Specification Table Schema

Relationship property

Proposal

1 – Requirement

- **Specify the relationship between two fields**
  - Three main link categories (see right):
    - derived, coupled, crossed

- **Example :**
  - Field « quarter » is derived from « month »
  - Field « name » is coupled to field « nickname »
  - Field « year » is crossed with field « semester »

- **Validation :**
  - Simple function (see below)
  - Requires all data
  - Test possible with each new input (derived and coupled) and not possible with crossed

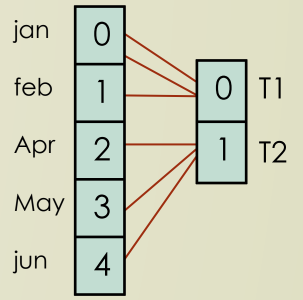
How to measure the link (see implementation example in last slide) ?

The evaluation is made by calculating **dist = len(set(zip(a,b)))** where a and b are array of the two fields *(python langage)*

```
dist   >=   max(len(set(a)), len(set(b)))
dist   <=   len(set(a)) *  len(set(b))
```

Quarter : [T1, T2, T2, T1, T2, T1] (a)
Month : [jan, apr, jun, feb, may, jan] (b)

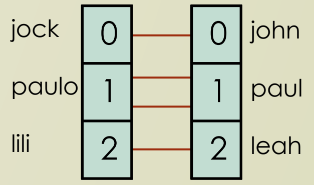
derived



```
if      dist == len(set(b))
and dist  >  len(set(a))
```

Name : [john, paul, leah, paul] (a)
Nickname : [jock, paulo, lili, paulo] (b)

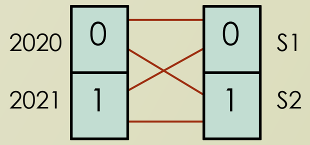
coupled



```
If      dist == len(set(b))
and  dist == len(set(a))
```

Year : [2020, 2020, 2021, 2021] (a)
Semester : [S1, S2, S1, S2] (b)

crossed



```
If dist ==  len(set(a)) *  len(set(b))
```

2 – Implementation (three options)

- **1 – New Field descriptor**

```
        « name »: « quarter »
        « relationship » : {
                « parent » : « month »,
                « link » : « derived »
                }
```

- **Pros**
  - No mixing with other descriptors
  - Consistent with a field view
- **Cons**
  - New descriptor

- **2 – New Constraints descriptor**

```
        « name »: « quarter »
        « constraints » : {
                « relationship » : {
                        « parent » : « month »,
                        « link » : « derived »
                }
        }
```

- **Pros**
  - The « constraints » property is consistent with the point
- **Cons**
  - The « crossed » link can't be validate at the data entry
  - Need to add a level in the properties tree

- **3 – New Table descriptor (other properties)**

```
    « relationship » : [
        {
                « fields »: « quarter »
                « parent » : « month »,
                « link » : « derived »
        } …
    ]
```

- **Pros**
  - New independant descriptor
- **Cons**
  - Relationships are described field by field

Option 1 seems to be the most suitable

3 – Text Proposal

Relationship

The relationship property MAY be used to define the dependency between another field. The relationship descriptor, if present, MUST be a JSON object and MUST contain two properties :

- parent : the property name of the field linked to
- link : the nature of the relationship between them

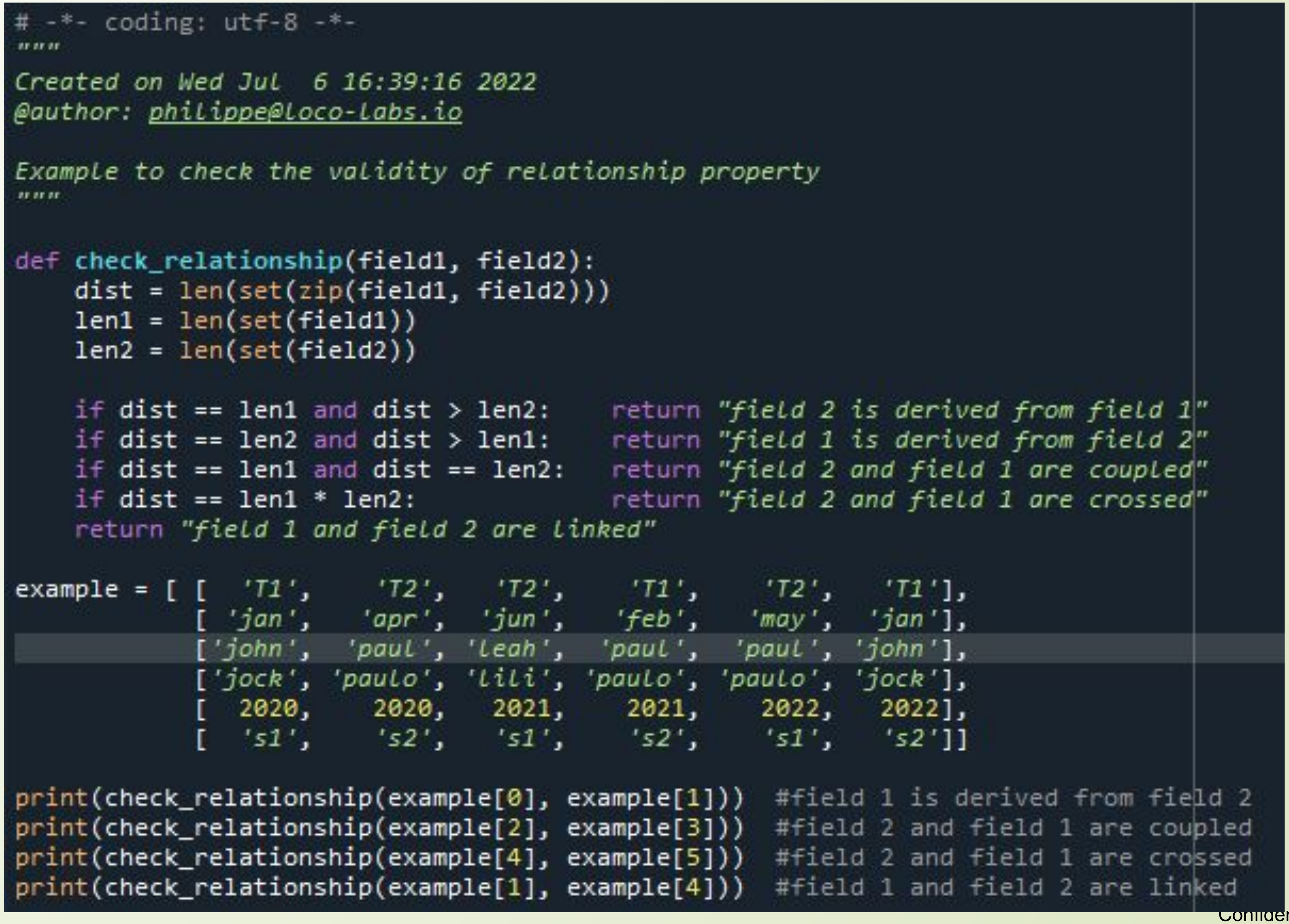
The link property value MUST be one of the three following :

- derived :
  - The field values are dependant on the values of parent field (a value in the parent field is associated with a single field value).
  - E.g. The « Quarter » field [T1, T2, T2, T1, T2, T1] is derived from the « month » field [jan, apr, jun, feb, may, jan]
  - i.e. if a new entry 'jun' is added, the corresponding « quarter » value must be 'T2'.

- coupled :
  - The field values are associated to the values of parent field (both fields are derived from each other).
  - E.g. The « Nickname" field [jock, paulo, lili, paulo] is coupled to the "name" field [john, paul, leah, paul]
  - i.e. if a new entry 'lili' is added, the corresponding « Name » value must be 'leah' just as if a new entry 'leah' is added, the corresponding « nickname » value must be 'lili'.

- crossed :
  - This relationship means that all the different values of the field are associated with all the different values of another field.
  - E.g. the "Year" Field [2020, 2020, 2021, 2021] is crossed to the "Semester" Field [S1, S2, S1, S2]
  - i.e the year 2020 is associated to semesters s1 and s2, just as the semester s1 is associated with years 2020 and 2021

4 - Check implementation Example



```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul  6 16:39:16 2022
@author: philippe@loco-labs.io

Example to check the validity of relationship property
"""


def check_relationship(field1, field2):
    dist = len(set(zip(field1, field2)))
    len1 = len(set(field1))
    len2 = len(set(field2))

    if dist == len1 and dist > len2:    return "field 2 is derived from field 1"
    if dist == len2 and dist > len1:    return "field 1 is derived from field 2"
    if dist == len1 and dist == len2:   return "field 2 and field 1 are coupled"
    if dist == len1 * len2:             return "field 2 and field 1 are crossed"
    return "field 1 and field 2 are linked"

example = [ [  'T1',    'T2',   'T2',    'T1',    'T2',   'T1'],
            [ 'jan',   'apr',  'jun',   'feb',   'may',  'jan'],
            ['john',  'paul', 'leah',  'paul',  'paul', 'john'],
            ['jock', 'paulo', 'lili', 'paulo', 'paulo', 'jock'],
            [  2020,    2020,   2021,    2021,    2022,   2022],
            [  's1',    's2',   's1',    's2',    's1',   's2']]

print(check_relationship(example[0], example[1]))  #field 1 is derived from field 2
print(check_relationship(example[2], example[3]))  #field 2 and field 1 are coupled
print(check_relationship(example[4], example[5]))  #field 2 and field 1 are crossed
print(check_relationship(example[1], example[4]))  #field 1 and field 2 are linked
```

Appendix – Indexed List

https://github.com/loco-philippe/Environnemental-Sensing /blob/main/documentation/Ilist_technical.pdf