GraphDB

Esercizi Reasoning

Manuel Fiorelli

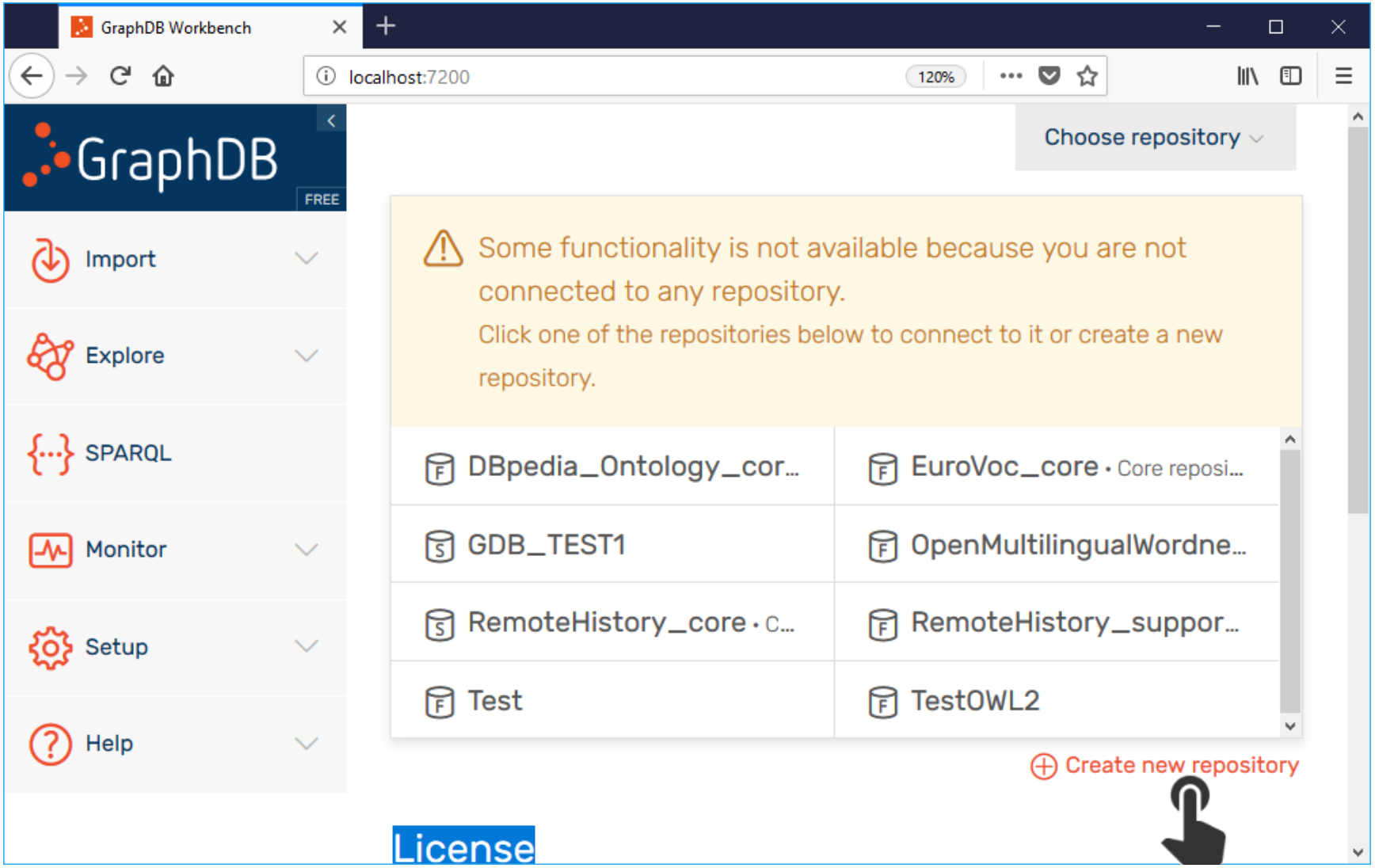
fiorelli@info.uniroma2.it

Creazione di un repository (1/6)

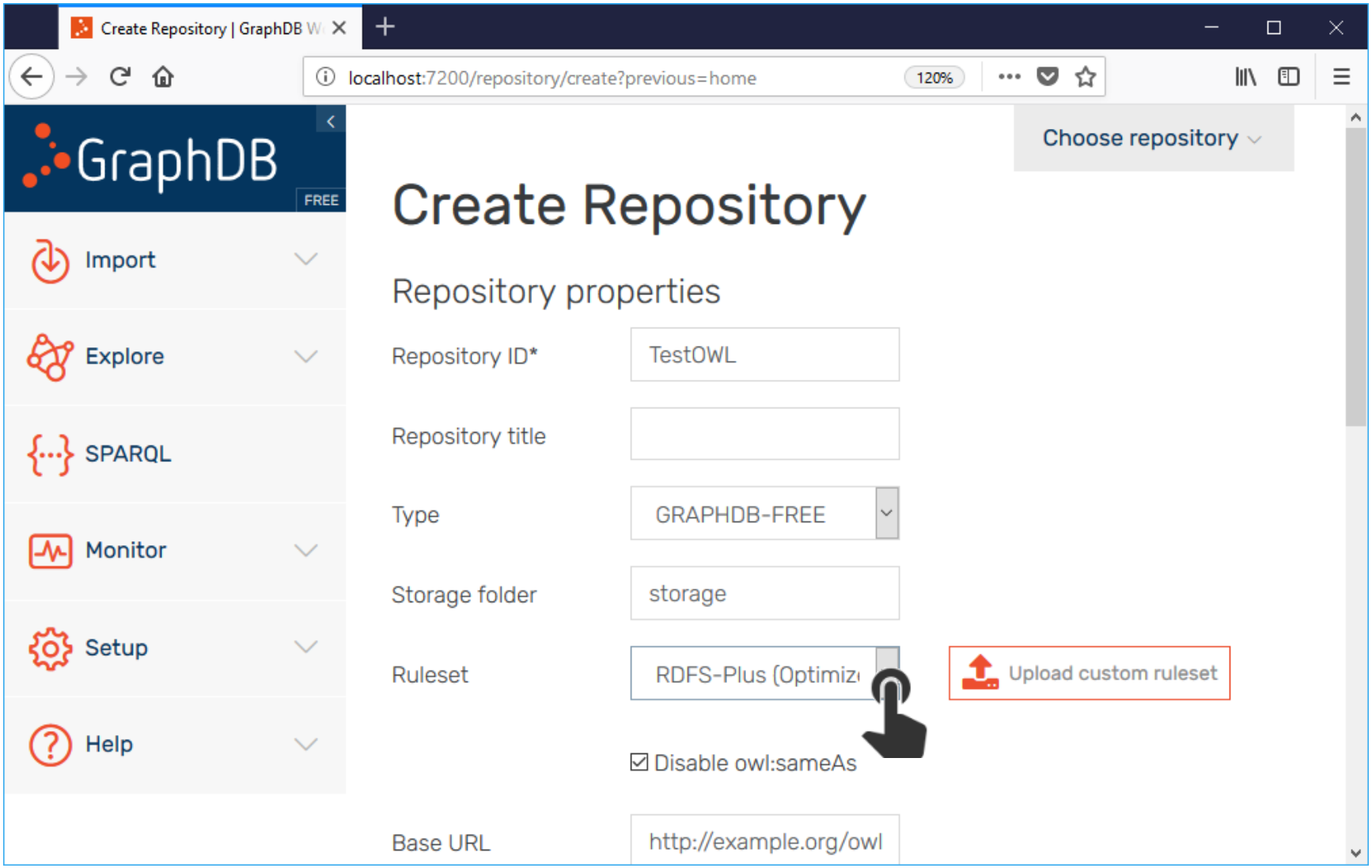
Per prima cosa dobbiamo creare un repository:

- **RepositoryID**: *TestOWL*
- **Ruleset**: *OWL2-RL*
  - Si noti che dopo aver selezionato questo ruleset, *la spunta sull'opzione disable owl:sameAs viene tolta* automaticamente (cioè l'ottimizzazione viene attivata)

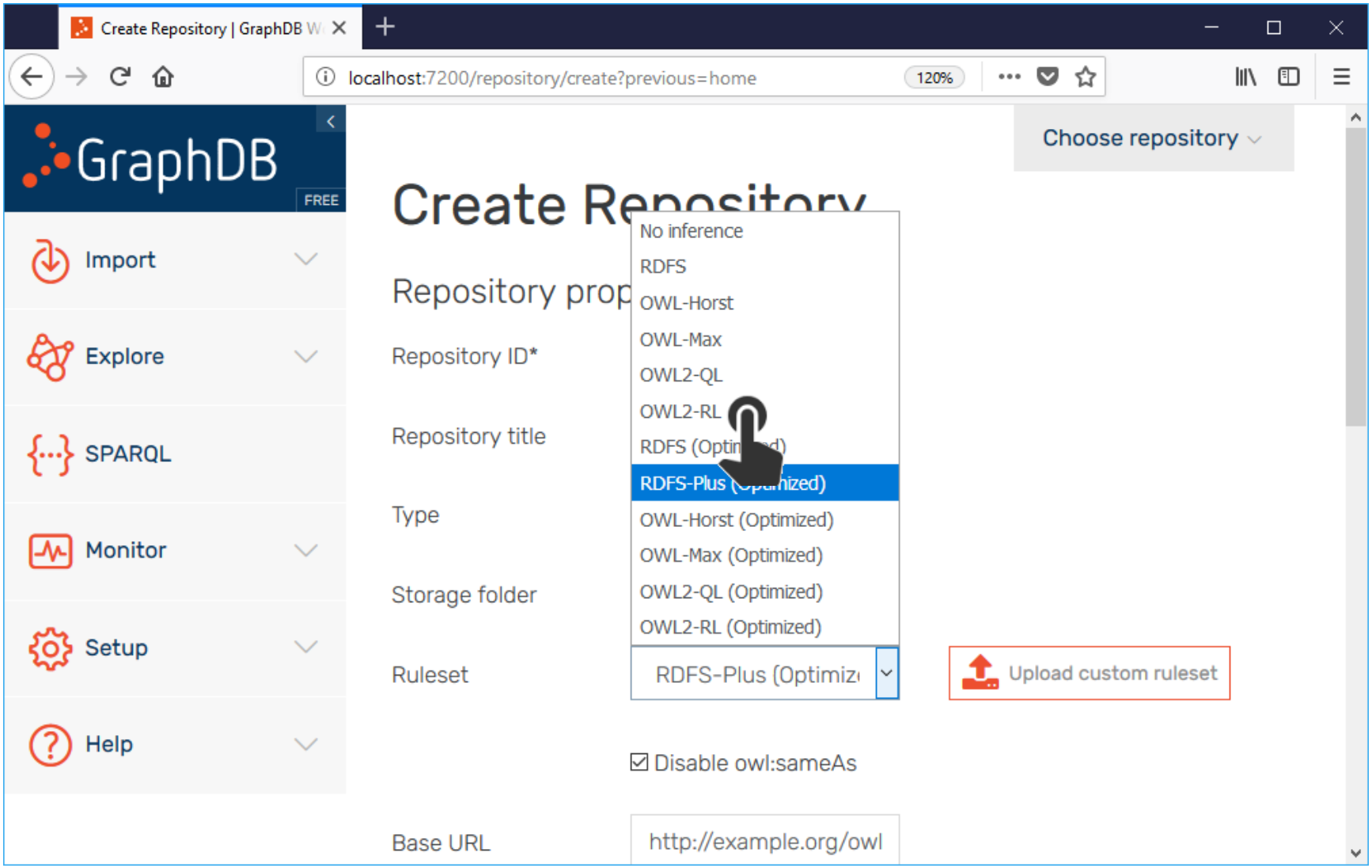
Creazione di un repository (2/6)



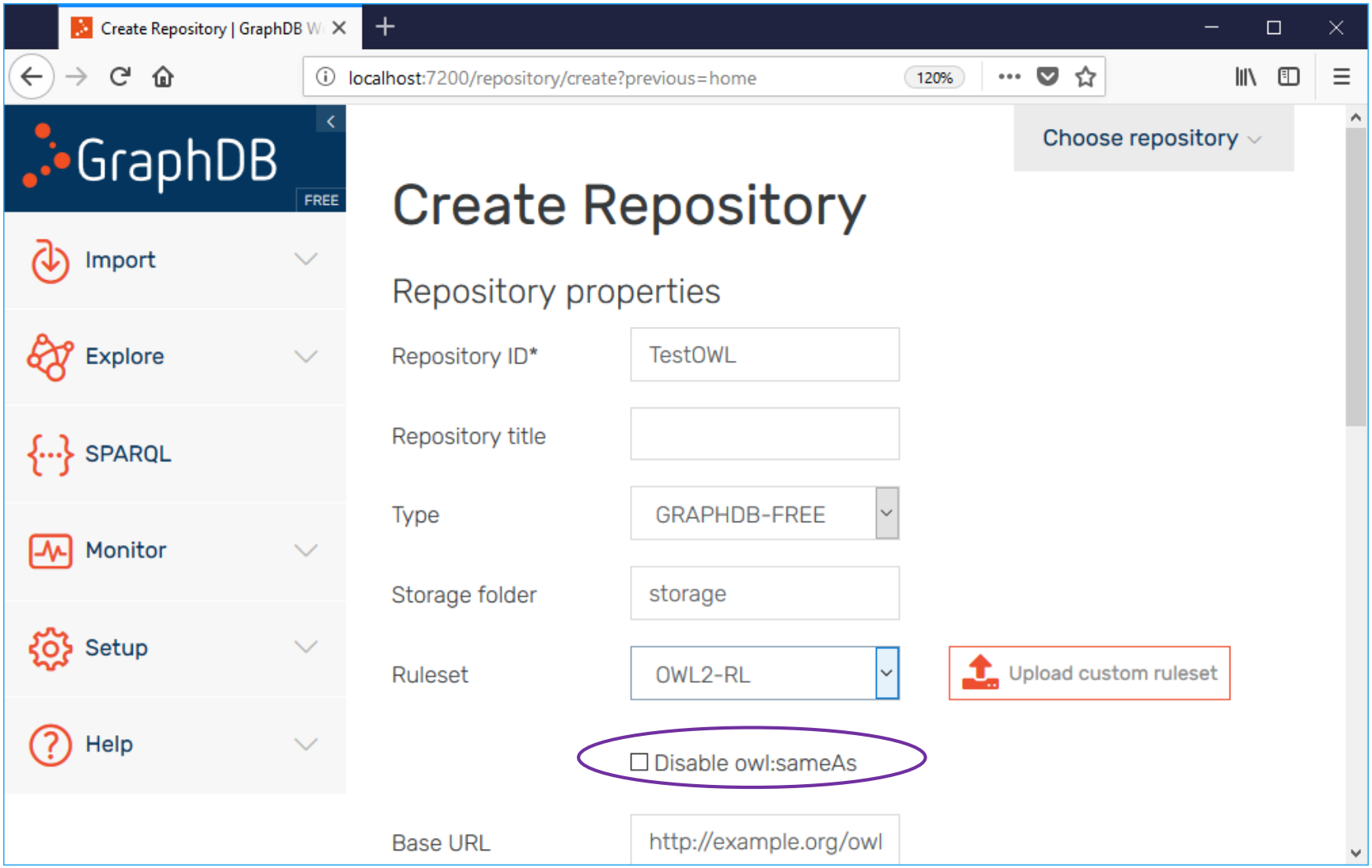
Creazione di un repository (3/6)



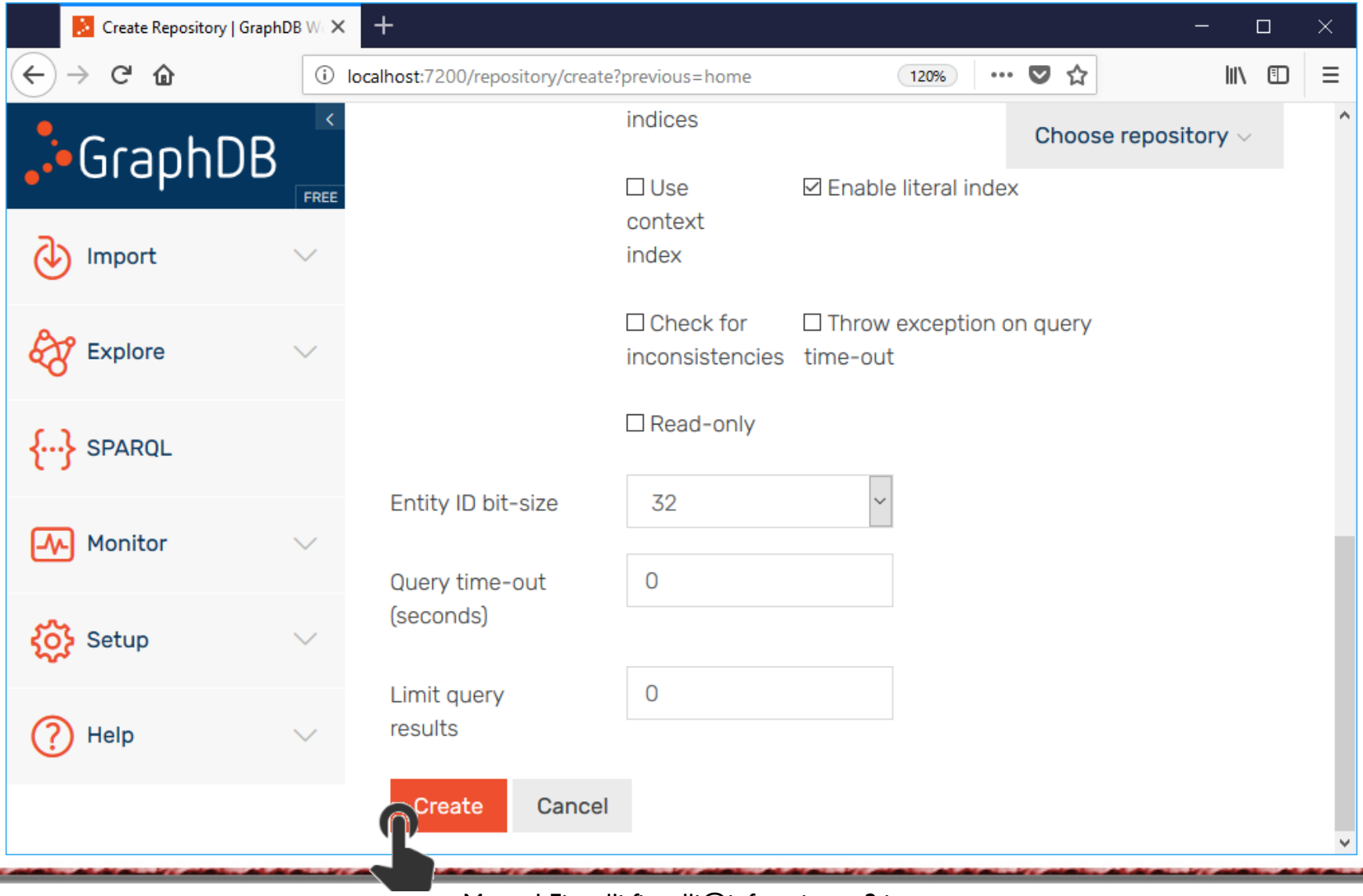
Creazione di un repository (4/6)



Creazione di un repository (5/6)

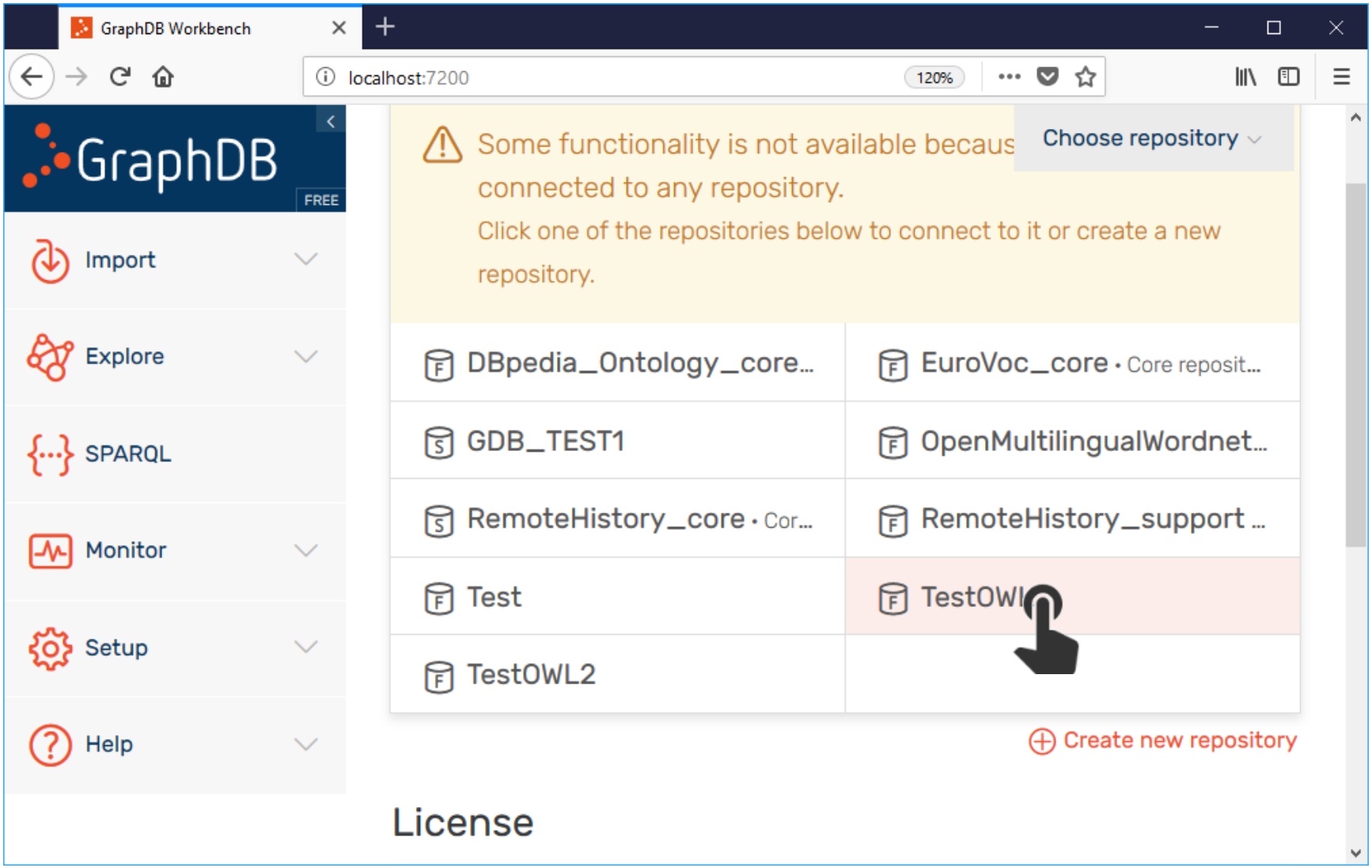


Creazione di un repository (6/6)

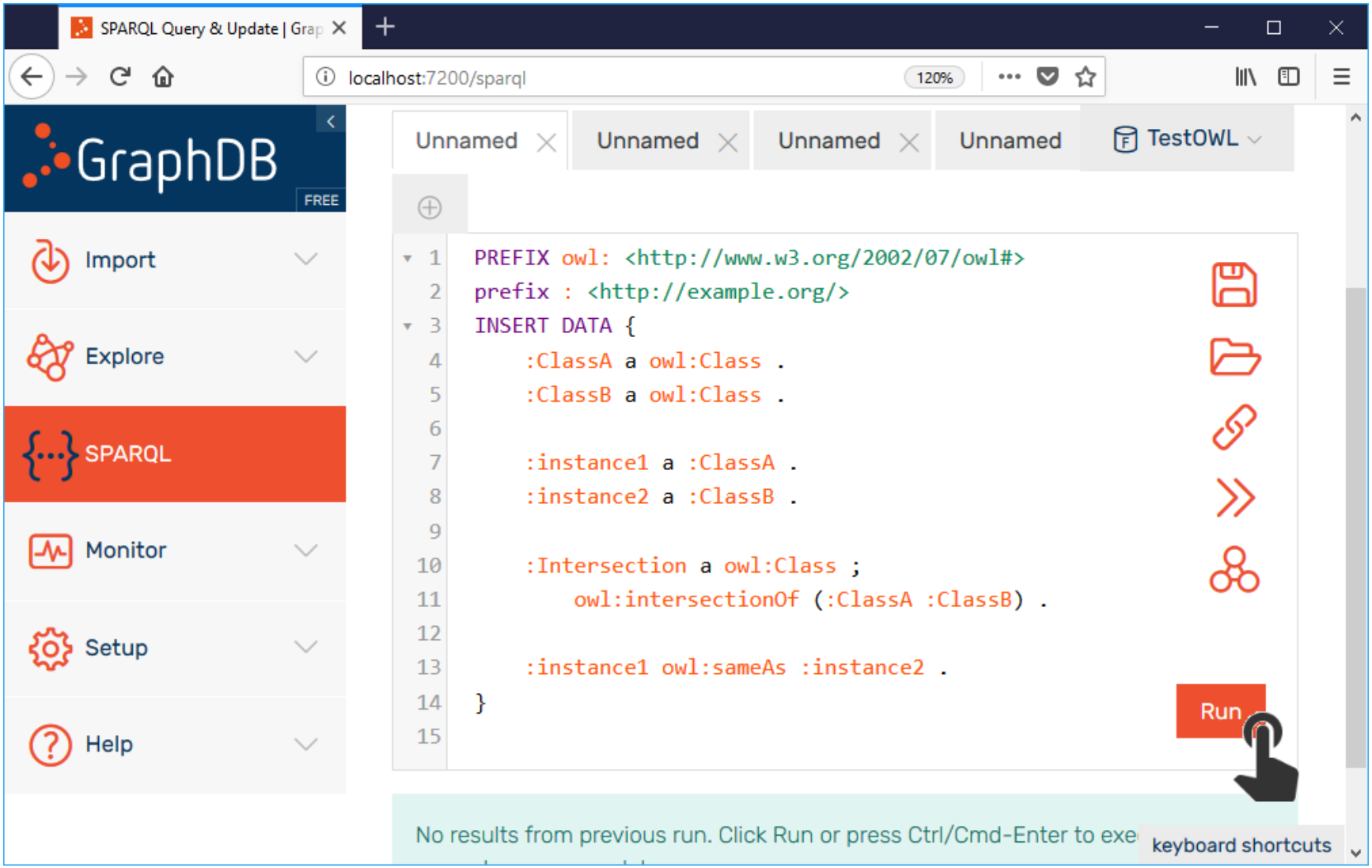


Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

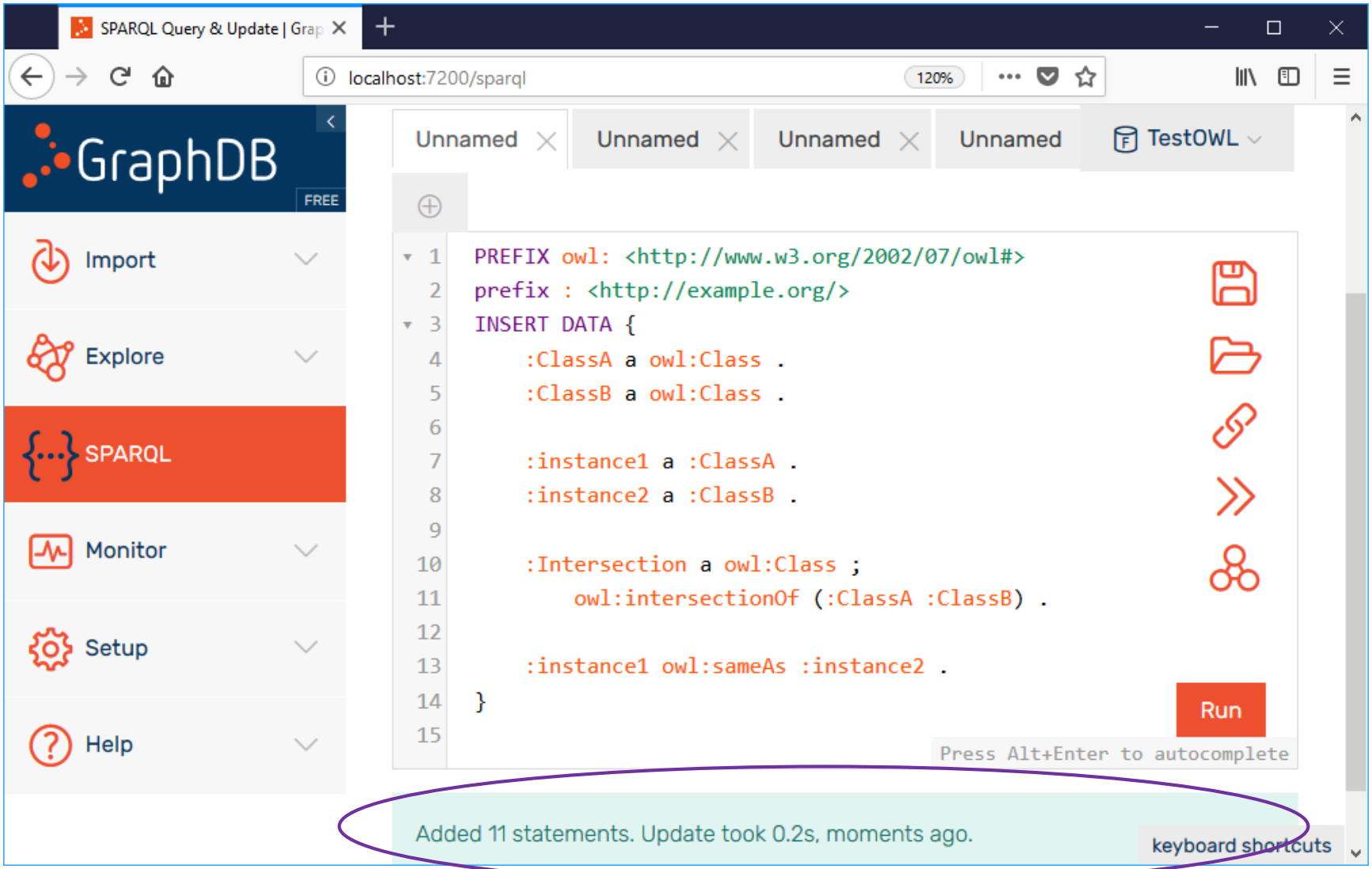
Selezione del repository



Aggiunta di triple RDF con INSERT DATA (1/2)



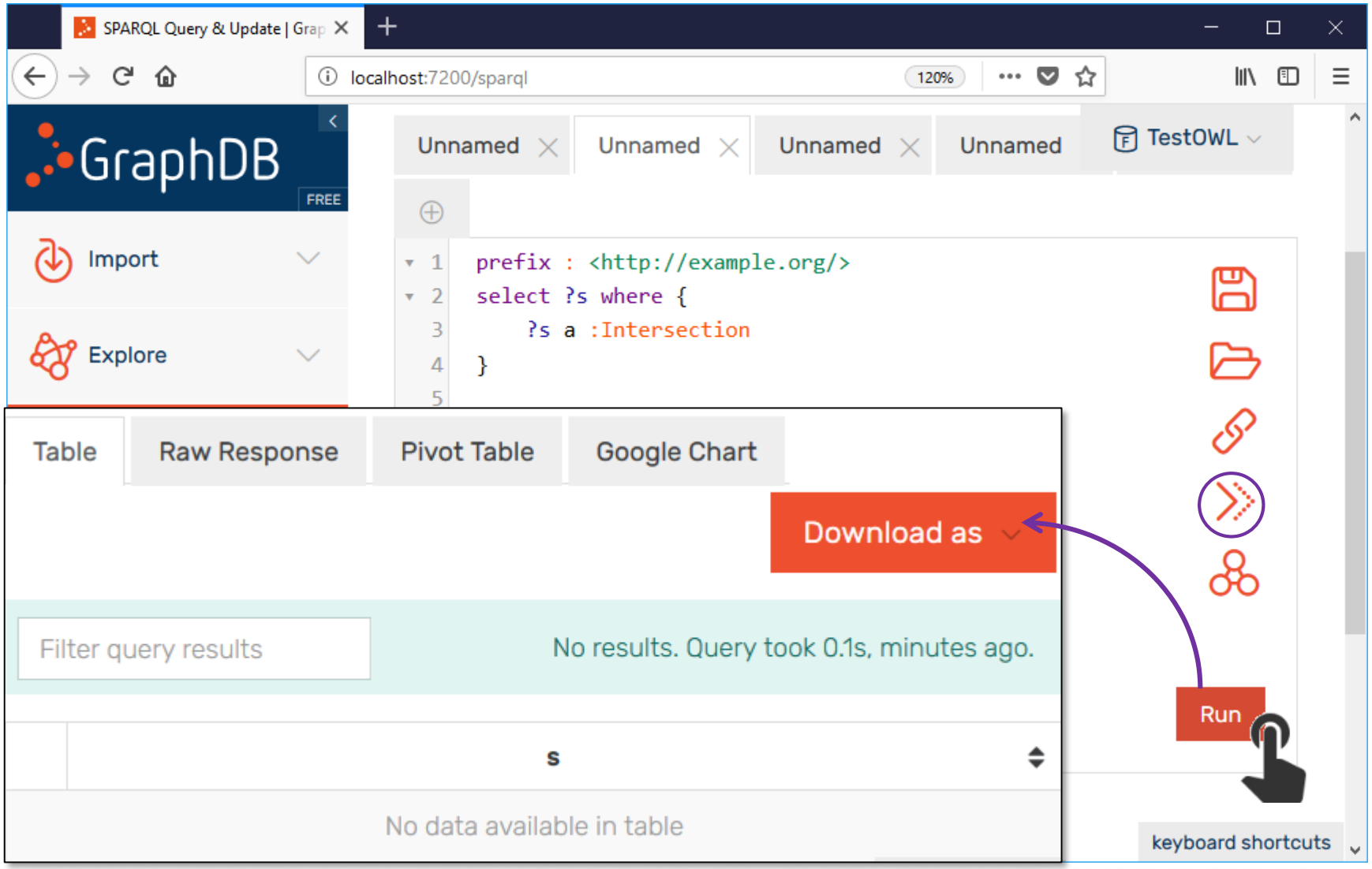
Aggiunta di triple RDF con INSERT DATA (2/2)



Spiegazione dei dati appena aggiunti

- Abbiamo definito due classi (*ClassA* e *ClassB*) e fornito un'istanza per ciascuna di esse (rispettivamente, *instance1* e *instance2*).
- Abbiamo quindi definito la classe *Intersection* come l'intersezione delle due classi citate di sopra
- Abbiamo asserito che le due istanze sono in effetti la stessa cosa (usando il predicato *owl:sameAs*)

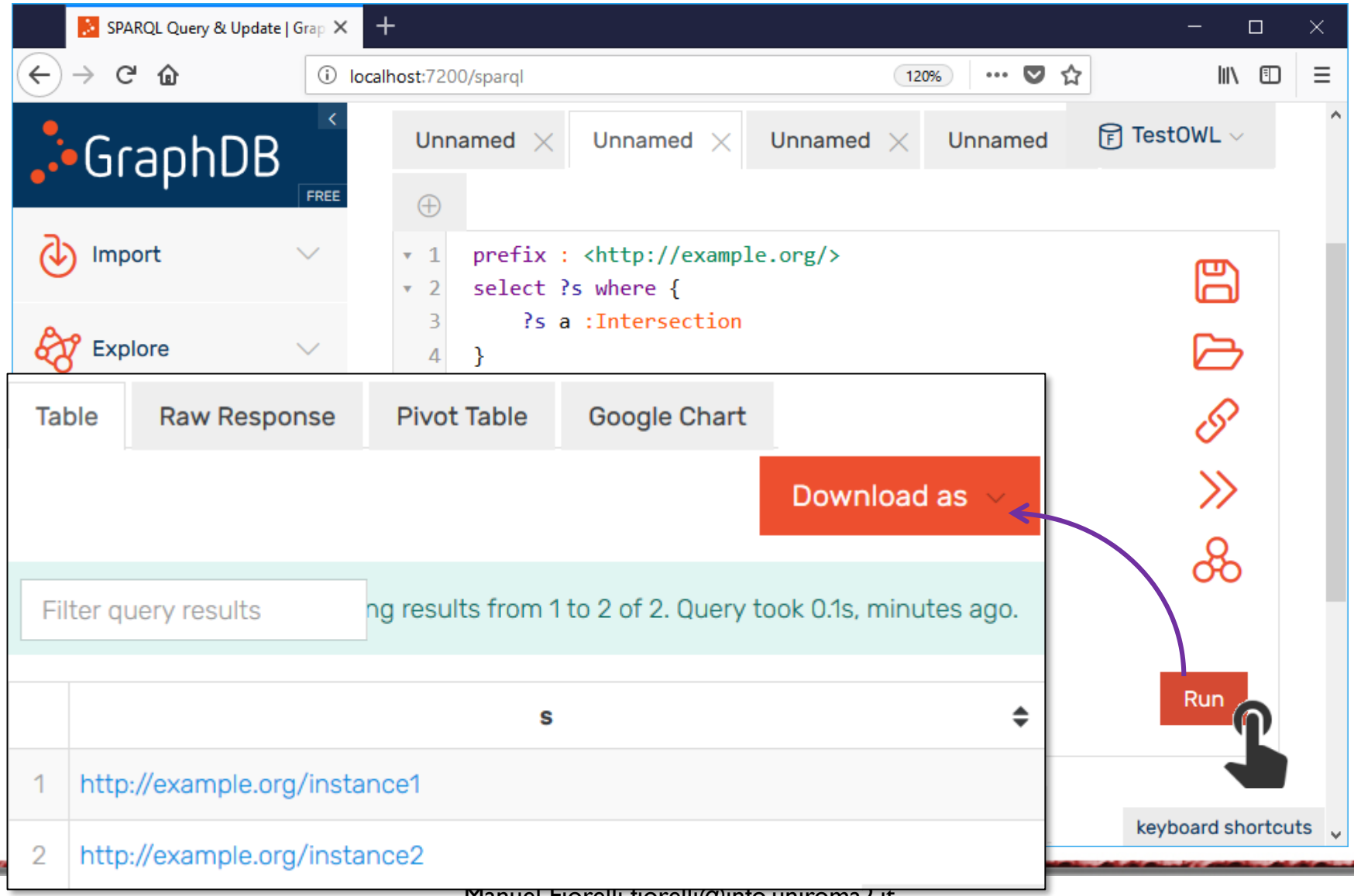
Istanze di *Intersection* senza inferenza (1/2)



Istanze di *Intersection* senza inferenza (2/2)

Nessuna risorsa è *esplicitamente dichiarata* essere un'istanza di Intersection.

Istanze di *Intersection* con inferenza e espansione owl:sameAs (1/2)



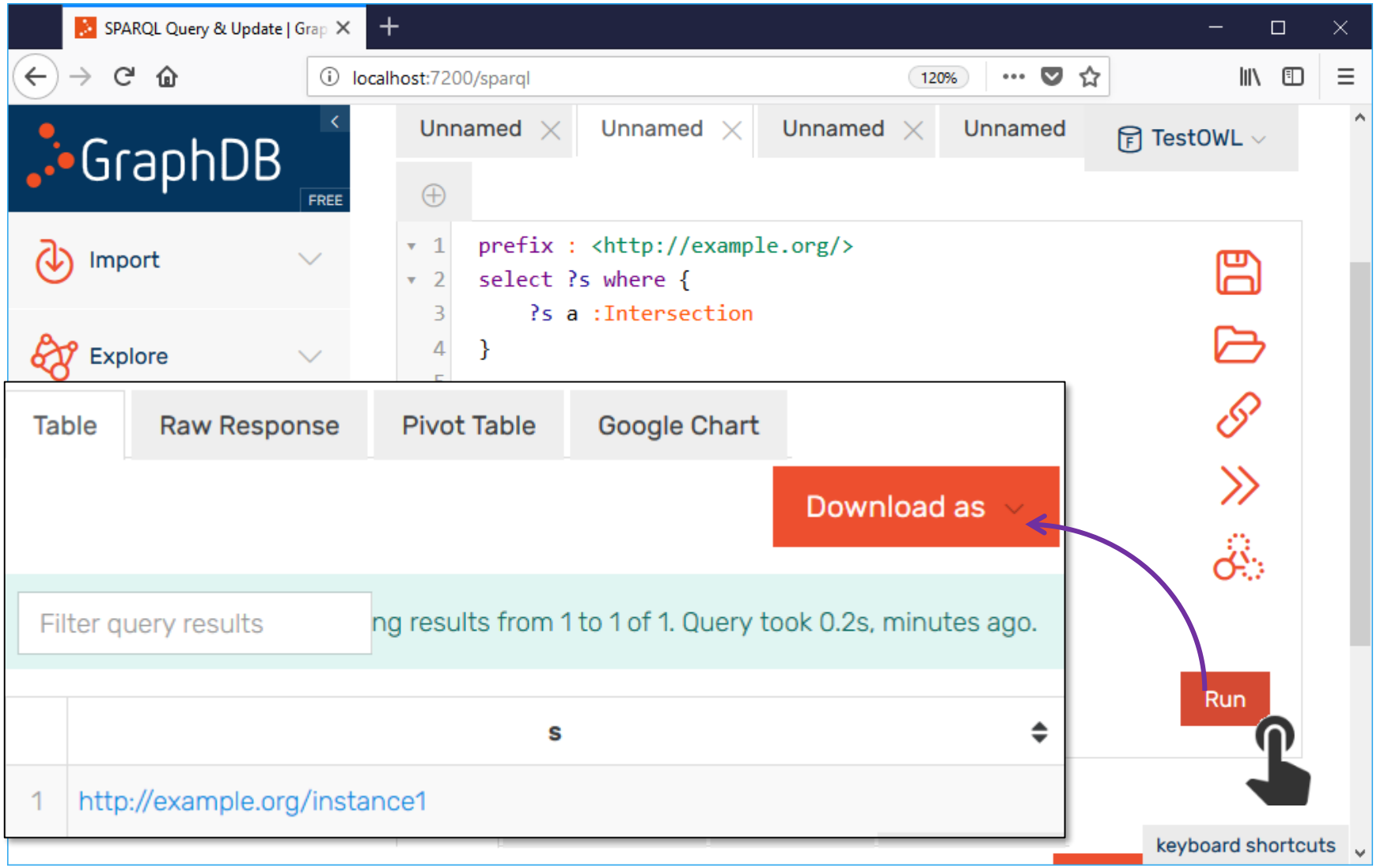
Manuel Fiorelli fiorelli@info.uniroma2.it
http://art.uniroma2.it/fiorelli

Istanze di *Intersection* con inferenza e espansione owl:sameAs (2/2)

instance1 è dichiarata esplicitamente di tipo *ClassA*, ma essendo stata dichiarata coincidente con *istance2*, che è di tipo *ClassB*, possiamo inferire che *instance1* è anche di tipo *ClassB*. Pertanto, si può inferire che *instance1* appartiene alla classe *Intersection*.

Un ragionamento analogo ci permette di inferire che *instance2* appartiene alla classe *Intersection*.

Istanze di *Intersection* con inferenza e senza espansione owl:sameAs (1/2)



Istanze di *Intersection* con inferenza e senza espansione owl:sameAs (2/2)

GraphDB possiede un'implementazione non rule-based del predicato *owl:sameAs*.

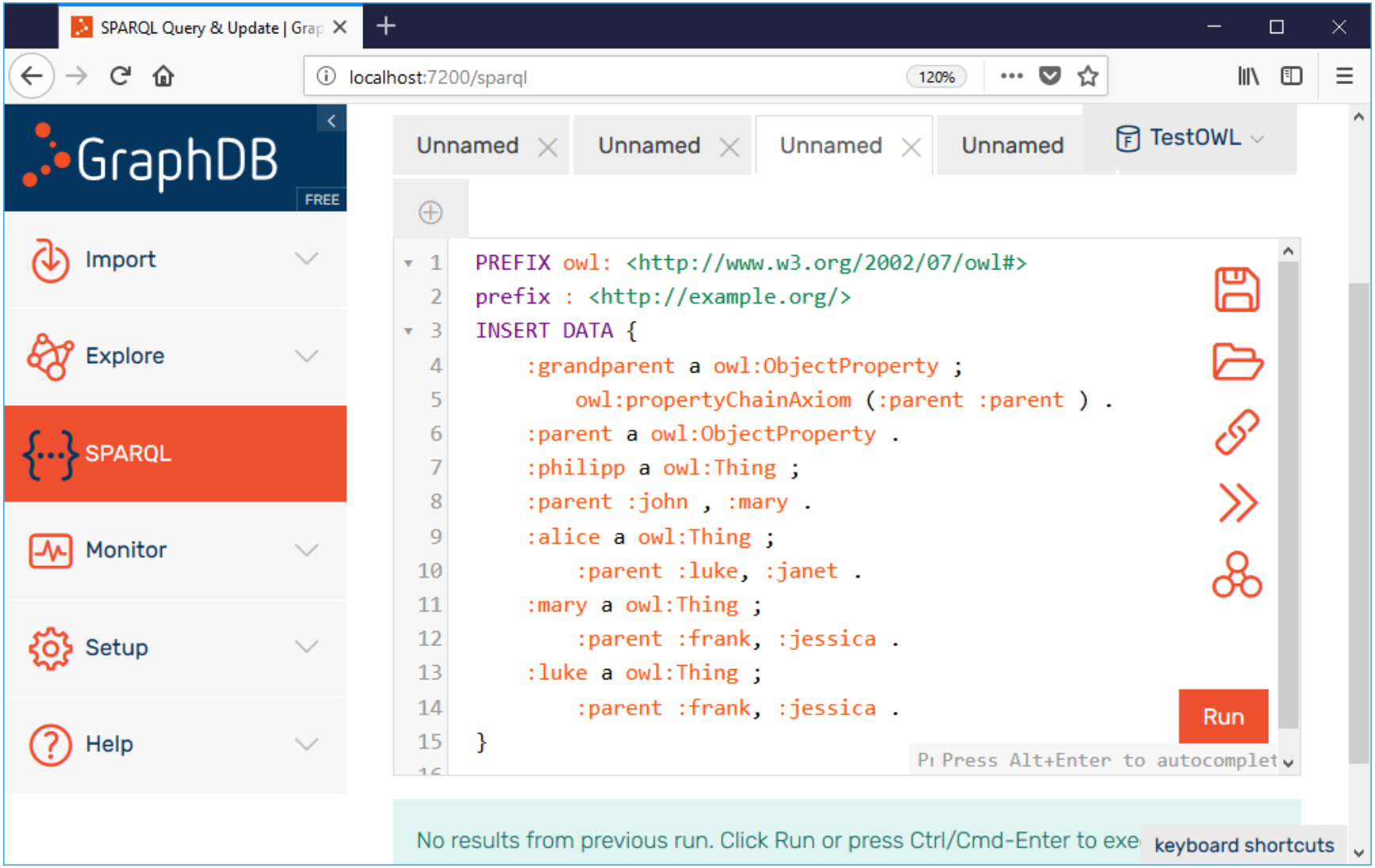
- Tiene traccia delle classi di equivalenza generate da *owl:sameAs*, associando a ciascuna di esse un *rappresentante* (tra le risorse che vi appartengono)
- Rappresenta ciascuno statement una sola volta sostituendo a ciascun componente i rappresentati delle classi di equivalenza cui appartengono
- Durante la valutazione di una query, GraphDB utilizza una sorta di backward-chaining enumerando tutti gli IRI equivalenti, garantendo la completezza dell'inferenza e dei risultati.
- L'implementazione di questa feature è tale da permettere di distinguere gli statement espliciti da quelli impliciti.

Per maggiori dettagli si consulti:

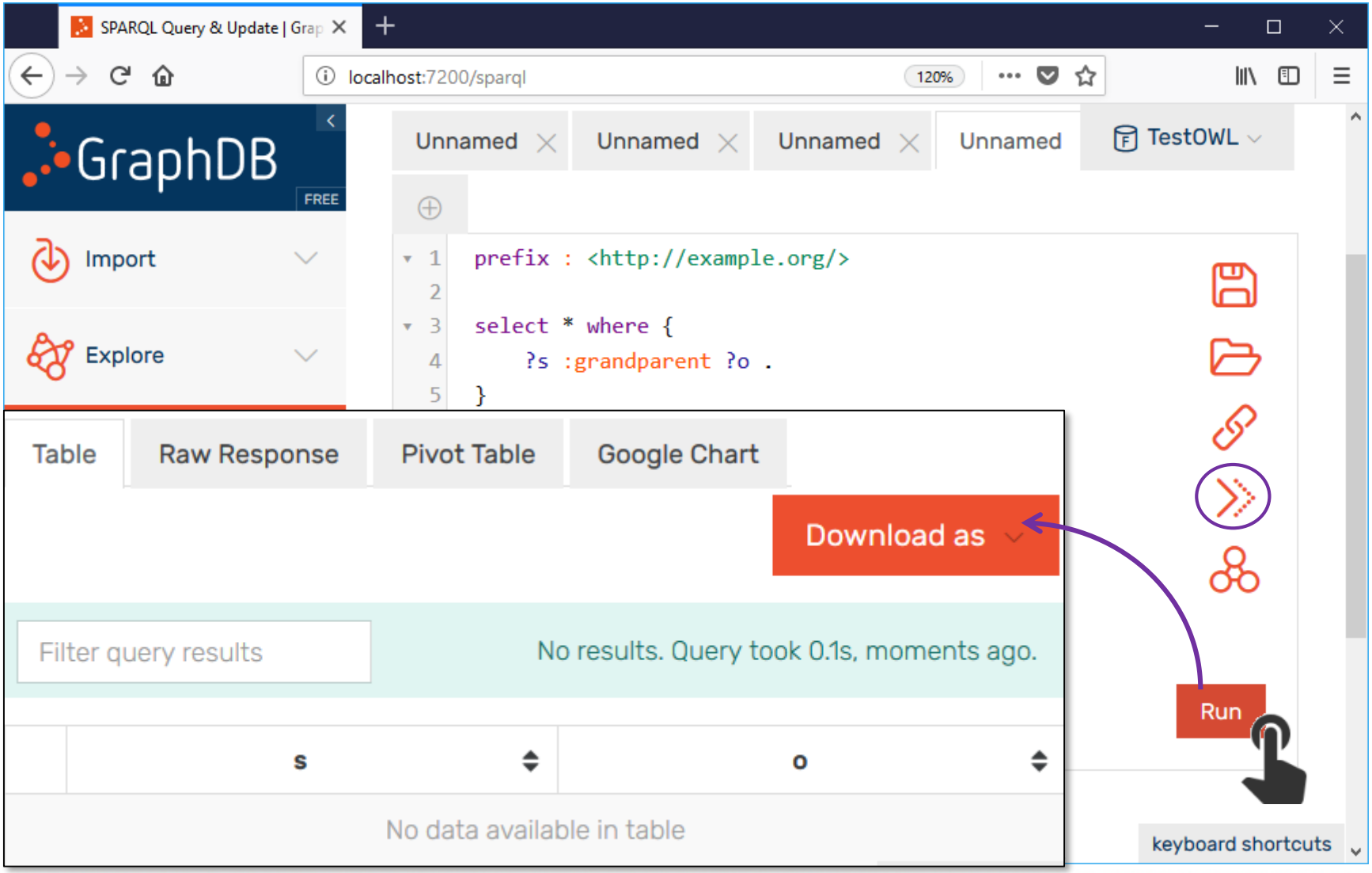
http://graphdb.ontotext.com/documentation/free/sameas-optimisation.html

Da notare l'uso della clausola *FROM onto:disable-sameAs* per disabilitare da dentro la query SPARQL l'espansione di *owl:sameAs* (utile quando si sottomette la query tramite API REST)

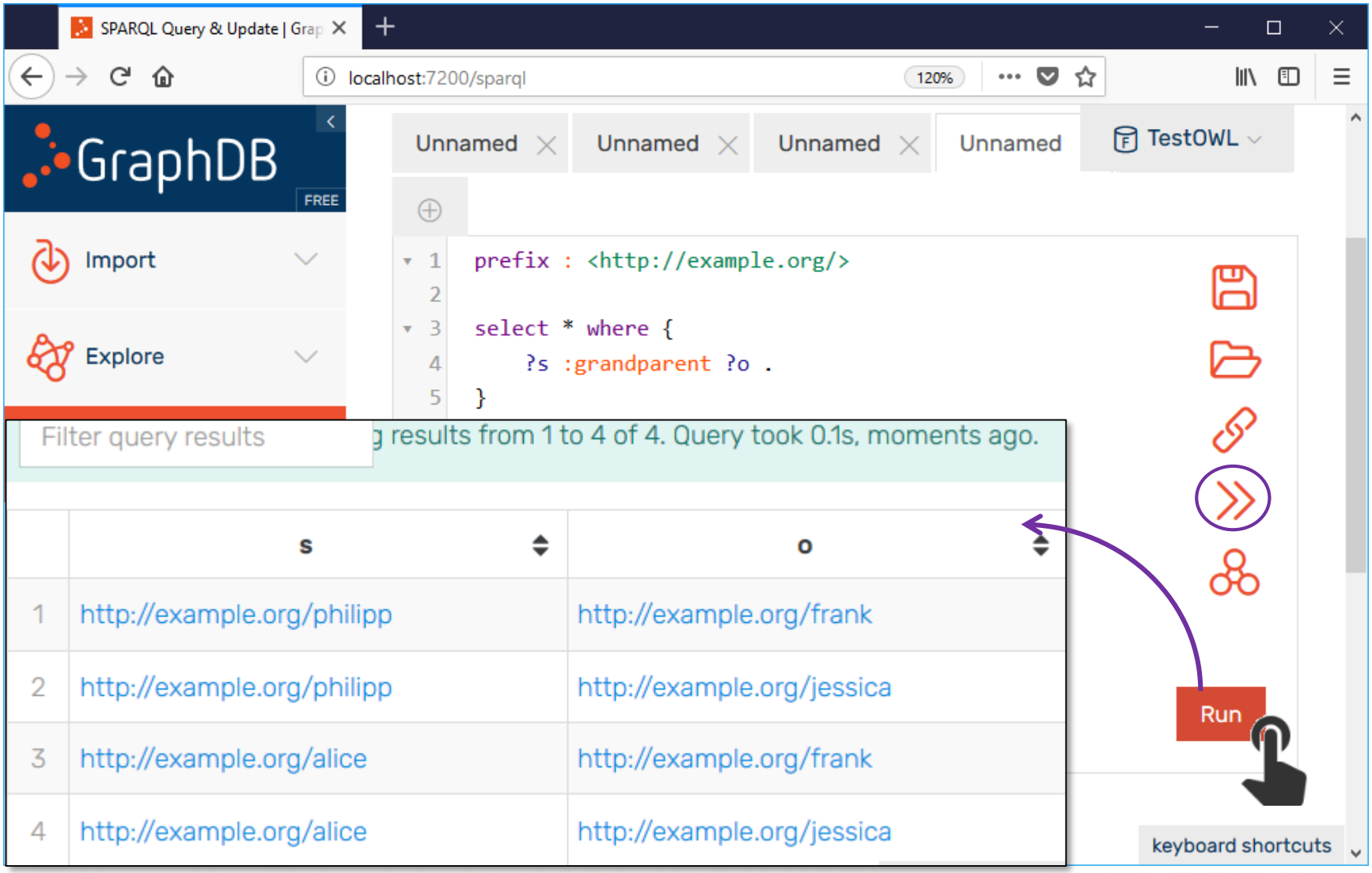
Aggiunta di altre triple (vedi esercizi OWL 9 e 10)



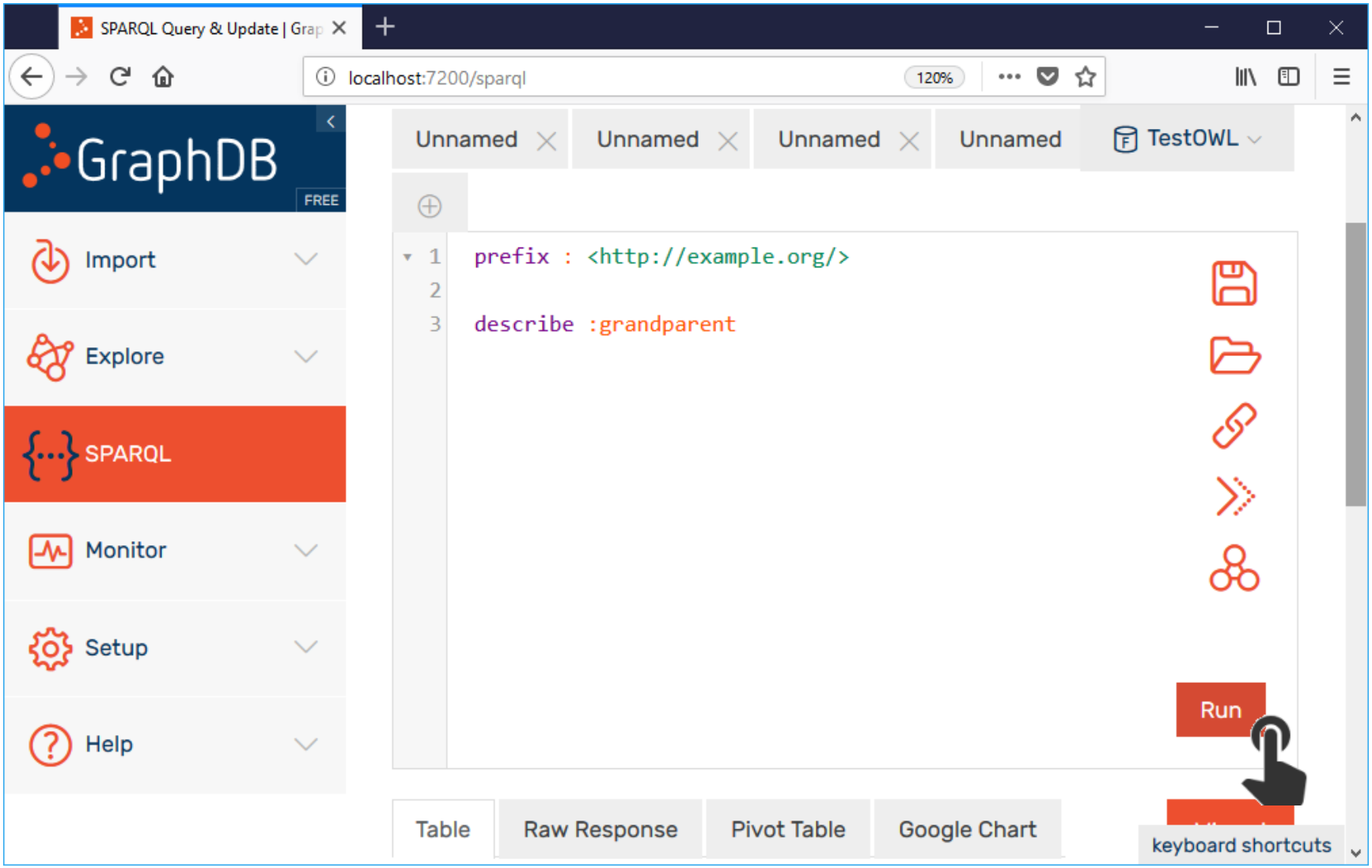
Query su property chain (senza inferenza)



Query su property chain (con inferenza)



Query *DESCRIBE* su *grandparent* (1/2)



Query *DESCRIBE* su *grandparent* (2/2)

Filter query results

ng results from 1 to 6 of 6. Query took 0.1s, minutes ago.

	subject	predicate	object
1	_:node10	rdf:first	http://example.org/parent
2	_:node10	rdf:rest	rdf:nil
3	_:node9	rdf:first	http://example.org/parent
4	_:node9	rdf:rest	_:node10
5	http://example.org/grandparent	rdf:type	owl:ObjectProperty
6	http://example.org/grandparent	owl:propertyChainAxiom	_:node9

L'oggetto della tripla col predicato *owl:propertyChainAxiom* è un blank node che rappresenta una RDF Collection attraverso un approccio molto simile a quello usato da Prolog: una lista contiene un elemento (*rdf:first*) e un riferimento al resto della lista (*rdf:rest*) (si noti che *rdf:nil* è la lista vuota)