Introducing JDBC for SPARQL

Twitter: @RobVesse
Email: rvesse@apache.org

About Me

- Software Engineer at YarcData, part of Cray Inc
- PMC Member and Committer on Apache Jena project
  - Joined project in January 2012
- Can also be found hanging out on other Apache mailing lists:
  - Incubator General
  - Giraph
  - Marmotta
  - Dev @ Community
- Interested in:
  - All things Semantic Web, RDF and SPARQL
  - Graphs (RDF or otherwise)
  - Big Data particularly wrt. graphs

Talk Overview

- Definitions
- Why JDBC for SPARQL?
- Introducing Jena JDBC
  - Where can I get this?
  - What is it?
  - Architecture
  - Connection Strings
  - Bridging the data model gap
  - Fudging the metadata
  - Supporting awkward tools
- Example Code
- Demo
- Alternative Options

Definitions

- **RDF**
  - Resource Description Framework
  - W3C Standard for describing arbitrary data about resources
- **SPARQL**
  - Recursive acronym - SPARQL Protocol and RDF Query Language
  - Set of W3C standards for querying, updating and accessing RDF databases
  - i.e. the SQL of the RDF database world
- **JDBC**
  - Java Database Connectivity
  - Standard Java API for communicating with databases
  - Part of the standard edition Java Runtime

Why JDBC for SPARQL?

What's the need?

- Graph analytics is one of the hot topics in the Big Data/ Analytics world right now
- However most popular analytics tooling is primarily geared around relational databases e.g.
  - Pentaho
  - Centrifuge
  - QlikView
  - Tableau
  - etc.
- Often the database API of choice is JDBC
  - Or ODBC
- Integrating SPARQL natively on a tool by tool basis is arduous, limits code re-use and open sourcing
  - May involve using semi-proprietary/restrictively licensed plugin APIs
- Implementing a JDBC driver lets people use SPARQL in any JDBC supporting tool

Introducing Jena JDBC

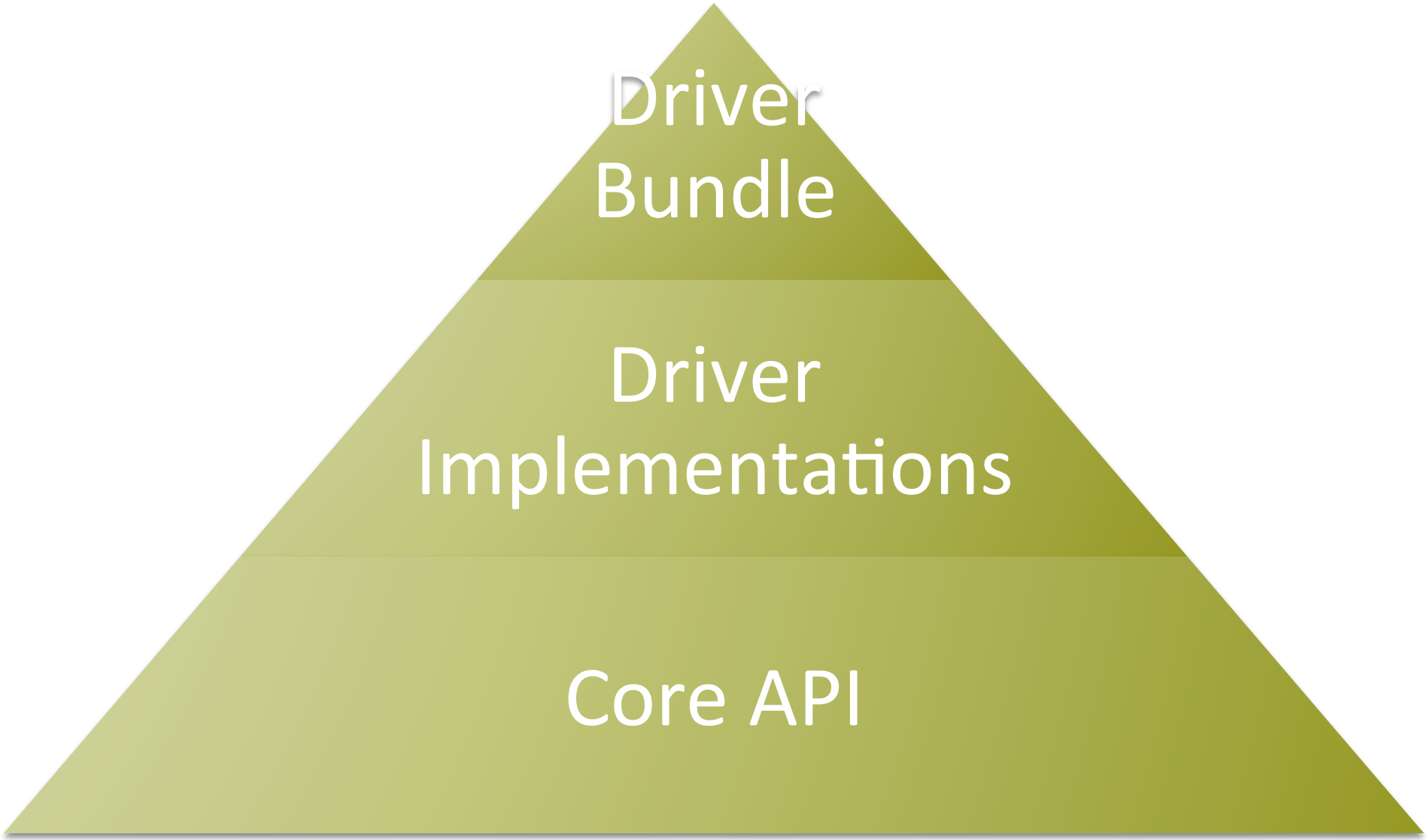
Where can I get this?

- Available as part of the Apache Jena project from the 2.11.0 release onwards
  - http://jena.apache.org
- NB - Not included in convenience binary packages
- Available via Maven
  - http://jena.apache.org/documentation/jdbc/artifacts.html
  - Group ID is **org.apache.jena**
  - Artifact IDs are of the form **jena-jdbc-foo** where **foo** is the specific module
  - Current version for these modules is **1.0.1**
- Documentation on our website
  - http://jena.apache.org/documentation/jdbc/index.html

What is it?

- API Framework for building SPARQL over JDBC drivers
- JDBC 4.0 API compatibility
  - Not JDBC 4.0 compliant
- JDBC Drivers for the major RDF/SPARQL back ends that Jena supports
  - In-Memory
  - TDB
  - Remote Endpoints
- Provides SPARQL over JDBC
  - i.e. allows executing SPARQL queries through the JDBC API
- An uber-jar driver bundle that can be dropped into applications to provide all the drivers and necessary dependencies

Architecture



Core API

- Provides high level abstract implementations of all the infrastructure we need to implement a SPARQL over JDBC driver e.g.
  - Drivers
  - Connections
  - Metadata
  - Statements
  - Result Sets
  - Data Typing
- Designed to be extensible and reused
- Most users won't need to know much about this

Driver Implementations & Bundle

- One for each major RDF/SPARQL backend the Apache Jena project supports
- These are the components most users care about
- In-Memory
  - Non-persistent though may be initialized from a file on disk
  - Useful for testing and prototyping
  - Maven artifact ID is **jena-jdbc-driver-mem**
- TDB
  - Persistent disk backed RDF database
  - Only driver that supports transactions
  - Maven artifact ID is **jena-jdbc-driver-tdb**
- Remote Endpoint
  - Any RDF database that implements the SPARQL HTTP Protocol
  - http://www.w3.org/TR/sparql11-protocol/
  - Maven artifact ID is **jena-jdbc-driver-remote**
- Bundle is a single convenience JAR containing all the drivers
  - Maven artifact ID is **jena-jdbc-driver-bundle**

Connection Strings

```
jdbc:jena:mem:dataset=file.nq

jdbc:jena:mem:empty=true

jdbc:jena:tdb:location=/path/to/data

jdbc:jena:remote:query=http://localhost:3030/ds/query&update=http://
localhost:3030/ds/update
```

- Jena JDBC drivers use a common jdbc:jena:**foo**: prefix
  - Where **foo** is a driver specific prefix e.g. **tdb**
- Various implementation specific parameters plus some general framework parameters e.g. **jdbc-compatibility** to control data typing behaviour
- See documentation for fuller connection string reference:
  - http://jena.apache.org/documentation/jdbc/drivers.html

Bridging the Data Model Gap

- **SPARQL has four query forms each with slightly different results**
  - Need to make each fit JDBCs tabular result set API
- **SELECT**
  - Tabular results - no translation necessary
- **ASK**
  - Single boolean result - single column with single row
- **CONSTRUCT/DESCRIBE**
  - RDF Graph result - represent the triples in tabular form
  - i.e. table with 3 columns - subject, predicate and object

Duck Typing

CC-BY-SA 3.0 - Wikimedia Commons
http://commons.wikimedia.org/wiki/File:Five_different_rubber_ducks.jpg

- JDBC assumes uniform column typing
- Not true of SPARQL results
- We make the exact data typing behavior configurable

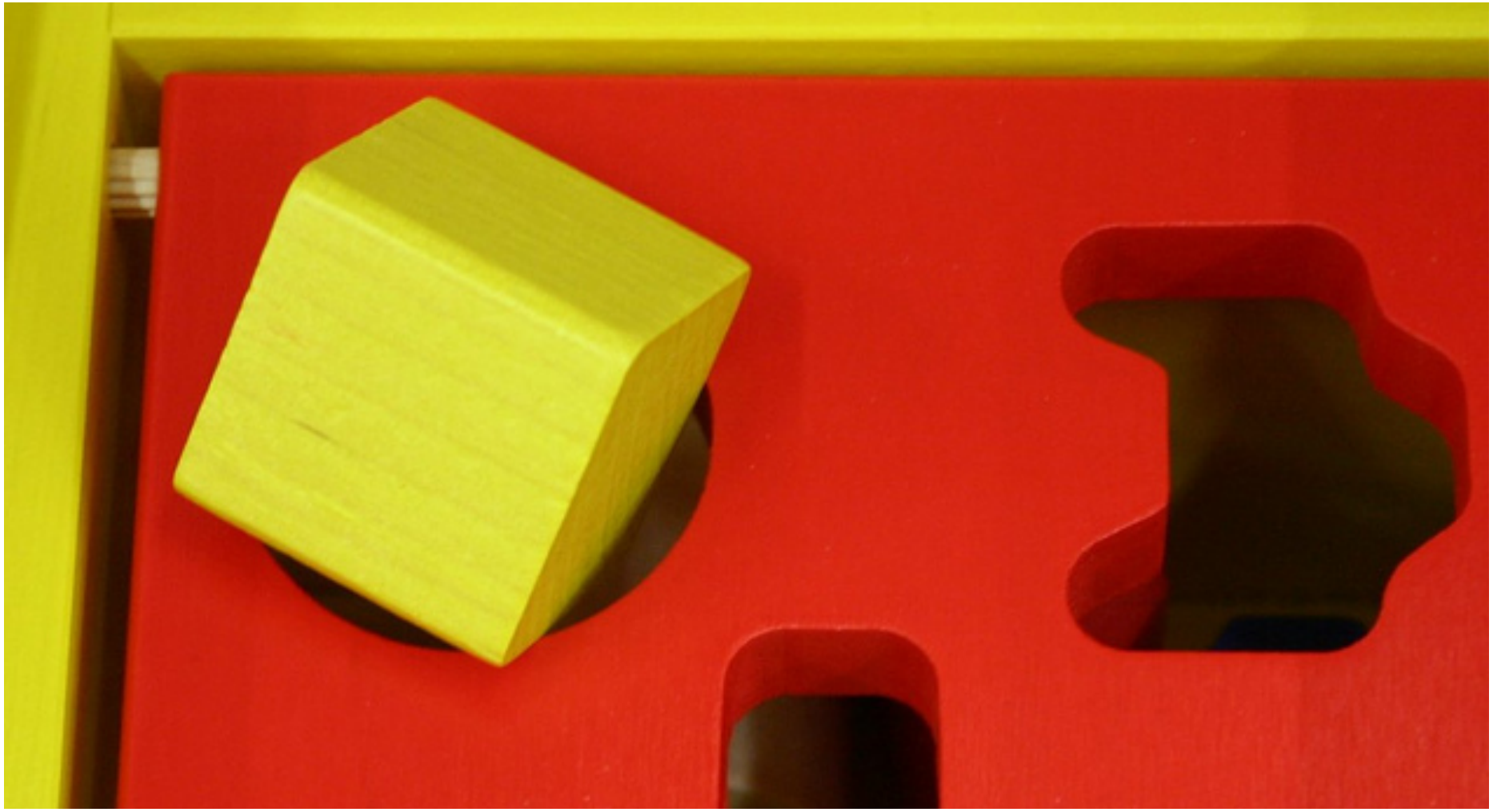
Fudging the metadata

Public Domain - Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Butter_tablet_fudge.jpg

Metadata support

- **Supported:**
  - Basic driver information
  - Type information
  - Function and keyword information
  - SQL language support
  - Result Set metadata
- **Not Supported:**
  - Table and Procedure information
  - Schema information
- **There are alternative solutions that do support more metadata but they represent a different architectural approach to the problem**

Supporting awkward tools



Pre and Post-Processing

- API allows for adding both pre and post-request processors to a connection
- Pre-processors can manipulate either the raw string or parsed Query/Update as appropriate
  - E.g. strip out extraneous syntax tools might add
- Post-processors can manipulate the raw results before they are returned
  - E.g. rewrite variable names to match what the tool expect

Example Code

Simple SPARQL over JDBC Example

```
// Open a Connection
Connection conn = DriverManager.getConnection("jdbc:jena:tdb:location=/
user/example/data/mydb/");

// Prepare a Statement
PreparedStatement stmt = conn.prepareStatement("SELECT * WHERE { GRAPH ?
{ ?s ?p ?o } }");
stmt.setURL(1, new URL("http://example/graph"));

// Execute the Statement
ResultSet rset = stmt.executeQuery();

// Process the results...

// Clean up as normal
rset.close();
conn.close();
```

Demo

Alternative Options

Alternative Libraries

- Our approach is not the only one available as open source
- Two similar approaches to ours
  - William Greenly's jdbc4sparql - http://code.google.com/p/jdbc4sparql/
  - Paul Gearon's scon - https://code.google.com/p/scon/wiki/Introduction
- An alternative approach is to map the RDF data into tables and translate SQL queries into SPARQL queries behind the scenes
  - Claude Warren's jdbc4sparql - https://github.com/Claudenw/jdbc4sparql

Questions?

Twitter: @RobVesse
Email: rvesse@apache.org

References

Topic	Link
Apache Jena Project	http://jena.apache.org
RDF 1.1 Specification	http://www.w3.org/TR/rdf11-concepts/
SPARQL 1.1 Specification	http://www.w3.org/TR/sparql11-overview/
Jena JDBC Documentation	http://jena.apache.org/documentation/jdbc/index.html
Jena JDBC Maven Artifacts Documentation	http://jena.apache.org/documentation/jdbc/artifacts.html
Jena JDBC Drivers Documentation - Supported Drivers and Connection String Parameters	http://jena.apache.org/documentation/jdbc/drivers.html
SPARQL 1.1 Protocol	http://www.w3.org/TR/sparql11-protocol/

Acknowledgments

Resource	License	Rightsholder	URL
Square Peg Round Hole Image	CC-BY-SA-NA 2.0	rosipaw	http://www.flickr.com/photos/rosipaw/4643095630/
Fudge Image	Public Domain	Wikimedia Commons	http://commons.wikimedia.org/wiki/File:Butter_tablet_fudge.jpg
Rubber Ducks Image	CC-BY-SA 3.0	Wikimedia Commons	http://commons.wikimedia.org/wiki/File:Five_different_rubber_ducks.jpg