QtSparql - A Gentle Introduction

Adrian Perez - aperez@igalia.com

March 2, 2011

igalia

Index

QtSparql
&
TrackerLiveQuery

Who's Who

- **QtSparql:**
  Thin library to access RDF stores. All of them in general (read: not just Tracker).
- **QtSparql-Tracker:**
  Driver for accessing Tracker from QtSparql. Provides some extensions for features only found in Tracker like e.g. change notifications.
- **QtSparql-Tracker-Live:**
  Batgadget built on top of QtSparql and the Tracker extensions, designed to supersede QtTracker.

QtSparql: All You Need to Know In a Single Slide

Familiar, database-like API:

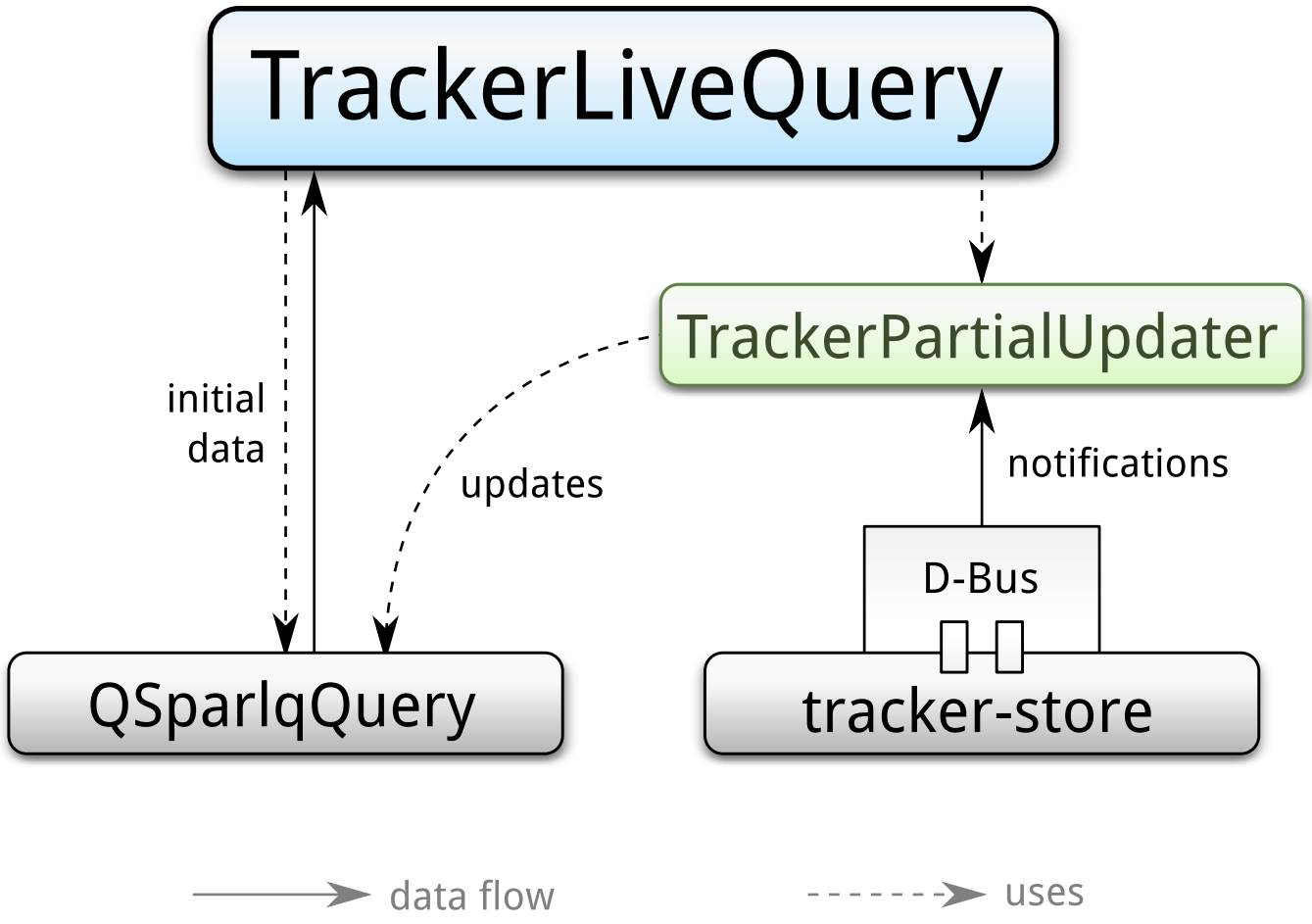
- `QSparqlConnection` (plus `QSparqlConnectionOptions`)
- `QSparqlQuery` (pass it to `QSparqlConnection::exec()`)
- `QSparqlResult` (note: cursor-like behavior).
  - Hint: Its memory usage is scarce.

Extras:

- Asynchronous by default.
  - Hint: `waitForFinished()`
  - Hint: `dataReady()`, `finished()`
- `QSparqlQueryModel` (really?)

igalia

Going Live: The Big Picture



Using TrackerLiveQuery

Initial data:

- "Initial" query: runs once, fills the model with data.
- Internally, a regular QSparqlQuery is used.
- The `tracker:id` attribute *must* appear in one column.

Getting updates:

- "Update" query: runs anytime a change notification arrives.
- To get notifications, TrackerChangeNotifier is used over D-Bus.
- Spawns a QSparqlQuery with a filter on the affected `tracker:id`.
- Updated data is merged into the model.

Going Live: How Queries Look Like? - I

Initial Query:

```
SELECT
  tracker:id(?urn) AS ?trackerid
  nie:url(?urn) AS ?url
WHERE {
  ?urn rdf:type nmm:Photo .
}
ORDER BY ?url
```

Going Live: How Queries Look Like? - II

Update Query:

```
SELECT
  tracker:id(?urn) AS ?trackerid
  nie:url(?urn) AS ?url
WHERE {
  ?urn rdf:type nmm:Photo .
  %FILTER
}
ORDER BY ?url
```

Going Live: How Queries Look Like? - III

Did you spot the difference?

Going Live: How Queries Look Like? - IV

%FILTER?

WTF???

Going Live: Knowing What To Update - I

The updater needs to run the query *only* for changed items:

- `%FILTER` gets replaced by a `FILTER` statement with the `tracked:id` of updated items.
- The "template" for the filter snippet is passed to `TrackerPartialUpdater::watchClass()`, like this:

```
FILTER (tracker:id(?urn) IN %LIST)
```

Going Live: Knowing What To Update - II

%LIST?

WTF???

igalia

Going Live: Knowing What To Update - III

The updater runs the query *only* for the changed items, and it has to know where to place the list of changed `tracker:id`:

- `%LIST` gets replaced by a list of identifiers.
- So, in the end the result is a filter like this:

```
FILTER (tracker:id(?urn) IN (5, 15, 29, 61, 125))
```

Going Live: Complete Update Query

Complete Update Query:

```
SELECT
  tracker:id(?urn) AS ?trackerid
  nie:url(?urn) AS ?url
WHERE {
  ?urn rdf:type nmm:Photo .
  FILTER (
    tracker:id(?urn) IN (5, 15, 29, 61, 125)
  )
}
ORDER BY ?url
```

SPARQL bits

SPARQL Is Like Maths - I

Maths

Given a set of equations, which values for the free variables solve the equations?

$$2x + 4y^5 - \sqrt{zk} = 0$$

$$x + 5y - zk^2 = 0$$

SPARQL Is Like Maths - II

SPARQL

Given a set of restrictions on data, which values for the free variables satisfy the restrictions?

```
SELECT ?url ?mimetype
WHERE {
  ?urn rdf:type nmm:Photo .
  ?urn nie:url ?url .
  ?urn nie:mimeType ?mimetype .
  FILTER (?urn nie:mimetype IN
    "image/jpeg", "image/png", "image/gif")
  ) .
}
```

SPARQL Bits - I

If you don't know the value of an attribute, you need a `WHERE` part in your query.
Example:

```
DELETE {
  ?urn nie:contentAccessed ?date .
}
WHERE {
  ?urn nie:contentAccessed ?date .
}
```

(Deletes access time for all elements in the store.)

SPARQL Bits - II

There are no `UPDATE` queries, `DELETE` followed by `INSERT` is needed. Fortunately, they can be glued in the same query (thus the same transaction).
Example:

```
DELETE { ?urn nie:contentAccessed ?date .  }
WHERE  { ?urn nie:contentAccessed ?date .  }
INSERT { ?urn nie:contentAccessed
              "2011-11-11T11:11:11"^^xsd:dateTime .
}
```

(Updates access time for all elements in the store.)

SPARQL Bits - III

Non-multivalued attributes must be updated. Thus deleted *and* re-inserted.
Example: See previous slide.

SPARQL Bits - IV

The URN (Uniform Resource Name) is not always the subject.
Especially when dealing with nested data like i.e. geolocation.
Example:

```
SELECT ?urn ?city ?country
WHERE {
  ?urn      slo:location      ?location .
  ?location slo:postalAddress ?address  .
  ?address  nco:locality      ?city     .
  ?address  nco:country       ?country  .
}
```

(Picks URNs and their geolocation information for all items.)

SPARQL Bits - V

Checking for the existence of some particular attribute for some particular subject can be done with an `ASK` query.
Example:

```
ASK {
  <urn-literal> nao:hasTag
      nao:predefined-tag-favorite .
}
```

(Works in general for any number of triplets.)

Examples
Q&A