SCULPT
A Schema Language for Tabular Data on the Web

Wim Martens  Frank Neven  Stijn Vansummeren
Bayreuth      Hasselt      Brussels

WWW 2015 Florence

this one is me
Tabular Data is...
Temperature readings from weather stations in Africa
Tabular Data is...

Temperature readings from weather stations in Africa

subject predicate object provenance
:e4 type PER :e4
:e4 mention "Bart" D00124 283-286
:e4 mention "Jojo" D00124 145-149 0.9
:e4 per:siblings :e7 D00124 283-286 173-179 274-281
:e4 per:age "10" D00124 180-181 173-179 182-191 0.9

US National Institute of Standards and Technology (NIST), Cold Start Knowledge Base Population Task

Examples taken from the W3C CSV on the Web WG use cases
Tabular Data is...
... text data that is structured in rows and columns

Temperature readings from weather stations in Africa

,ENTEBBE AIR, FT PORTAL, GONDOKORO, GULU, HOIMA, JINJA, KABALE, MASAKA, MASINDI, MBALE, MBARARA, MOROTO
1905.04, 25.60, 24.17, 34.67,-99.00,-99.00, 29.06,-99.00, 24.50,-99.00,-99.00, 26.50,-99.00,-99.00
1905.13, 27.30, 25.61, 36.50,-99.00,-99.00, 31.44,-99.00, 26.67,-99.00,-99.00, 27.00,-99.00,-99.00
1905.21, 24.90, 25.44, 37.39,-99.00,-99.00, 28.78,-99.00, 25.06,-99.00,-99.00, 27.17,-99.00,-99.00
1905.29, 25.10, 25.56, 35.28,-99.00,-99.00, 28.72,-99.00, 24.33,-99.00,-99.00, 27.67,-99.00,-99.00
1905.38, 24.30,-99.00, 33.06,-99.00,-99.00, 28.78,-99.00, 25.08,-99.00,-99.00, 27.54,-99.00,-99.00

Examples taken from the W3C CSV on the Web WG use cases
Tabular Data

A lot of data on the Web is tabular
Tabular Data

A lot of data on the Web is **tabular**

spreadsheets
Tabular Data

A lot of data on the Web is **tabular**

- spreadsheets
- comma-separated-value files (CSV)
Tabular Data

A lot of data on the Web is tabular spreadsheets comma-separated-value files (CSV) HTML tables
Tabular Data

A lot of data on the Web is **tabular**

- spreadsheets
- comma-separated-value files (CSV)
- HTML tables
- ...

Tabular Data

A lot of data on the Web is tabular

- spreadsheets
- comma-separated-value files (CSV)
- HTML tables
- ...

“Over 90% of open data is tabular”

-Jeni Tennison
(Open Data Institute and W3C CSV on the Web WG)
But...
But...

...tabular data / CSV has many irregularities
But...

...tabular data / CSV has many irregularities because there is no standard
“2/3 of 'CSV' files on data.gov.uk are not machine-readable [in an elegant way]”

-Jeni Tennison
(Open Data Institute and W3C CSV on the Web WG)
But...

...tabular data / CSV has many irregularities because there is no standard

“2/3 of 'CSV' files on data.gov.uk are not machine-readable [in an elegant way]”

-Jeni Tennison

(Open Data Institute and W3C CSV on the Web WG)

This is why the W3C is working on a standard for CSV
In the end, you'd like to have a simple way to:

describe

and

manipulate

tabular data
What do I mean by that?
What do I mean by that?
What do I mean by that?
What do I mean by that?
What do I mean by that?

You want a tool to describe the logical structure of the cells

So that you can
• define a schema
• select, transform, export cell values

You need a schema language, a.k.a. meta data format
W3C Meta Data, current state

The W3C is currently working on a metadata format for tabular data
The W3C is currently working on a metadata format for tabular data. It allows to specify the format of cells, but its capabilities are limited.
The W3C is currently working on a metadata format for tabular data. It allows to specify the format of cells, but its capabilities are limited. On the structural level, it can basically say:

- cell x contains the XSD data type [...]  
- the column of cell x contains the XSD data type [...]

W3C Meta Data, current state
W3C Meta Data, current state

The W3C is currently working on a metadata format for tabular data

It allows to specify the format of cells, but its capabilities are limited

On the structural level, it can basically say:

- cell x contains the XSD data type [...]
- the column of cell x contains the XSD data type [...]

1905.04, 25.60, 24.17, 34.67, -99.00, [...]  
1905.13, 27.30, 25.61, 36.50, -99.00, [...]  
1905.21, 24.90, 25.44, 37.39, -99.00, [...]  
1905.29, 25.10, 25.56, 35.28, -99.00, [...]  
1905.38, 24.30, -99.00, 33.06, -99.00, [...]  
1905.46, 25.30, -99.00, 32.44, -99.00, [...]  

(W3C Use Cases and Requirements, Use Case 3)
One Data Type Per Column?

(W3C Use Cases and Requirements, Use Case 13)

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124</td>
</tr>
</tbody>
</table>
One Data Type Per Column?
(W3C Use Cases and Requirements, Use Case 13)

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:parent</td>
<td>:e9</td>
<td>D00124</td>
</tr>
</tbody>
</table>

Here, we see different data types per column:
- document IDs
- positions in a document
- certainty values
You want to reason about these different types in a column.
What can research do to help?
SCULPT
SChema for Un-Locking and Processing Tabular data

“Every block of stone has a statue inside it and it is the task of the sculptor to discover it”

-Michelangelo

SCULPT is about describing the statue

(Image: Caricato da Sailko, public domain, wikipedia)
"Every block of data has structure inside it and it is the task of the schema-designer to describe it"

-Michelangelo
(what he probably meant)

The schema-designer is the Michelangelo of the data

SCULPT is about describing the statue
SCULPT

A simple language

Serves as a schema language for tabular data

Serves as a basis for node selection, for a transformation language

Built on solid foundations

Hopefully, serves as a source of inspiration
Describe the structure of tabular data in three stages:

1. It describes what the cells are  
   (trivial)

2. It describes the content of **single cells**  
   (easy)

3. It describes the **relationship between cells**  
   (main machinery)
SCULPT, general principle

Describe the structure of tabular data in three stages:

1. It describes what the cells are
   by specifying row / column delimiter

2. It describes the content of single cells
   each cell's content is matched against a regex,
   which we'll interpret as a "data type"

(Alternatively, one could also use
XML Schema single types)
Describe the structure of tabular data in three stages:

After the 2nd stage, the each cell of the table has a **set of datatypes**

<table>
<thead>
<tr>
<th>Timestamp,</th>
<th>Temperature,</th>
<th>Temperature,</th>
<th>Temperature,</th>
<th>Dummy,</th>
</tr>
</thead>
<tbody>
<tr>
<td>1905.04,</td>
<td>25.60,</td>
<td>24.17,</td>
<td>34.67,</td>
<td>-99.00,</td>
</tr>
<tr>
<td>1905.13,</td>
<td>27.30,</td>
<td>25.61,</td>
<td>36.50,</td>
<td>-99.00,</td>
</tr>
<tr>
<td>1905.21,</td>
<td>24.90,</td>
<td>25.44,</td>
<td>37.39,</td>
<td>-99.00,</td>
</tr>
<tr>
<td>1905.29,</td>
<td>25.10,</td>
<td>25.56,</td>
<td>35.28,</td>
<td>-99.00,</td>
</tr>
<tr>
<td>1905.38,</td>
<td>24.30,</td>
<td>-99.00,</td>
<td>33.06,</td>
<td>-99.00,</td>
</tr>
<tr>
<td>[...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**SCULPT, general principle**

We describe the structure of tabular data in three stages:

3. It describes the relationship between cells

This is the core of SCULPT

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>ENTEBBE AIR</th>
<th>FT PORTAL</th>
<th>GONDOKORO</th>
<th>GULU</th>
<th>[...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Dummy</td>
<td>[...]</td>
</tr>
<tr>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Dummy</td>
<td>[...]</td>
</tr>
<tr>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Temperature</td>
<td>Dummy</td>
<td>[...]</td>
</tr>
<tr>
<td>Temperature</td>
<td>Temperature</td>
<td>Dummy</td>
<td>Temperature</td>
<td>Dummy</td>
<td>[...]</td>
</tr>
</tbody>
</table>
SCULPT, general principle

We describe the structure of tabular data in three stages:

3. It describes the relationship between cells

This is the core of SCULPT

| Timestamp, Temperature, Temperature, Temperature, Temperature, Dummy, [...] |
| Timestamp, Temperature, Temperature, Temperature, Temperature, Dummy, [...] |
| Timestamp, Temperature, Temperature, Temperature, Temperature, Dummy, [...] |
| Timestamp, Temperature, Temperature, Temperature, Temperature, Dummy, [...] |
| Timestamp, Temperature, Dummy, Temperature, Dummy, [...] |

row(1) -> Empty, ENTEBBE AIR, FT PORTAL, GONDOKORO, GULU, [...] 

col(ENTEBBE AIR) -> Temperature 

col(FT PORTAL) -> Temperature | Dummy
SCULPT, general principle

We describe the structure of tabular data in three stages:

3. It describes the relationship between cells

General idea:
the schema has a set of rules

\[ \text{<selection expression>} \rightarrow \text{<content expression>} \]

selects a region
(set of cells)
in the table
describes how the region should look like

This is the core of SCULPT
SCULPT, general principle

We describe the structure of tabular data in three stages:

3. It describes the relationship between cells

General idea:
the schema has a set of rules

\(<\text{selection expression}\> \rightarrow \ <\text{content expression}\>

selects a region
(set of cells)
in the table

describes how the region should look like

If the data satisfies these rules, it is valid / well-formed

This is the core of SCULPT
Token Structure (1/3)

<table>
<thead>
<tr>
<th>selection expression</th>
<th>-&gt;</th>
<th>content expression</th>
</tr>
</thead>
</table>
<selection expression>  ->  <content expression>

region is matched
row by row
Token Structure (3/3)

<selection expression> => <content expression>

entire region is matched

(, *, )*
### Data:

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124</td>
</tr>
</tbody>
</table>

### Schema:

% Simple datatypes

```
<table>
<thead>
<tr>
<th>word</th>
<th>entity-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf-uri</td>
<td>rdf-lit</td>
</tr>
<tr>
<td>doc-ID</td>
<td>position</td>
</tr>
<tr>
<td></td>
<td>certainty</td>
</tr>
</tbody>
</table>
```

% Rules
```
row(1) -> subject, predicate, object, provenance
col(subject) -> rdf-uri
col(predicate) -> word | rdf-uri
col(object) -> rdf-lit | rdf-uri | entity-type

down+(right*(provenance)) -> (doc-ID, position*, certainty?)*```
How are Rules Defined?

In the rules

\[
\text{<selection expression> } \rightarrow \text{ <content expression>}
\]

\[
\text{<selection expression> } \Rightarrow \text{ <content expression>}
\]

the **selection expression** is based on core **XPath**

(it selects nodes)

**XPath** is powerful, expressive,
and has linear time evaluation
How are Rules Defined?

In the rules

\[
\text{<selection expression>} \rightarrow \text{<content expression>}
\]

\[
\text{<selection expression>} \Rightarrow \text{<content expression>}
\]

the selection expression is based on core **XPath** (it selects nodes)

the content expression is just

a regular expression using datatypes

**XPath** is powerful, expressive,
and has linear time evaluation
SCULPT is Simple, Powerful, and Efficient

Theorem:
Given a tabular document $D$ and a Sculpt schema $S$, we can test in linear-time* if $D$ satisfies $S$

*combined complexity
SCULPT is Simple, Powerful, and Efficient

Theorem:
Given a tabular document $D$ and a Sculpt schema $S$, we can test in linear-time* if $D$ satisfies $S$

*combined complexity

Theorem:
Streaming validation works too

(precise statement can be found in the paper)
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:parent</td>
<td>:e9</td>
<td>D00124</td>
</tr>
</tbody>
</table>

If you can identify and select regions, transforming them becomes easy.
SCULPT is Simple, Powerful, and Efficient
and is a basis for a transformation language

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124</td>
</tr>
<tr>
<td>:e4</td>
<td>per:parent</td>
<td>:e9</td>
<td>D00124</td>
</tr>
</tbody>
</table>

If you can identify and select regions, transforming them becomes easy

, ENTEBBE AIR, FT PORTAL, GONDOKORO, GULU, [...]  
1905.04, 25.60, 24.17, 34.67, -99.00, [...]  
1905.13, 27.30, 25.61, 36.50, -99.00, [...]  
1905.21, 24.90, 25.44, 37.39, -99.00, [...]  
1905.29, 25.10, 25.56, 35.28, -99.00, [...]  
1905.38, 24.30, -99.00, 33.06, -99.00, [...]  
[...]  

SITE[down*::Temperature]  
SITE = [A-Z]*
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

This is not about
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

This is not about a bunch of crazy researchers.
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language.

This is not about a bunch of crazy researchers trying to get some weird feature in some standard.
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

This is not about a bunch of crazy researchers trying to get some weird feature in some standard

Actually,
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

This is not about a bunch of crazy researchers trying to get some weird feature in some standard

Actually, we're seeing a challenge in the language
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language

This is **not** about a bunch of crazy researchers trying to get some weird feature in some standard

Actually, we're seeing a challenge in the language and we're seeing how it can be addressed
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language we hope that the W3C is listening and can take some inspiration from us
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language. We hope that the W3C is listening and can take some inspiration from us to make their meta-data format.
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language. We hope that the W3C is listening and can take some inspiration from us to make their meta-data format more expressive.
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language. We hope that the W3C is listening and can take some inspiration from us to make their meta-data format more expressive, better capable of dealing with their use cases,
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language. We hope that the W3C is listening and can take some inspiration from us to make their meta-data format more expressive, better capable of dealing with their use cases, without a cost in complexity.
SCULPT is Simple, Powerful, and Efficient and is a basis for a transformation language.

We hope that the W3C is listening and can take some inspiration from us to make their meta-data format more expressive, better capable of dealing with their use cases, without a cost in complexity.

Thank you!
Backup
### SCULPT by Example

**Data:**

<table>
<thead>
<tr>
<th>subject</th>
<th>predicate</th>
<th>object</th>
<th>provenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>:e4</td>
<td>type</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Bart&quot;</td>
<td>D00124 283-286</td>
</tr>
<tr>
<td>:e4</td>
<td>mention</td>
<td>&quot;Jojo&quot;</td>
<td>D00124 145-149 0.9</td>
</tr>
<tr>
<td>:e4</td>
<td>per:siblings</td>
<td>:e7</td>
<td>D00124 283-286 173-179 274-281</td>
</tr>
<tr>
<td>:e4</td>
<td>per:age</td>
<td>&quot;10&quot;</td>
<td>D00124 180-181 173-179 182-191 0.9</td>
</tr>
</tbody>
</table>

The complete schema:

- **Col delim:** \t
- **Row delim:** \n
% Tokens / data types

- rdf-uri = [a-zA-Z0-9]*:[a-zA-Z0-9]*
- rdf-lit = "[a-zA-Z0-9]*"
- doc-ID = D[0-9]{5}
- position = [0-9]{3}-[0-9]{3}
- certainty = [0-9].[0-9]
- word = [a-z]*
- entity-type = PER | ORG | GPE

% Rules

- row(1) -> subject, predicate, object, provenance
- col(subject) -> rdf-uri
- col(predicate) -> word | rdf-uri
- col(object) -> rdf-lit | rdf-uri | entity-type
- down+(right*(provenance))
  -> (doc-ID, position*, certainty?)*