

**NAME**

tracker-sparql - Use SparQL to query the Tracker databases.

**SYNOPSIS**

**tracker sparql** -q <*sparql*> [-u] | -f <*file*>

**tracker sparql** -t [*class*] [-s <*needle*>] [-p]

**tracker sparql** [-c] [-p] [-x] [-n [*class*]] [-i [*property*]] [-s <*needle*>]

**tracker sparql** [--get-longhand <*class*>] [--get-shorthand <*class*>]

**DESCRIPTION**

This command allows probing of the current database schema (also known as ontology) and running low level queries or updates on the data set. In terms of the database ontology, it's easy to find out what properties are indexed for speed, or notified on changes, what classes are available and the properties belonging to those classes. There are also visual tools to display an ascii tree layout of the classes and their relationships to each other.

When the caller runs a query, the query is in RDF and SPARQL. This can be done two ways. Either by providing a *file* with the query or by providing a string with the *sparql* query.

The *file* argument can be either a local path or a URI. It also does not have to be an absolute path.

**OPTIONS**

**-f, --file=<*file*>**

Use a *file* with SPARQL content to query or update.

**-q, --query=<*sparql*>**

Use a *sparql* string to query the database with.

**-u, --update**

This has to be used with **--query**. This tells "tracker sparql" to use the SPARQL update extensions so it knows it isn't a regular data lookup request. So if your query is intended to change data in the database, this option is needed.

**-c, --list-classes**

Returns a list of classes which describe the ontology used for storing data. These classes are also used in queries. For example, <http://www.w3.org/2000/01/rdf-schema#Resource> is one of many classes which should be returned here.

**-x, --list-class-prefixes**

Returns a list of classes and their related prefixes. Prefixes are used to make querying a lot simpler and are much like an alias. For example, <http://www.w3.org/2000/01/rdf-schema#Resource> has the prefix *rdfs* so queries can be cut down to:

```
"SELECT ?u WHERE { ?u a rdfs:Resource }"
```

**-p, --list-properties=[class]**

Returns a list of properties which pertain to a *class*. You can use both formats here for the *class*, either the full name <http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#Video> or the shortened prefix name *nfo:Video*.

This gives the following result:

```
$ tracker sparql -p nfo:Video
```

Properties: 2

<http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#frameRate>

<http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#frameCount>

These properties *nfo:frameRate* and *nfo:frameCount* can then be used in queries.

See also **--tree** and **--query**.

**-n, --list-notifies=[class]**

Returns a list of classes which are notified over D-Bus about any changes that occur in the database. The *class* does not have to be supplied here. This is optional and filters the results according to any argument supplied. With no *class*, all classes are listed.

**-i, --list-indexes=[property]**

Returns a list of properties which are indexed in the database. Indexes improves query speed but also add an indexing penalty. The *property* does not have to be supplied here. This is optional and filters the results according to any argument supplied. With no *property*, all properties are listed.

**-t, --tree=[class]**

Prints a tree showing all parent classes of *class* in the ontology. The *class* can be provided in

shorthand or longhand (see **--get-shorthand** and **--get-longhand** for details). For example:

```
$ tracker sparql -t nmo:MMSMessage
ROOT
+-- rdfs:Resource (C)
| +-- nie:InformationElement (C)
| | +-- nfo:Document (C)
| | | +-- nfo:TextDocument (C)
| | | | '-- nmo:Message (C)
| | | | | +-- nmo:PhoneMessage (C)
| | | | | | '-- nmo:MMSMessage (C)
```

If no *class* is given, the entire tree is shown.

The **--search** command line option can be used to highlight parts of the tree you're looking for. The search is case insensitive.

The **--properties** command line option can be used to show properties for each class displayed, for example:

```
$ tracker sparql -t nfo:FileDataObject -p
ROOT
+-- rdfs:Resource (C)
| --> http://purl.org/dc/elements/1.1/contributor (P)
| --> http://purl.org/dc/elements/1.1/coverage (P)
| --> http://purl.org/dc/elements/1.1/creator (P)
| --> http://purl.org/dc/elements/1.1/date (P)
| --> http://purl.org/dc/elements/1.1/description (P)
| --> http://purl.org/dc/elements/1.1/format (P)
| --> http://purl.org/dc/elements/1.1/identifier (P)
| --> http://purl.org/dc/elements/1.1/language (P)
| --> http://purl.org/dc/elements/1.1/publisher (P)
| --> http://purl.org/dc/elements/1.1/relation (P)
| --> http://purl.org/dc/elements/1.1/rights (P)
| --> http://purl.org/dc/elements/1.1/source (P)
| --> http://purl.org/dc/elements/1.1/subject (P)
| --> http://purl.org/dc/elements/1.1/title (P)
| --> http://purl.org/dc/elements/1.1/type (P)
| --> nao:deprecated (P)
| --> nao:hasTag (P)
```

```

| --> nao:identifier (P)
| --> nao:isRelated (P)
| --> nao:lastModified (P)
| --> nao:numericRating (P)
| --> rdf:type (P)
| --> rdfs:comment (P)
| --> rdfs:label (P)
| --> tracker:added (P)
| --> tracker:damaged (P)
| --> tracker:modified (P)
| +-+ nie:DataObject (C)
| | --> nfo:belongsToContainer (P)
| | --> nie:byteSize (P)
| | --> nie:created (P)
| | --> nie:dataSource (P)
| | --> nie:interpretedAs (P)
| | --> nie:isPartOf (P)
| | --> nie:lastRefreshed (P)
| | --> nie:url (P)
| | --> tracker:available (P)
| | +-+ nfo:FileDataObject (C)
| | | --> nfo:fileCreated (P)
| | | --> nfo:fileLastAccessed (P)
| | | --> nfo:fileLastModified (P)
| | | --> nfo:fileName (P)
| | | --> nfo:fileOwner (P)
| | | --> nfo:fileSize (P)
| | | --> nfo:hasHash (P)
| | | --> nfo:permissions (P)

```

**-s, --search=<needle>**

Returns a list of classes and properties which partially match *needle* in the ontology. This is a case insensitive match, for example:

```
$ tracker sparql -s text
```

Classes: 4

<http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#TextDocument>

<http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#PlainTextDocument>

```
http://www.semanticdesktop.org/ontologies/2007/03/22/nfo#PaginatedTextDocument
http://www.tracker-project.org/temp/nmm#SynchronizedText
```

Properties: 4

```
http://www.tracker-project.org/ontologies/tracker#fulltextIndexed
http://www.semanticdesktop.org/ontologies/2007/01/19/nie#plainTextContent
http://www.semanticdesktop.org/ontologies/2007/03/22/nmo#plainTextMessageContent
http://www.tracker-project.org/temp/scal#textLocation
```

See also **--tree**.

### **--get-shorthand=<class>**

Returns the shorthand for a class given by a URL. For example:

```
$ tracker sparql --get-shorthand http://www.semanticdesktop.org/ontologies/2007/03/22/nmo#plainTextMessageC
nmo:plainTextMessageContent
```

### **--get-longhand=<class>**

Returns the longhand for a class given in the form of CLASS:PROPERTY. For example:

```
$ tracker sparql --get-longhand nmm:MusicPiece
http://www.tracker-project.org/temp/nmm#MusicPiece
```

## **ENVIRONMENT**

### **TRACKER\_SPARQL\_BACKEND**

This option allows you to choose which backend you use for connecting to the database. This choice can limit your functionality. There are three settings.

With "**direct**" the connection to the database is made directly to the file itself on the disk, there is no intermediary daemon or process. The "**direct**" approach is purely *read-only*.

With "**bus**" the **tracker-store** process is used to liase with the database queuing all requests and managing the connections via an IPC / D-Bus. This adds a small overhead *BUT* this is the only approach you can use if you want to *write* to the database.

With "**auto**" the backend is decided for you, much like it would be if this environment variable was undefined.

**EXAMPLES**

List all classes

```
$ tracker sparql -q "SELECT ?cl WHERE { ?cl a rdfs:Class }"
```

List all properties for the Resources class (see --list-properties)

```
$ tracker sparql -q "SELECT ?prop WHERE {  
  ?prop a rdf:Property ;  
  rdfs:domain <http://www.w3.org/2000/01/rdf-schema#Resource>  
}"
```

List all class namespace prefixes

```
$ tracker sparql -q "SELECT ?prefix ?ns WHERE {  
  ?ns a tracker:Namespace ;  
  tracker:prefix ?prefix  
}"
```

List all music files

```
$ tracker sparql -q "SELECT ?song WHERE { ?song a nmm:MusicPiece }"
```

List all music albums, showing title, track count, and length in seconds.

```
$ tracker sparql -q "SELECT ?title COUNT(?song)  
  AS songs  
  SUM(?length) AS totallength  
  WHERE {  
    ?album a nmm:MusicAlbum ;  
    nie:title ?title .  
    ?song nmm:musicAlbum ?album ;  
    nfo:duration ?length  
  } GROUP BY ?album"
```

List all music from a particular artist

```
$ tracker sparql -q "SELECT ?song ?title WHERE {  
    ?song nmm:performer [ nmm:artistName 'Artist Name' ] ;  
    nie:title ?title  
}"
```

Set the played count for a song

```
$ tracker sparql -u -q "DELETE {  
    <file:///home/user/Music/song.mp3> nie:usageCounter ?count  
} WHERE {  
    <file:///home/user/Music/song.mp3> nie:usageCounter ?count  
} INSERT {  
    <file:///home/user/Music/song.mp3> nie:usageCounter 42  
}"
```

List all image files

```
$ tracker sparql -q "SELECT ?image WHERE { ?image a nfo:Image }"
```

List all image files with a specific tag

```
$ tracker sparql -q "SELECT ?image WHERE {  
    ?image a nfo:Image ;  
    nao:hasTag [ nao:prefLabel 'tag' ]  
}"
```

List all image files created on a specific month and order by date

```
$ tracker sparql -q "SELECT ?image ?date WHERE {  
    ?image a nfo:Image ;  
    nie:contentCreated ?date .  
    FILTER (?date >= '2008-07-01T00:00:00' &&  
            ?date < '2008-08-01T00:00:00')  
} ORDER BY ?date"
```

**SEE ALSO**

**tracker-sql(1), tracker-store(1), tracker-info(1).**

**<http://nepomuk.semanticdesktop.org/>**

**<http://www.w3.org/TR/rdf-sparql-query/>**