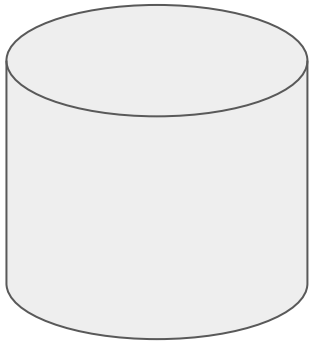
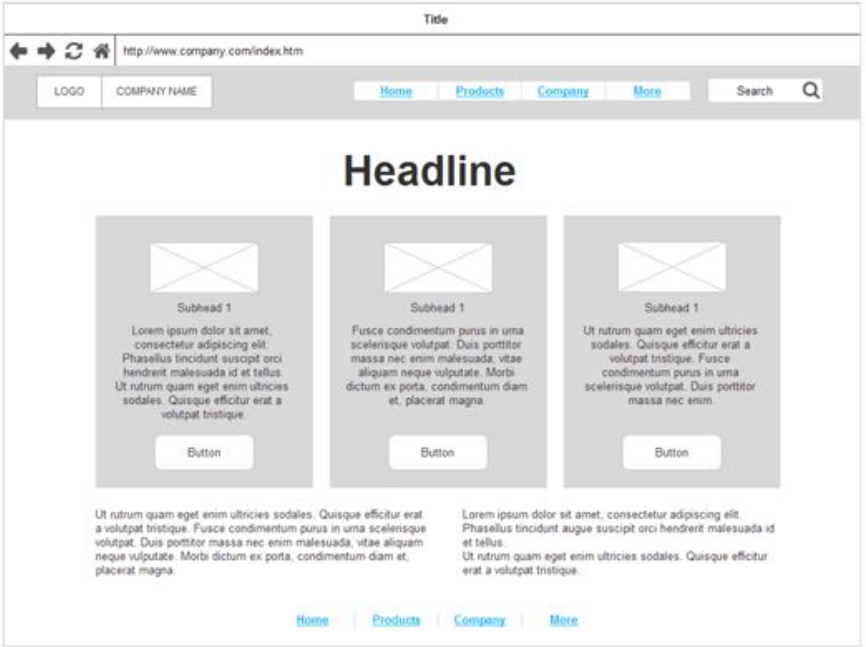
Computational pipelines

NEH Institute 2022

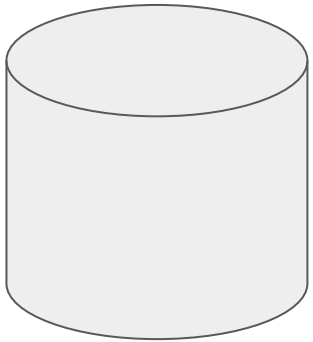
From XML to digital edition



XML database

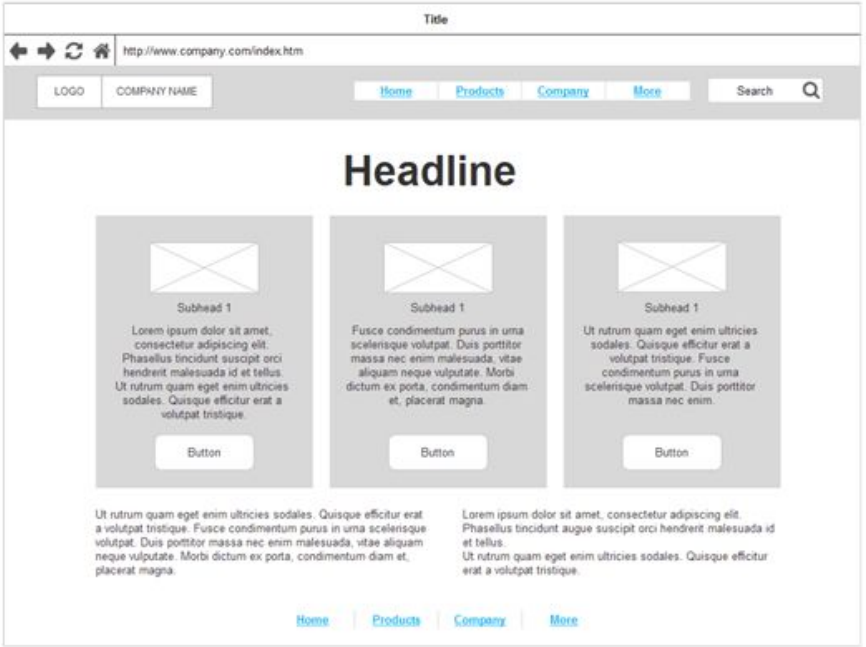


From XML to digital edition

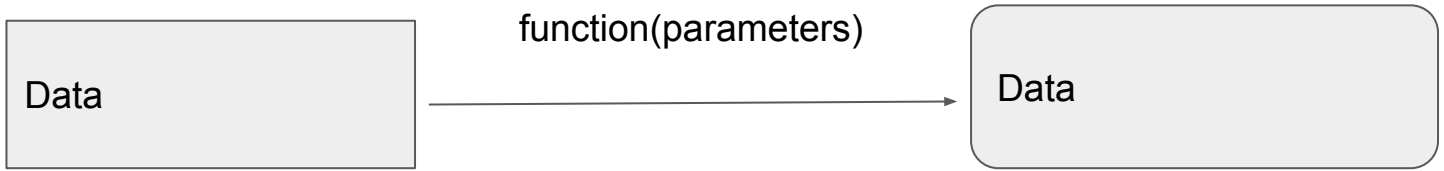


XML database

Computational pipeline



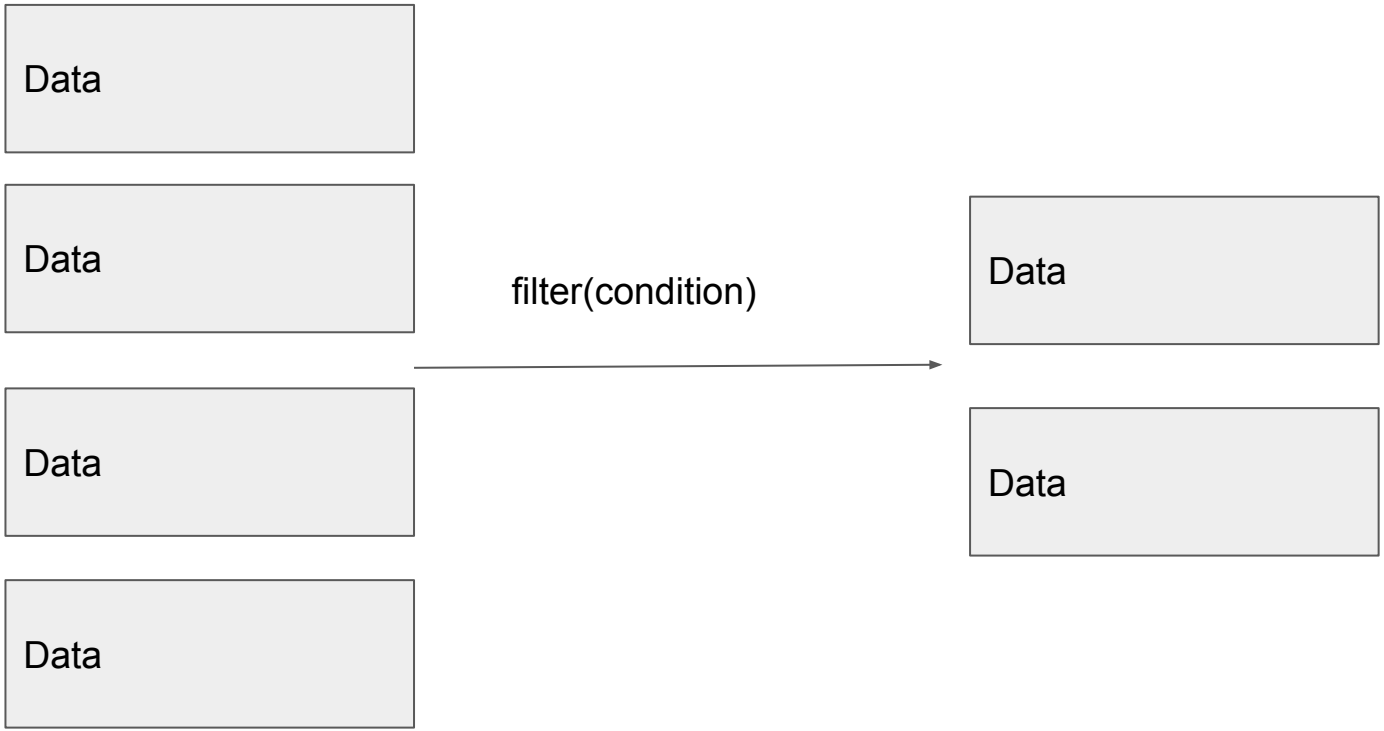
Function



Different types of functions

- Filtering
- Sorting
- Mapping
- Reduction
- Gather
- Scatter

Function: filtering



Example: filter

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

Example: filter (II)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Students with names starting with ‘M’.

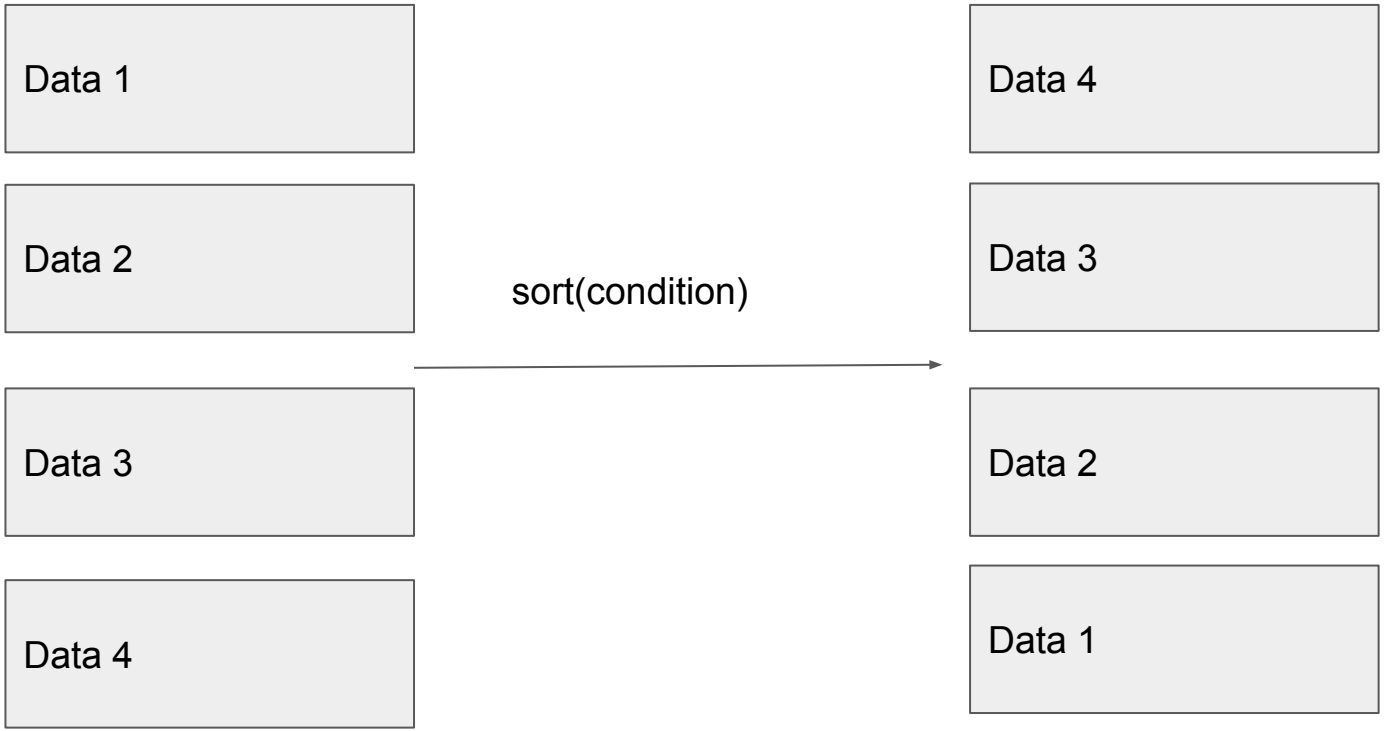
Example: filter (III)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Students with names starting with ‘M’.

```
<students>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Function: sorting



Example: sorting

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

Example: sorting (II)

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

Sort by name

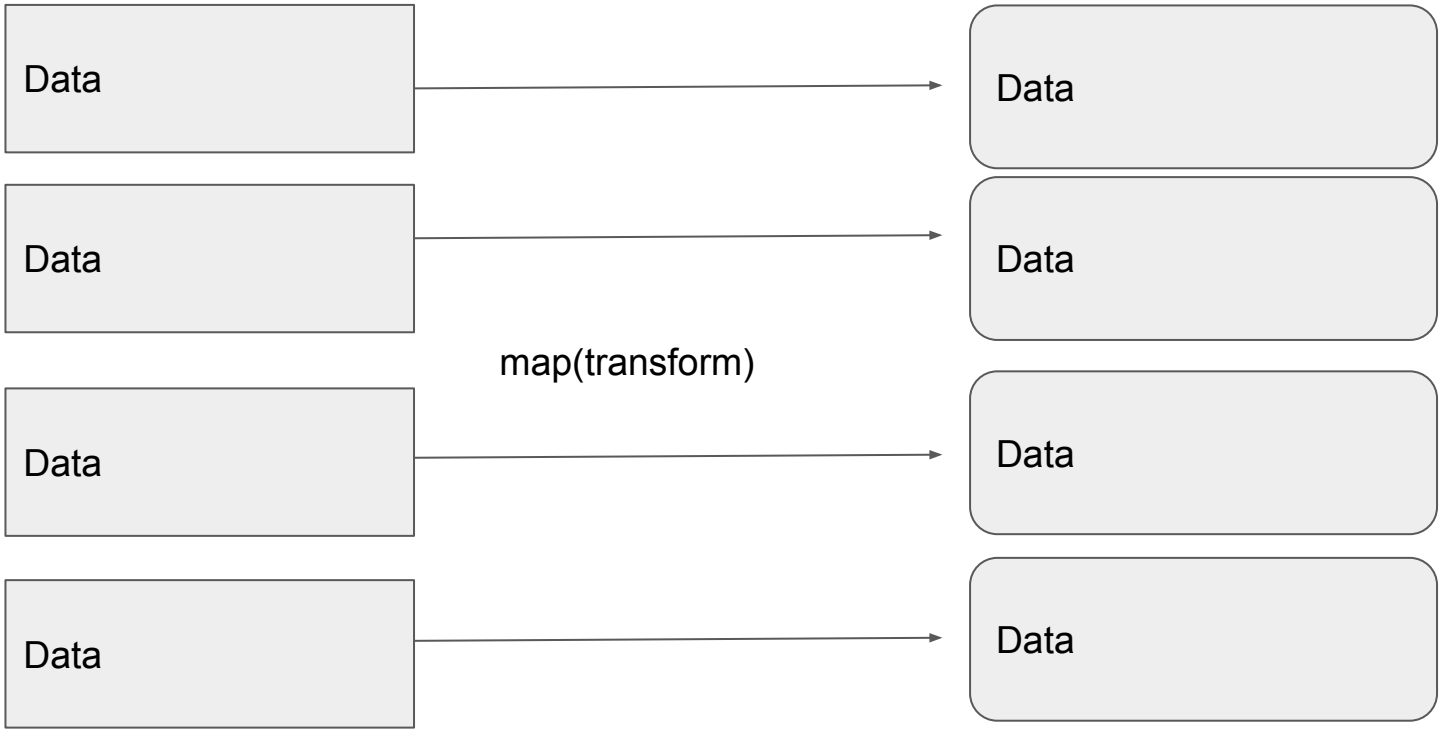
Example: sorting (III)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Sort by name

```
<students>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
</students>
```

Function: mapping



Example: mapping names to uppercase (I)

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

Example: mapping names to uppercase (II)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Take the name of each student and convert it to uppercase

Example: mapping names to uppercase (III)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Take the name of each student and convert it to uppercase

```
<students>
 <student>
  <name>RICK GRIMES</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>DARYL DIXON</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>MAGGIE</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Example: visualisation; map to HTML

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

Example: visualisation; map to HTML (II)

```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Convert each student element to a row with each field as a cell

Example: visualisation; map to HTML (III)

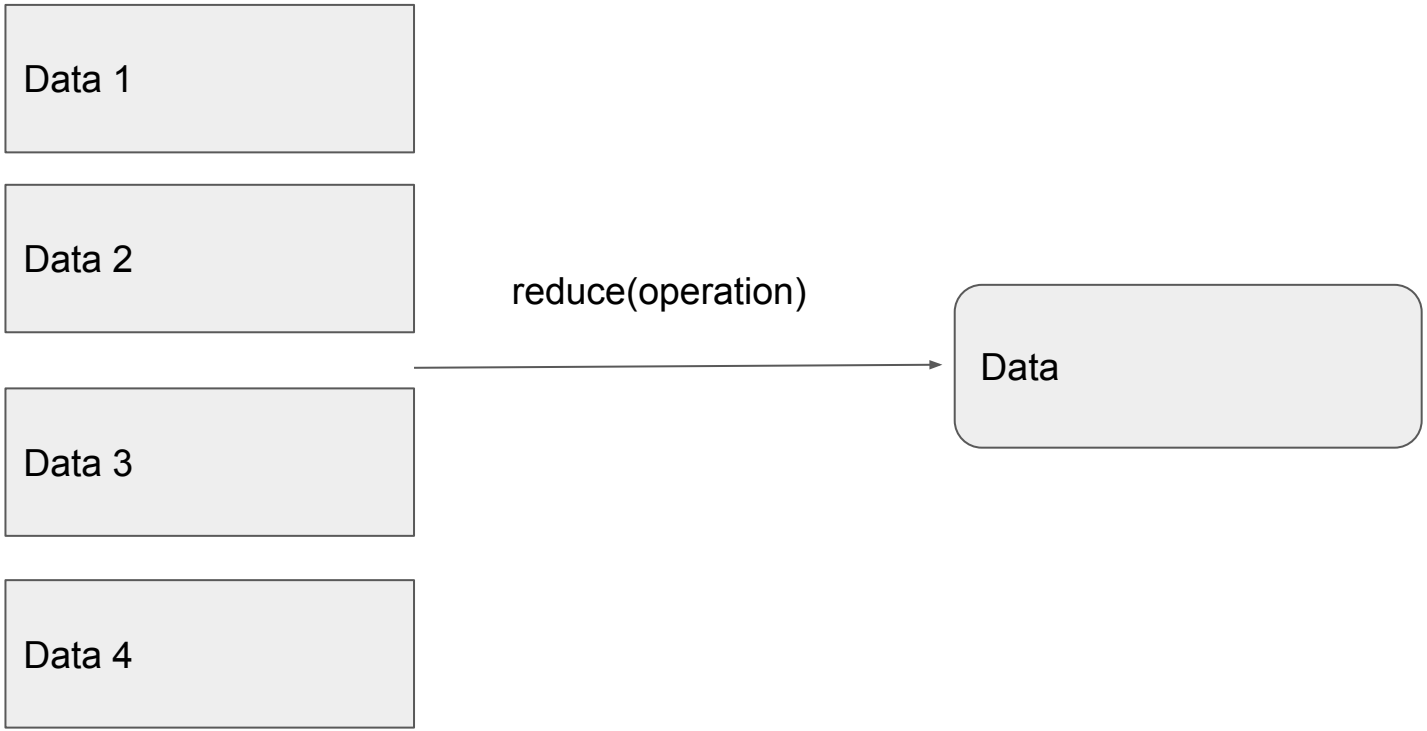
```
<students>
 <student>
  <name>Rick Grimes</name>
  <age>35</age>
  <subject>Maths</subject>
 </student>
 <student>
  <name>Daryl Dixon</name>
  <age>33</age>
  <subject>Science</subject>
 </student>
 <student>
  <name>Maggie</name>
  <age>36</age>
  <subject>Arts</subject>
 </student>
</students>
```

Convert each student element to a row with each field as a cell

```
<table>
<tr>
  <th>Name</th>
  <th>Age</th>
  <th>Subject</th>
</tr>
<tr>
  <td>Rick Grimes</td>
  <td>35</td>
  <td>Maths</td>
</tr>
<tr>
  <td>Daryl Dixon</td>
  <td>33</td>
  <td>Science</td>
</tr>
<tr>
  <td>Maggie</td>
  <td>36</td>
  <td>Arts</td>
</tr>
</table>
```

Name	Age	Subject
Rick Grimes	35	Maths
Daryl Dixon	33	Science
Maggie	36	Arts

Function: reduction



Example: reduction; summation (I)

<expenses>
<item>36</item>
<item>50</item>
<item>7</item>
</expenses>

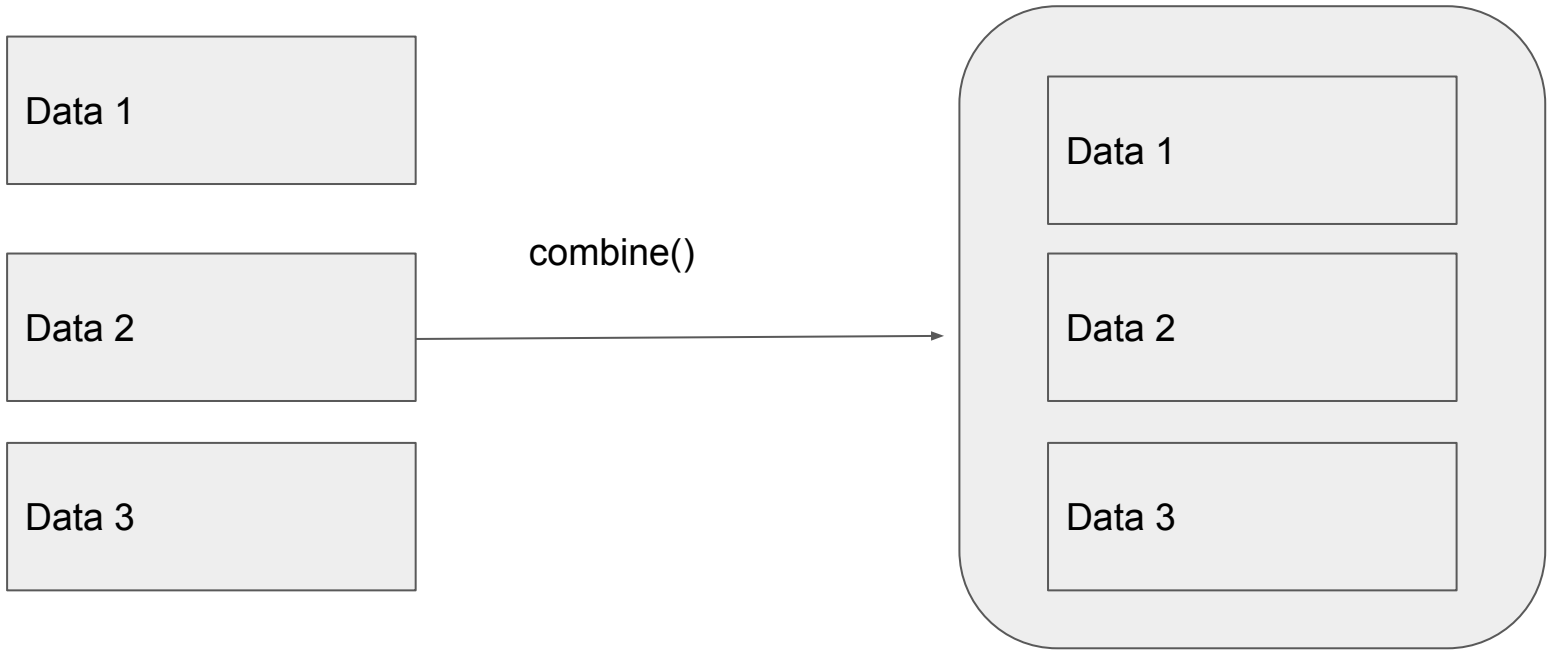
Example: reduction; summation (II)

```
<expenses>
<item>36</item>
<item>50</item>
<item>7</item>
</expenses>
```

Add up the costs of all the items keeping a total

```
36
50
7
---
93
```

Function: gather



Example: Gather; combining address and coordinates (I)

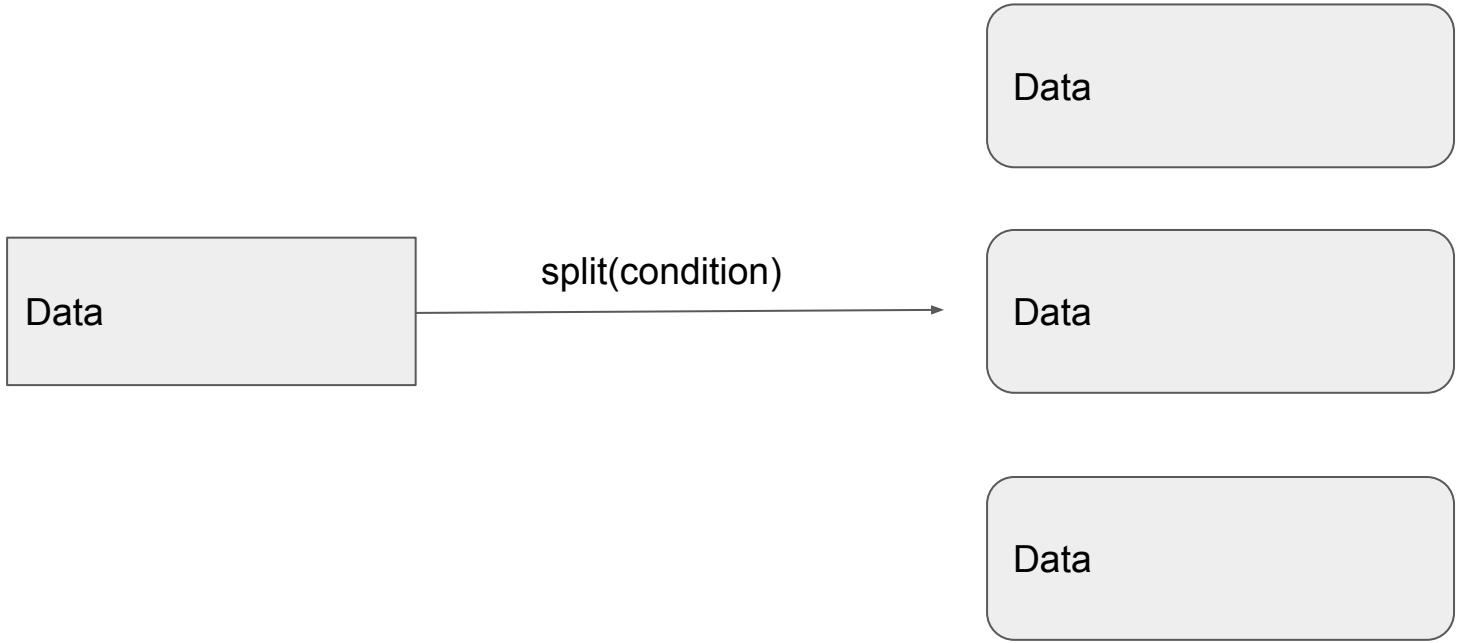
```
<students>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <city>New York</city>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <city>Amsterdam</city>
 </student>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <city>Pittsburgh</city>
 </student>
</students>
```

```
<gazetteer>
<cities>
  <city name="New York City" coordinates="40°42′46″N 74°
00′22″W"/>
  <city name="Amsterdam" coordinates="52°22′N 4°54′E"/>
  <city name="Pittsburgh" coordinates="40°26′23″N 79°58′35″W">
</cities>
</gazetteer>
```

Example: Gather; combining address and coordinates(II)

```
<students>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <city coordinates="40°42′46″N 74°00′22″W">New York</city>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <city coordinates="52°22′N 4°54′E">Amsterdam</city>
 </student>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <city coordinates="40°26′23″N 79°58′35″W">Pittsburgh</city>
 </student>
</students>
```

Function: scatter



Example: scatter; splitting a sentence into tokens (I)

<sentence>The red fox jumped over the lazy brown dog</sentence>

Example: scatter; splitting a sentence into tokens (II)

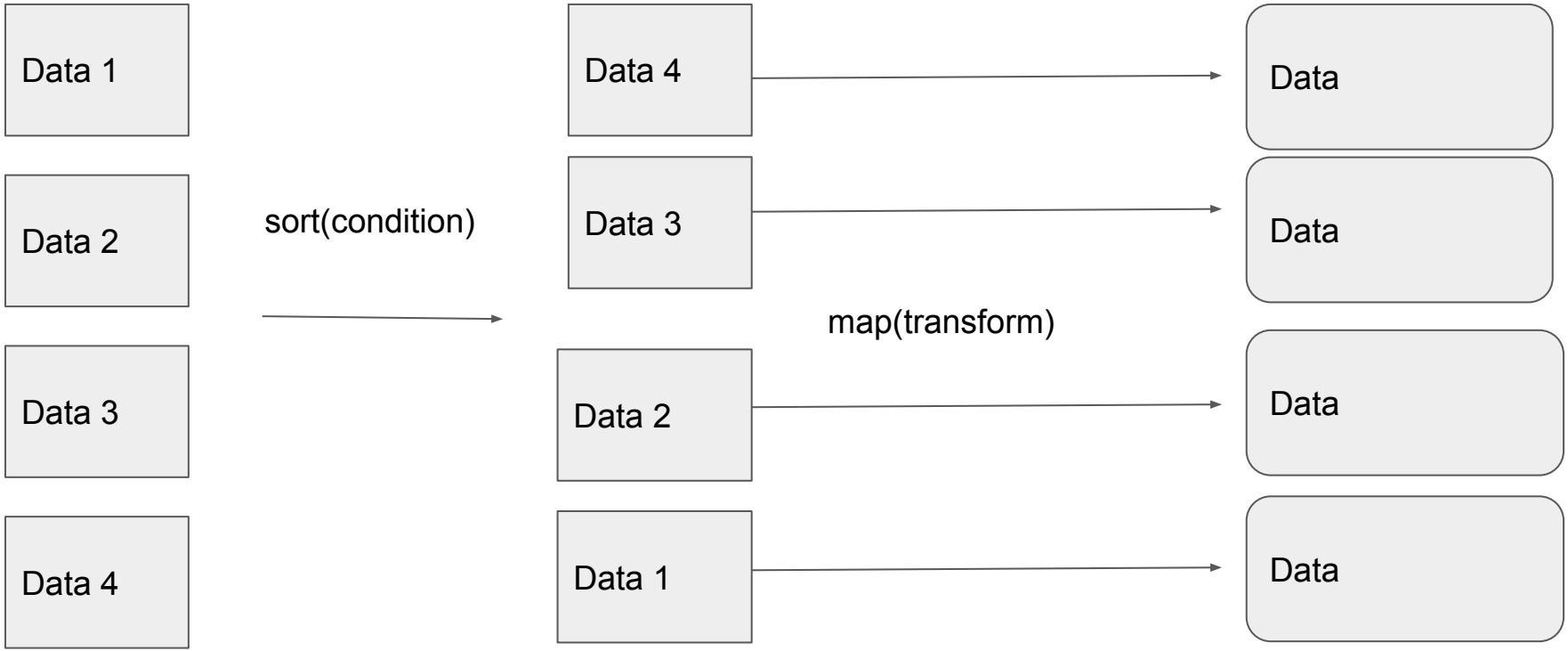
<sentence>The
red fox jumped
over the lazy
brown
dog</sentence>

<tokens>
<token>The</token>
<token>red</token>
<token>fox</token>
<token>jumped</token>
<token>over</token>
<token>the</token>
<token>lazy</token>
<token>brown</token>
<token>dog</token>
</tokens>

Computational Pipeline

Chain functions

Computational pipeline: sort and visualise



Example: Combining sort and map to HTML (I)

```
<students>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
</students>
```

First sort by name

```
<students>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
</students>
```

Example: Combining sort and map to HTML (II)

```
<students>
 <student>
   <name>Daryl Dixon</name>
   <age>33</age>
   <subject>Science</subject>
 </student>
 <student>
   <name>Maggie</name>
   <age>36</age>
   <subject>Arts</subject>
 </student>
 <student>
   <name>Rick Grimes</name>
   <age>35</age>
   <subject>Maths</subject>
 </student>
</students>
```

Then map students to table

Example: Combining sort and map to HTML (III)

```
<table>
<tr>
  <th>Name</th>
  <th>Age</th>
  <th>Subject</th>
</tr>
<tr>
  <td>Daryl Dixon</td>
  <td>33</td>
  <td>Science</td>
</tr>
<tr>
  <td>Maggie</td>
  <td>36</td>
  <td>Arts</td>
</tr>
<tr>
  <td>Rick Grimes</td>
  <td>35</td>
  <td>Maths</td>
</tr>
</table>
```

Name	Age	Subject
Daryl Dixon	33	Science
Maggie	36	Arts
Rick Grimes	35	Maths

Resources

https://pypi.org/project/functional-pipeline/

https://docs.marklogic.com/guide/cpf/pipelines

Description of function: https://www.progress.com/tutorials/xquery/functions

Wiki examples: https://en.wikipedia.org/wiki/XQuery

HTML table: https://www.w3schools.com/html/html_tables.asp