RRDF gotcha

mja@statgroup.dk

2016-02-18

Contents

Setup **1**

Example on what not to do **1**

Setup

First load the package.

```
library(rrdf)
library(rrdfancillary)
```

Example on what not to do

When the exactly same triples are inserted - only one triple remains.

```
store1<- new.rdf(ontology=FALSE)

sparql.rdf( store1, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
## <0 x 0 matrix>
```

```
SPARQLinsert<- '
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT DATA
{
  <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
  <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
}
'
cat(SPARQLinsert,"\n")
```

```
##
## PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
## INSERT DATA
## {
##   <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
##   <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
## }
##
```

```
update.rdf( store1, SPARQLinsert )
```

```
## [1] TRUE
```

```
sparql.rdf( store1, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

Now, of course, it is always better only to store the values once, when it is intended to store one copy.

However, as I thougth that only one trippel is stored, so I was less carefull in some of the code.

Here is what Apache/Jena does when using the RRDF interface.

```
store2<- new.rdf(ontology=FALSE)
add.data.triple(
    store2,
    subject="http://example.org/subject1",
    predicate="http://example.org/property1",
    data="mytext",
    lang="en"
)

add.data.triple(
    store2,
    subject="http://example.org/subject1",
    predicate="http://example.org/property1",
    data="mytext",
    type="string"
)
```

Now query the store:

```
sparql.rdf( store2, "select ?s ?p ?o where {?s ?p ?o}"  )
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
## [2,] "http://example.org/subject1" "http://example.org/property1" "mytext"
```

The two rows look identical. The next query also show language and datatype associate with the object.

```
sparql.rdf( store2, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
## [2,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
## [2,] "en" "http://www.w3.org/1999/02/22-rdf-syntax-ns#langString"
```

The same tripel appears twice! That learned me that the language and data type are important. They make a difference, so to speak.

Now using the same datatype, `string`, gives two triples again.

```
store3<- new.rdf(ontology=FALSE)
add.data.triple(
    store3,
    subject="http://example.org/subject1",
    predicate="http://example.org/property1",
    data="mytext",
    type="string"
)

add.data.triple(
    store3,
    subject="http://example.org/subject1",
    predicate="http://example.org/property1",
    data="mytext",
    type="string"
)

sparql.rdf( store3, "select ?s ?p ?o where {?s ?p ?o}"  )
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
## [2,] "http://example.org/subject1" "http://example.org/property1" "mytext"
```

```
sparql.rdf( store3, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }")
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
## [2,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
## [2,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

Mixing `INSERT DATA` and RRDF `add.data.triple gives` same result - two triples.

```
store4<- new.rdf(ontology=FALSE)

SPARQLinsert<- '
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT DATA
{
  <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
}
'
update.rdf( store4, SPARQLinsert )
```

```
## [1] TRUE
```

```
sparql.rdf( store4, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

One triple inserted, one triple in the store. Fine!

Now add one triple - exactly the same as the previos.

```
add.data.triple(
    store4,
    subject="http://example.org/subject1",
    predicate="http://example.org/property1",
    data="mytext",
    type="string"
)

sparql.rdf( store4, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
##      s                              p                              o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
## [2,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
## [2,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

Two triples in the store.

What if doing two `INSERT DATA`?

```
store5<- new.rdf(ontology=FALSE)

sparql.rdf( store5, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }
```

```
## <0 x 0 matrix>
```

```
SPARQLinsert<- '
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT DATA
{
  <http://example.org/subject1>  <http://example.org/property1> "mytext"^^xsd:string .
}
'
update.rdf( store5, SPARQLinsert )
```

```
## [1] TRUE
```

```
sparql.rdf( store5, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }")
```

```
##      s                             p                             o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

One triple - as expected, only triple was inserted.

```
update.rdf( store5, SPARQLinsert )
```

```
## [1] TRUE
```

```
sparql.rdf( store5, "select ?s ?p ?o (lang(?o) as ?lang) (datatype(?o) as ?datatype) where {?s ?p ?o  }")
```

```
##      s                             p                             o
## [1,] "http://example.org/subject1" "http://example.org/property1" "mytext"
##      lang datatype
## [1,] ""   "http://www.w3.org/2001/XMLSchema#string"
```

One triple - as expected, as the triple already existed.

Lessons learned:

- Apache/Jena interface R to Java behaves differently than Apache Jena handling of Update Scripts.
- Be carefull when changing code.