SocialEvent*.

Ogni istanza di *TrackDay* è associata ad un *RaceCircuit* classe che rappresenta un circuito dove avrà luogo l'evento. Del circuito vengono memorizzati dati prettamente "informativi" come nome, lunghezza (in metri), numero di curve a destra e sinistra e la città dove si trova utilizzando un'istanza di *City*, importata da **DBpedia**.

Il Manager, organizzatore del *TrackDay* può essere un'organizzazione per questo motivo è sottoclasse di *Organization*, importata dall'ontologia Friend Of a Friend (FOAF). All'evento è associata la classe *Participant* che identifica un partecipante al track day che può essere un privato oppure un team, con il primo sottoclasse della classe *Person* importata nuovamente dall'ontologia **FOAF**.

Sempre all'evento è associata la classe *Round* che rappresenta il turno pista ideato per definire dei momenti dove i partecipanti possono utilizzare il tracciato specificando un orario di inizio turno e di fine turno. Il *Round* può essere di due tipi *JuniorRound* o *SeniorRound* pensati per separare i partecipanti in base alle loro abilità ed esperienze di guida. Da notare il collegamento tra le classi *TrackDay*, *Participant* e *Round* che rappresenta al meglio il fulcro dell'evento. In ogni *TrackDay*, come spiegato in precedenza, vengono creati dei turni temporizzati per evitare che i partecipanti possano girare tutti contemporaneamente e per distinguere tra piloti amatoriali ed esperti, successivamente a questa distinzione vengono collegati i *Participant* ai *Round* più idonei.

L'ultima parte del grafo contiene le classi *Vehicle* e *Tyres* dove la prima identifica il veicolo utilizzato dal partecipante per girare in pista mentre la seconda specifica le gomme montate sul veicolo. Per ognuna sono state create delle sottoclassi che rappresentano delle specializzazioni, difatti sotto il punto di vista dei veicoli sono state effettuate 3 separazioni in kart, veicoli stradali e veicoli da corsa mentre per le gomme sono state create le classi che identificano gomme stradali, gomme sportive, gomme da corsa.

La modellazione effettuata tramite RDF/RDFS è osservabile nel dettaglio all'interno del file *trackdayevent.rdf* allegato. Esso è stato scritto manualmente tramite l'editor rdfEditor con sintassi RDF/XML, ed è stato poi

correttamente validato tramite l'RDF Validator del W3C raggiungibile al sito https://www.w3.org/RDF/Validator/

Capitolo 3

Sviluppo dell'ontologia - Protégé

Dovendo specificare ulteriori vincoli ed esprimere una semantica più dettagliata riguardo alle informazioni inserite, il dominio è stato infine modellato con OWL.
Per la moderazione dell'ontologia si è scelto di usare Protégé, un software free e open source, sviluppato dallo Stanford Center. Protègè, come Eclipse, è un framework per il quale vari progetti mettono a disposizione dei plugin; inoltre fornisce un'interfaccia grafica per definire ontologie. Essa comprende anche classificatori deduttivi per convalidare la consistenza dei modelli e inferire nuove informazioni sulla base dell'analisi di una ontologia. In particolare Protègè consente di modellare:

- Classi OWL;
- Object Property: proprietà che legano due oggetti;
- Data Property: proprietà che legano un oggetto ad un dato definito nel dizionario;
- Gerarchie.

3.1 Modellazione dell'ontologia

Il primo passo per la modellazione dell'ontologia è la scelta del nome. Per questo progetto si è scelto Track Day Event, il cui acronimo è TDE. L'IRI dell'ontologia è http://www.unibo.it/lucapascucci/ontology/trackdayevent-ontology. Ovviamente si tratta di un IRI fittizio in quanto il progetto è a scopo didattico. Tuttavia, potrebbe essere sostituito in futuro con un

IRI adeguato, semplicemente modificandolo dall'interfaccia di Protégé. Si inseriscono poi quattro annotazioni, le prime due utilizzando il vocabolario Dublin Core:

- dc:Title : contiene il titolo dell'ontologia;
- dc:Description : contiene la descrizione dell'ontologia;
- rdfs:isDefinedBy : contiene l'autore dell'ontologia;
- owl:versionInfo : contiene il numero della versione dell'ontologia.

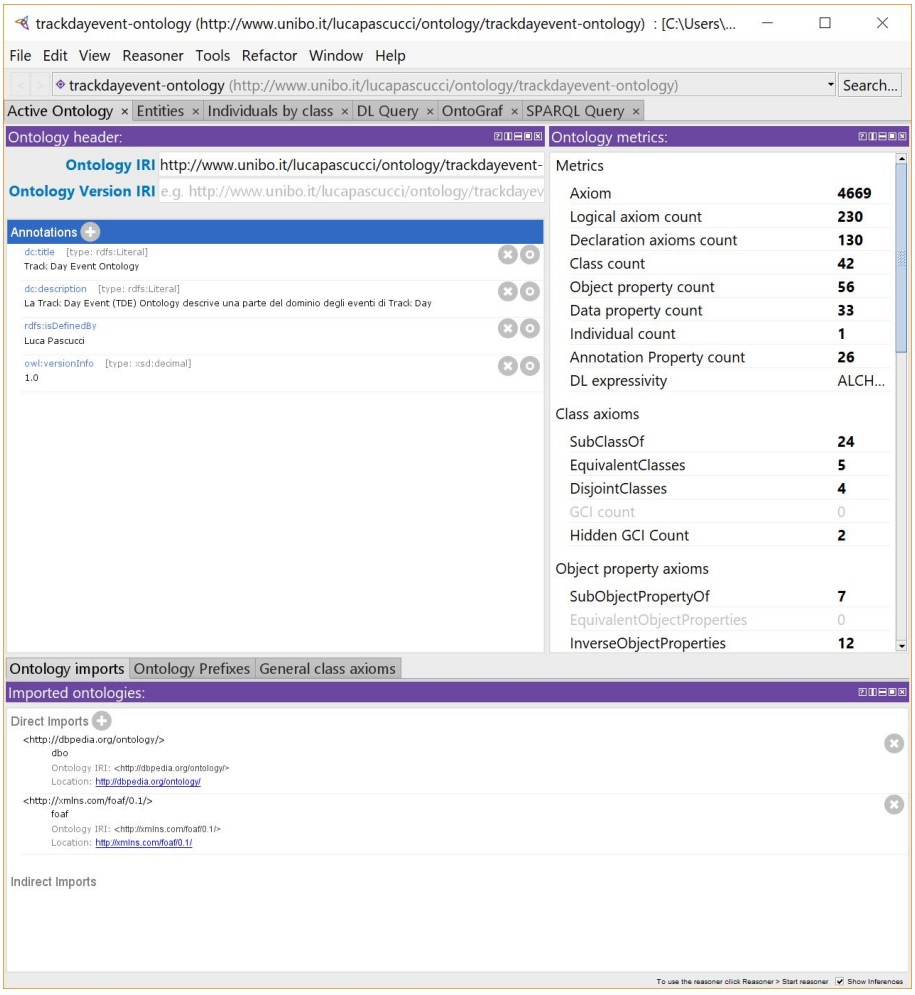

Figura 3.1: Schermata principale di Protégé che mostra le informazioni generali sull'ontologia

La modellazione dell'ontologia procede poi con:

- Importazione delle ontologie esterne;
- Modellazione delle classi e relative gerarchie;
- Modellazione delle object property e data property;
- Modellazione di vincoli e restrizioni tramite OWL;
- Inserimento degli individui nell'ontologia.

3.2 Importazione di ontologie esterne

L'importazione di ontologie esterne consente di collegare l'ontologia del progetto al mondo esterno, in modo da integrarla con i linked data presenti attualmente sulla rete. I principali metodi per creare questi collegamenti sono:

- Utilizzare la classe rdfs:subClassOf: a partire da una classe di un'ontologia importata, si crea una sotto classe per quella classe, che viene poi utilizzata per l'ontologia;
- Utilizzare la proprietà owl:equivalentClass: questa proprietà built-in collega una descrizione di una classe a quella di un'altra classe. Il significato di tale assioma è che le descrizioni delle due classi hanno la stessa estensione;
- Utilizzare la proprietà owl:sameAs: questa proprietà built-in collega un individuo ad un altro individuo. owl:sameAs indica che due URI si riferiscono attualmente alla stessa cosa e, di conseguenza, che gli individui hanno la stessa identità.

Nel corso di questo progetto verrà utilizzato il primo metodo proposto, ovvero la creazione di sottoclassi per classi importate. Protégé offre la possibilità di importare ontologie esterne sia da file salvato sul pc locale, che da remoto fornendogli l'URL.

Come precedentemente illustrato nello scenario iniziale, sono state importate le seguenti ontologie:

- **FOAF**: Friend Of A Friend è un'ontologia atta a descrivere persone, con le loro attività e le relazioni con altre persone e oggetti. FOAF

permette a gruppi di persone di descrivere quel fenomeno noto come social network senza la necessità di un database centralizzato. FOAF è un vocabolario descrittivo espresso in Resource Description Framework (RDF) ed è definita usando Web Ontology Language (OWL);

- **dbo**: DBpedia è un progetto che mira a estrarre il contenuto strutturato dalle informazioni, creato come parte del progetto Wikipedia. Queste informazioni strutturate viene quindi reso disponibile sul World Wide Web. DBpedia consente agli utenti di interrogare semanticamente le relazioni e le proprietà associate con le risorse di Wikipedia, tra cui i collegamenti ad altri insiemi di dati correlati.

3.3 Modellazione delle classi

Tutte le classi che verranno modellate saranno sottoclassi della classe owl:Thing. Si modellano le seguenti classi:

- TrackDay
- Manager, sottoclasse della classe foaf:Organization
- RaceCircuit
- Participant, suddivisa a sua volta in:
  - Team
  - Private Participant, sottoclasse della classe foaf:Person
- Vehicle, suddivisa a sua volta in:
  - Kart
  - Road
  - Racing
- Tyres, suddivisa a sua volta in:
- RoadTyres
- Semi-SlickTyres
- SlickTyres
- Round, suddivisa a sua volta in:

- JuniorRound
- SeniorRound

In seguito sono state esplicitate le disgiunzioni tra le varie classi. In sintesi, un assioma di disgiunzione tra due classi indica che un elemento non può essere istanza di entrambe le classi. Nel caso in esame ad esempio è utile specificare che *JuniorRound* e *SeniorRound* (entrambi sottoclassi di Round) sono disgiunte, non ci potrà quindi essere un'istanza che è allo stesso tempo *JuniorRound* e *SeniorRound*.

Infine vengono importate le classi *City* e *SportsEvent* da **DBpedia**, e *Person* e *Organization* da **FOAF**. Sono incluse anche alcune classi automaticamente aggiunte dopo l'import di **FOAF**, che tuttavia non sono utilizzate.

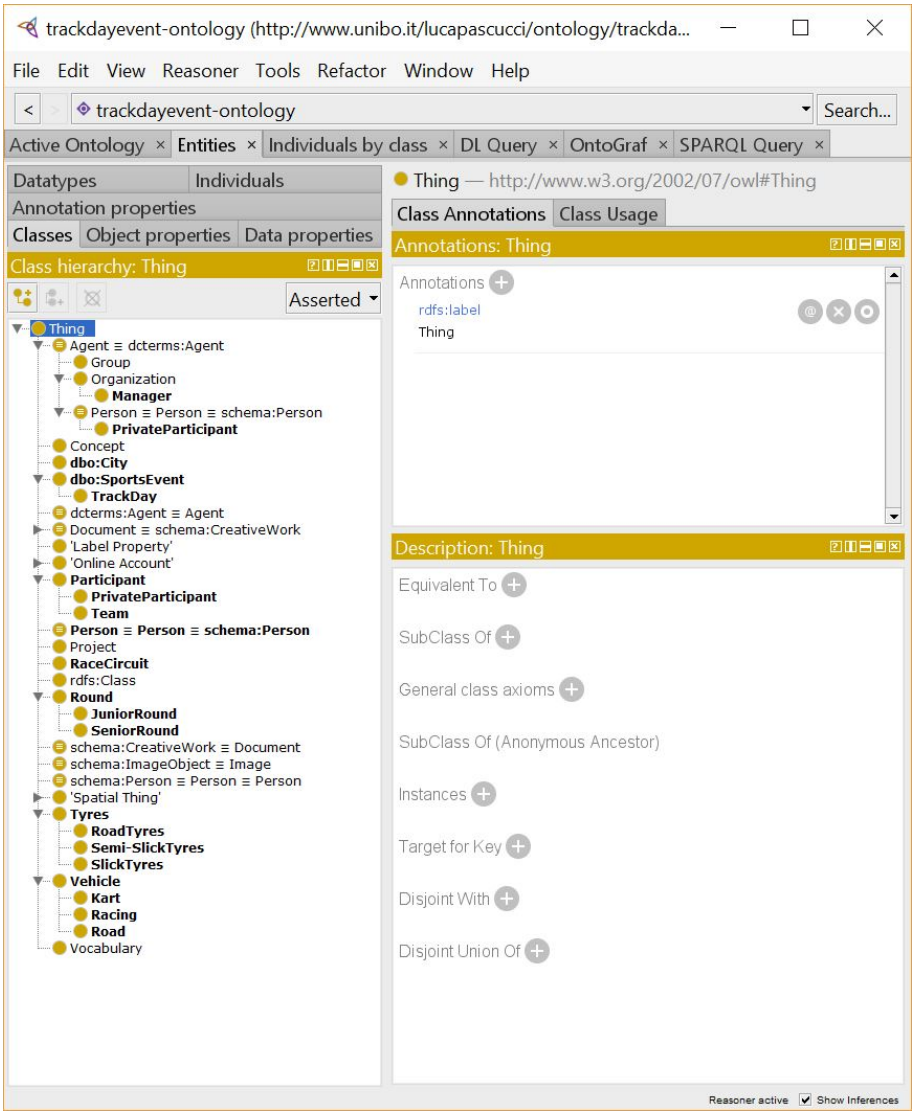

Figura 3.2: Gerarchia delle classi

3.4 Modellazione delle Object Property e Data Property

Di seguito viene mostrato l'insieme delle Object Property. Sono incluse anche alcune proprietà automaticamente aggiunte dopo l'import di FOAF, che tuttavia non sono utilizzate.

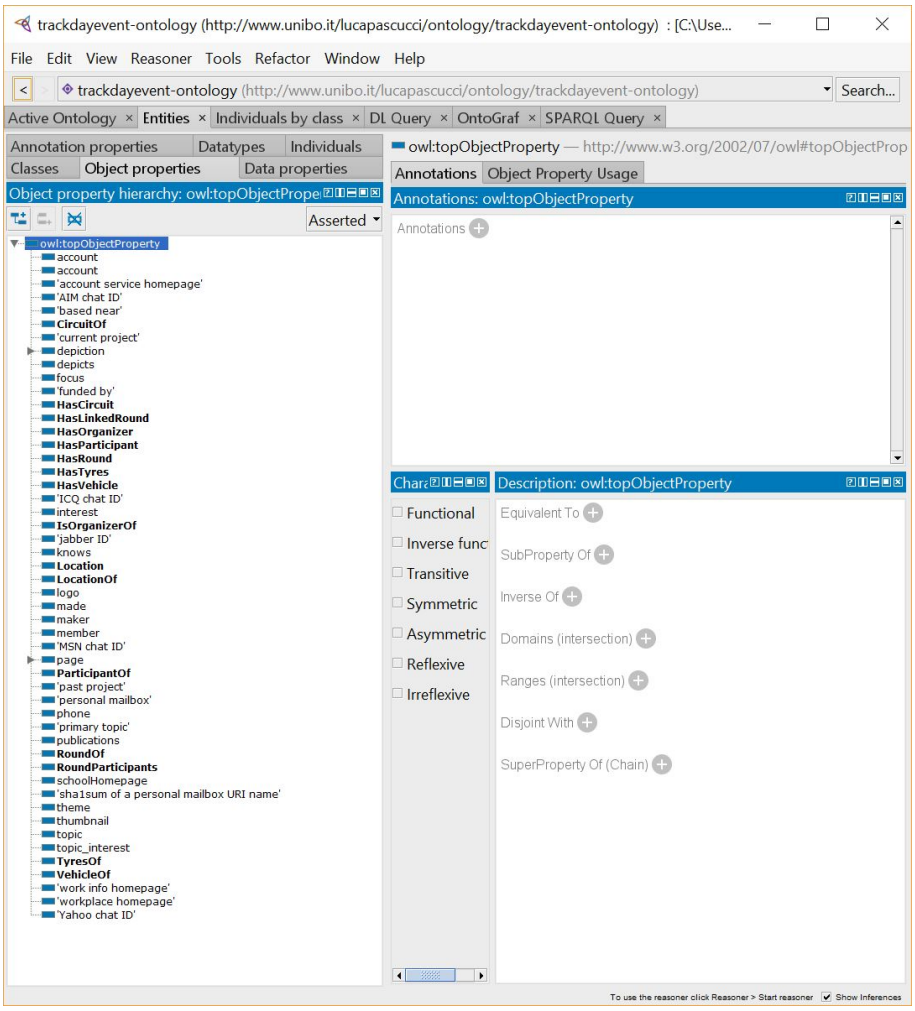

Figura 3.3: Object Properties

Per ogni Object property si definisce il Domain e il range. Se una proprietà è l'inversa di una già definita, è sufficiente riempire il campo Inverse Of, selezionando la proprietà precedente: il reasoner sarà in grado di riconoscere il rispettivo domain e range tramite inferenza come vedremo in seguito.

Di seguito viene riportato l'insieme delle Data Property. Come per le Object Property, anche qua vengono incluse automaticamente alcune proprietà dopo l'import di FOAF.

3.5 Restrizioni su Object Property e Data Property

Dopo aver definito la gerarchia delle classi e le proprietà con i relativi domain e range, sono state applicate le corrette restrizioni sulle cardinalità delle proprietà. Di seguito viene riportato l'esempio sulla classe *TrackDay* che ha le seguenti caratteristiche:

- Deve essere organizzata esattamente da 1 *Manager*;
- Deve avere esattamente 1 data di svolgimento;
- Deve avere esattamente 1 *RaceCircuit* dove avverrà l'evento;
- Deve avere almeno un *Round* di giri in pista;
- Deve avere almeno un *Participant* all'evento;
- Deve avere un costo espresso tramite *Price*;
- Deve avere esattamente un nome.

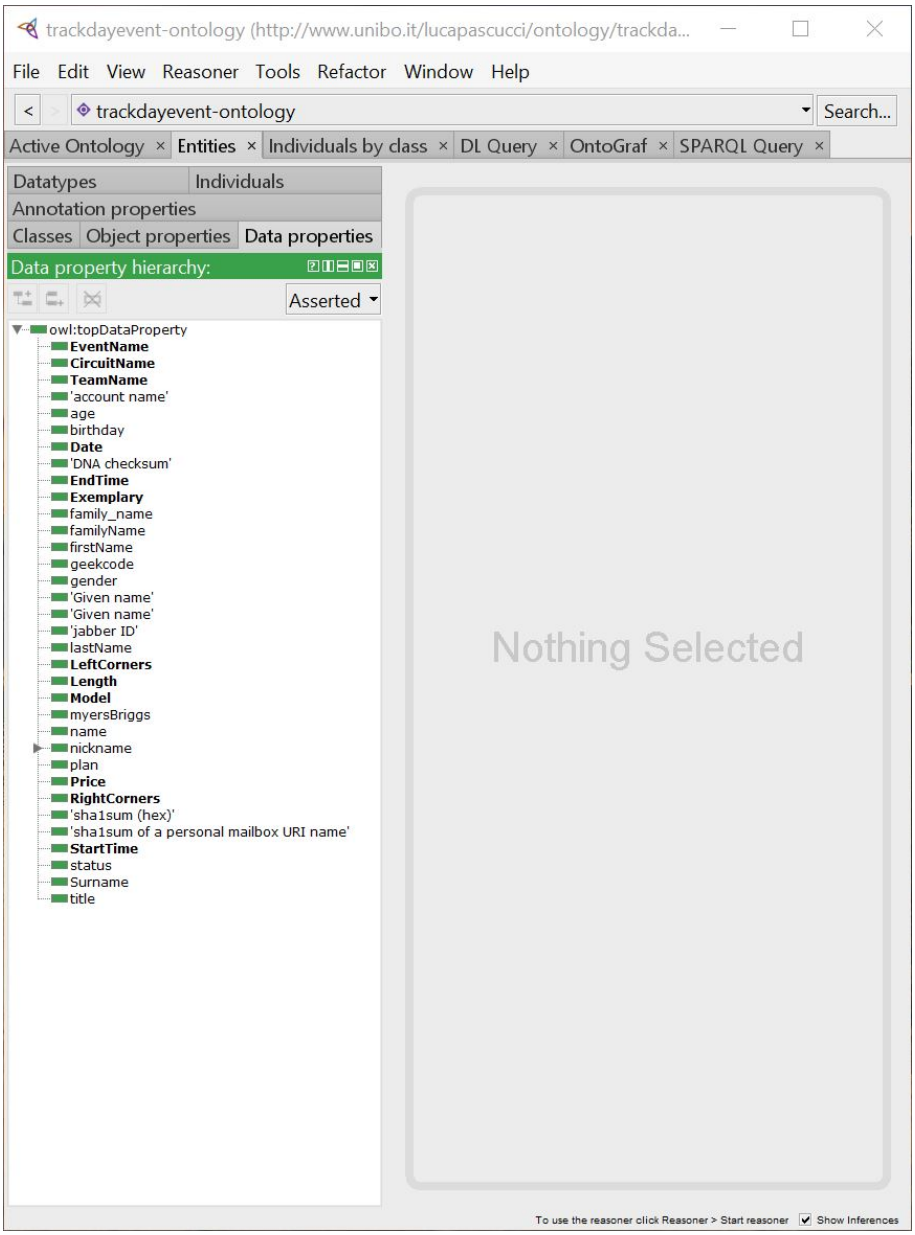

Figura 3.4: Data Properties

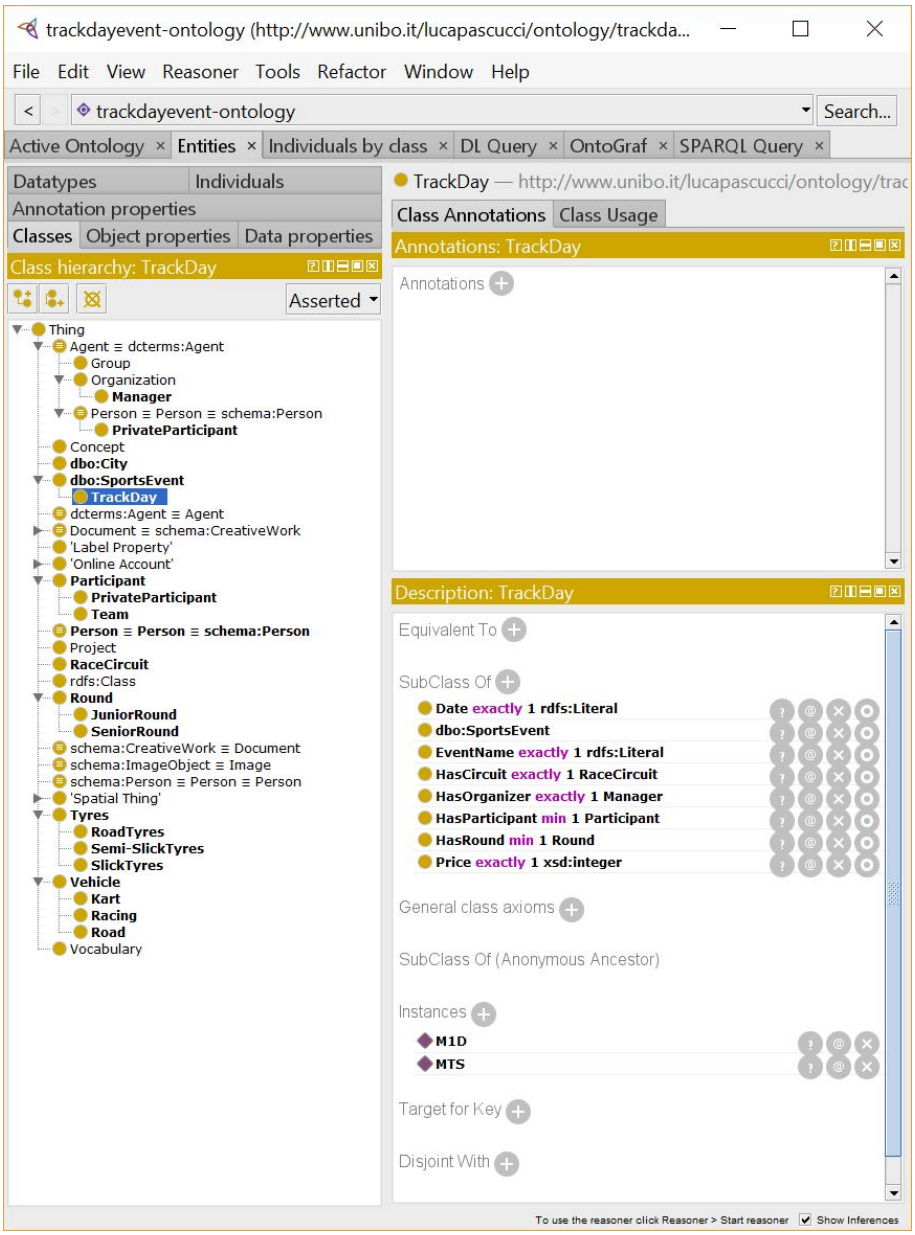

Figura 3.5: Classe *TrackDay* nel dettaglio, con le sue restrizioni

3.6 Inserimento individui

L'inserimento degli individui corrisponde alla creazione delle varie istanze delle varie classi. Per ogni individuo è possibile inserire delle asserzioni sulle proprietà, collegando i diversi individui tramite le proprietà precedentemente definite. In figura viene mostrata la sezione di Protégé dedicata alla creazione e modifica degli individui.

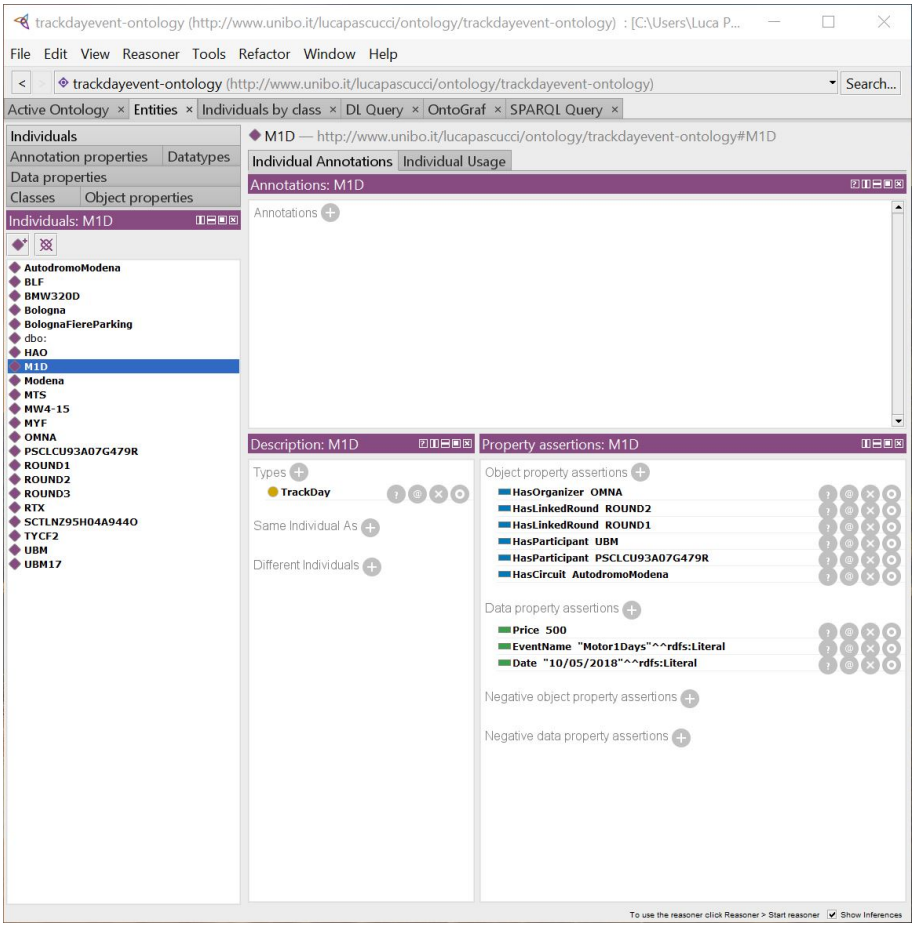

Figura 3.6: Individui dell'ontologia, con dettaglio su un individuo di tipo *TrackDay*

Capitolo 4

Ontologia completa

OntoGraf fornisce la possibilità di navigare in maniera interattiva le relazioni di ontologie OWL. Nella figura sottostante viene mostrato il grafo dell'ontologia completa estratta tramite questo plugin.

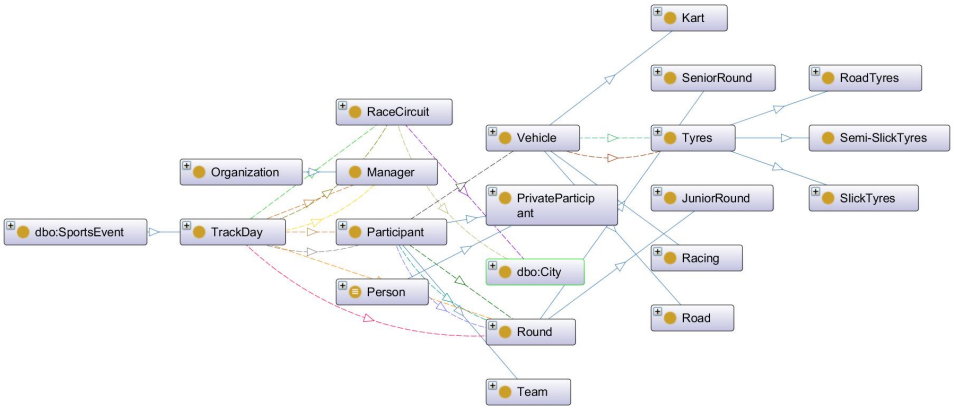

Figura 4.1: Grafo dell'ontologia visualizzato con OntoGraf

Capitolo 5

L'utilizzo del reasoner

Un reasoner è un software in grado di svolgere dei ragionamenti su delle basi di conoscenza adeguatamente formalizzate. Il ragionamento è inteso come la capacità di elaborare la base di conoscenza secondo alcune regole, in modo da validare ed analizzare la base di conoscenza stessa.

Il reasoner può essere utilizzato sia per la validazione, ovvero il controllo di coerenza interna della base di conoscenza, ma anche per l'inferenza. Per mostrare più nel dettaglio le potenzialità del reasoner, sono state create due classi: *CheapTrackDay* e *ExpensiveTrackDay*.

La prima rappresenta un evento di Track day con prezzo minore di 200 euro e la seconda con un prezzo maggiore di 200 euro. Come si può notare dall'immagine sottostante, è sufficiente esprimente la nuova classe come equivalente a: *TrackDay* and (*Price* some xsd:integer[>= 200]).
Facendo partire il reasoner, esso riconosce che la nuova classe è sottoclasse di *TrackDay* e trova anche un'istanza definita precedentemente, tramite inferenza.

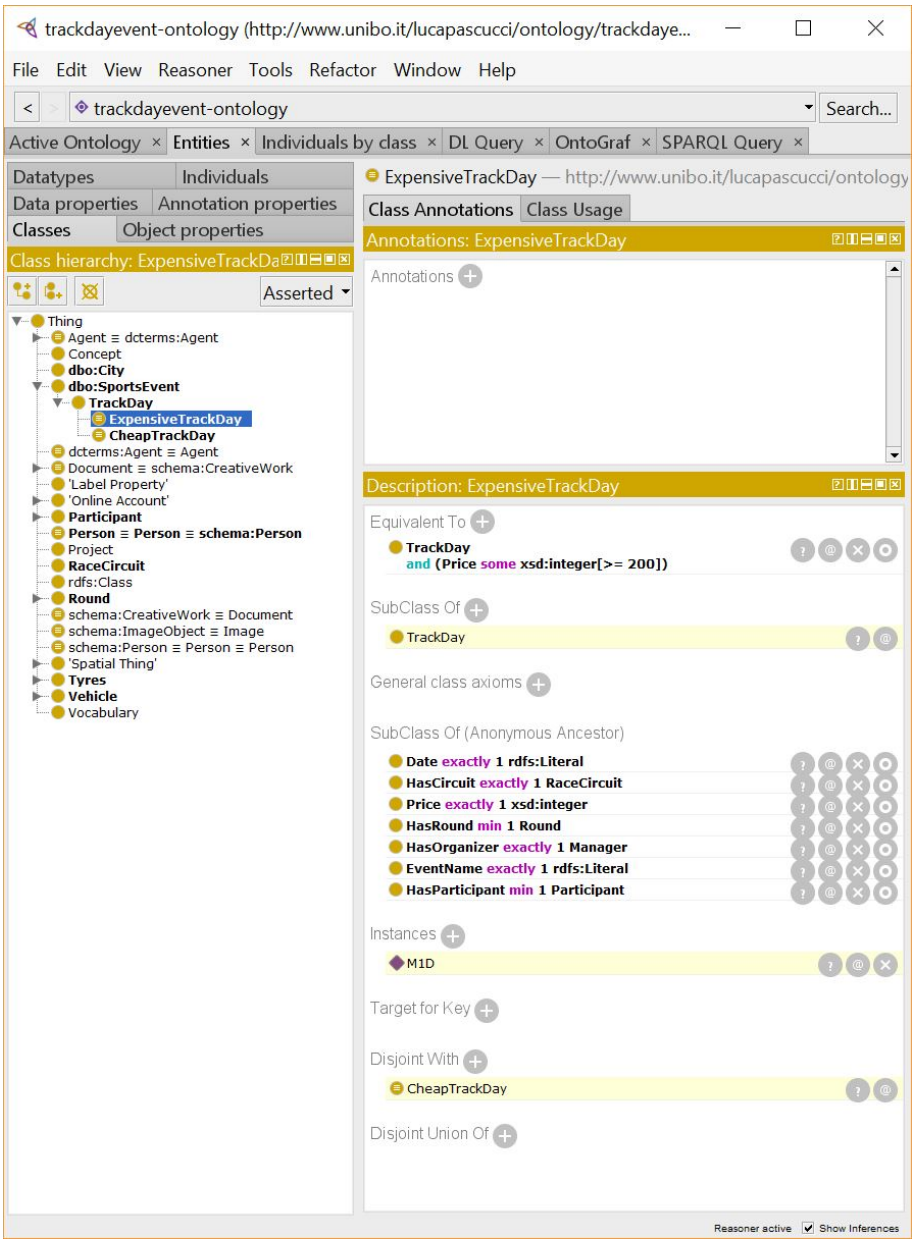

Figura 5.1: Utilizzo del reasoner sulla classe *ExpensiveTrackDay*

Capitolo 6

Query SPARQL

SPARQL (acronimo ricorsivo di SPARQL Protocol And RDF Query Language) è un linguaggio di interrogazione standardizzato e interoperabile per i dati rappresentati tramite RDF. Soddisfa o supera l'SQL nelle sue capacità e potenza, pur mantenendo gran parte della sintassi familiare ed è uno degli elementi chiave delle tecnologie legate al paradigma noto come web semantico consentendo l'estrazione di informazioni dalle basi di conoscenza distribuite sul web.

Sono state eseguite delle query sull'ontologia prodotta per poter effettuare un ulteriore controllo e valutare se i risultati ottenuti sono coerenti con quelli attesi. Di seguito vengono mostrate alcune delle query eseguite.

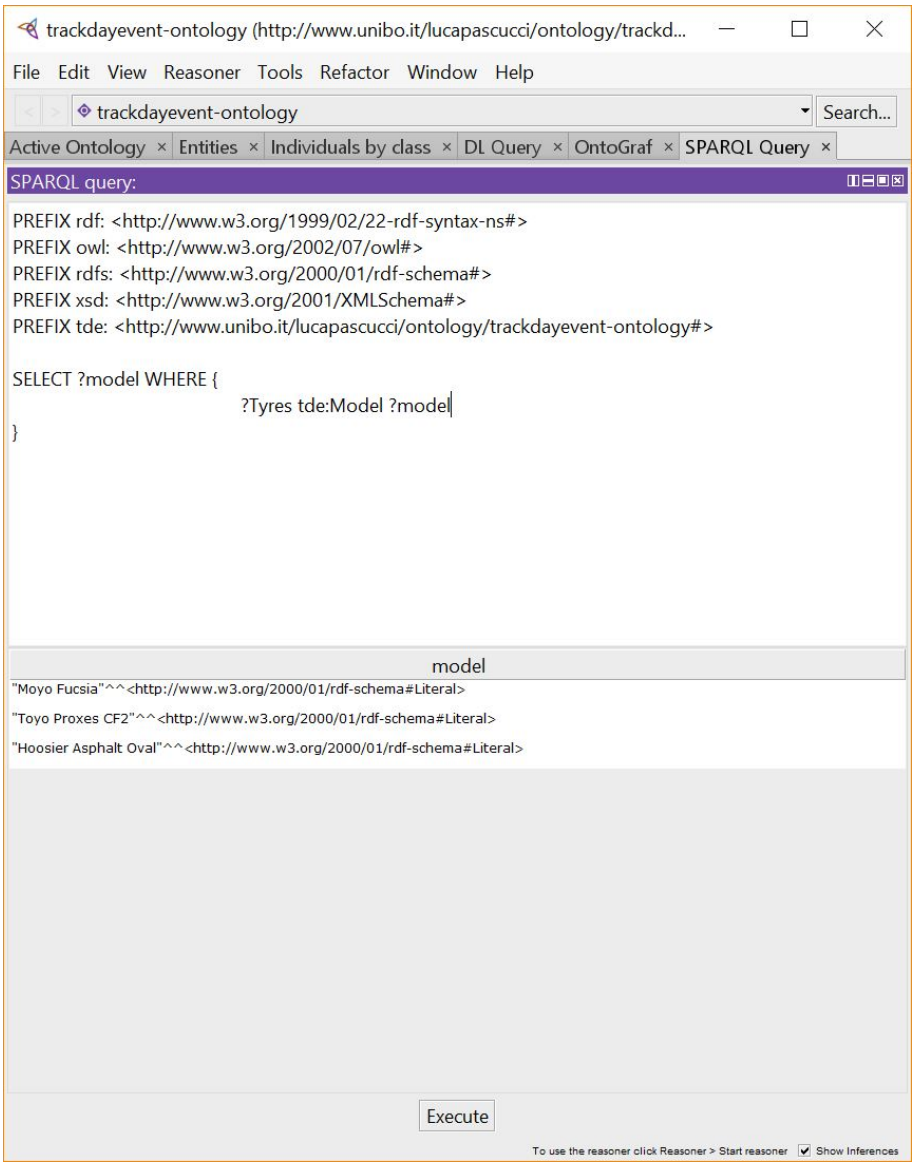

Figura 6.1: Lista di tutti i modelli di gomme

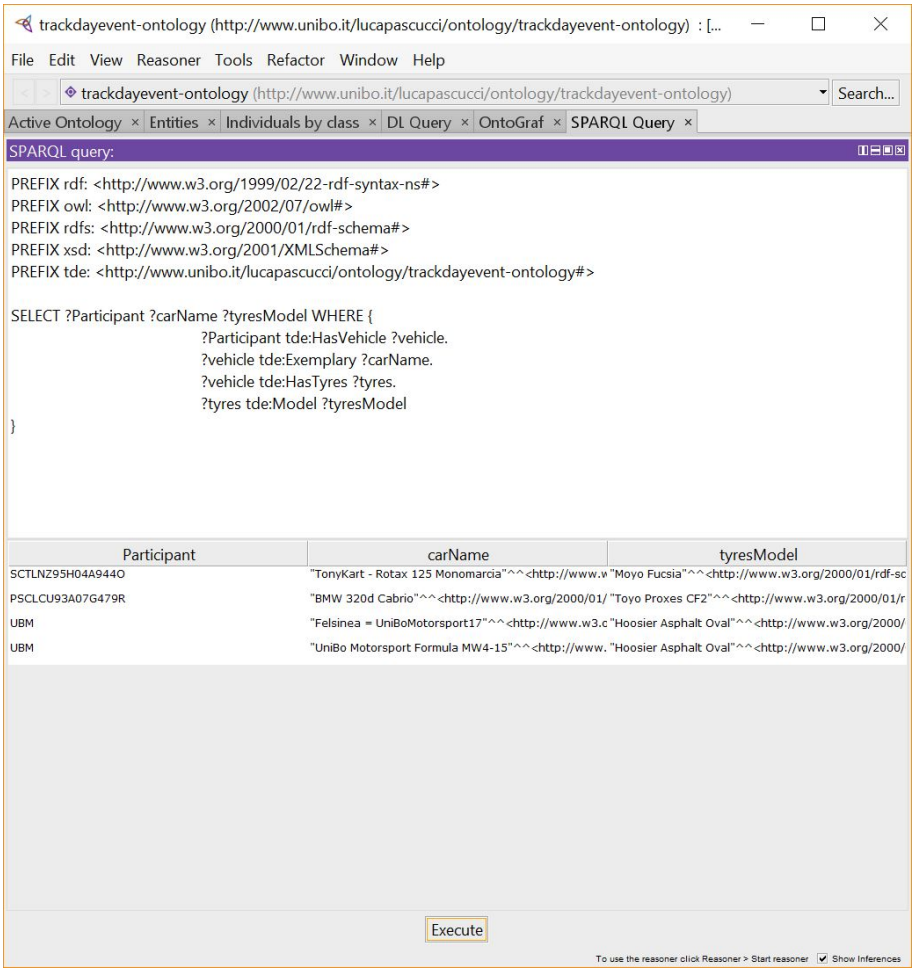

Figura 6.2: Lista di tutti i partecipanti con relativi veicoli e gomme montate dai veicoli

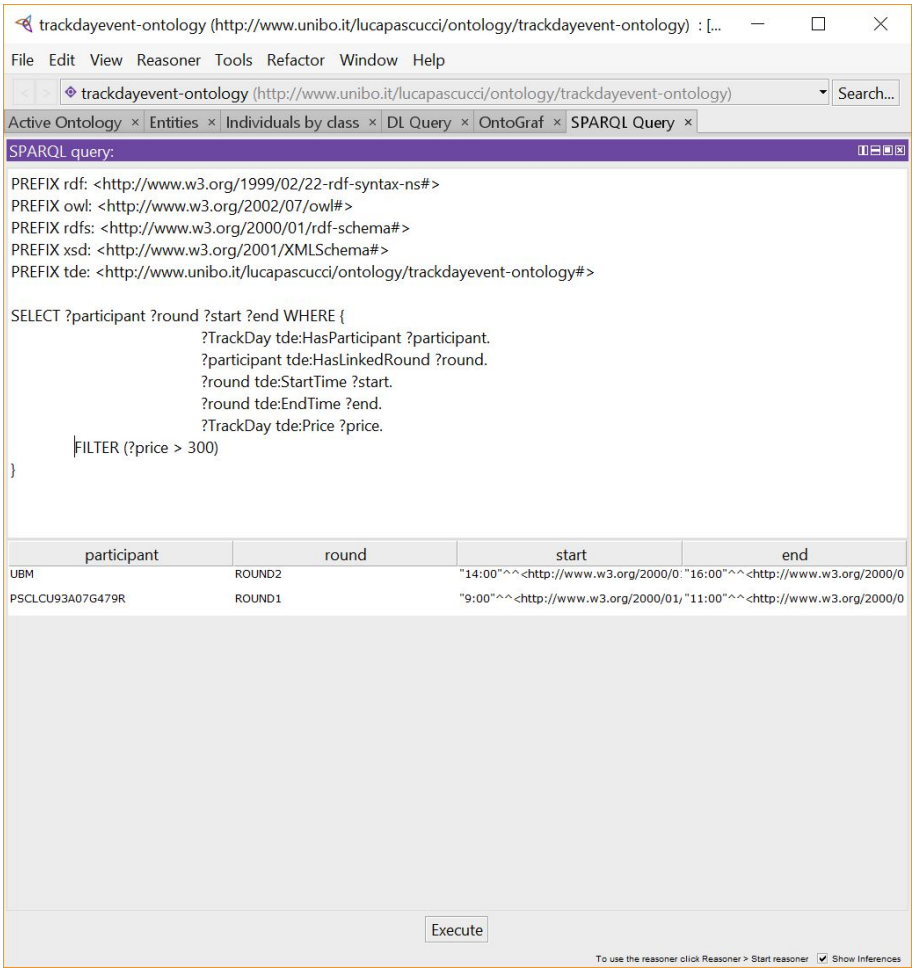

Figura 6.3: Lista dei partecipanti, round e relative ore di inizio e fine turno, ai TrackDay con prezzo maggiore di 300 euro

Capitolo 7

Conclusione

La realizzazione di questo progetto ha permesso di approfondire e mettere in pratica i concetti appresi a lezione e rappresenta un buon esempio delle potenzialità offerte dal web semantico, il cui scopo principale ricordiamo essere quello di collegare il più possibile i dati e renderli interpretabili dalle macchine. È stato approfondito inoltre l'utilizzo del software open source Protégé, punto di riferimento per lo sviluppo di ontologie.
Questa ontologia è anche un buon punto di partenza per poter realizzare nuove ontologie. Di seguito sono elencati i possibili miglioramenti al progetto:

- Inserimento di classi per modellare caratteristiche tecniche del tracciato (Tipologia asfalto, dislivello assoluto, anno di realizzazione, ...);
- Inserimento di classi per modellare i tempi sul giro;
- Inserimento di classi per modellare tutti i dati atmosferici collegati al tracciato e che influiscono sulle prestazioni (Temperatura aria, temperatura asfalto, condizioni meteorologiche, direzione e intensità vento, ...);
- Inserimento di classi per migliorare la rappresentazione dei veicoli (motore, telaio, freni, peso, tipologie di gomme, ...);
- Inserimento di classi per modellare il setup dei veicoli (camber, convergenza, ripartizione frenante, ripartizioni peso, tipologia rapporti, pressione gomme, ...).