

# RDB2RDF mapping with D2RQ and D2R Server

Richard Cyganiak

Presentation to W3C RDB2RDF WG, 10 Nov 2009

# Topics

1. The D2RQ project
2. The D2RQ mapping language
3. Requirements for RDB2RDF

# I. The project

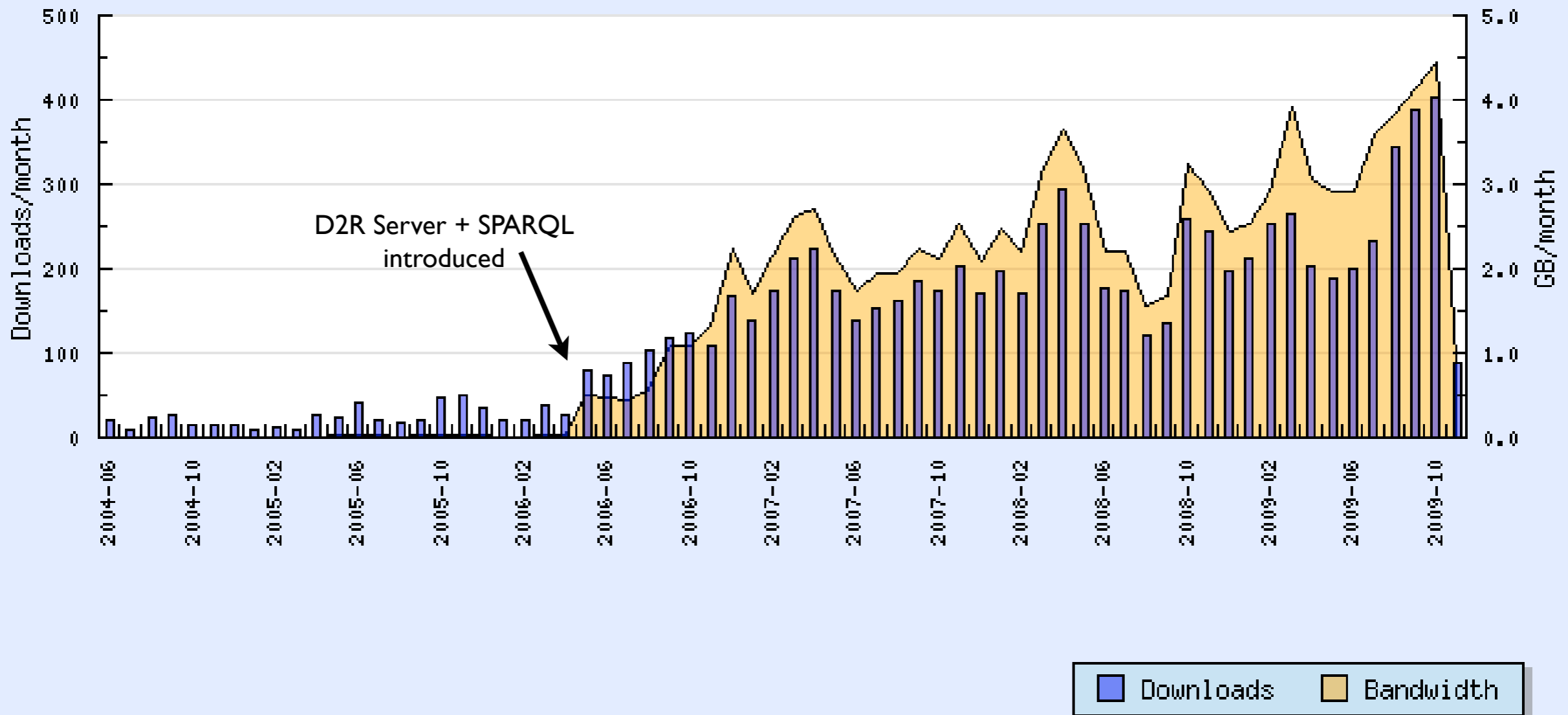
# D2RQ

- DB-to-RDF mapper written in Java
- In: any JDBC database
- Out: SPARQL, Linked Data, ETL, Jena API
- GPL, popular, easy to get started
- Axiom: We never modify the database

# The project

- Started 2004 (roots: 2002) by Chris Bizer at FU Berlin; later me at FU and HP Labs; today Christian Becker, Andy Langeegger
- 250+ downloads/month, 8700+ total
- mailing list at ~20 msgs/month, 1000+ total
- In LOD cloud, LinkedMDB, LODD, TopBraid Composer

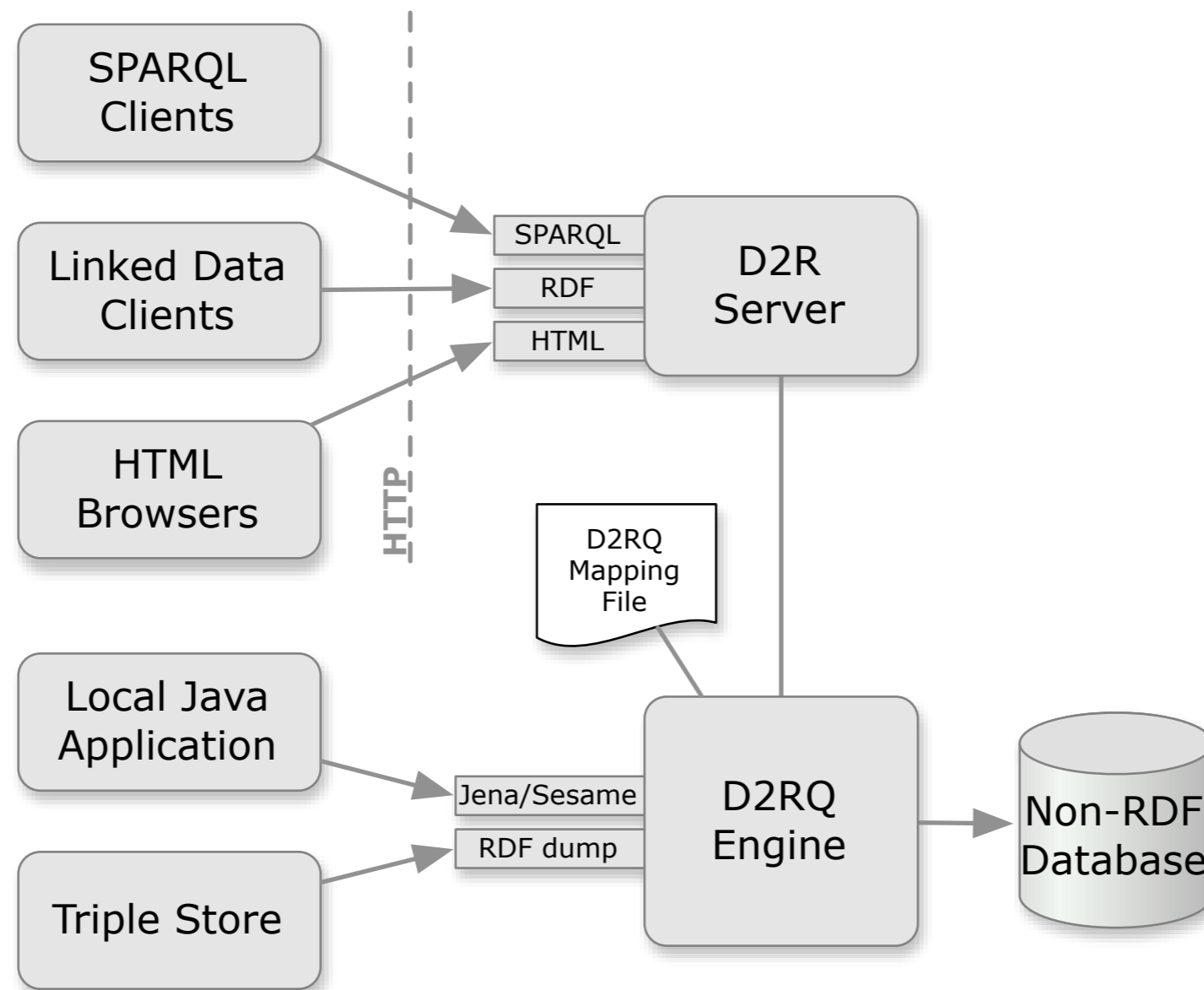
# Download History for All Files For D2RQ and D2R Server All Time



Generated 2009-11-10 16:18:38 UTC

Copyright SourceForge.net

# Architecture



# Architecture (2)

- maps DB to *virtual* RDF graph
- easy to offer arbitrary interfaces to the RDF graph
- most requested: SPARQL and RDF dumps

# 2. Mapping language

# Mapping language

- N3 based syntax
- Very flexible
- Language is not trivial, wish we had a GUI
- Usual workflow: auto-generate mapping from DB schema, then customize

# Flexible mappings!

- Properties of one class from multiple tables
- Several classes in the same table
- Value translations, SQL expressions
- Arbitrary joins and SQL conditions

# To SQL or not to SQL?

- Users want to deal with complexity by using their SQL knowledge
- They want to write arbitrary SQL queries
- We don't want to parse SQL (*painful, DB differences*)
- We force users to decompose their query into small fragments

# Mapping process

1. Define DB connection
2. Define your entities
3. Add properties to entites
4. Link entities together
5. Advanced stuff: conditions, joins, value translations

# I. Define DB connection

```
map:MyDatabase a d2rq:Database;  
  d2rq:jdbcDSN "jdbc:mysql://localhost/mydb";  
  d2rq:jdbcDriver "com.mysql.jdbc.Driver";  
  d2rq:username "user";  
  d2rq:password "password".
```

## 2. Define your entities

```
map:People a d2rq:ClassMap;  
  d2rq:uriPattern "http://.../people/@@User.ID@@".
```

(SQL fragments in red)

```
map:People a d2rq:ClassMap;  
  d2rq:uriPattern "http://.../people/@@User.ID@@";  
  d2rq:condition "User.deleted=0".
```

```
map:People a d2rq:ClassMap;  
  d2rq:bNodeIdColumns "User.ID";  
  d2rq:condition "User.deleted=0".
```

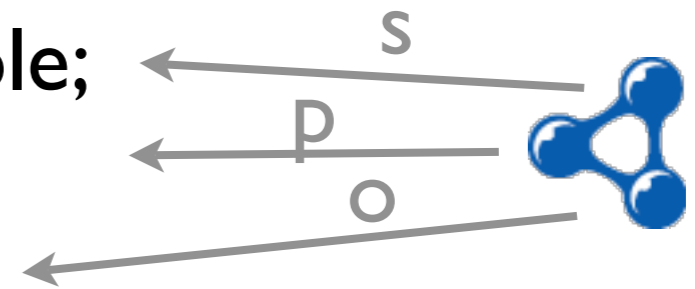
# 3. Add properties to entities

```
map:People a d2rq:ClassMap;  
  d2rq:uriPattern "http://.../people/@@User.ID@@";  
  d2rq:condition "User.deleted=0";  
  d2rq:class foaf:Person .
```

(SQL fragments in red, RDFS/OWL vocabulary in blue)

map:People a d2rq:ClassMap .

map:name a d2rq:PropertyBridge;  
d2rq:belongsToClassMap map:People;  
d2rq:property foaf:nick;  
d2rq:column "User.name".



map:mbox a d2rq:PropertyBridge;  
d2rq:belongsToClassMap map:People;  
d2rq:property foaf:mbox;  
d2rq:uriPattern "mailto:@@User.email@@".



```
map:mbox_sha1 a d2rq:PropertyBridge;  
d2rq:belongsToClassMap map:People;  
d2rq:property foaf:mbox_sha1sum;  
d2rq:sqlExpression
```

```
“SHA1(CONCAT('mailto:', User.email))”.
```

# 4. Link your entities

```
map:Photos a d2rq:ClassMap;  
  d2rq:uriPattern "http://.../photo/@@Photo.ID@@";  
  d2rq:class foaf:Image .
```

```
map:photo a d2rq:PropertyBridge;  
  d2rq:belongsToClassMap map:People;  
  d2rq:property foaf:made;  
  d2rq:uriPattern "http://.../photo/@@Photo.UserID@@".
```

(Photo.UserID is a foreign key to User.ID)

# Better, less repetition

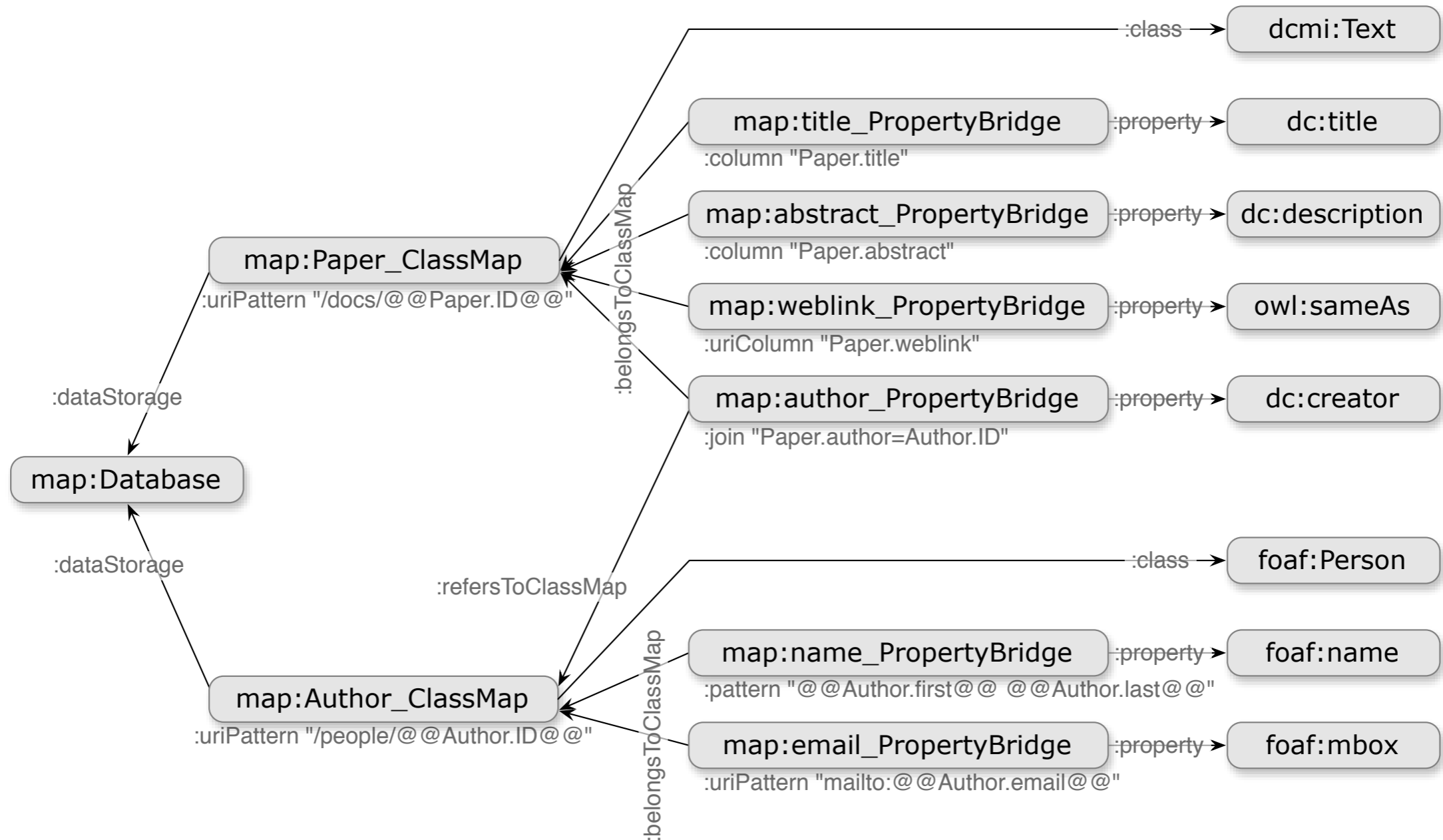
```
map:photo a d2rq:PropertyBridge;  
  d2rq:belongsToClassMap map:People;  
  d2rq:property foaf:made;  
  d2rq:join "User.ID = Photo.UserID";  
  d2rq:refersToClassMap map:Photos .
```

# Better, less repetition

```
map:photo a d2rq:PropertyBridge;  
  d2rq:belongsToClassMap map:People;  
  d2rq:property foaf:made;  
  d2rq:join "User.ID = Photo.UserID";  
  d2rq:refersToClassMap map:Photos .
```

(also d2rq:alias for self-joins)

# Mapping file overview



# 3. RDB2RDF Requirements

# Syntax?

- Turtle, XML, SPARQL-like, SQL-like?
- Should be human-writable
- Would like to avoid parsing SQL
- “SQL Query + RDF template” vs.  
“RDF Graph + SQL fragment”

# Expressivity?

- Arbitrary SQL for value transforms and conditions
- Dynamic properties
- Char-separated lists within values
- Transformation tables (for type codes)

# DB compatibility?

- Syntax rules for table/column names (spacing, case sensitivity)
- Datatypes
- Extension functions
- “AS”, “LIMIT”, “CONCAT”

# Links

- D2RQ homepage  
<http://www4.wiwiss.fu-berlin.de/bizer/d2rq/>
- D2RQ manual & language spec  
<http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/>
- Mailing list  
[d2rq-map-devel@lists.sourceforge.net](mailto:d2rq-map-devel@lists.sourceforge.net)