MASWS – Constructing RDF Datasets

Kate Byrne

7th February 2013
Outline

1. Converting Data to RDF
   - available tools and guidance
   - basic RDB2RDF conversion

2. RDF Schema Design
   - basic principles
   - “table as class; column as predicate”? or a variant?

3. Grounding Your RDF Resources
Warning!

A lot of this is just my opinion... ...you have to keep your own eyes open and brain in gear.
Converting Data to RDF

- Tim Berners-Lee proposed the Semantic Web in May 1994
- whatever have we been doing since then?
- Turning existing data into RDF is still tricky
It’s all about triplets!
Just a few years back, the W3C Semantic Web FAQ site, www.w3.org/RDF/FAQ#relodb:

“How do I export my data from a Relational Database?”

“This is one of the active areas of R&D, and no final answer is yet available…”

W3C Incubator Group on RDB2RDF

Reported June 2009:
www.w3.org/wiki/Rdb2RdfXG/StateOfTheArt
Current W3C Guidance on RDB2RDF

  - direct mapping – simple transformation
  - R2RML – customised mapping; RDF ‘views’ over SQL
  - ‘Recommendation’ status achieved in late September 2012
- RdfAndSQL Wiki: [www.w3.org/wiki/RdfAndSql](http://www.w3.org/wiki/RdfAndSql)
  - various tools available – D2R, SquirrelRDF, Virtuoso, etc.
  - unlikely to be using R2RML yet
- This lecture: not the W3C syntax but the underlying issues in RDB2RDF conversion, using practical example
RCAHMS
The Royal Commission on the Ancient and Historical Monuments of Scotland

- One of Scotland’s 'National Collections'
- Record of archaeological sites and historic buildings in Scotland
- 300,000 sites + 1 million archive items (photos, maps, plans..)
- Structured RDBMS plus associated text documents
- Where we are now, 7 George Square: canmore.rcahms.gov.uk/en/site/74113/
The photo, drawing and map all pertain to Dirleton Castle
Dirleton Cottage and Jamie’s Neuk are also on the parish map
There is no archive associated with Drem Airfield
Database Conversion – Growing Spider Plants

Relational Database

<table>
<thead>
<tr>
<th>siteNo</th>
<th>name</th>
<th>parish</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dirleton Castle</td>
<td>Dirleton</td>
<td>scheduled</td>
</tr>
<tr>
<td>2</td>
<td>Dirleton Cottage</td>
<td>Dirleton</td>
<td>listed</td>
</tr>
<tr>
<td>3</td>
<td>Drem Airfield</td>
<td>Dirleton</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Jamie’s Neuk</td>
<td>Dirleton</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>siteNo</th>
<th>arcNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>1</td>
<td>103</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>103</td>
</tr>
</tbody>
</table>

ARCHIVE

<table>
<thead>
<tr>
<th>arcNo</th>
<th>arcType</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>photo</td>
<td>North face</td>
</tr>
<tr>
<td>102</td>
<td>drawing</td>
<td>Site plan</td>
</tr>
<tr>
<td>103</td>
<td>map</td>
<td>Parish map</td>
</tr>
</tbody>
</table>

RDF

Converting to RDF
A Simple Example – One Database Record

Table as Class, Column as Predicate conversion

<table>
<thead>
<tr>
<th>siteNo</th>
<th>name</th>
<th>parish</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dirleton Castle</td>
<td>Dirleton</td>
<td>defence</td>
</tr>
<tr>
<td>2</td>
<td>Dirleton Cottage</td>
<td>Dirleton</td>
<td>residential</td>
</tr>
<tr>
<td>3</td>
<td>Drem Airfield</td>
<td>Dirleton</td>
<td>military</td>
</tr>
<tr>
<td>4</td>
<td>Jamie’s Neuk</td>
<td>Dirleton</td>
<td>military</td>
</tr>
</tbody>
</table>

@prefix : <http://www.ltg.ed.ac.uk/tether/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

- Each row, or instance, of SITE forms a new offshoot –
- central node, surrounded by cluster of attributes
- Principles same whether from RDBMS or .csv dump
With real data we have a **lot** of attributes
Storing RDF triples

 Traverse links by “self-joins” on table.
1 Converting Data to RDF
   - available tools and guidance
   - basic RDB2RDF conversion

2 RDF Schema Design
   - basic principles
   - “table as class; column as predicate”? or a variant?

3 Grounding Your RDF Resources
Good Design is *Simple!*

Edinburgh University Gliding Club: [gliding.eusu.ed.ac.uk](http://gliding.eusu.ed.ac.uk/)
### RDBMS Schema Design

**Real-world entities map to database tables (relations)**

- Each row (tuple) describes an **instance**
- Each column contains values for an **attribute**
- Entity relationships expressed as table joins, on keys

<table>
<thead>
<tr>
<th>siteNo</th>
<th>name</th>
<th>parish</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dirleton Castle</td>
<td>Dirleton</td>
<td>scheduled</td>
</tr>
<tr>
<td>2</td>
<td>Dirleton Cottage</td>
<td>Dirleton</td>
<td>listed</td>
</tr>
<tr>
<td>3</td>
<td>Drem Airfield</td>
<td>Dirleton</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Jamie's Neuk</td>
<td>Dirleton</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>siteNo</th>
<th>archNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>1</td>
<td>103</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>103</td>
</tr>
</tbody>
</table>

**ARCHIVE**

<table>
<thead>
<tr>
<th>archNo</th>
<th>category</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>photo</td>
<td>North face</td>
</tr>
<tr>
<td>102</td>
<td>drawing</td>
<td>Site plan</td>
</tr>
<tr>
<td>103</td>
<td>map</td>
<td>Parish map</td>
</tr>
</tbody>
</table>
Querying RDBMS Tables with SQL

```sql
select s.siteNo, s.name, a.category
from site s
  left join site-arch sa on s.siteNo=sa.siteNo
  left join archive a on sa.archive=a.archNo
where s.parish='Dirleton'
order by s.siteNo;
```

---

**SITE**

<table>
<thead>
<tr>
<th>siteNo</th>
<th>name</th>
<th>parish</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dirleton Castle</td>
<td>Dirleton</td>
<td>scheduled</td>
</tr>
<tr>
<td>2</td>
<td>Dirleton Cottage</td>
<td>Dirleton</td>
<td>listed</td>
</tr>
<tr>
<td>3</td>
<td>Drem Airfield</td>
<td>Dirleton</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Jamie’s Neuk</td>
<td>Dirleton</td>
<td></td>
</tr>
</tbody>
</table>

**SITE-ARCH**

<table>
<thead>
<tr>
<th>siteNo</th>
<th>archNo</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101</td>
</tr>
<tr>
<td>1</td>
<td>102</td>
</tr>
<tr>
<td>1</td>
<td>103</td>
</tr>
<tr>
<td>2</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>103</td>
</tr>
</tbody>
</table>

**ARCHIVE**

<table>
<thead>
<tr>
<th>archNo</th>
<th>category</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>photo</td>
<td>North face</td>
</tr>
<tr>
<td>102</td>
<td>drawing</td>
<td>Site plan</td>
</tr>
<tr>
<td>103</td>
<td>map</td>
<td>Parish map</td>
</tr>
</tbody>
</table>

---

**Dirleton Castle**

- Scheduled
- Photo: North face

**Dirleton Cottage**

- Listed
- Drawing: Site plan

**Drem Airfield**

- Map: Parish map

**Jamie’s Neuk**

- Map: Parish map
RDF Schema Design

- Hierarchy of Classes models the real-world entities (rdf:Class)
- Predicates model relationships between them (rdf:Property)
- RDFS lets you define structure around these:
  - rdfs:subClassOf, rdfs:subPropertyOf, domains and ranges

Eg: *hasDateOfBirth* might be...

:s:hasDateOfBirth rdfs:domain :Person .
s:hasDateOfBirth rdfs:range :Date .

- Use OWL if more detailed structure needed:
  - owl:inverseOf, owl:SymmetricProperty, owl:TransitiveProperty
- Note: Title Case for Classes, lower case for properties
Querying Basic RDF Conversion with SPARQL

```
select ?siteNo ?name ?category
where {
  ?siteNo :parish "Dirleton" .
  ?siteNo :name  ?name .
  optional {
    ?siteNo :siteArch ?archNo .
  }
}
```
Basic RDF Conversion v Good Schema Design

- “Table as Class” means one rdf:Class per RDBMS table
  - flat and relatively small Class hierarchy
- “Column as predicate” generates lots of Properties
  - complex graph structure
- Querying requires detailed knowledge of RDBMS schema
- What have we gained over RDBMS?
- Issues with bnodes, nulls, literals, data-bloat,…
- Let’s examine some of the problems, explore alternatives
What’s the Purpose of Converting to RDF?

- Related data stores are everywhere
  - RCAHMS archaeological sites – NMS excavation finds
- RDF is designed for linking
  - but graph must be queryable, nodes must be mergeable
Issues with RDF Conversion

1. Primary keys and bnodes
2. Values as literals or as resources?
3. Redundant nodes generated by foreign keys
4. Dealing with n-ary relations
5. A triple, S-P-O, should be ’noun-verb-noun’

6. Easy ones:
   - Nulls – skip them
   - Coded values – dereference them
1. To Bnode or Not to Bnode?

- Centre clusters around primary keys; drop the horrid bnodes
- Note: no longer 'table as class', strictly speaking
2. Literals or Resources?
2. Literals or Resources? *ctd*

- Graph is sterilised at literals – no further links
- Encode database values as URIs
  - all?
  - just those that could ever be Subject of triple
  - how to distinguish?
- If all values encoded we get some unlikely URIs:

**Is this useful?**

PhotoDesc field value: ' #5: 6”x4” neg, B&W’ becomes
http://www.ex.com/Pdesc/%235:%206%22%20x%22%20neg%2C%20B%26W
3. Redundant Nodes at Relational Joins

- **SITE**
  - siteNo
  - name
  - parish

- **ARCHIVE**
  - arcNo
  - arcType
  - description

- **SITE-ARC**
  - siteArcNo
  - siteNo
  - arcNo

3 million triples saved in my 22 million triple dataset
4. Dealing with n-ary Relations

*cf.* RDBMS Relations

- RDF is binary relations: `hasParish(Drem_Airfield, Dirleton)`
- Not all relations are binary, e.g. finding something:

  Mr. Peter Moar found a small hoard of 5 polished stone knives on the same patch in May 1946.

\[
\text{eventRel(eventAgent, eventAgentRole, eventDate, eventPatient, eventPlace)}
\]

\[
\text{find123(“Mr. Peter Moar”, “May 1946”, “polished stone knives”, “Hill of Shurton”)}
\]

Split event relation into RDF triples:

\[
\text{:find123 :hasAgent :peterMoar .}
\]

\[
\text{:find123 :hasDate :may1946 .}
\]

\[
\text{:find123 :hasPatient :polishedStoneKnives .}
\]

\[
\text{:find123 :hasLocation :hillOfShurton .}
\]
5. To Be or To Do? Nouns or Verbs?

- Simple binary relation: \textit{eats(Person, Food)}
- Instance: \textit{eats(Kate, chocolate)}

What you might expect −

<table>
<thead>
<tr>
<th>Property</th>
<th>eatsPerson</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kate</td>
<td>choc</td>
<td>eats</td>
</tr>
</tbody>
</table>

What you get −

<table>
<thead>
<tr>
<th>Eats</th>
<th>Person</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kate</td>
<td>choc</td>
<td></td>
</tr>
</tbody>
</table>

- Properties (verbs) and Classes (nouns) change places
5. Nouns or Verbs? *ctd*

eats(Kate, chocolate)

What you might expect –

- **Property**
  - **Person**
  - eats
  - **Food**
  - domain
  - range

Kate → eats → choc

domain range

typetype
type
eats(Kate, chocolate)

What you get –

**Eats**

<table>
<thead>
<tr>
<th>Person</th>
<th>Food</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kate</td>
<td>choc</td>
</tr>
</tbody>
</table>

**Class**

- eats
  - type
- **Person**
  - hasAgent
- **Food**
  - hasPatient

My variant –

- **eatingEvent1**
  - **Person**
  - hasAgent
  - type
  - Kate

- **Food**
  - hasPatient
  - type
  - choc
Nouns or Verbs? – Conclusion

- 'Column as Class' not 'Column as predicate'
- Database columns are *things*: name, parish, archive category
  - so express them as Classes in RDF
  - merge related Classes under generic parent
- RDF predicates express relationships between instances
  - keep as generic as possible: `hasLocation`, `hasAgent`, `hasDate`
  - make the set as small as possible – easier to query
- Use `rdf:type` (ie Class membership) to provide granularity
Example of RDF Schema on these Principles – *Tether*

<table>
<thead>
<tr>
<th>siteNo</th>
<th>name</th>
<th>parish</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dirleton Castle</td>
<td>Dirleton</td>
<td>defence</td>
</tr>
<tr>
<td>2</td>
<td>Dirleton Cottage</td>
<td>Dirleton</td>
<td>residential</td>
</tr>
<tr>
<td>3</td>
<td>Drem Airfield</td>
<td>Dirleton</td>
<td>military</td>
</tr>
<tr>
<td>4</td>
<td>Jamie’s Neuk</td>
<td>Dirleton</td>
<td>military</td>
</tr>
</tbody>
</table>

@prefix : <http://www.ltg.ed.ac.uk/tether/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
1. Converting Data to RDF
   - available tools and guidance
   - basic RDB2RDF conversion

2. RDF Schema Design
   - basic principles
   - “table as class; column as predicate”? or a variant?

3. Grounding Your RDF Resources
Linking 'Linked Data' is Actually Pretty Hard

- Direct link means spotting identical node in separate graph
- How? String matching? Clues from context?
Using the LOD Cloud as Intermediary – 'Authority Nodes'

grounding local URIs against "authority" nodes is the next big challenge!
Finally – Some Useful References

- Tim Berners-Lee: www.w3.org/DesignIssues/RDB-RDF.html
- RDB2RDF Working Group: www.w3.org/2001/sw/rdb2rdf/
- Use Cases and Requirements: www.w3.org/TR/rdb2rdf-ucr/
- RdfAndSQL Wiki: www.w3.org/wiki/RdfAndSql
- R2RML: www.w3.org/TR/r2rml/
- (Chapter 5 of my PhD thesis: hdl.handle.net/1842/3781)