MASWS – Relational Databases and RDF

Kate Byrne

8th February 2008
Outline

Information Management and Relational Databases
   some history
   relational database basics

Converting Relational Databases to RDF
   why would you want to?
   how? – the basic conversion procedure

What’s Wrong With the Basic Procedure?
   bnodes
   literals
   nouns or verbs?

Useful Websites
Milestones in Information Management

- 1450s:
  - Gutenberg invents movable type printing
  - (Aside: 2008 marks quincentenary of printing in Scotland. See http://www.500yearsofprinting.org/)

- 1950s:
  - First modern computers come into use

- 1970s:
  - The Internet emerges
  - Database management systems begin to be widely used
  - Ted Codd proposes the Relational Model

- 1990s:
  - Tim Berners-Lee invents the World Wide Web (1990)
  - First version of RDF becomes W3C Recommendation
  - Tim Berners-Lee proposes the Semantic Web (1999)
Milestones in Information Management

• 1450s:
  • Gutenberg invents movable type printing
  • (Aside: 2008 marks quincentenary of printing in Scotland. See http://www.500yearsofprinting.org/)

• 1950s:
  • first “modern” computers come into use

• 1970s:
  • the Internet emerges
  • database management systems begin to be widely used
  • Ted Codd proposes the Relational Model

• 1990s:
  • Tim Berners-Lee invents the World Wide Web (1990)
  • first version of RDF becomes W3C Recommendation
  • Tim Berners-Lee proposes the Semantic Web (1999)
Milestones in Information Management

- **1450s:**
  - Gutenberg invents movable type printing

- **1950s:**
  - First “modern” computers come into use

- **1970s:**
  - The Internet emerges
  - Database management systems begin to be widely used
  - Ted Codd proposes the Relational Model

- **1990s:**
  - Tim Berners-Lee invents the World Wide Web (1990)
  - First version of RDF becomes W3C Recommendation
  - Tim Berners-Lee proposes the Semantic Web (1999)
Milestones in Information Management

- **1450s:**
  - Gutenberg invents movable type printing

- **1950s:**
  - first “modern” computers come into use

- **1970s:**
  - the Internet emerges
  - database management systems begin to be widely used
  - Ted Codd proposes the Relational Model

- **1990s:**
  - Tim Berners-Lee invents the World Wide Web (1990)
  - first version of RDF becomes W3C Recommendation
  - Tim Berners-Lee proposes the Semantic Web (1999)
Milestones in Information Management

• 1450s:
  • Gutenberg invents movable type printing
  • (Aside: 2008 marks quincentenary of printing in Scotland. See http://www.500yearsofprinting.org/)

• 1950s:
  • first “modern” computers come into use

• 1970s:
  • the Internet emerges
  • database management systems begin to be widely used
  • Ted Codd proposes the Relational Model

• 1990s:
  • Tim Berners-Lee invents the World Wide Web (1990)
  • first version of RDF becomes W3C Recommendation
  • Tim Berners-Lee proposes the Semantic Web (1999)
Relational Database Basics

Relations – Tables – Entities – ?Classes

### 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>

### 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>
## Relational Database Basics

### Relations – Tables – Entities – ?Classes

<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>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>

- Each row (tuple) describes an **instance**
- Each column contains values for an **attribute**
- Null values (generally) permitted
- A relation is a **set** of tuples:  
  \[
  \text{site}(\text{siteNo}, \text{name}, \text{parish}, \text{status})
  \]
Relational Database Basics

Relations – Tables – Entities – ?Classes

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>

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>

- Each row (tuple) describes an **instance**
- Each column contains values for an **attribute**
- Null values (generally) permitted
- A relation is set of tuples: site(siteNo, name, parish, status)
Relational Database Basics

Relations – Tables – Entities – ?Classes

### 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>

### 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>

- Each row (tuple) describes an **instance**
- Each column contains values for an **attribute**
- Null values (generally) permitted
- A relation is set of tuples: `site(siteNo, name, parish, status)`
Relational Database Basics

Relations – Tables – Entities – ?Classes

<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>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>

- Each row (tuple) describes an **instance**
- Each column contains values for an **attribute**
- Null values (generally) permitted
- A relation is **set** of tuples:
  
  `site(siteNo, name, parish, status)`
Relational Database Basics

Relational Joins

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.
### Relational Database Basics

#### Relational Joins

<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>

<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>

- 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
Relational Database Basics

Relational Joins

- 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
### Relational Database Basics

#### Relational Joins

<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>

<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>

- 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.
Where is RDF stored anyway?

• If your RDF graph is too big to fit in memory...
  • ...you store it in a relational database
  • Many triple stores available: Jena, Kowari, Sesame, ...
  • These all use database back-ends, typically MySQL
    • 3-column table, for Subject, Predicate, Object
    • sometimes GraphId too
    • physical implementations differ
• Relational database query language is SQL
• ➞ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

• If your RDF graph is too big to fit in memory...
• ...you store it in a relational database
  • Many triple stores available: Jena, Kowari, Sesame,...
  • These all use database back-ends, typically MySQL
    • 3-column table, for Subject, Predicate, Object
    • sometimes GraphId too
    • physical implementations differ
  • Relational database query language is SQL
  • ⇒ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
  - physical implementations differ
- Relational database query language is SQL
- ⇒ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
  - physical implementations differ
- Relational database query language is SQL
- ⇒ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
  - physical implementations differ
- Relational database query language is SQL
- $\Rightarrow$ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
    - physical implementations differ
- Relational database query language is SQL
- \( \Rightarrow \) SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

• If your RDF graph is too big to fit in memory...
• ...you store it in a relational database
• Many triple stores available: Jena, Kowari, Sesame,...
• These all use database back-ends, typically MySQL
  • 3-column table, for Subject, Predicate, Object
  • sometimes GraphId too
  • physical implementations differ
• Relational database query language is SQL
• ⇒ SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
  - physical implementations differ
- Relational database query language is SQL
  - \( \Rightarrow \) SPARQL has to be translated (invisibly) into SQL
Where is RDF stored anyway?

- If your RDF graph is too big to fit in memory...
- ...you store it in a relational database
- Many triple stores available: Jena, Kowari, Sesame,...
- These all use database back-ends, typically MySQL
  - 3-column table, for Subject, Predicate, Object
  - sometimes GraphId too
  - physical implementations differ
- Relational database query language is SQL
- $\Rightarrow$ SPARQL has to be translated (invisibly) into SQL
Storing RDF triples

- traverse links by “self-joins” on table
Terminology

RDF triple:

- **subject** → **object**
  - **predicate**

(Or, if you prefer, **Subject** − **Verb** − **Object**.)

In database terms:

- **rowId** → **value**
  - **attribute**

- **:site1** → **"Dirleton Castle"**
  - **:name**

Note that **:name** just depends on schema designer’s choice, whereas **"Dirleton Castle"** is real data.
Terminology

RDF triple:

(subject, predicate, object)

(Or, if you prefer, Subject – Verb – Object.)

In database terms:

(rowId, attribute, value)

:site1 :name "Dirleton Castle"

• Note that :name just depends on schema designer’s choice...

• ...whereas “Dirleton Castle” is real data...
Terminology

RDF triple:

subject \[\rightarrow\] predicate \[\rightarrow\] object

(Or, if you prefer, Subject – Verb – Object.)

In database terms:

rowId \[\rightarrow\] attribute \[\rightarrow\] value

:site1 :name "Dirleton Castle"

• Note that :name just depends on schema designer’s choice...
• ...whereas “Dirleton Castle” is real data
Information Management and Relational Databases
  some history
  relational database basics

Converting Relational Databases to RDF
  why would you want to?
  how? – the basic conversion procedure

What’s Wrong With the Basic Procedure?
  bnodes
  literals
  nouns or verbs?

Useful Websites
The Hidden/Invisible/Deep Web Problem

• Most data is (still) in databases, especially “good” data:
  • carefully curated datasets, built over decades/centuries
  • like CANMORE – architecture and archaeology across Scotland – at http://www.rcahms.gov.uk/
• Web crawlers can’t see inside databases –
• – unless you “expose” individual search results:
• http://www.rcahms.gov.uk/pls/portal/newcanmore.newcandig_p_coll_details?p.arcnumlink=778111
The Hidden/Invisible/Deep Web Problem

- Most data is (still) in databases, especially “good” data:
  - carefully curated datasets, built over decades/centuries
  - like CANMORE – architecture and archaeology across Scotland – at http://www.rcahms.gov.uk/
- Web crawlers can’t see inside databases –
- unless you “expose” individual search results:
The Hidden/Invisible/Deep Web Problem

- Most data is (still) in databases, especially “good” data:
  - carefully curated datasets, built over decades/centuries
  - like CANMORE – architecture and archaeology across Scotland – at http://www.rcahms.gov.uk/

- Web crawlers can’t see inside databases –
- – unless you “expose” individual search results:
The Hidden/Invisible/Deep Web Problem

- Most data is (still) in databases, especially “good” data:
  - carefully curated datasets, built over decades/centuries
  - like CANMORE – architecture and archaeology across Scotland – at http://www.rcahms.gov.uk/

- Web crawlers can’t see inside databases –
  - unless you “expose” individual search results:
The Hidden/Invisible/Deep Web Problem

- Most data is (still) in databases, especially “good” data:
  - carefully curated datasets, built over decades/centuries
- Web crawlers can’t see inside databases –
- – unless you “expose” individual search results:
The Hidden/Invisible/Deep Web Problem

- Most data is (still) in databases, especially “good” data:
  - carefully curated datasets, built over decades/centuries
  - like CANMORE – architecture and archaeology across Scotland – at http://www.rcahms.gov.uk/
- Web crawlers can’t see inside databases –
- – unless you “expose” individual search results:
Linked Data

- **Related databases everywhere:**
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- **Interconnecting relational databases is hard:**
  - you need to know the schema in detail

- security issues
- complex networking protocols – not http
- whereas RDF was designed for data linking...
Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

Interconnecting relational databases is hard:
  - you need to know the schema in detail
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- Interconnecting relational databases is hard:
  - you need to know the schema in detail
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records
- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you cannot query without knowing attribute names
    - with SPARQL, you can
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Linked Data

• Related databases everywhere:
  • RCAHMS archaeological sites – NMS excavation finds
  • company merger; personnel records

• Interconnecting relational databases is hard:
  • you need to know the schema in detail
    • with SQL, you cannot query without knowing attribute name
    • with SPARQL, you can
  • security issues
  • complex networking protocols – not http
  • whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you **cannot** query without knowing attribute name
    - with SPARQL, you can
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you **cannot** query without knowing attribute name
    - with SPARQL, you can

- security issues
- complex networking protocols – not http
- whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you **cannot** query without knowing attribute name
    - with SPARQL, you can
  - security issues
    - complex networking protocols – not http
    - whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records
- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you cannot query without knowing attribute name
    - with SPARQL, you can
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Linked Data

- Related databases everywhere:
  - RCAHMS archaeological sites – NMS excavation finds
  - company merger; personnel records

- Interconnecting relational databases is hard:
  - you need to know the schema in detail
    - with SQL, you **cannot** query without knowing attribute name
    - with SPARQL, you can
  - security issues
  - complex networking protocols – not http
  - whereas RDF was designed for data linking...
Dataset Linking in RDF

- Same node appears in two graphs?
- graphs are automatically linked
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso

- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
Alternatives to Conversion

- You don’t *have* to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
  - All need the database schema available
  - Principles same as for full conversion
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
Alternatives to Conversion

- You don’t have to instantiate the database as RDF:
  - query SQL database using SPARQL – eg SquirrelRDF, R2D2
  - virtual RDF graph interface for relational dbs – eg D2RQ
  - hybrid “middleware” database engine – eg Virtuoso
- Saves duplicating data – but lots of work at query time
- All need the database schema available
- Principles same as for full conversion
How to Convert DBs - W3C Guidance

- RDF has been around since 1999...
- From the W3C Semantic Web FAQ site, http://www.w3.org/RDF/FAQ#reldb:

  “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.”

- Much discussion, and various prototypes, eg D2R, dbview
- My view of conversion? Think spider plant.
How to Convert DBs - W3C Guidance

• RDF has been around since 1999...

• From the W3C Semantic Web FAQ site, http://www.w3.org/RDF/FAQ#reldb:

  “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.”

• Much discussion, and various prototypes, eg D2R, dbview

• My view of conversion? Think spider plant.
How to Convert DBs - W3C Guidance

- RDF has been around since 1999...
- From the **W3C Semantic Web FAQ** site, [http://www.w3.org/RDF/FAQ#relodb](http://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."

- Much discussion, and various prototypes, eg D2R, dbview
- My view of conversion? Think spider plant.
How to Convert DBs - W3C Guidance

- RDF has been around since 1999...
- From the **W3C Semantic Web FAQ** site, http://www.w3.org/RDF/FAQ#reldb:

  "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."

- Much discussion, and various prototypes, eg D2R, dbview
- My view of conversion? Think spider plant.
How to Convert DBs - W3C Guidance

- RDF has been around since 1999...
- From the W3C Semantic Web FAQ site, http://www.w3.org/RDF/FAQ#reldb:
  
  "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."

- Much discussion, and various prototypes, eg D2R, dbview
- My view of conversion? Think spider plant.
How to Convert DBs - W3C Guidance

- RDF has been around since 1999...
- From the **W3C Semantic Web FAQ** site, http://www.w3.org/RDF/FAQ#reldb:

  "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."

- Much discussion, and various prototypes, eg D2R, dbview
- My view of conversion? Think spider plant.
DB 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>

### Site Plan

<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
### A Simple Example – One Database Record

<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>

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

```rdfs
:rdfs:Class

:site

  rdf:type

  "Dirleton Castle"

  :name

  "1"

  :SiteNo

  "Dirleton Castle"

  :classification

  "defence"

  :parish

  "Dirleton"

```

---

**RelDB-to-RDF conversion problems websites**

---

**School of Informatics**
Generated Graph for a Real Database Record
Information Management and Relational Databases
  some history
  relational database basics

Converting Relational Databases to RDF
  why would you want to?
  how? – the basic conversion procedure

What’s Wrong With the Basic Procedure?
  bnodes
  literals
  nouns or verbs?

Useful Websites
To Bnode or Not to Bnode?

- duck typing – is it good data management?
- primary keys: important data items need direct reference
- needs schema knowledge
To Bnode or Not to Bnode?

- **Duck typing** – is it good data management?
- **Primary keys**: important data items need direct reference
- Needs schema knowledge
To Bnode or Not to Bnode?

- duck typing – is it good data management?
- primary keys: important data items need direct reference
- needs schema knowledge
Literals or Resources?
Literals or Resources?
Literals or Resources?

- RelDB-to-RDF conversion problems
- Websites

```
rdfs:Class :site
<siteName>
"East Lothian"
:location

<:classification>
"defence"
:parish
"Dirleton"

:location
"East Lothian"

:dirletonCastle
"Dirleton Castle"
:rdfs:label
"Dirleton Castle"
```

```
rdfs:Class :site
<siteName>
"East Lothian"
:location

<:classification>
"defence"
:parish
"Dirleton"

:location
"East Lothian"

:dirletonCastle
"Dirleton Castle"
:rdfs:label
"Dirleton Castle"
```
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc - '#5: 6"x4" neg, B&W'
  - http://www.ltg.ed.ac.uk/naavarb/photodesc#%235:
    %206%22x4%22%20neg%2C%20B%26W
  - # and & need special care
- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
  - Encode database values as URIs
  - Some unlikely URIs:
    - PhotoDesc = '#5: 6''x4'' neg, B&W'
    - http://www.ltg.ed.ac.uk/maevedb/photodesc/#%235:
      %206%22x4%22%20neg%2C%20B%26W
    - # and & need special care
  - What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
  - Some unlikely URIs:
    - PhotoDesc = '#5: 6''x4'' neg, B&W'
    - http://www.ltg.ed.ac.uk/maswsdb/photosdesc#%235:
      %206%22x4%22%20neg%2C%20B%26W
    - # and & need special care
- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc – ‘#5: 6”x4” neg, B&W’
  - http://www.ltg.ed.ac.uk/maswsdb/photodesc#%235:
    %206%22x4%22%20neg%2C%20B%26W
  - # and & need special care
- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc – ‘#5: 6”x4” neg, B&W’
    - http://www.ltg.ed.ac.uk/maswsdb/photodesc#%235:%206%22x4%22%20neg%2C%20B%26W
  - # and & need special care
- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc – ’#5: 6” x 4” neg, B&W’
  - http://www.ltg.ed.ac.uk/maswsdb/photodesc#%235: %206%22x4%22%20neg%2C%20B%26W
- # and & need special care
- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc – ’#5: 6”x4” neg, B&W’
  - http://www.ltg.ed.ac.uk/maswsdb/photodesc#/235:%206%22x4%22%20neg%2C%20B%26W
  - # and & need special care

- What are these URIs for?
Literals or Resources?

- Avoid literals!
- Graph is sterilised at literals – no further links
- Encode database values as URIs
- Some unlikely URIs:
  - PhotoDesc – '#5: 6'' x 4'' neg, B&W'
  - http://www.ltg.ed.ac.uk/maswsdb/photodesc#%235:%206%22x4%22%20neg%20neg%20B%20W
  - # and & need special care
- What are these URIs for?
Some Numbers

- Database with around 1 million records
- 235 million potential key-attribute-value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
  - Inwards: :site1 :name :dirletonCastle
  - Outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key–attribute–value triples
  - 15 million after pre-joining, denormalising, removing nulls
  - 50 million once schema triples added –
  - – 3 triples per database value
    - Inwards: :site1 :name :dirletonCastle
    - Outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key–attribute–value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
    - Inwards: rdfs:label :dirletonCastle
    - Outwards: rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key-attribute-value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
    - Inwards: site1 :name :dirletonCastle
    - Outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key-attribute-value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
    - inwards: :site1 :name :dirletonCastle .
    - outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key-attribute-value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
    - inwards: :site1 :name :dirletonCastle .
    - outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
Some Numbers

• Database with around 1 million records
• 235 million potential key-attribute-value triples
• 15 million after pre-joining, denormalising, removing nulls
• 50 million once schema triples added –
  • – 3 triples per database value
    • inwards: :site1 :name :dirletonCastle .
    • outwards: rdf:type, rdfs:label
• Multiple self-joins on 50 million row table
Some Numbers

- Database with around 1 million records
- 235 million potential key-attribute-value triples
- 15 million after pre-joining, denormalising, removing nulls
- 50 million once schema triples added –
  - 3 triples per database value
    - inwards: :site1 :name :dirletonCastle .
    - outwards: rdf:type, rdfs:label
- Multiple self-joins on 50 million row table
To Be or To Do? Nouns or Verbs?

- Properties (verbs) and Classes (nouns) change places
- Simple binary relation: `eats(Person, Food)`
- Instance: `eats(Kate, chocolate)`

What you might expect –

- `Property` `Person` `Food`
- `domain` `range`
- `type` `type`
- `Kate` `choc`
- `eats` `eats`
To Be or To Do? Nouns or Verbs?

- Properties (verbs) and Classes (nouns) change places
- Simple binary relation: `eats(Person, Food)`
- Instance: `eats(Kate, chocolate)`

What you might expect −

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

What you get −

<table>
<thead>
<tr>
<th>Class</th>
<th>type</th>
<th>type</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eats</td>
<td></td>
<td>Person</td>
<td>Food</td>
</tr>
<tr>
<td>Kate</td>
<td></td>
<td></td>
<td>choc</td>
</tr>
</tbody>
</table>
Assisted Automatic Conversion

eats(Kate, chocolate)

What you might expect –

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>

My variant –

School of Informatics
Putting Design into the Schema

- Standard approach duplicates data (bad!):
  - database attribute name becomes RDF predicate
  - database attribute name appears in target node (data source)
  - database attribute name becomes RDF Class name
- Instead, design attribute set; no data source in node name
Putting Design into the Schema

- **Standard approach duplicates data (bad!):**
  - database attribute name becomes RDF predicate
  - database attribute name appears in target node (data source)
  - database attribute name becomes RDF Class name
- Instead, design attribute set; no data source in node name
Putting Design into the Schema

- Standard approach duplicates data (bad!):
  - database attribute name becomes RDF predicate
  - database attribute name appears in target node (data source)
  - database attribute name becomes RDF Class name
- Instead, design attribute set; no data source in node name
Putting Design into the Schema

- Standard approach duplicates data (bad!):
  - database attribute name becomes RDF predicate
  - database attribute name appears in target node (data source)
  - database attribute name becomes RDF Class name
- Instead, design attribute set; no data source in node name
Putting Design into the Schema

- Standard approach duplicates data (bad!):
  - database attribute name becomes RDF predicate
  - database attribute name appears in target node (data source)
  - database attribute name becomes RDF Class name
- Instead, design attribute set; no data source in node name
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
  - Coded values
  - Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Other Issues We Don’t Have Enough Time For!

- Hash fragments in URIs
  - big debate raging
  - what will your URIs serve?
- Should the conversion process be reversible?
  - full data source references in all URIs
  - is the relational database the master copy?
- Data bloat:
  - 3 triples per database value
  - short fields become long URIs
- Null fields
- Coded values
- Redundant nodes produced by foreign keys
- Data maintenance!
Useful websites

- RDFAndSQL Wiki: http://esw.w3.org/topic/RdfAndSql
- Tim Berners-Lee design note: http://www.w3.org/DesignIssues/RDB-RDF.html
- D2RQ (Chris Bizer): http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
- dbview.py (Dan Connolly): http://dig.csail.mit.edu/2006/dbview/dbview.py
- Linked Data: http://linkeddata.org/
Useful websites

- RDFAndSQL Wiki: [http://esw.w3.org/topic/RdfAndSql](http://esw.w3.org/topic/RdfAndSql)
Useful websites

- RDFAndSQL Wiki: http://esw.w3.org/topic/RdfAndSql
- Tim Berners-Lee design note: http://www.w3.org/DesignIssues/RDB-RDF.html
- D2RQ (Chris Bizer): http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
- dbview.py (Dan Connolly): http://dig.csail.mit.edu/2006/dbview/dbview.py
- Linked Data: http://linkeddata.org/
Useful websites

- RDFAndSQL Wiki: [http://esw.w3.org/topic/RdfAndSql](http://esw.w3.org/topic/RdfAndSql)
Useful websites

• RDFAndSQL Wiki:  http://esw.w3.org/topic/RdfAndSql
• Tim Berners-Lee design note:  
  http://www.w3.org/DesignIssues/RDB-RDF.html
• “Publishing databases on the Semantic Web”:  
  http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/publishing/
• D2RQ (Chris Bizer):  
  http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
• dbview.py (Dan Connolly):  
  http://dig.csail.mit.edu/2006/dbview/dbview.py
• Linked Data:  http://linkeddata.org/
Useful websites

- RDFAndSQL Wiki: http://esw.w3.org/topic/RdfAndSql
- Tim Berners-Lee design note:
  http://www.w3.org/DesignIssues/RDB-RDF.html
- “Publishing databases on the Semantic Web”:
  http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/publishing/
- D2RQ (Chris Bizer):
  http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
- dbview.py (Dan Connolly):
  http://dig.csail.mit.edu/2006/dbview/dbview.py
- Linked Data: http://linkeddata.org/