Flexible Custom Formulas: How to Use VisibleRows and VisibleColumns

Jānis Plūme
Things [were] not evident when writing formulas of report-wide measures

- Report row filter by condition
- Removed members
- Bookmarked members
- Dimension used both in rows/columns and pages
Outline

• How to work with visible rows in formulas

• Use cases
Standard calculations from what you see on report

VisibleRowsSet()
VisibleColumnsSet()
CurrentTuple()
Standard calculations from what you see on report

[Image of a data visualization interface with a table showing issues created and average issues created by quarter for the year 2020.]
How visible rows work?
How to see visible rows?

```
<table>
<thead>
<tr>
<th>Issue Type</th>
<th>Issues created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>564</td>
</tr>
<tr>
<td>Change</td>
<td>29</td>
</tr>
<tr>
<td>Data task</td>
<td>140</td>
</tr>
<tr>
<td>Epic</td>
<td>92</td>
</tr>
<tr>
<td>Feature</td>
<td>13</td>
</tr>
<tr>
<td>Improvement</td>
<td>5</td>
</tr>
<tr>
<td>Incident</td>
<td>44</td>
</tr>
<tr>
<td>Problem</td>
<td>21</td>
</tr>
<tr>
<td>Risk</td>
<td>8</td>
</tr>
<tr>
<td>Service Request</td>
<td>25</td>
</tr>
<tr>
<td>Story</td>
<td>796</td>
</tr>
<tr>
<td>Sub-task</td>
<td>171</td>
</tr>
<tr>
<td>Test</td>
<td>5</td>
</tr>
<tr>
<td>Test Execution</td>
<td>5</td>
</tr>
<tr>
<td>Test Plan</td>
<td>2</td>
</tr>
<tr>
<td>Test task</td>
<td>131</td>
</tr>
</tbody>
</table>
```
How to see visible rows?

```python
SetToStr(VisibleRowsSet()) =

{[Issue Type].[Bug],
 [Issue Type].[Change],
 [Issue Type].[Data task],
 [Issue Type].[Epic],
 [Issue Type].[Feature],
 [Issue Type].[Improvement],
 [Issue Type].[Incident],
 [Issue Type].[Problem],
 [Issue Type].[Risk],
 [Issue Type].[Service Request],
 [Issue Type].[Story],
 [Issue Type].[Sub-task],
 [Issue Type].[Test],
 [Issue Type].[Test Execution],
 [Issue Type].[Test Plan],
 [Issue Type].[Test task]}
```
How to see visible rows?

### Issue Type
- **Demo**

<table>
<thead>
<tr>
<th>Issue Type</th>
<th>Issues created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>3</td>
</tr>
<tr>
<td>Story</td>
<td>18</td>
</tr>
<tr>
<td>Sub-task</td>
<td>4</td>
</tr>
</tbody>
</table>

**Pages**
- Nonempty

**Rows**
- Nonempty
- Issue Type
  - Select individual members
  - All hierarchy level members
    - Select all members at level
      - Issue Type

**Columns**
- Measures
  - Table
How to see visible rows?

SetToStr(VisibleRowsSet()) =

{[Issue Type].[Bug],
 [Issue Type].[Story],
 [Issue Type].[Sub-task]}
Multiple dimensions in rows

Visible rows is the set of tuples with crossjoined members
Multiple dimensions in rows

Visible rows is the set of tuples with crossjoined members.

```plaintext
SetToStr(VisibleRowsSet()) =

{([Issue Type].[Story], [Status].[In Progress]),
 ([Issue Type].[Story], [Status].[To Do]),
 ([Issue Type].[Story], [Status].[Done]),
 ([Issue Type].[Sub-task], [Status].[In Progress]),
 ([Issue Type].[Sub-task], [Status].[To Do]),
 ([Issue Type].[Bug], [Status].[To Do]),
 ([Issue Type].[Bug], [Status].[Done])
}
```
Multiple dimensions in rows

Visible rows is the set of tuples with crossjoined members

SetToStr(VisibleRowsSet()) =

{([Issue Type].[Bug], [Status].[All Statuses]),
([Issue Type].[Bug], [Status].[To Do]),
([Issue Type].[Bug], [Status].[Done]),
([Issue Type].[Story], [Status].[All Statuses]),
([Issue Type].[Story], [Status].[To Do]),
([Issue Type].[Story], [Status].[In Progress]),
([Issue Type].[Story], [Status].[Done]),
([Issue Type].[Sub-task], [Status].[All Statuses]),
([Issue Type].[Sub-task], [Status].[To Do]),
([Issue Type].[Sub-task], [Status].[In Progress])}
Accessing elements from visible rows

- Access by index
- CurrentTuple()
- Filter/Generate
Accessing elements from visible rows

- Access by index in the set
- `CurrentTuple()`
- `Filter/Generate`

```c
TupleToStr(VisibleRowsSet().Item(2))
= ([Issue Type].[Bug],[Status].[Done])
```
Accessing elements from visible rows

- Access by index in the set
- CurrentTuple()
- Filter/Generate

```csharp
TupleToStr(VisibleRowsSet().Item(2).Item(1))
= [Status].[Done]
```
Accessing elements from visible rows

- Access by index in the set
- `CurrentTuple()`
- `Filter/Generate`

```csharp
TupleToStr( CurrentTuple( VisibleRowsSet() ) )
```
Accessing elements from visible rows

• Access by index in the set
• CurrentTuple()
• Filter/Generate

```ruby
SetToStr(
    Filter(VisibleRowsSet(),
        [Status].CurrentHierarchyMember.Level.Name="Status"
    )
)
=

{([Issue Type].[Bug], [Status].[To Do]),
 ([Issue Type].[Bug], [Status].[Done]),
 ([Issue Type].[Story], [Status].[To Do]),
 ([Issue Type].[Story], [Status].[In Progress]),
 ([Issue Type].[Story], [Status].[Done]),
 ([Issue Type].[Sub-task], [Status].[To Do]),
 ([Issue Type].[Sub-task], [Status].[In Progress])}
```
Accessing elements from visible rows

- Access by index in the set
- CurrentTuple()
- Filter/Generate

```csharp
SetToStr(
    Generate(
        Filter(VisibleRowsSet(),
            [Status].CurrentHierarchyMember.Level.Name="Status"),
        [Status].CurrentHierarchyMember
    )
) =

{[Status].[To Do],
 [Status].[Done],
 [Status].[In Progress]}
```
Some use cases
All properties for any dimension in the rows

[Measures].[Sprint all properties] = 1 [Sprint].CurrentMember.AllProperties
All properties for any dimension in the rows

```plaintext
[Measures].[Universal All properties] =

1 CurrentTuple(VisibleRowsSet()).item(0).AllProperties
```
## Creating sub-totals

<table>
<thead>
<tr>
<th>Issue Type</th>
<th>Status</th>
<th>Issues created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>All Statuses</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>1</td>
</tr>
<tr>
<td>Story</td>
<td>All Statuses</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>In Progress</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>9</td>
</tr>
<tr>
<td>Sub-task</td>
<td>All Statuses</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>In Progress</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue Type</th>
<th>Status</th>
<th>Issues created</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story</td>
<td>In Progress</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>9</td>
</tr>
<tr>
<td>Sub-task</td>
<td>In Progress</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>2</td>
</tr>
<tr>
<td>Bug</td>
<td>To Do</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>1</td>
</tr>
</tbody>
</table>
Creating sub-totals

Sum(
  Generate(
    VisibleRowsSet(),
    [Status].CurrentHierarchyMember
  ),
  [Measures].[Issues created]
)

{[Status].[In Progress],
 [Status].[To Do],
 [Status].[Done]}

Sum(
  Generate(
    VisibleRowsSet(),
    CurrentTuple(VisibleRowsSet()).Item(1)
  ),
  [Measures].[Issues created]
)
Creating sub-totals

Sum (Generate (VisibleRowsSet(), [Status].CurrentHierarchyMember), [Measures].[Issues created])

{[Status].[In Progress], [Status].[To Do], [Status].[Done]}

<table>
<thead>
<tr>
<th>Issue Type</th>
<th>Status</th>
<th>Issues created</th>
<th>Sub total by Issue type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story</td>
<td>In Progress</td>
<td>1</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>Sub-task</td>
<td>In Progress</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>To Do</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Bug</td>
<td>To Do</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Done</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Use of sub-totals

Count the number of second level members under the each first level member
Self-adjusting average
Self-adjusting average

```sql
CASE WHEN [Time].CurrentHierarchyMember.Level.Name='Month'
THEN Sum(
    Filter(
        VisibleRowsSet(),
        [Time].CurrentHierarchyMember.Level.Name="Day"
    ),
    [Measures].[Total resolution days]
) / Sum(
    Filter(
        VisibleRowsSet(),
        [Time].CurrentHierarchyMember.Level.Name="Day"
    ),
    [Measures].[Issues resolved]
)
ELSE [Measures].[Average resolution days]
END
```
Show visually issues on timeline whose cycle time exceeds the 75th percentile
Show visually issues on timeline whose cycle time exceeds the 75th percentile

<table>
<thead>
<tr>
<th>Time</th>
<th>Average resolution days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 01 2021</td>
<td>DA-179: 42.13</td>
</tr>
<tr>
<td></td>
<td>DA-181: 39.56</td>
</tr>
<tr>
<td></td>
<td>DA-190: 28.02</td>
</tr>
<tr>
<td></td>
<td>DB-189: 26.74</td>
</tr>
<tr>
<td></td>
<td>DB-190: 25.46</td>
</tr>
<tr>
<td></td>
<td>DG-174: 46.54</td>
</tr>
<tr>
<td>Jan 04 2021</td>
<td>DB-185: 34.75</td>
</tr>
<tr>
<td></td>
<td>DG-171: 55.26</td>
</tr>
<tr>
<td>Jan 06 2021</td>
<td>DA-188: 35.58</td>
</tr>
<tr>
<td></td>
<td>DA-191: 31.73</td>
</tr>
<tr>
<td></td>
<td>DB-174: 50.96</td>
</tr>
<tr>
<td></td>
<td>DB-177: 47.11</td>
</tr>
<tr>
<td>Jan 08 2021</td>
<td>DA-105: 188.33</td>
</tr>
<tr>
<td></td>
<td>DB-103: 189.98</td>
</tr>
<tr>
<td></td>
<td>DB-169: 59.38</td>
</tr>
<tr>
<td></td>
<td>DB-193: 28.61</td>
</tr>
</tbody>
</table>
Calculate percentile from all issues resolved in period?

Shortages of the solution:
1) Iterates full set of issues, although the calculation is from the rows already filtered in the report
2) Relies that page filter is by year
Calculate percentile from all issues resolved in period?

Shortages of the solution:
1) Iterates full set of issues, although the calculation is from the rows already filtered in the report
2) Relies that page filter is by year
Avoiding re-calculation

```
[Measures].[Percentile 75] =
1  ConstantValue(
2  Percentile(
3    VisibleRowsSet(),
4    [Measures].[Total resolution days],
5    75
6  )
7 ))
```
Thank you!
Questions?

https://eazybi.com/accounts/1000/dashboards

community.eazybi.com

support@eazybi.com