D-REPR: A Language For Describing And Mapping Diversely-Structured Data Sources To RDF

Binh Vu, Jay Pujara, and Craig Knoblock
Motivating example

No uniform method to access data

CSV

<table>
<thead>
<tr>
<th>Indicator Name</th>
<th>Units</th>
<th>2005</th>
<th>2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cargo Traffic (M.Ton Million tonne)</td>
<td>6915000.00</td>
<td>7100000.00</td>
<td></td>
</tr>
<tr>
<td>Cargo Traffic (M.Ton Million tonne)</td>
<td>204929.00</td>
<td>182810.00</td>
<td></td>
</tr>
<tr>
<td>Container Traffic (TEU)</td>
<td>5432353.3</td>
<td>5489586.00</td>
<td></td>
</tr>
<tr>
<td>Cargo Traffic, Annus Volume</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cargo Traffic, Annus %</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

JSON

```json
{
  "url": "https://example.com/obj/abc-123",
  "title": "Example Title",
  "sitters": [
    {
      "name": "Jane Doe",
      "born_died_date": "1 Jan 1990 - 31 Dec 2020"
    },
    {
      "name": "John Smith",
      "born_died_date": "20 Jan 2000 - 29 Dec 2029"
    }
  ]
}
```

NETCDF

Graph showing data dimensions and time series.
Motivating example

Need one method to access all types of data

Language for describing dataset
Heterogeneous datasets

- Multiple **formats**: CSV, JSON, XLSX, NetCDF4, ...

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

```
[  
  {  
    "indicator": "LIFE_0035",
    "age group": "< 1 year",
    "gender": "male",
    "year": "2016",
    "value": 57.7
  },  
  {  
    "indicator": "LIFE_0035",
    "age group": "< 1 year",
    "gender": "female",
    "year": "2016",
    "value": 59.6
  }
]
```

```
<obs>
  <ob>
    <indicator>LIFE_0035</indicator>
    <age_group>&lt;1 year</age_group>
    <gender>male</gender>
    <year>2016</year>
    <value>57.7</value>
  </ob>
  <ob>
    <indicator>LIFE_0035</indicator>
    <age_group>&lt;1 year</age_group>
    <gender>female</gender>
    <year>2016</year>
    <value>59.6</value>
  </ob>
</obs>
```
### Heterogeneous datasets

- **Same format, multiple layouts**

<table>
<thead>
<tr>
<th>Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Gender</th>
<th>Year</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt; 1 year</td>
<td>Male</td>
<td>2016</td>
<td>57.7</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>&lt; 1 year</td>
<td>Female</td>
<td>2016</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>Male</td>
<td>2016</td>
<td>60.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>Female</td>
<td>2016</td>
<td>62.1</td>
</tr>
</tbody>
</table>

| LIFE_0035 |

<table>
<thead>
<tr>
<th>LIFE_0035</th>
<th>Age Group</th>
<th>Gender</th>
<th>Observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt; 1 year</td>
<td>Male</td>
<td>2016</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>Female</td>
<td>2016</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Gender</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt; 1 year</td>
<td>Male</td>
<td>57.7</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>&lt; 1 year</td>
<td>Female</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>Male</td>
<td>60.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>Female</td>
<td>62.1</td>
</tr>
</tbody>
</table>
Related work

• Mapping nested relational datasets:
  – RML (Dimou et al, 2014), KR2RML (Slepicka et al 2015), xR2RML (Michel et al, 2015), etc.
  – Can handle multiple formats but only work for nested relational model layout

• Mapping tabular datasets:
  – XLWrap (Langegger et al, 2009), M2 (O’Connor et al, 2010), T2WML (Szekely et al, 2019)
  – Can handle multiple layouts, but support only tabular formats
Contributions

• A generic language to easily for describing and mapping heterogeneous datasets to RDF
  – It’s capable of mapping wide variety of data sources and goes beyond the set of sources that existing languages support.

• The language is extensible to new formats and layouts

• An efficient engine to convert datasets to RDF
Our approach

Step 1: define resources

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

Step 2: define attributes

Step 3: join attributes to tables

Step 4: semantic modeling

```json
{
  "indicator": "LIFE_0035",
  "url": "http://apps.who.int/.../indicator.aspx?iid=35"
},
{
  "indicator": "LIFE_0029",
  "url": "http://apps.who.int/.../indicator.aspx?iid=29"
}
```
Step 1: Resources

- A resource can be a physical file, SQL table, etc.

Syntax:

```
resources:
  <resource_id>:
    type: <resource_type>
  preprocessing:
  attributes:
  alignments:
  semantic_model:
```

Example:

```
resources:
  life_tbl:
    type: csv
  indicators:
    type: json
```

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

```
{
  "indicator": "LIFE_0035",
  "url": "http://apps.who.int/.../indicator.aspx?iid=35"
},
{
  "indicator": "LIFE_0029",
  "url": "http://apps.who/int/.../indicator.aspx?iid=29"
}
```
Step 2: Attributes

- Containing values that belong to a group
- Syntax

```plaintext
> resources: ...
> preprocessing: ...
  attributes:
    <attribute_id>:
      [resource_id]: <resource_id>
      path: <json_path>
      [unique]: false
      [missing_values]: [<value_0>, <value_1>, ...]
> alignments: ...
> semantic_model: ...
```

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

```plaintext
attributes:
  year:
    resource_id: life_tbl
    path: [$[0][2:]
  gender:
    resource_id: life_tbl
    path: [$[1][2:]
  indicator_col:
    resource_id: life_tbl
    path: [$[2:][0]
  age_group:
    resource_id: life_tbl
    path: [$[2:][1]
  observation:
    resource_id: life_tbl
    path: [$[2:][2:]
> indicator: ...
> url: ...
```
Step 2: Attributes

indicators.json

```json
{
    "indicator": "LIFE_0035",
    "url": "http://apps.who/int/.../indicator.aspx?iid=35"
},
{
    "indicator": "LIFE_0029",
    "url": "http://apps.who/int/.../indicator.aspx?iid=29"
}
```

attributes:
- year: ...
- gender: ...
- indicator_col: ...
- age_group: ...
- observation: ...

indicator:
- resource_id: indicators
- path: $[::].indicator
- unique: true

url:
- resource_id: indicators
- path: $[::].url
- unique: true
Step 3: Alignments

- Explicitly specifying the layout through alignments

- For linking across resources

<table>
<thead>
<tr>
<th>Indicator</th>
<th>male</th>
<th>female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gender</th>
<th>observation</th>
<th>indicator_col</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>57.7</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>female</td>
<td>59.6</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>male</td>
<td>60.6</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>female</td>
<td>62.1</td>
<td>LIFE_0035</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>gender</th>
<th>observation</th>
<th>indicator_col</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>57.7</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>female</td>
<td>59.6</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>male</td>
<td>60.6</td>
<td>LIFE_0035</td>
</tr>
<tr>
<td>female</td>
<td>62.1</td>
<td>LIFE_0035</td>
</tr>
</tbody>
</table>

```
life_tbl.csv
```

```
http://...?iid=35
http://...?iid=29
```
Step 3: Alignments

• Join by value (equi-join)

```json
{
  "indicator": "LIFE_0035",
  "url": "http://apps.who/int/.../indicator.aspx?iid=35"
},
{
  "indicator": "LIFE_0029",
  "url": "http://apps.who/int/.../indicator.aspx?iid=29"
},
```

• Syntax

```
  resources: ...
  preprocessing: ...
  attributes: ...
    alignments:
      - type: <join_type>
        source: <attribute_id>
        target: <attribute_id>
        # ..optional arguments depends on the alignment type..
  semantic_model: ...
```

<table>
<thead>
<tr>
<th>Indicator</th>
<th>Age Group</th>
<th>2016 Male</th>
<th>2016 Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

Step 3: Alignments

- Join by positions in the dataset

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>2016</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Indicator</td>
<td>Age Group</td>
<td>male</td>
<td>female</td>
</tr>
<tr>
<td>2</td>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>3</td>
<td>LIFE_0035</td>
<td>1–4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

**alignments:**
- **type:** value...
- **type:** dimension
  - **source:** observation
  - **target:** gender
- **aligned_dims:**
  - **source:** 1
  - **target:** 1

**dimension 0 (row)**
**dimension 1 (column)**
Step 3: Alignments

- Join by positions in the dataset

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>2016</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Indicator</td>
<td>Age Group</td>
<td>male</td>
<td>female</td>
</tr>
<tr>
<td>2</td>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
<td>59.6</td>
</tr>
<tr>
<td>3</td>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
<td>62.1</td>
</tr>
</tbody>
</table>

alignments:

- type: value
  - source: indicator_col
  - target: observation
- type: dimension
  - source: observation
  - target: indicator_col

dimension 0 (row)  dimension 1 (column)
Step 3: Alignments

• Join by positions in the dataset

Sample data

```json
[{  
  "departments": {  
    "people": [{  
      "name": "Peter",  
      "phone": "213-266-2777"
    },  
    {  
      "name": "John",  
      "phone": "222-222-2222"
    }  
   }  
  }  
}
```

Aligned in dimensions 0 and 3

- Name: `${departments.people.*.name}`
- Phone: `${departments.people.*.phone}`

Dimensions:

- Dimension 0
- Dimension 1
- Dimension 2
- Dimension 3
- Dimension 4
Step 3: Alignments

• Easy to incorporate new alignment function

• Users only need to define the minimum number of joins (N-1) because the engine can infer the rest via composition.
Step 4: Semantic Model

- Using ontologies to describe your data (classes and predicates)

<table>
<thead>
<tr>
<th>gender</th>
<th>observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>57.7</td>
</tr>
<tr>
<td>female</td>
<td>59.6</td>
</tr>
<tr>
<td>male</td>
<td>60.6</td>
</tr>
<tr>
<td>female</td>
<td>62.1</td>
</tr>
</tbody>
</table>

Syntax:

```
semantic_model:
  data_nodes:
    observation: qb:Observation:1--sdmx-m:obsValue
    year: qb:Observation:1--sdmx-d:refPeriod
    indicator: eg:Indicator:1--eg:code
    url: eg:Indicator:1--drepr:uri
  relations:
    - qb:Observation:1--eg:indicator--eg:Indicator:1
```
Step 4: Semantic Model

• Users can create arbitrary semantic model, even when using attributes across multiple resources
Data cleaning (optional)

- Users can write python function to clean or transform the data

```python
preprocessing:
  - type: pmap
    input:
      resource: life_tbl
      path: $[0][2:]
    code: |
      if value == ":
        return context.get_left_value(index)
      return value
```

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>2016</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>2016</td>
</tr>
<tr>
<td>1</td>
<td>Indicator</td>
<td>Age Group</td>
<td>male</td>
</tr>
<tr>
<td>2</td>
<td>LIFE_0035</td>
<td>&lt;1 year</td>
<td>57.7</td>
</tr>
<tr>
<td>3</td>
<td>LIFE_0035</td>
<td>1-4 years</td>
<td>60.6</td>
</tr>
</tbody>
</table>

- Can re-use functions or existing libraries
Evaluation

• Coverage of D-REPR
  – Randomly sampling 700 datasets from data.gov
  – Modeling datasets of different formats and layouts

a. Children and Family Health

```json
{
  "columns": [
    {"name": "teenbir10", "description": "Teen Birth Rate ... (2010)", "..."},
    {"name": "teenbir11", "description": "Teen Birth Rate ... (2011)", "..."},
    ...
  ],
  "data": [
    ["...", "Allendale/Irvington/S. Hilton", "55.0", "58.1", "..."],
    ["...", "Beechfield/Ten Hills/West Hills", "42.8", "21.4", "..."],
    ...
  ]
}
```

b. Sugar production by sugar beet and sugarcane processors

```
<table>
<thead>
<tr>
<th></th>
<th>JAN</th>
<th>FEB</th>
<th>MAR</th>
<th>APR</th>
<th>MAY</th>
<th>JUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>From domestic sugar beets</td>
<td>661,586</td>
<td>485,126</td>
<td>423,775</td>
<td>337,473</td>
<td>216,526</td>
<td>82,987</td>
</tr>
<tr>
<td>From imported sugar beets</td>
<td>0</td>
<td>37,160</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Subtotal</td>
<td>661,586</td>
<td>522,287</td>
<td>423,775</td>
<td>337,473</td>
<td>216,526</td>
<td>82,987</td>
</tr>
<tr>
<td>Cane production:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Florida</td>
<td>321,414</td>
<td>253,438</td>
<td>242,560</td>
<td>92,302</td>
<td>47,237</td>
<td>0</td>
</tr>
<tr>
<td>Subtotal</td>
<td>378,919</td>
<td>283,190</td>
<td>289,237</td>
<td>108,826</td>
<td>68,504</td>
<td>30,903</td>
</tr>
<tr>
<td>Total</td>
<td>1,040,505</td>
<td>805,476</td>
<td>713,012</td>
<td>446,298</td>
<td>285,030</td>
<td>113,889</td>
</tr>
</tbody>
</table>
```

Cannot be modeled with Nested Relational Models!
Evaluation

- Runtime of D-REPR engine (ms)
  - Mapping large CSV files (row-based table) containing (name, phone, address)
  - Generating 1.3m triples / second (15 times faster than KR2RML)

<table>
<thead>
<tr>
<th>Tools</th>
<th>5,000</th>
<th>10,000</th>
<th>20,000</th>
<th>40,000</th>
<th>80,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-REPR</td>
<td>33.44</td>
<td>69.84</td>
<td>132.00</td>
<td>267.50</td>
<td>551.24</td>
</tr>
<tr>
<td>KR2RML</td>
<td>1368.00</td>
<td>1776.33</td>
<td>3276.66</td>
<td>4990.33</td>
<td>8305.33</td>
</tr>
<tr>
<td>Morph</td>
<td>4812.00</td>
<td>14949.66</td>
<td>65961.33</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Discussion and Future work

• A novel **generic** data representation language: D-REPR
  – Uses a declarative approach
  – Works for heterogeneous datasets of different formats and layouts

• Open source: [https://github.com/usc-is-i2/d-repr](https://github.com/usc-is-i2/d-repr)

• Future work:
  – (Semi-)automatically generating D-REPR models
  – UI for annotating datasets
  – Improving efficiency of D-REPR’s engine by doing parallel processing
References


• [3] xR2RML: Franck Michel, Loïc Djimou, Catherine Faron Zucker, and Johan Montagnat. 2015. Translation of relational and non-relational databases into RDF with xR2RML
