# Environnemental Sensing

**Ilist**

Concepts and principles

# 0 – Main

- **Indexed list**
  - 0 - Presentation
- **Objectives**
  - Structure optimization
    - 1 - Structure understanding
    - 2 - Structure optimization
    - 3 - Size optimization
  - Integrate process
    - 4 - Building process
    - 4 - Interface tools
- **Associated tools**
  - 5 - Data format
  - 5 - Exchange format
- **Extension**
  - 6 - Environmental sensing
  - 6 - Sensor acquisition

# 0 - Ilist (Indexed list)

## What is Ilist ?

**List of values :**

+ Age : [12, 28, 39, 58]

**List of indexes :**

Name : [Paul, John, Lea, Cat]

City : [Paris, Metz, Rennes, Bollène]

....

| Name | city | Age |
|---|---|---|
| Paul | Paris | 12 |
| John | Metz | 28 |
| Lea | Rennes | 39 |
| Cat | Bollène | 58 |

*Example : csv file, measurement, log, matrix*

## Why Ilist ?

- The majority of work processes are underpinned by Sheets
- The main Open-data format is CSV
- Existing tools process data but not data structures

***Such tool daesn't exist !***

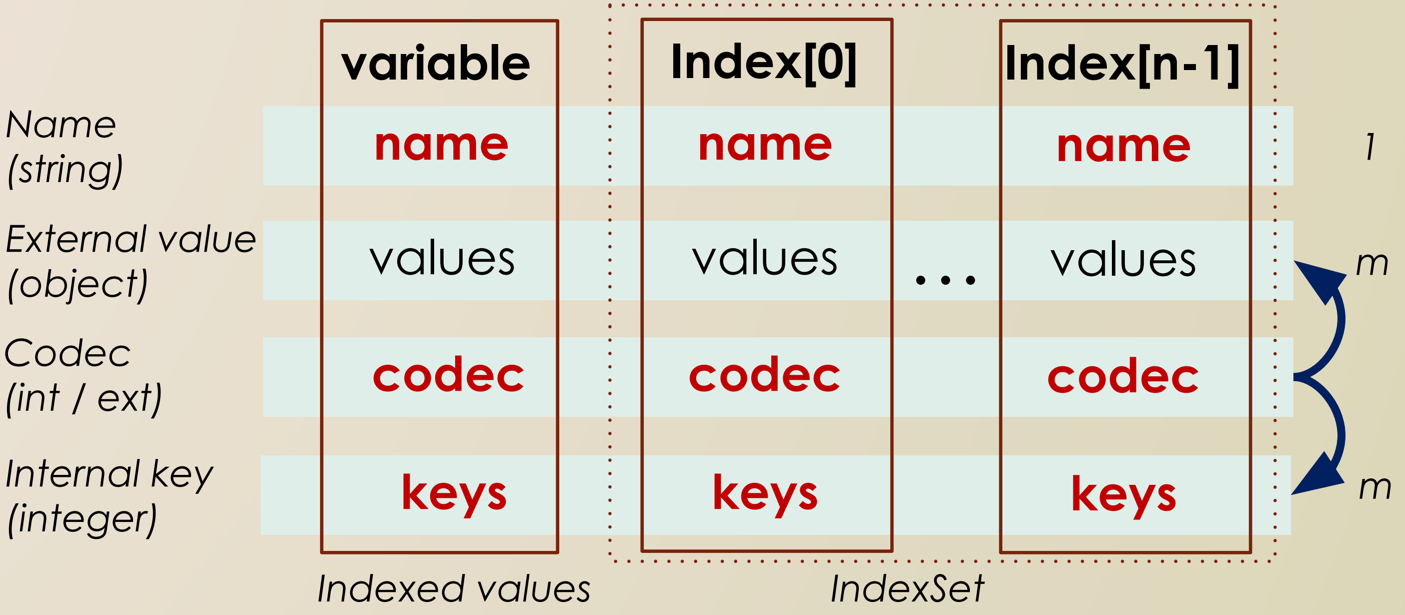
# 0 – Ilist structure

**Red** : static value
Black : dynamic value

n : number of indexes
m : number of values

## Two levels

- **External values**
  *(every kind of object)*
- **Internal keys**
  (no duplication)



| | variable | Index[0] | | Index[n-1] | |
|---|---|---|---|---|---|
| *Name (string)* | **name** | **name** | | **name** | *1* |
| *External value (object)* | values | values | . . . | values | *m* |
| *Codec (int / ext)* | **codec** | **codec** | | **codec** | |
| *Internal key (integer)* | **keys** | **keys** | | **keys** | *m* |

*Indexed values*

*IndexSet*

## Example

### Structure analysis

*variable*

*indexes*

| score |
|---|
| name |
| age |
| subject |

***name***

*External*

| 10 | 12 | 15 |
|---|---|---|
| Paul | Lea | Lea |
| 16 | 15 | 15 |
| math | math | english |

***values***

codec

*Internal*

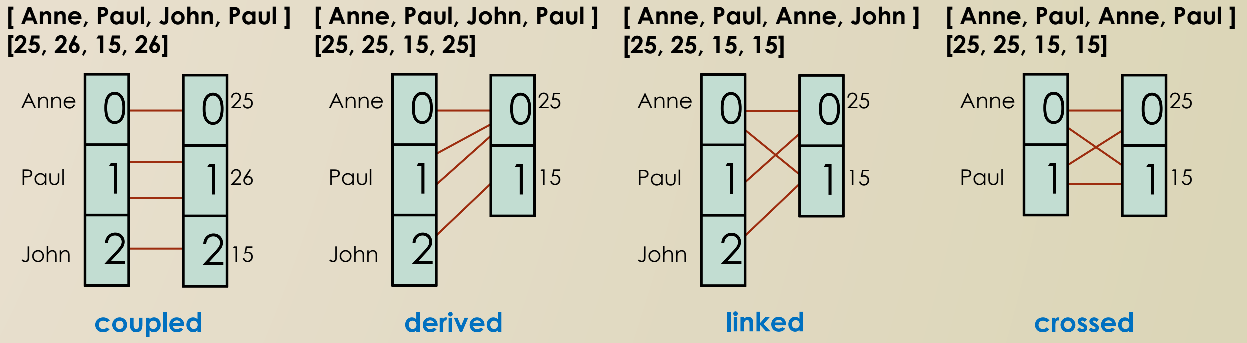
| 0 | 1 | 2 |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

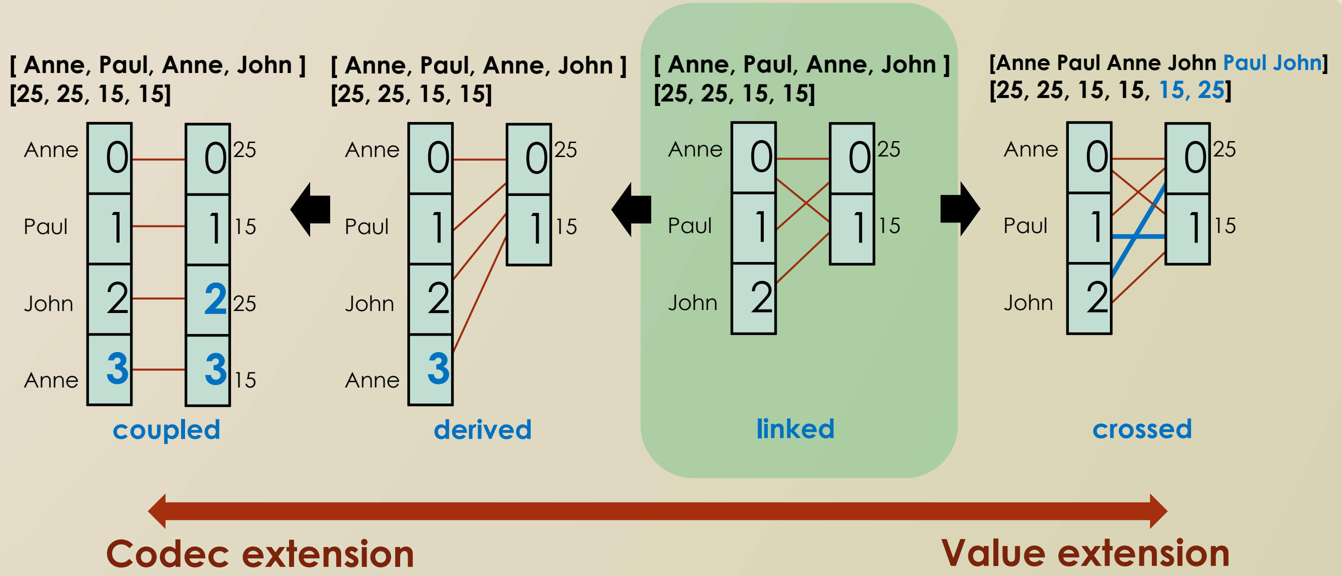
***keys***

# 1 – Structure understanding

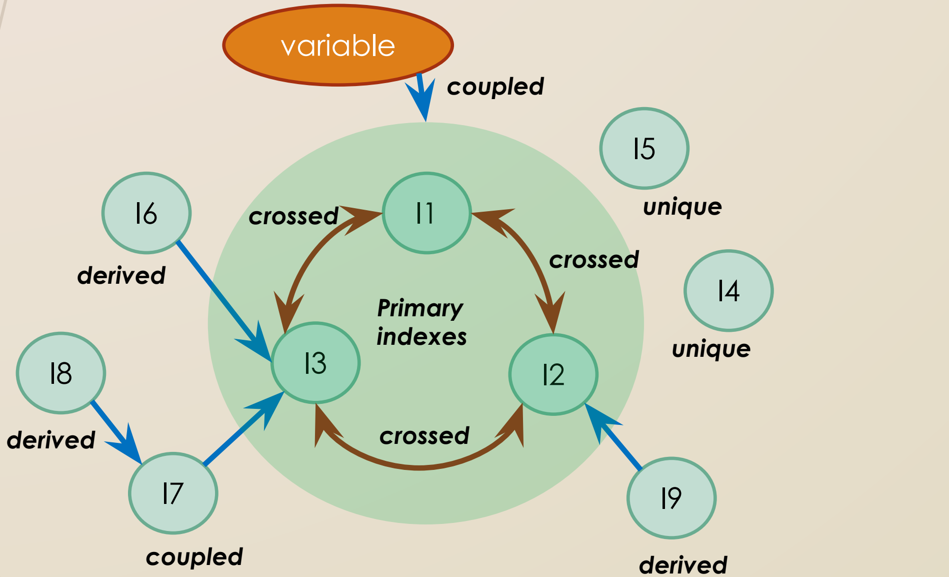
- **Relationship analysis**
  - Index qualification
  - Index relationship
- **Data structuration**
  - Canonical format
  - Convergence

## Index relationship





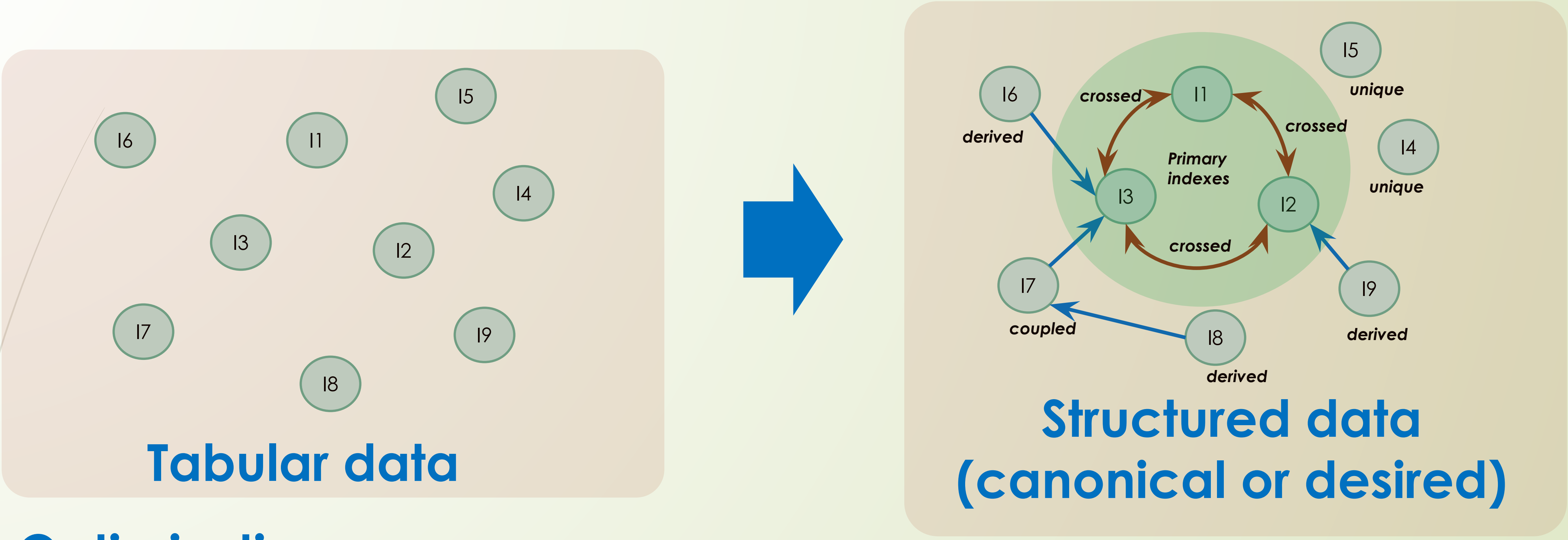
## Convergence



## Canonical structure

Confidential C

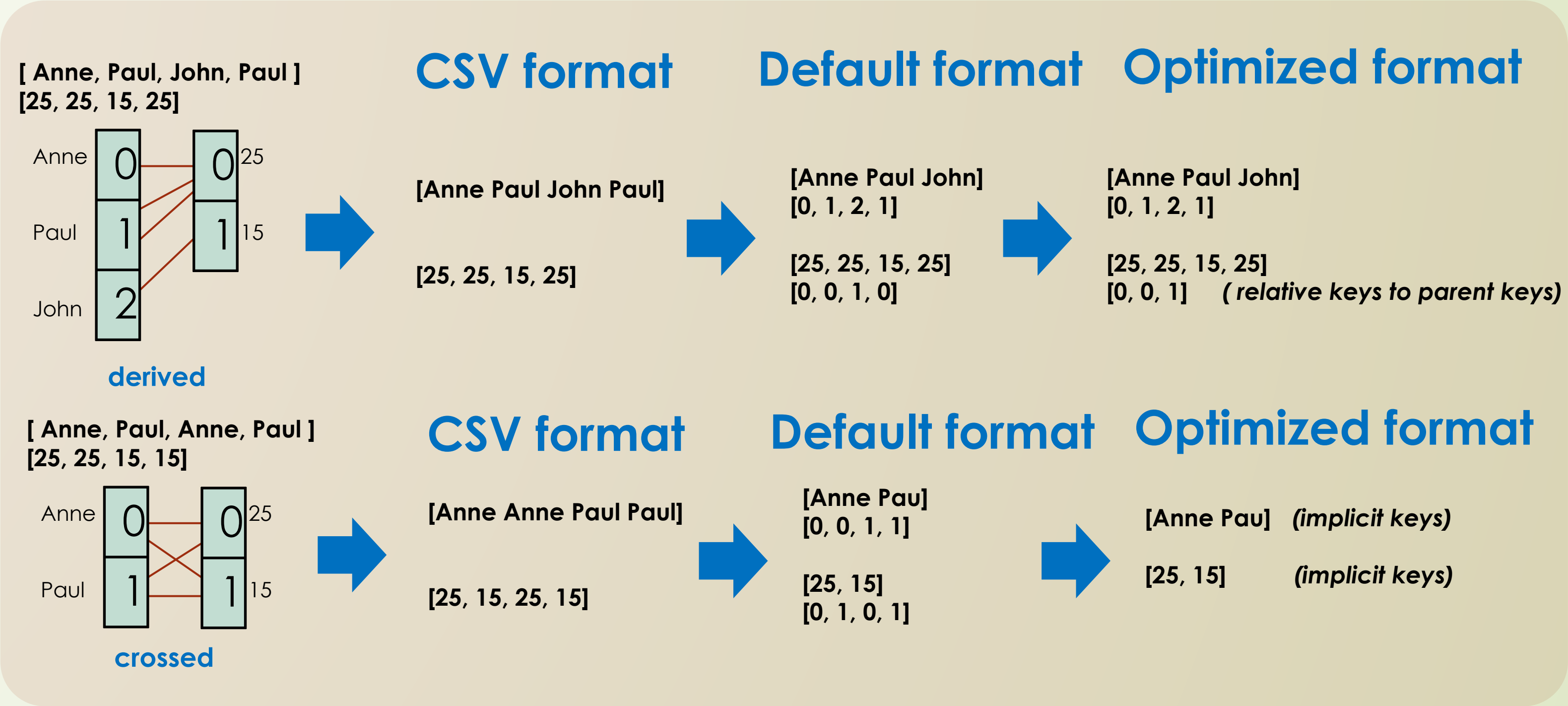
# 2 – Structure optimization



- **Optimization**
  - minimization of additional data to achieve canonical structure
- **Consistency**
  - identification of additional data to achieve the desired structure

# 3 – Size optimization

- **Canonical structure**
  - Minimal structure
- **Minimal size**
  - No multiple value
  - Keys optimization
- **Exchange format**
  - Text : JSON format
  - Binary : CBOR (RFC 8949)



[ Anne, Paul, John, Paul ]
[25, 25, 15, 25]

derived

**CSV format**

[Anne Paul John Paul]

[25, 25, 15, 25]

**Default format**

[Anne Paul John]
[0, 1, 2, 1]

[25, 25, 15, 25]
[0, 0, 1, 0]

**Optimized format**

[Anne Paul John]
[0, 1, 2, 1]

[25, 25, 15, 25]
[0, 0, 1] *( relative keys to parent keys)*

[ Anne, Paul, Anne, Paul ]
[25, 25, 15, 15]

crossed

**CSV format**

[Anne Anne Paul Paul]

[25, 15, 25, 15]

**Default format**

[Anne Pau]
[0, 0, 1, 1]

[25, 15]
[0, 1, 0, 1]

**Optimized format**

[Anne Pau] *(implicit keys)*

[25, 15] *(implicit keys)*

**Example** : Open-data - french charging point (EVSE)

**7.5 Mo** – 11 000 rows – 49 columns

**Analysis** :

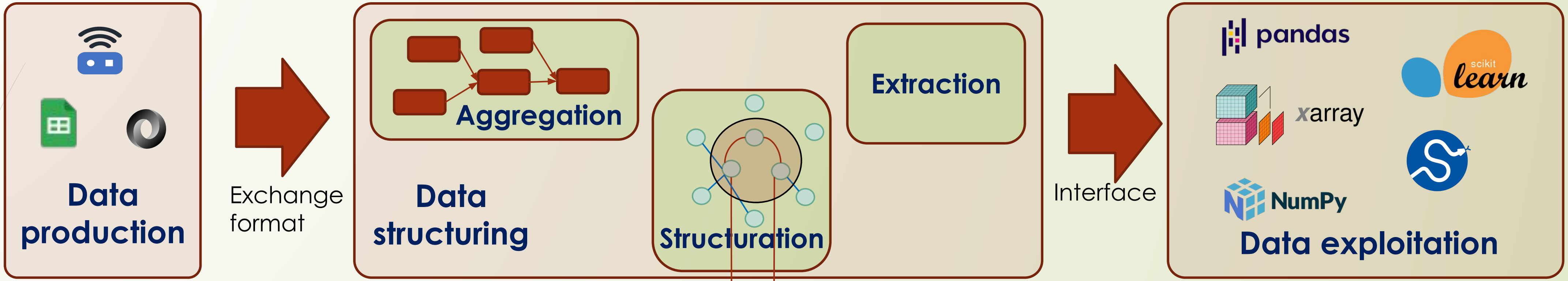
Indexes : 1 coupled, 6 derived, 1 crossed, 41 linked

Canonical format : 1 crossed, 48 derived

**File size** :

| | |
|---|---|
| Default : | 3.7 Mo |
| Optimized : | 2.5 Mo |
| CBOR optimized : | **1.7 Mo (gain : 77% !)** |

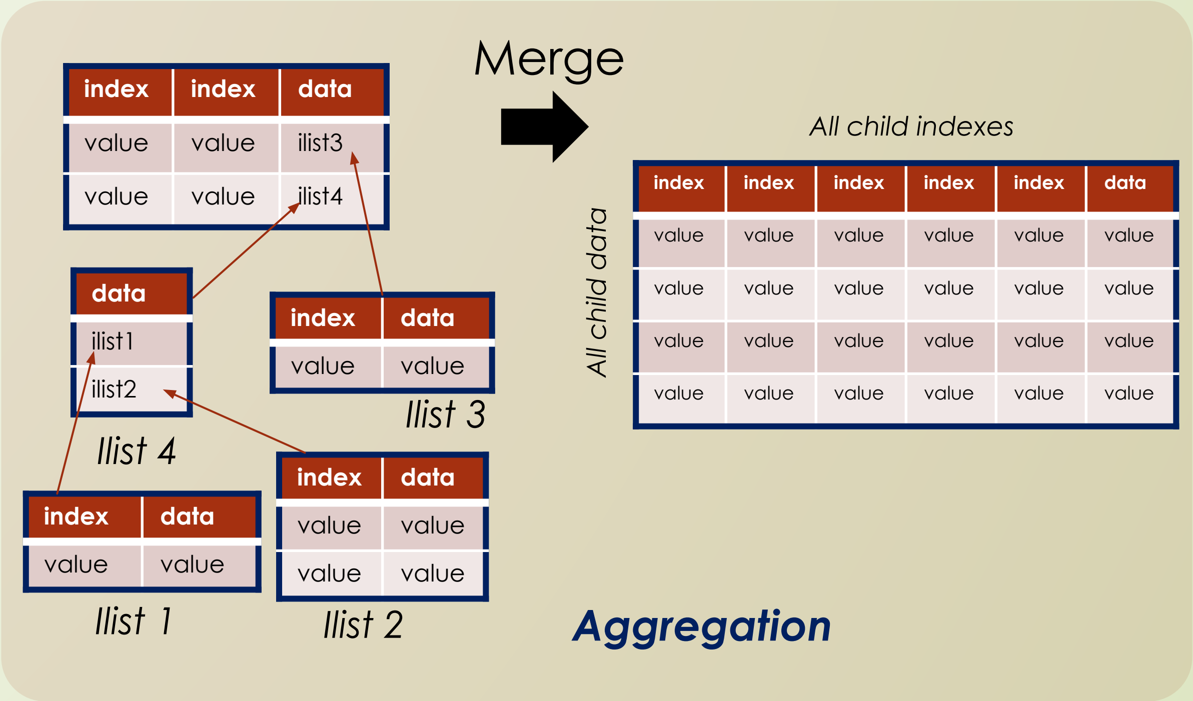
# 4 – Integrate process



- **Data production interface**
  - Exchange format (Json, Bluetooth, CSV)

- **Aggregation / merge functions**
  - Adapted to projects / organizations
  - Add information without altering

- **Export to analysis tools**
  - Canonical structure compatibility



| index | index | data |
|---|---|---|
| value | value | ilist3 |
| value | value | ilist4 |

ilist 4

| data |
|---|
| ilist1 |
| ilist2 |

ilist 3

| index | data |
|---|---|
| value | value |

ilist 1

| index | data |
|---|---|
| value | value |

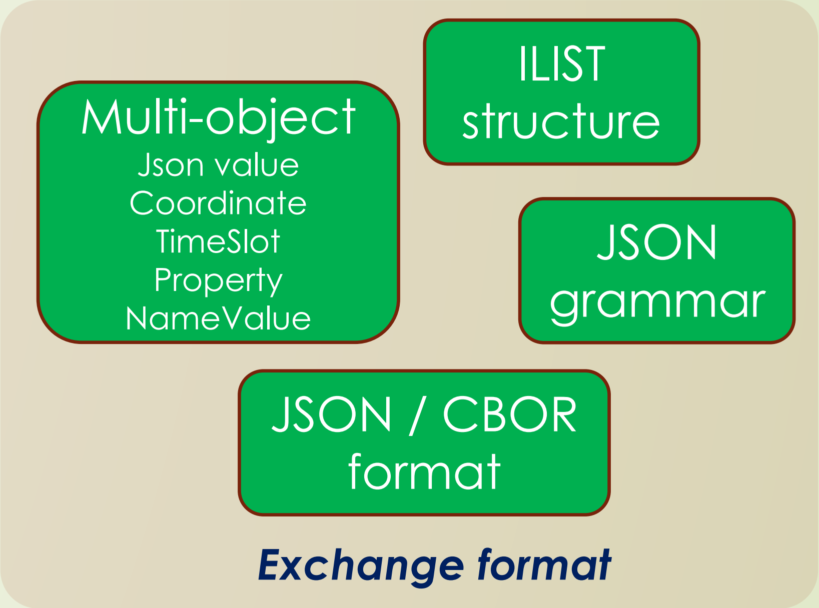
ilist 2

| index | data |
|---|---|
| value | value |
| value | value |

| index | index | index | index | index | data |
|---|---|---|---|---|---|
| value | value | value | value | value | value |
| value | value | value | value | value | value |
| value | value | value | value | value | value |
| value | value | value | value | value | value |

# 5 – Data format

- **Large set of objects**
  - New format (timeslot, property)
- **JSON representation**
  - Exchange format



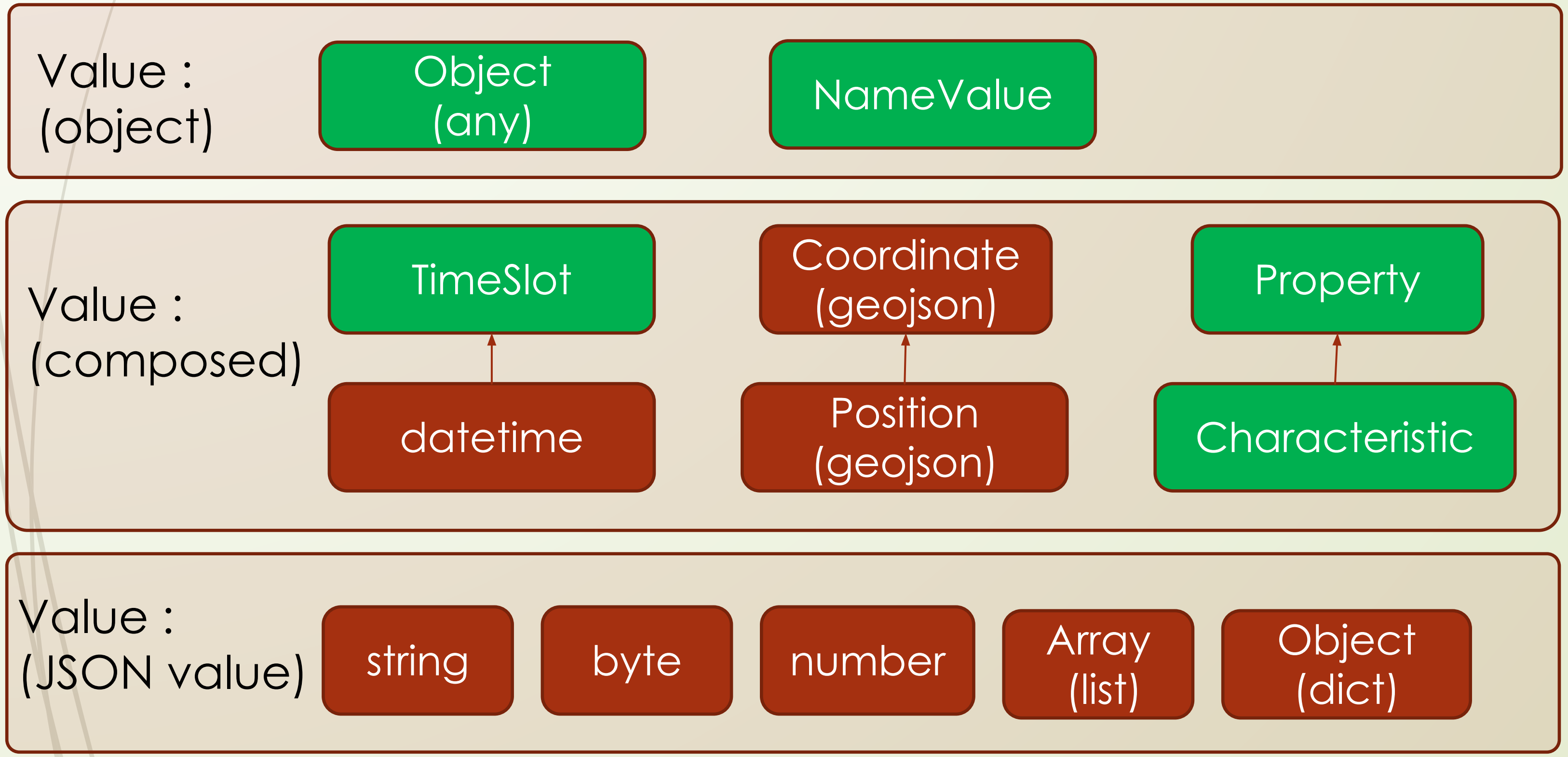
Multi-object
Json value
Coordinate
TimeSlot
Property
NameValue

ILIST structure

JSON grammar

JSON / CBOR format

***Exchange format***



Value : (object)

Object (any)

NameValue

Value : (composed)

TimeSlot

datetime

Coordinate (geojson)

Position (geojson)

Property

Characteristic

Value : (JSON value)

string

byte

number

Array (list)

Object (dict)

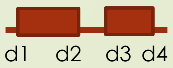
**NameValue**

{ ‘Paris’ : [2,4, 48,9] }

**Object**

{‘object name’: object value }

**TimeSlot**



[ [ [d1,d2] ], [ [d3,d4] ] ]

**Property**

{‘char’:’PM10, ‘unit’: ‘kg/m3’, …}
*(Char -> i. e. BLE characteristic)*

# 6 – Ilist extension

- **Observation**
  - Ilist specialization with three main indexes :
    - Datation index (Timeslot), Location index (coordinate), Property
  - Conformance with ISO 19156 : Observation & Measurement

- **Sensor acquisition**
  - Integration of Bluetooth Environmental Sensing Profile (extension in 2021)
  - Reduced exchange format for micro-controlers

- **Open-data**
  - Tool to define data structuring (tabular data)
  - Consistency measurement tool (tabular data)

# Appendix

Concepts and principles

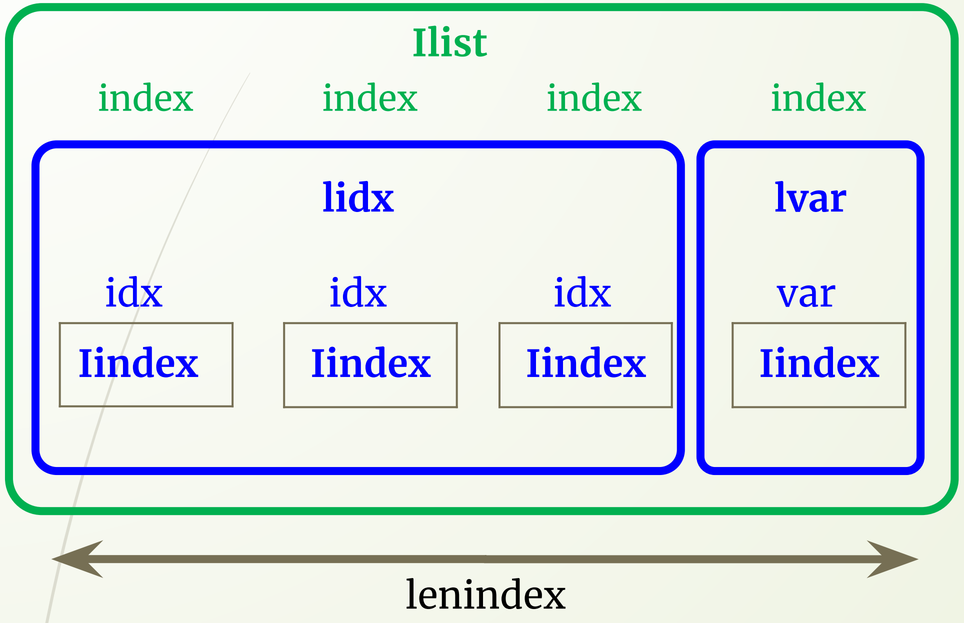
**1 - Index analysis**

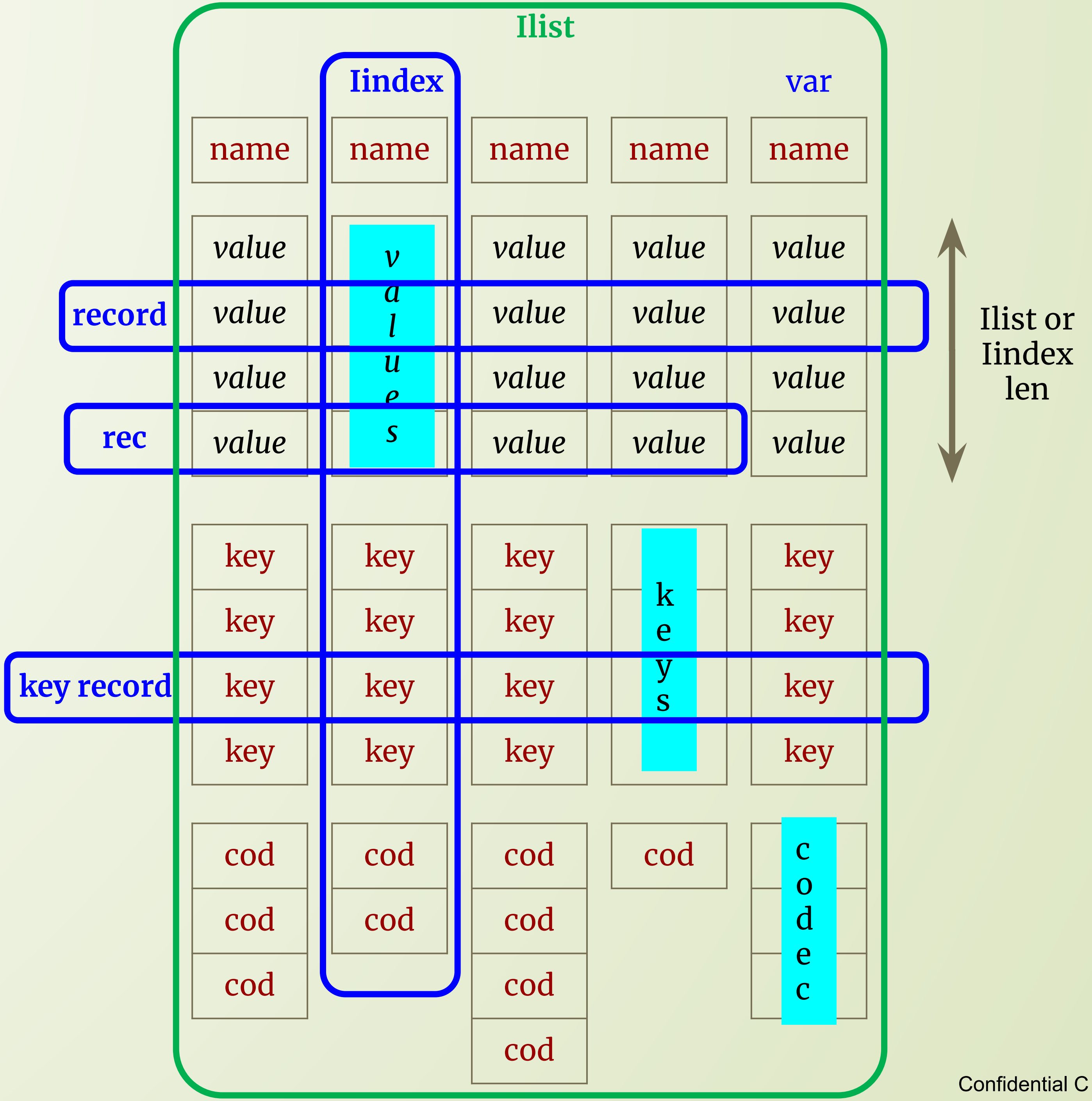
**2 - Matrix generation**

**3 - Aggregation**

**4 – Format, storage**

# 0 – Terminology


Ilist
index index index index
lidx
lvar
idx idx idx
var
Iindex Iindex Iindex Iindex
lenindex


Ilist
Iindex
var
name name name name name
value value value value
record value value value value
value value value value
rec value value value value
values
key key key key
key key key key
key record key key key key
key key key key
keys
cod cod cod cod
cod cod cod
cod cod
cod
codec
Ilist or Iindex len

val : external json representation of internal value

*Italic: dynamic value*

# 1 - Index categories

| | | | |
|---|---|---|---|
| **Values** | **[ Anne, Paul, Anne]** | **[ Anne, Anne, Anne]** | **[ Anne, Paul, Anne]** |
| **Length**<br>**(number of values)** | **3** | **3** | **3** |
| **Codec**<br>**(row)** | 0<br>1<br>2 | 0 | 0<br>1 |
| **Type codec** | *full* | *unique* | *default* |
| **Property** | Rate : 1<br>Disttomax : 0 | Rate : 0<br>Disttomin : 0 | Rate : 0<br>Disttomin : 0 |
| **Representation** | **Codec :[Anne, Paul, Anne]**<br>**Keys: implicit (full keys)** | **Codec :[Anne]**<br>**Keys: implicit** | **Codec :[Anne, Paul ]**<br>**Keys: [ 0, 1, 1 ]** |

## Definition :

**Default codec :**
list of different values
**Full codec :**
list of values

## Indicators :

**M = len(values)**
**m = len(set(values))**
**x = len(codec)**

**Rate : (M – x) / (M – m)**
**Dist to min : x - m**
**Dist to max : M - x**

- **Properties**
  - Any index have a default codec and a full codec
  - Default are the shortest codec, full are the longest codec

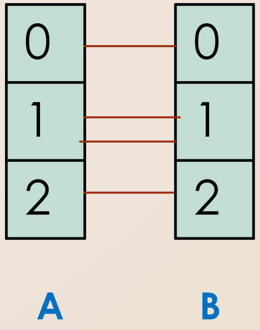
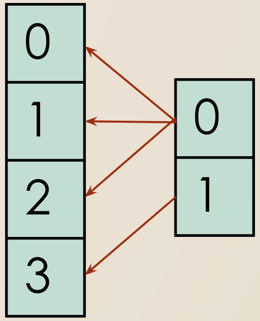
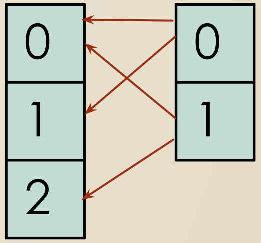
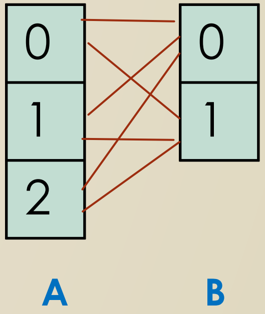
| | |
|---|---|
| M = 0 | null |
| m = 1 | unique |
| m = M > 1 | complete |
| m < M = x | full |
| x = m < M | default |
| m < x < M | mixed |

*A codec defines the correspondence between values and keys (e.g.) :*
- *1 : Anne*
- *0 : Paul*
- *2 : John*

*A codec may not be bijective (e.g.) :*
- *0 : Anne*
- *1 : Paul*
- *2 : Anne*

C

# 1 - Relationship categories

| | | | | | |
|---|---|---|---|---|---|
| Values | A<br>B | [ Anne, Paul, John, Paul ]<br>[25, 26, 15, 26] | [ Anne, Paul, John, Lea ]<br>[25, 25, 25, 12 ] | [ Anne, Paul, Anne, Lea ]<br>[25, 25, 12, 12 ] | [Anne, Anne, Anne, Paul, Paul, Paul]<br>[25, 12, 25, 12, 25, 12 ] |
| Codec (row) | |  |  <br>B derived from A<br>A derive B |  <br>B linked from A<br>A link B |  |
| Type | | coupled | derived<br>(asymmetrical) | linked<br>(asymmetrical) | crossed |
| Property | | Rate : 0<br>Disttomin : 0<br>diff : 0 | Rate : 0<br>Disttomin : 0<br>0 < diff < min | 0 < Rate < 1<br>min < dist < Max<br>0 <= diff < min | Rate : 1<br>Disttomax : 0<br>0 <= diff < min |
| Keys | B | Relative<br>(equal Keys A) | Relative<br>to keys A | Absolute | Matrix order |
| | | **Codec :[25, 26, 15, 35]**<br>**Keys: implicit** | **Codec :[25, 12]**<br>**Keys: relative**<br>**[0,0,0,1]** | **Codec :[25, 12]**<br>**Keys: absolute**<br>**[0,0,1,1]** | **Codec :[25, 12]**<br>**Keys: implicit**<br>***([0,1,0,1,0,1])*** |

## Indicators :

**Max = len(i1) * len(i2)**
**min = max(len(i1), len(i2)**
**diff = abs(len(i1) – len(i2))**
**x = len(index(v1, v2))**

**Rate : (x – m) / (M – m)**
**Dist to min : x – m**
**Dist to coup : 2x - 2m + diff**
**Dist to max : M - x**

## *Relative derived keys :*

*Length:*

- *length(parent.codec)*

*Values:*

- Keyder(parent.key(i)) = key(i)

# 1 - relationship properties

- Type and Indicators are independant of Values (order or value) and dependant of Codec and Keys
- If one index is complete, all the indexes are derived or coupled from it
- If one index is unique, it is derived from all other indexes
- If A is derived (coupled) from B and B is derived (coupled) from C, A is derived (coupled) from C
- If A is coupled to B, all the relationships with other indexes are identical
- **Keys can be deduced with coupled or crossed relationship**

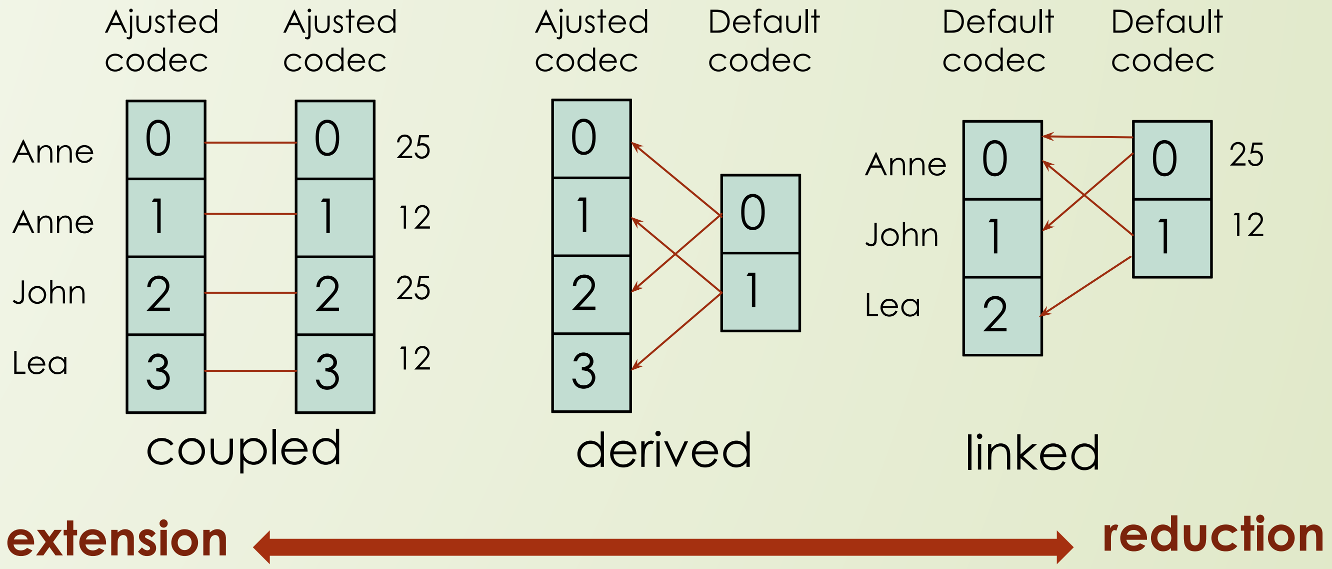
# 1 – Relationship adjustement

- **Codec reduction / extension**
  - Codec changed
  - Values unchanged

**Reduction is usefull to minimize codec size**

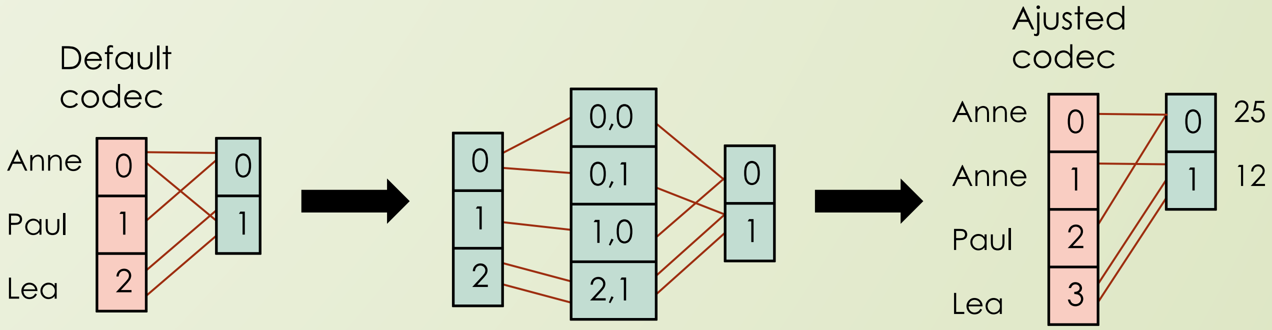
**Extension is usefull to increase values readibility (like csv data)**

**[ Anne, Anne, John, Lea ]**
**[25, 12, 25, 12 ]**



- **Codec adjustement**
  - Codec is ajusted to the other codec
  - Other index is derived or coupled to the ajusted index
  - If A is derived from B and if B is adjusted to C, A is still derived from B

**Keys can be deduced from keys parent**

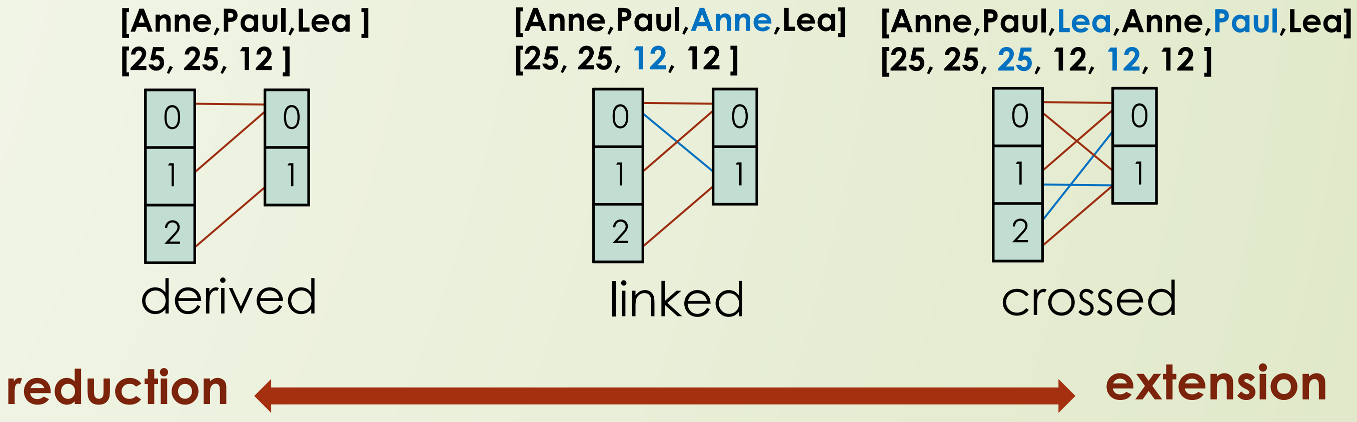


# 1 – Relationship adjustement

- **Values reduction / extension**
  - Codec unchanged
  - Values changed

**Extension is usefull to generate matrix**

**Reduction is usefull to increase codec readibility**



**[Anne,Paul,Lea ]**
**[25, 25, 12 ]**

derived

**[Anne,Paul,Anne,Lea]**
**[25, 25, 12, 12 ]**

linked

**[Anne,Paul,Lea,Anne,Paul,Lea]**
**[25, 25, 25, 12, 12, 12 ]**

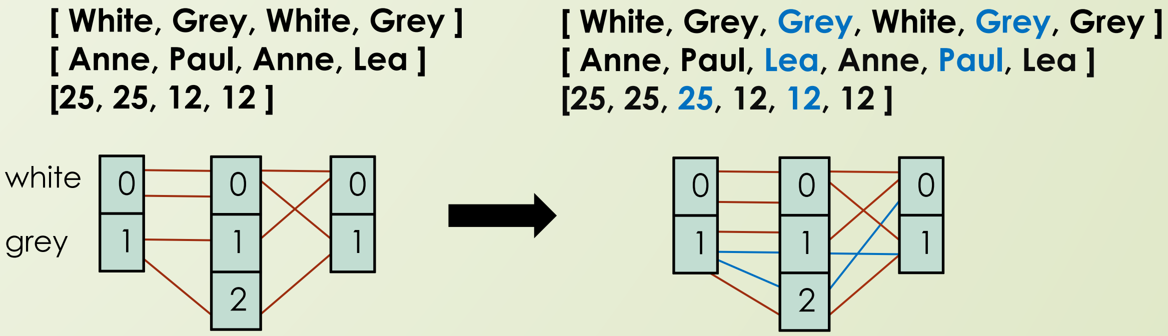
crossed

**reduction**

**extension**

- **Propagation**
  - Values reduction / extension can be propagated to derived ou coupled indexes

**Extension can't be propagated to crossed or linked Indexes.**



**[ White, Grey, White, Grey ]**
**[ Anne, Paul, Anne, Lea ]**
**[25, 25, 12, 12 ]**

**[ White, Grey, Grey, White, Grey, Grey ]**
**[ Anne, Paul, Lea, Anne, Paul, Lea ]**
**[25, 25, 25, 12, 12, 12 ]**

# 1 – Representation

- **Codec representation**
  - List of values (or dict key/value)
  - List of unique value + list of keys
- **Keys representation**
  - Absolute : List of integer (index of codec value)
  - Relative : List of integer (index of other keys)
  - Implicit : Automatic list (i.e. with full codec)
- **Iindex /variable Formats**
  - Simple format (codec)
  - Complete format (codec + keys)
  - Coupled format (codec + parent)
  - Derived format (codec + parent + keys)
    - Keys = index of parent keys

## Json Example

**[ ‘Anne’, ‘Anne’, ‘John’, ‘Paul’ ]**

**[ [ ‘Anne’, ‘John’, ‘Paul’], [0, 0, 1, 2 ] ]**

**[ 2, 2, 3 ]** → **[‘John’, ‘John’, ‘Paul’ ]**

**[ 0, 1, 2, 3 ]** ← **if full codec**

**[ ‘Anne’, ‘Anne’, ‘John’, ‘Paul’, ‘John’]** *(full)*

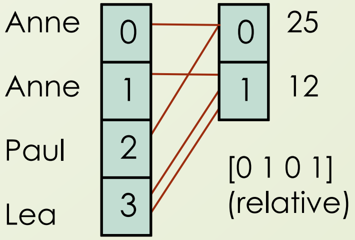
**[ ‘Anne’, ‘John’, ‘Paul’]** *(default)*

**[ [ ‘Anne’, ‘John’, ‘Paul’], [0, 0, 1, 2, 1 ] ]**

**[ [ ‘Anne’, ‘John’, ‘Paul’], parent ]**

**[ [ ‘Anne’, ‘John’, ‘Paul’], parent, [0, 0, 1, 2, 1 ] ]**

[ Anne,Anne, Lea, Paul, Lea ]
[ 12, 25, 12, 25, 12 ]



Derived Iindex : [ [ 25, 12 ], parent, [0, 1, 0, 1] ] *(derived)*
Parent index : [ Anne, Anne, Paul, Lea ], [0, 1, 3, 2, 3]] *(complete)*

*or*

Parent index : [ [Anne, Paul, Lea ], [0,0,1,2] ], [0, 1, 3, 2, 3]] *(complete)*

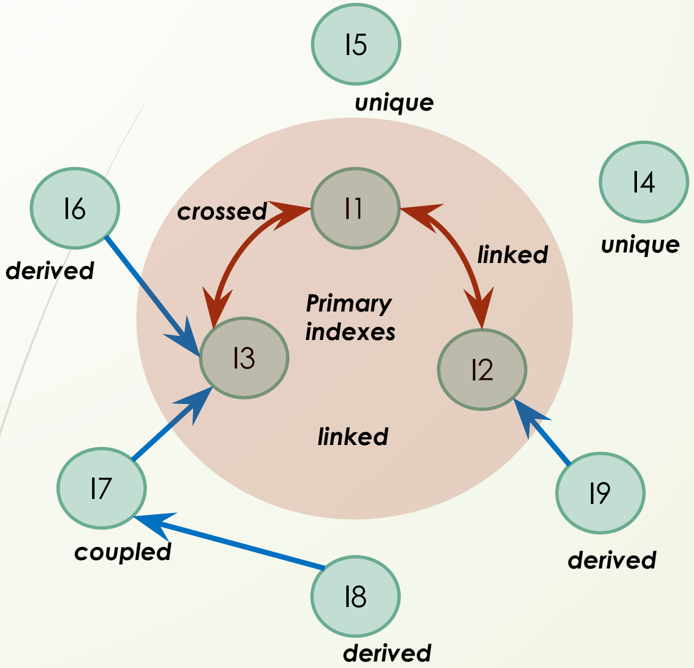
# 2 – IndexSet (list of indexes with same length)

- **Index definition**
  - An index is **secondary** if it's derived or coupled from at least one other index
  - An Index is **primary** if it's not secondary
  - If the index is secondary, the **parent** index is the first index with the lowest disttomin in the list of coupling or derivating indexes
  - If the index is primary, the **parent** index is the first index with the lowest disttomin in the list of primary indexes (or itself if the index is the first crossed primary)
  - The **precursor** index is the first Primary index in the indexing tree

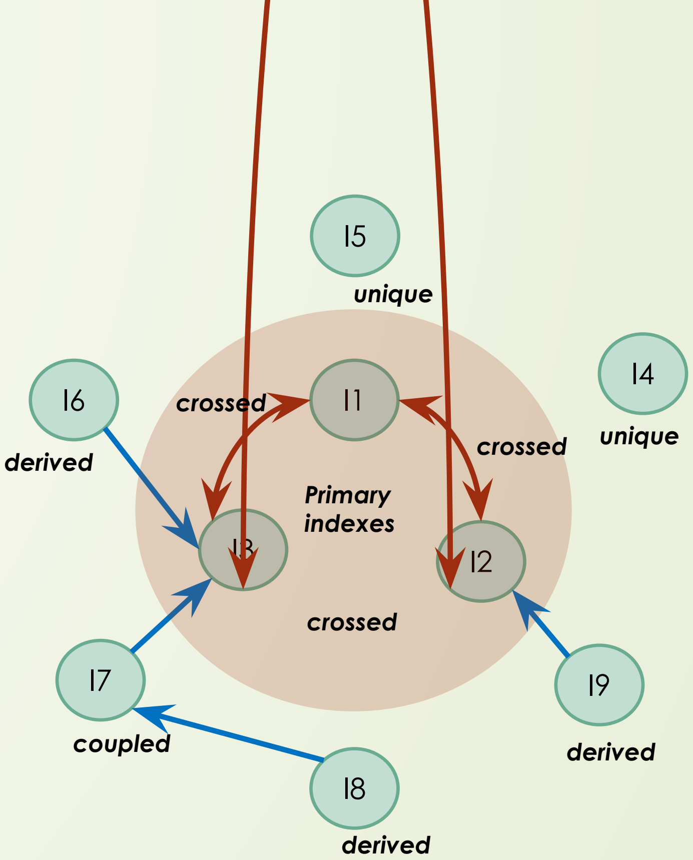
- **IndexSet definition**
  - **Dimension** : number of primary indexes
  - **Complete** : An indexSet is complete if all the primary indexes are crossed with each other primary index

- **Properties**
  - **The number of values of a full indexset is the product of the primary indexes length**
  - **A complete IndexSet can be transformed in a Matrix with the dimension of the indexset**
  - **Keys data is unnecessary in a complete indexset whithout derived codec**
  - **Dimension can be reduced by codec extension**
  - **Dimension can be increased by values extension**

# 2 – Structure



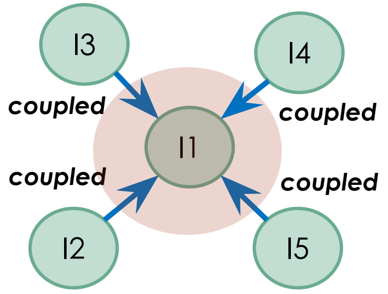
Canonical structure (default codec)



Complete structure (adjusted codec and values)

**In a complete format, Keys are:**

- **Implicit for Primary, Unique and Coupled indexes**
- **Relative for Derived indexes**



**CSV format**

- **Properties**
  - Each indexset has a **canonical structure** (at least one primary index)
  - **Complete data** is obtained by crossing all the primary indexes (values extension)
  - Complete indexset can be transformed in **Matrix** (full codec for secondary indexes)
  - **CSV format** is a canonical structure with one primary index and any coupled indexes, all indexes have full codec

# 2 - Example

### 3 columns are linked

- Full name
- Course
- Examen

### 3 columns are derived

- First name
- Last name
- Group

### 1 column is coupled

- Surname

### 1 column is unique

- Year

### ratio

- Name – Course : 37,5 %
- Name – Examen : 62,5 %
- Course – Examen : 83,7 %

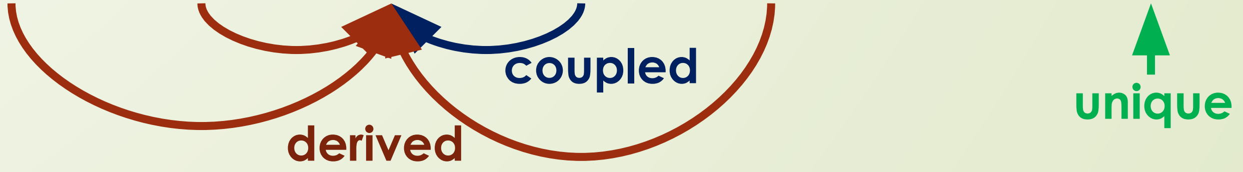
IndexSet

**37% almost derived or coupled**

**83% almost crossed**

Data

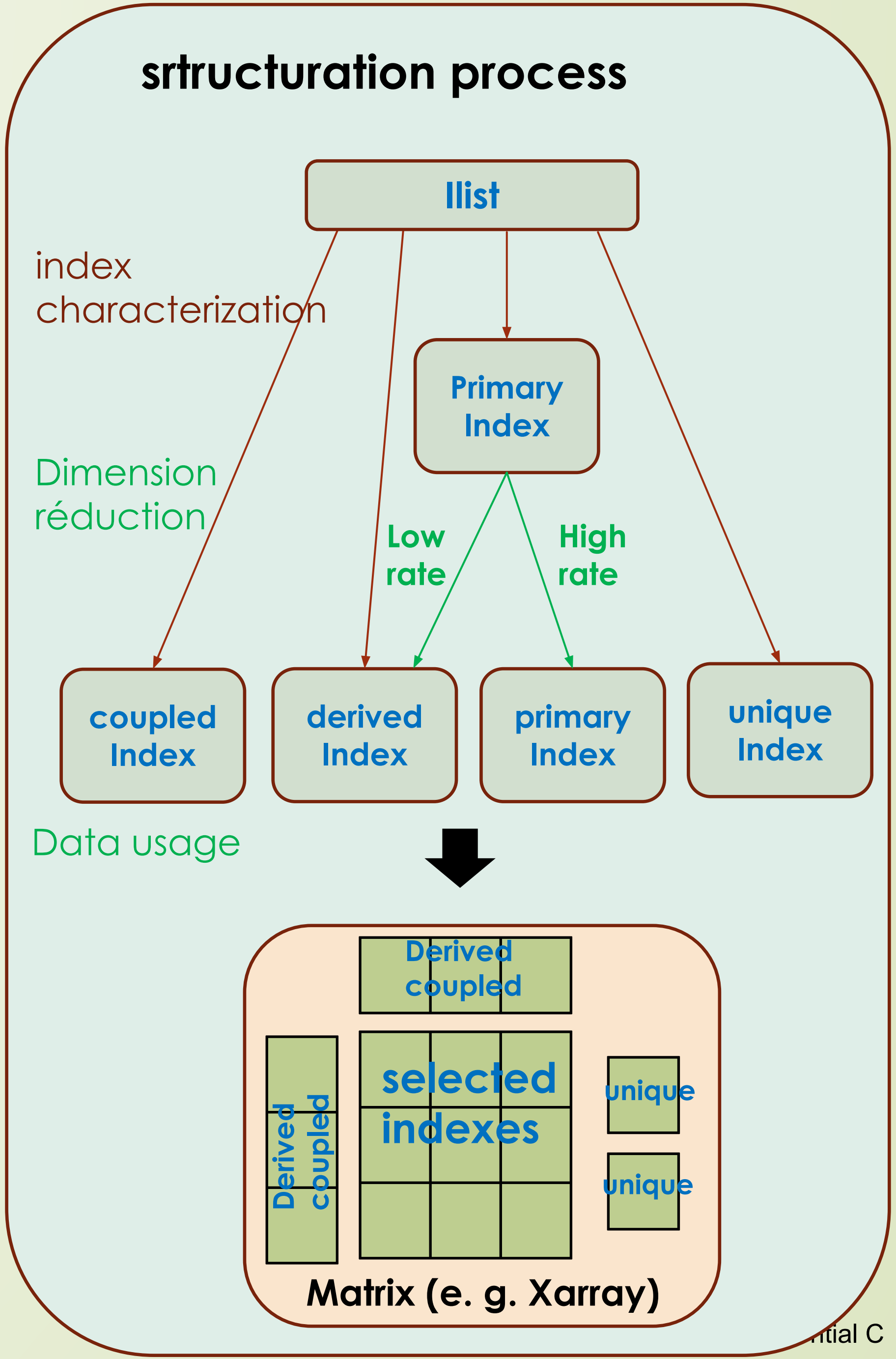
| first name | last name | full name | surname | group | course | year | examen | score |
|---|---|---|---|---|---|---|---|---|
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t1 | 11 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t2 | 13 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t3 | 15 |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t2 | 10 |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t3 | 12 |
| Philippe | White | Philippe White | heisenberg | gr2 | math | 2021 | t1 | 15 |
| Philippe | White | Philippe White | heisenberg | gr2 | english | 2021 | t2 | 8 |
| Camille | Red | Camille Red | saul | gr3 | software | 2021 | t3 | 17 |
| Camille | Red | Camille Red | saul | gr3 | software | 2021 | t2 | 18 |
| Camille | Red | Camille Red | saul | gr3 | english | 2021 | t1 | 2 |
| Camille | Red | Camille Red | saul | gr3 | english | 2021 | t2 | 4 |
| Philippe | Black | Philippe Black | gus | gr3 | software | 2021 | t3 | 18 |
| Philippe | Black | Philippe Black | gus | gr3 | english | 2021 | t1 | 6 |



# 2 – Structuration process

- **Objectives**
  - Data understanding
  - Unconsistent data identification
  - Size reduction
  - Tranfer to analysis tools (e.g. Pandas, Xarray)

- **Analysis**
  - **Index characterization**
    - Identification of primary indexes
    - Association of secondary indexes to primary indexes
  - **Linked indexes analysis**
    - Low rate (i.e. < 0,1) = almost derived index
      -> transform to derived index (codec extension)
      -> or values correction
    - High rate (i.e. > 0,9) = almost crossed index
      -> transform to crossed index
      -> or values correction
- **Data usage**
  - **Dimension reduction (if necessary)**
    - Primary index merging (rather low rate)
  - **Export**
    - Matrix generation
    - Storage



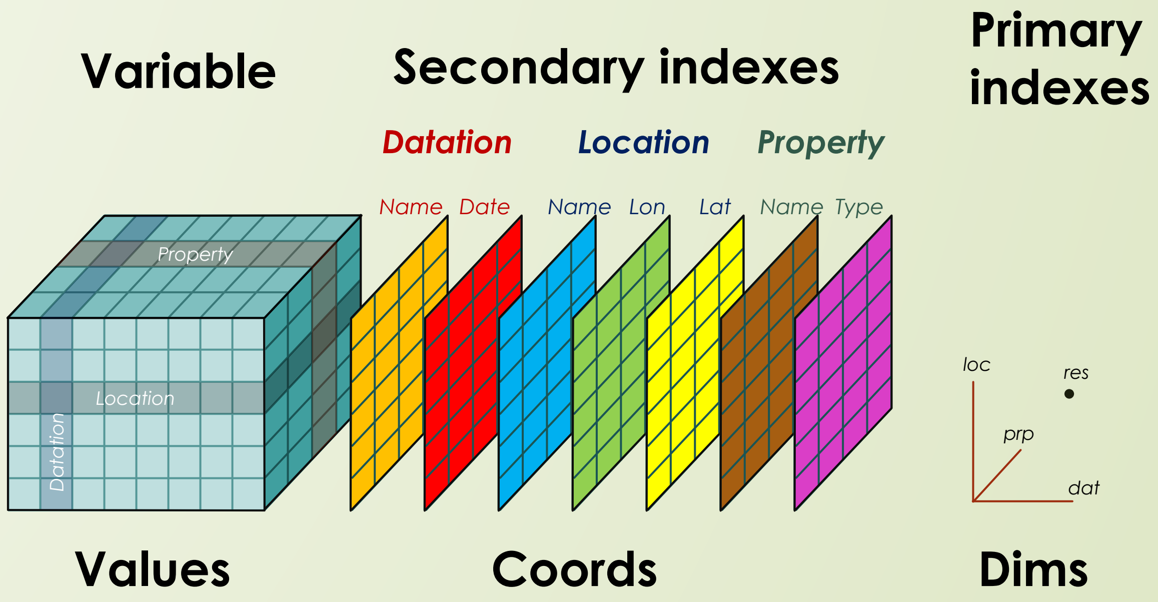
# Example : Xarray – mapping

## > Xarray

- Values : data matrix(ex. numpy ndarray)
- Coords : list of indexes: (dims, data, attrs)
- Dims : names of dimensions
- Attrs : attribut dictionnary (data or coord)
- Name

## > Ilist Mapping

- Dims : Primary indexes
- Values : Variable values
- Coords : Secondary indexes
- Attrs : Unique indexes
- Name : Ilist name



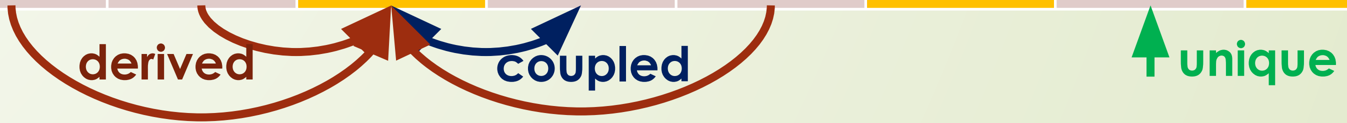
# 2 - Example

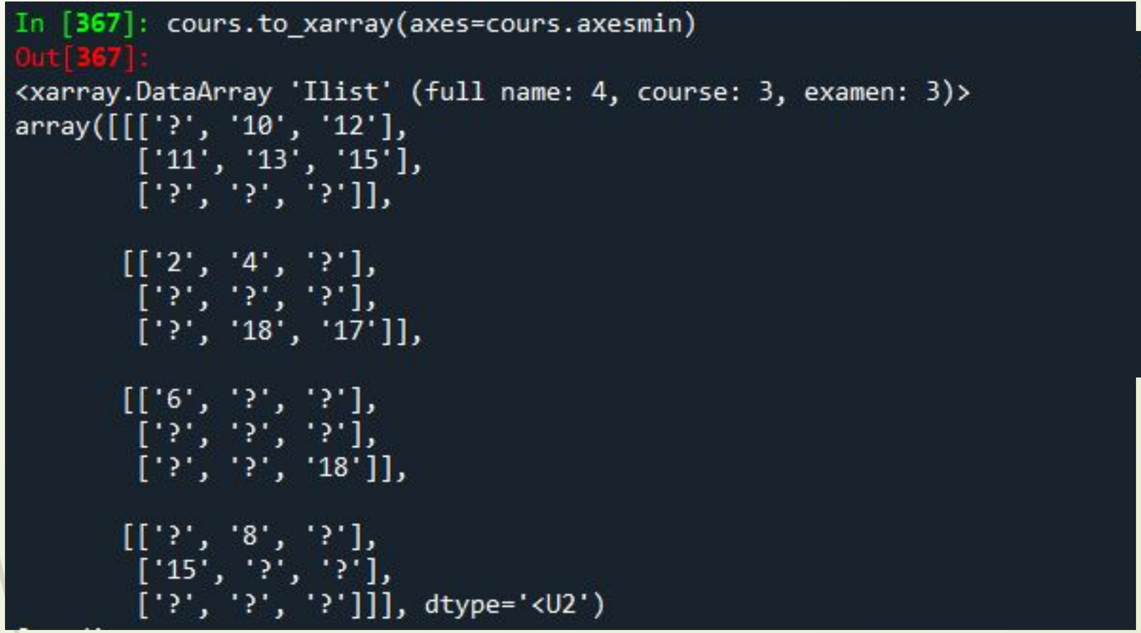
*to_xarray* function :

- Primary crossed (values extension)
- Secondary coupled (full codec)

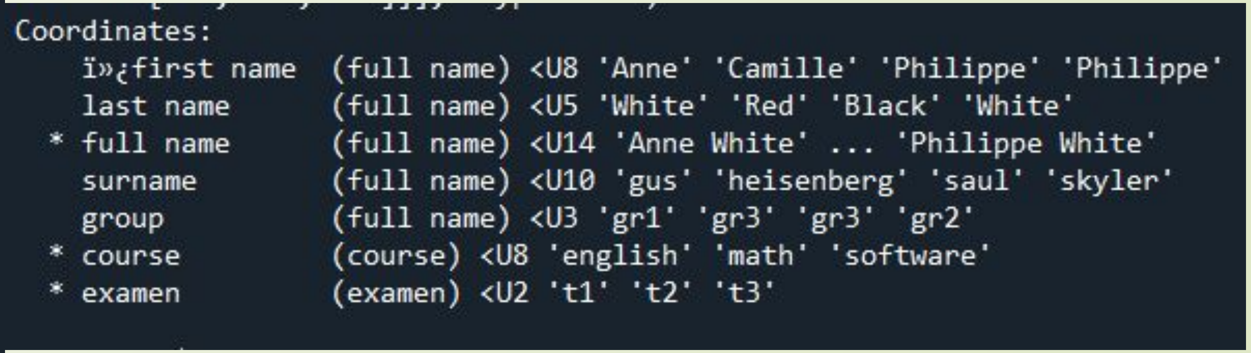
| first name | last name | full name | surname | group | course | year | examen | score |
|---|---|---|---|---|---|---|---|---|
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t1 | - |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t2 | 10 |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t3 | 12 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t1 | 11 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t2 | 13 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t3 | 15 |
| ***Anne*** | ***White*** | ***Anne White*** | ***skyler*** | ***gr1*** | ***software*** | ***2021*** | ***t1*** | **-** |
| ***Anne*** | ***White*** | ***Anne White*** | ***skyler*** | ***gr1*** | ***software*** | ***2021*** | ***t2*** | **-** |
| ***Anne*** | ***White*** | ***Anne White*** | ***skyler*** | ***gr1*** | ***software*** | ***2021*** | ***t3*** | **-** |

completed



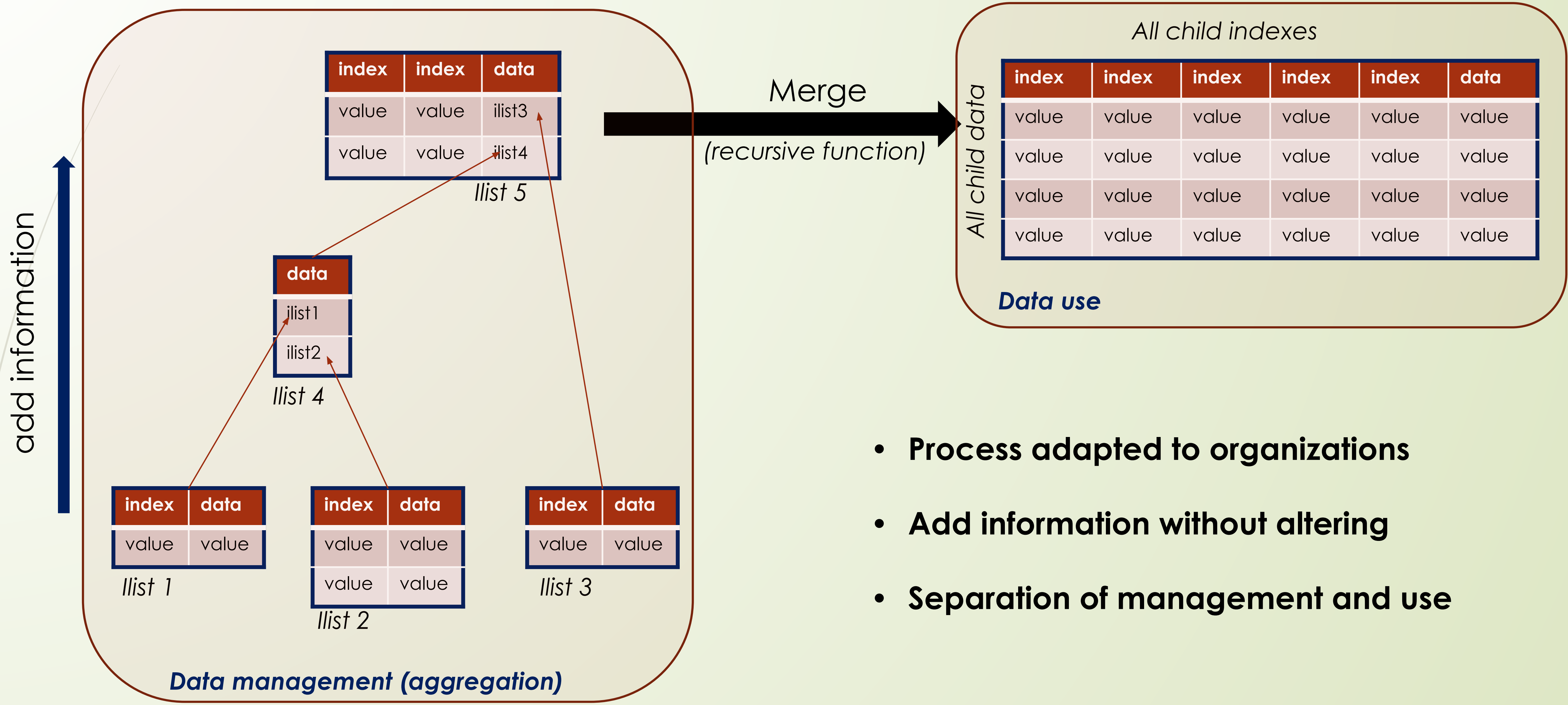


```
In [367]: cours.to_xarray(axes=cours.axesmin)
Out[367]:
<xarray.DataArray 'Ilist' (full name: 4, course: 3, examen: 3)>
array([[['?', '10', '12'],
        ['11', '13', '15'],
        ['?', '?', '?']],

       [['2', '4', '?'],
        ['?', '?', '?'],
        ['?', '18', '17']],

       [['6', '?', '?'],
        ['?', '?', '?'],
        ['?', '?', '18']],

       [['?', '8', '?'],
        ['15', '?', '?'],
        ['?', '?', '?']]], dtype='<U2')
```



```
Coordinates:
    ï»¿first name   (full name) <U8 'Anne' 'Camille' 'Philippe' 'Philippe'
    last name      (full name) <U5 'White' 'Red' 'Black' 'White'
  * full name      (full name) <U14 'Anne White' ... 'Philippe White'
    surname        (full name) <U10 'gus' 'heisenberg' 'saul' 'skyler'
    group          (full name) <U3 'gr1' 'gr3' 'gr3' 'gr2'
  * course         (course) <U8 'english' 'math' 'software'
  * examen         (examen) <U2 't1' 't2' 't3'
```

# 3 - Building process



llist 1

| index | data |
|---|---|
| value | value |

llist 2

| index | data |
|---|---|
| value | value |
| value | value |

llist 3

| index | data |
|---|---|
| value | value |

llist 4

| data |
|---|
| ilist1 |
| ilist2 |

llist 5

| index | index | data |
|---|---|---|
| value | value | ilist3 |
| value | value | ilist4 |

All child indexes

| index | index | index | index | index | data |
|---|---|---|---|---|---|
| value | value | value | value | value | value |
| value | value | value | value | value | value |
| value | value | value | value | value | value |
| value | value | value | value | value | value |

- **Process adapted to organizations**
- **Add information without altering**
- **Separation of management and use**

# 3 - Example

IndexSet | Data

aw

| course | year | examen | score |
|---|---|---|---|
| math | 2021 | t1 | 11 |
| math | 2021 | t2 | 13 |
| math | 2021 | t3 | 15 |
| english | 2021 | t2 | 10 |
| english | 2021 | t3 | 12 |

pw

| course | year | examen | score |
|---|---|---|---|
| math | 2021 | t1 | 15 |
| english | 2021 | t2 | 8 |

cr

| course | year | examen | score |
|---|---|---|---|
| software | 2021 | t3 | 17 |
| software | 2021 | t2 | 18 |
| english | 2021 | t1 | 2 |
| english | 2021 | t2 | 4 |

pb

| course | year | examen | score |
|---|---|---|---|
| software | 2021 | t3 | 18 |
| english | 2021 | t1 | 6 |

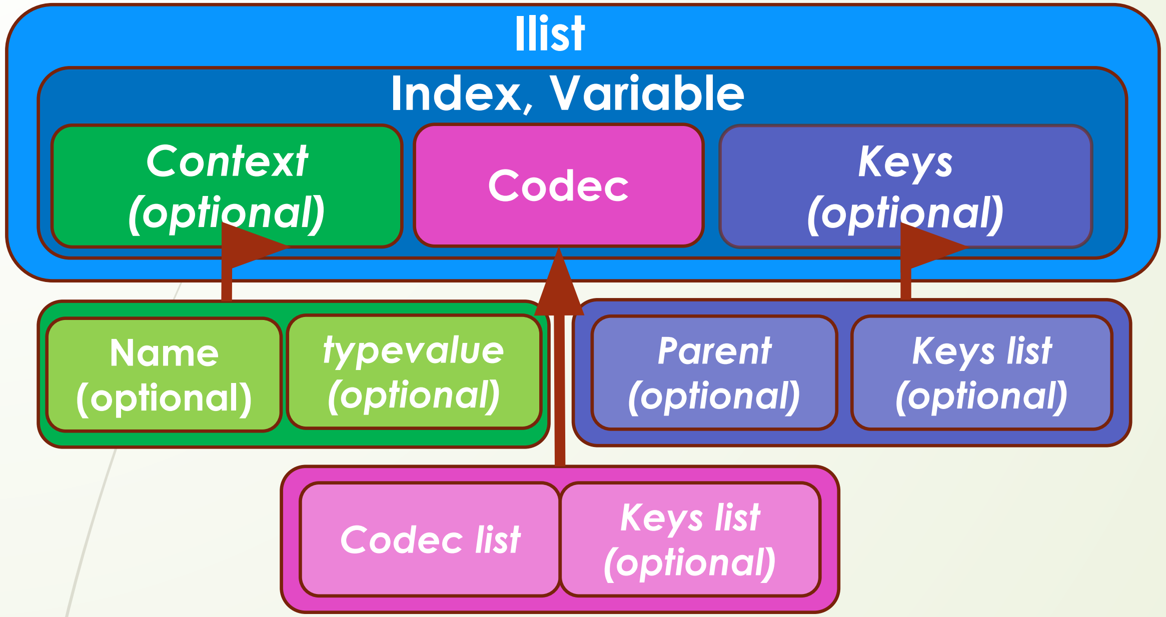
total

| first name | last name | full name | surname | group | file |
|---|---|---|---|---|---|
| Anne | White | Anne White | skyler | gr1 | aw |
| Philippe | White | Philippe White | heisenberg | gr2 | pw |
| Camille | Red | Camille Red | saul | gr3 | cr |
| Philippe | Black | Philippe Black | gus | gr3 | pb |

total.merge()

| first name | last name | full name | surname | group | course | year | examen | score |
|---|---|---|---|---|---|---|---|---|
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t1 | 11 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t2 | 13 |
| Anne | White | Anne White | skyler | gr1 | math | 2021 | t3 | 15 |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t2 | 10 |
| Anne | White | Anne White | skyler | gr1 | english | 2021 | t3 | 12 |
| Philippe | White | Philippe White | heisenberg | gr2 | math | 2021 | t1 | 15 |
| Philippe | White | Philippe White | heisenberg | gr2 | english | 2021 | t2 | 8 |
| Camille | Red | Camille Red | saul | gr3 | software | 2021 | t3 | 17 |
| Camille | Red | Camille Red | saul | gr3 | software | 2021 | t2 | 18 |
| Camille | Red | Camille Red | saul | gr3 | english | 2021 | t1 | 2 |
| Camille | Red | Camille Red | saul | gr3 | english | 2021 | t2 | 4 |
| Philippe | Black | Philippe Black | gus | gr3 | software | 2021 | t3 | 18 |
| Philippe | Black | Philippe Black | gus | gr3 | english | 2021 | t1 | 6 |

# 4 – JSON Representation



Ilist

Index, Variable

*Context (optional)*

Codec

*Keys (optional)*

Name (optional)

*typevalue (optional)*

*Parent (optional)*

*Keys list (optional)*

*Codec list*

*Keys list (optional)*

**Ilist:** JSON Array (or JSON value if only one value)
**Index:** JSON Array (or JSON value if only one value)

**Context:** JSON Object (or JSON string if only one value)
**Codec:** JSON Array (or JSON value if only one value)
**Keys:** JSON Array (or JSON value if only one value)

**Name, Typevalue:** string
**Codec list:** JSON Array
**keys list:** JSON Array
**Parent:** Integer

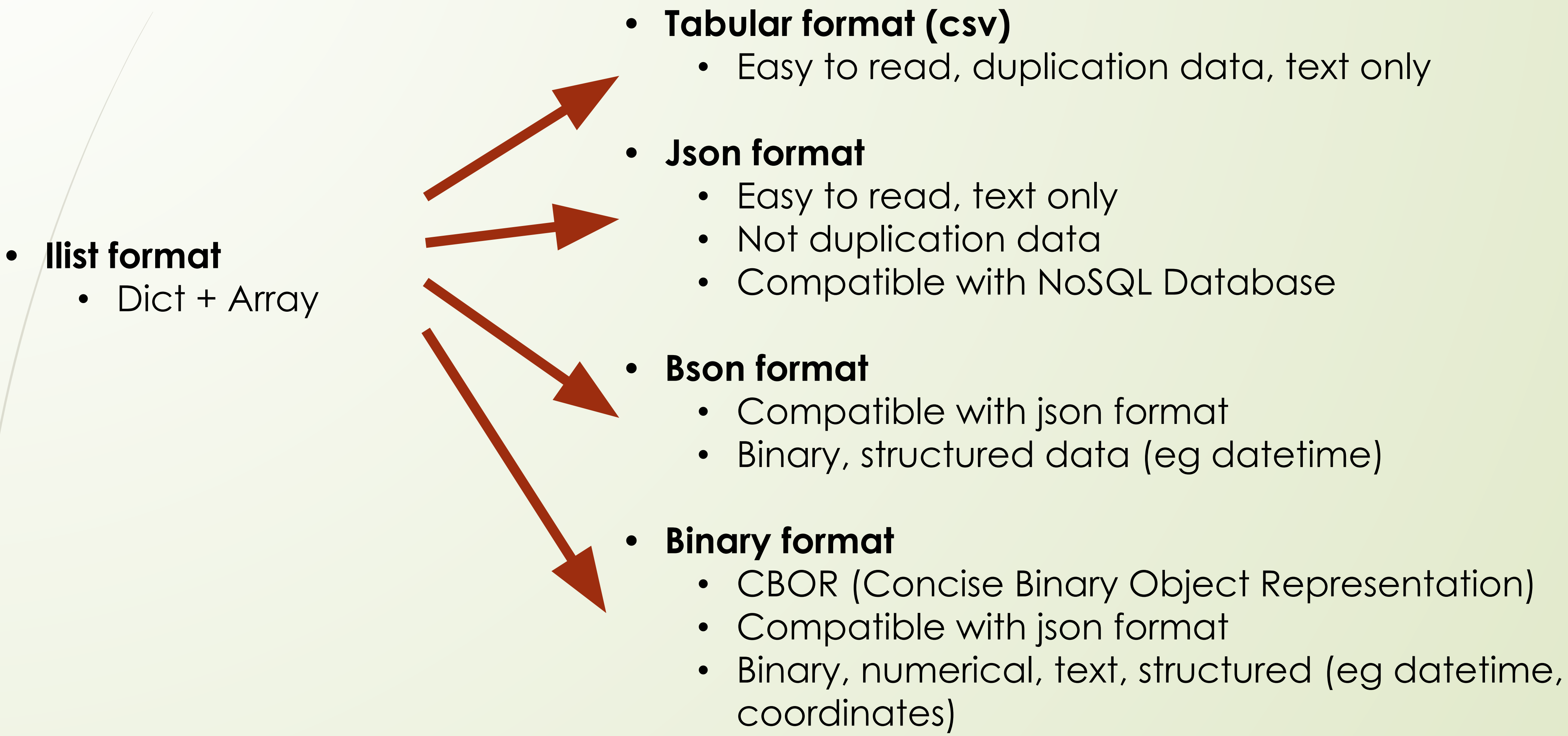
**Format**

**Text (JSON text), Binary (CBOR)**

**Example Index : Name : ‘team1’**
**Values : [ ‘Anne’, ‘Anne’, ‘John’, ‘Paul’, ‘John’]**

- **Simple format (without name)**
  **[‘Anne’, ‘Anne’ ‘John’, ‘Paul’, ‘John’]**
  -> Full codec (e.g. csv format)
- **Simple format (with name)**
  **[‘team1’, [‘Anne’, ‘John’, ‘Paul’]]**
  -> Default codec (e.g. crossed index)
- **Complete format (with name)**
  **[‘team1’, [‘Anne’, ‘John’, ‘Paul’], [0,0,1,2,1]]**
  -> Default codec, name, absolute keys
- **Coupled format (with name)**
  **[‘team1’, [‘Anne’, ‘John’, ‘Paul’, ‘John’], 2 ]**
  -> Adjusted codec, parent id
- **Derived format (with name)**
  **[‘team1’, [‘Anne’, ‘John’, ‘Paul’], [2, [0,1,2,1]]]**
  -> Default codec, parent id, relative keys
- **Unique format**
  **[‘team1’, [‘Anne’]] (with name) [‘Anne’] (without name)**
  -> Default codec (= full codec)

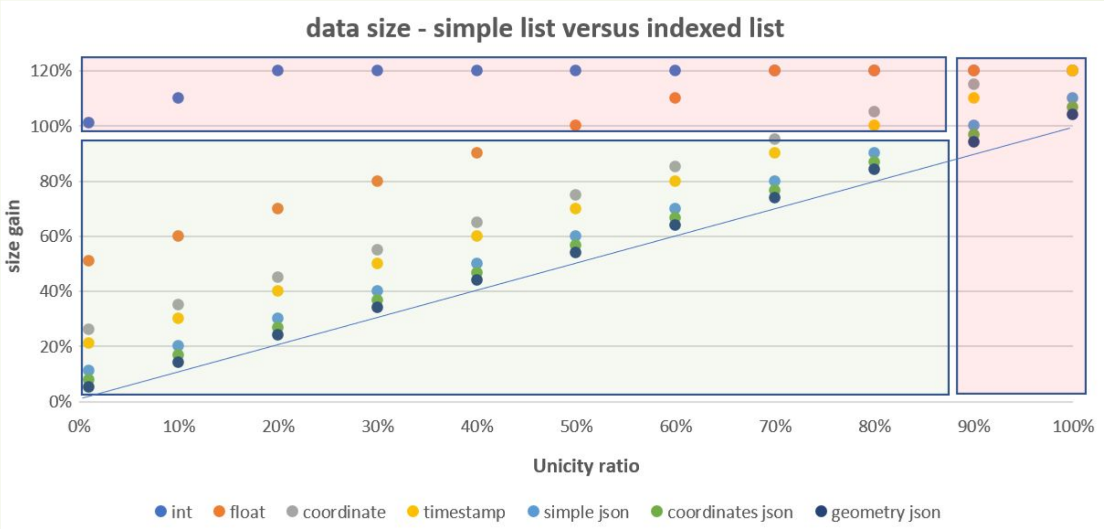
# 4 – format



- **Ilist format**
  - Dict + Array

- **Tabular format (csv)**
  - Easy to read, duplication data, text only
- **Json format**
  - Easy to read, text only
  - Not duplication data
  - Compatible with NoSQL Database
- **Bson format**
  - Compatible with json format
  - Binary, structured data (eg datetime)
- **Binary format**
  - CBOR (Concise Binary Object Representation)
  - Compatible with json format
  - Binary, numerical, text, structured (eg datetime, coordinates)

# 4 – Ilist size and indicators



- **Simple list size = nv * sv**
  - nv : number of values
  - sv : mean value size = sizesimple / nv
- **Indexed list size = (nv – nc) * sc + nc * sv**
  - nc : number of different values
  - sc: mean coding size = (size – nc * sv) / (nv – nc)
- **Gain = (Simple size - indexed size) / simple size = (1 – ul) * (1 – ol)**
  - **OL = sc / sv (object lightness) [0, 1] (data complexity)**
  - **UL = nc / nv (unicity level) [0, 1] (data quality)**
- **Properties**
  - If object lightness and unicity level are low, the indexed list size is lower than simple list size
    - e.g. : OL = 0.1 , UL = 0.2 => **Gain = 72 %**
- **In a Ilist with data more complex than numerical data, the json (or binary) format has a smaller size than a tabular format**

| Object lightness | l | OL |
|---|---|---|
| int | 2 | 1,00 |
| float, int32 | 4 | 0,50 |
| coordinate | 8 | 0,25 |
| string(10) (eg. timestamp) | 10 | 0,20 |
| simple json element (eg key/value) | 20 | 0,10 |
| structured json element (eg coordinates) | 30 | 0,07 |
| complex json element (eg geometry) | 50 | 0,04 |

**E.g. previous example :**

- csv : 2 418 bytes
- json : 1 496 bytes
- binary (CBOR) : 697 bytes

# structure

name :

- $xxx: simple (int, obj, str, tuple, bool, list-> tuple, dict-> str)
- xxx: ESValue (from_obj)

lindex:

- default name = $xxx
- default typevalue : NamedValue

from_obj:

- decodevalue -> class, name, val
- decodeval -> classval

decodeval:

- int, bool: NamedValue
- dict : PropertyValue
- list:
  - extraire 1e val
  - float, int : LocationValue
  - date : DatationValue
  - sinon: test Datation sinon Named