ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declarative Amsterdam

Adding Scripting with side-effects to RumbleDB

Developed by David-Marian Buzatu

in collaboration with

Prof. Gustavo Alonso, Dr. Ghislain Fourny

DINFK

Systems@ETH Zürich

Contents

- Introduction and background
- Motivation
- XQSX scripting implementation
- Numpy and pandas
- Experiments
- Future work
- Q&A

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

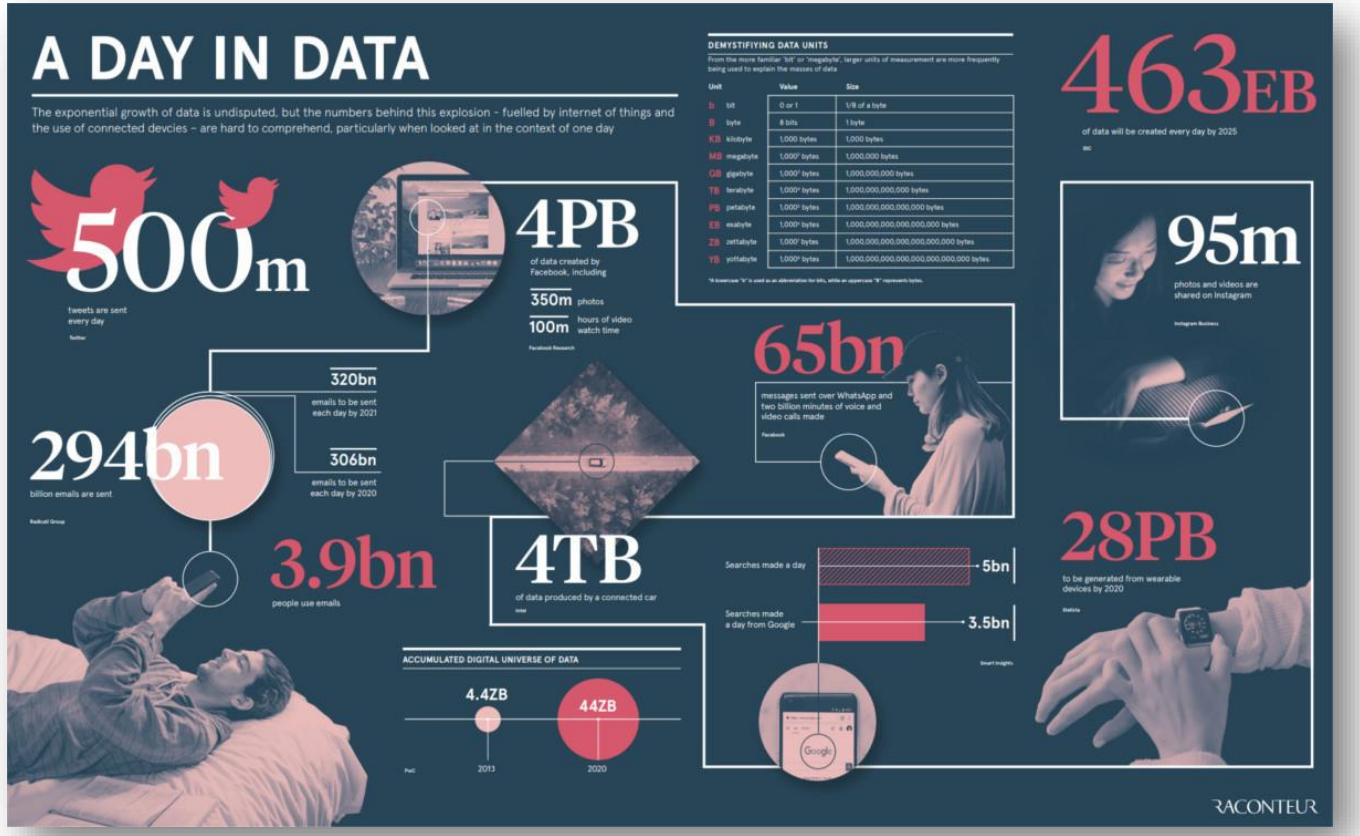
D INFK

Systems@ETH Zürich

Introduction and background

Today's world generates Exabytes of data **every day**!

- That is 10^{18} bytes
- A single machine is not enough any more



*Courtesy of Raconteur

DINFK

Systems@ETH Zürich

How can this be standardized?

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DESCRIBE DATA IN TERMS OF *VOLUME*, *VELOCITY* AND *VARIETY*

CATEGORIZE DATA AS *STRUCTURED*, *UNSTRUCTURED* OR *SEMI-STRUCTURED*

Data representation

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Tables

Structured data

Text, image, video files

Unstructured data

Tree structures – **JSON** and **XML**

Semi-structured data

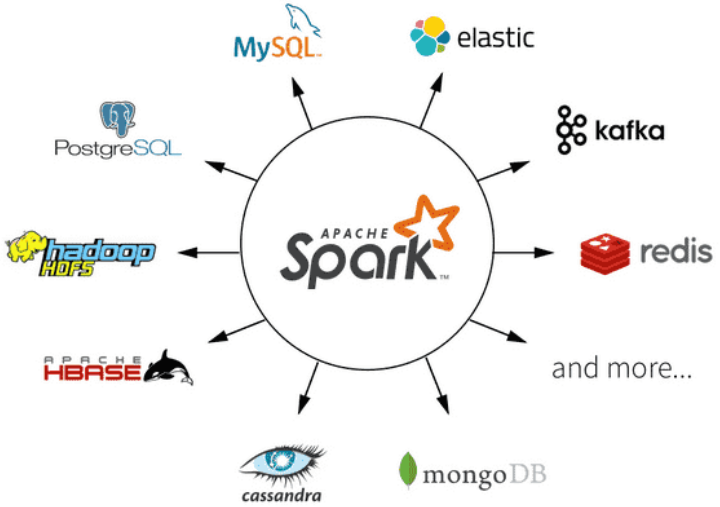
D INFK

Systems@ETH Zürich

Data analysis

- With data on multiple machines, tools need to:
  - Interact with data remotely
  - Ease interaction
  - Handle results in chunks
  - Work with different storage systems

  …
- Apache Spark resolves these issues
  - But is not **abstract** enough!



JSONiq

- A query language for JSON type data
- SQL-like interaction using a declarative language

```
for $access-entry in $access-logs
group by $url := $access-entry.url
return
{
  "url": $url,
  "response_time": avg($access.response_time)
}
```

Example JSON

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

RumbleDB

- Query engine for large, heterogeneous, and nested collections of JSON data
- Automatically select how to perform queries:
  - Locally
  - Distributed
- Uses JSONiq to ease interaction
  - Users write queries
  - RumbleDB decides how to run the query
- Handles terabyte range of data

```
for $x in (1, 2, 2, "1", "1", "2", true, null)
group by $y := $x
return {"key": $y, "content": [$x]}
```

RumbleDB invocation relying on Apache Spark

DINFK

Systems@ETH Zürich

Motivation

- Extend the capabilities of RumbleDB beyond querying
  - Allowing data to be stored
  - Allowing multiple changes to data
  - Allowing complex algorithms (sorting)
  - Allowing external interactions
  - Allowing imperative uses of JSONiq
- Enrich analysis capabilities
  - For RDDs
  - For DataFrames

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scripting and side-effects

Follows the specification provided in XQuery Scripting Extension 1.0

DINFK

Systems@ETH Zürich

Scripting and side-effects

Follows the specification provided in XQuery Scripting Extension 1.0

Expressions become:

Sequential (side-effecting)

Non-sequential (non-side-effecting)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scripting and side-effects

Follows the specification provided in XQuery Scripting Extension 1.0

Expressions become:

Sequential (side-effecting)

Non-sequential (non-side-effecting)

New instructions are introduced

While loops

Break, continue and exit statements

Local variable declaration

```
variable $a as xs:integer := 0;
variable $b as xs:integer := 1;
variable $c as xs:integer := $a + $b;
variable $fibseq as xs:integer* := ($a, $b);
while ($c < 100) {
        $fibseq := ($fibseq, $c);
        $a := $b;
        $b := $c;
        $c := $a + $b;
}
```

Example script for computing Fibonacci numbers smaller than 100

DINFK

Systems@ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Example usage - logs

```
declare function local:validate-and-log($username as xs:string) {
        variable $log as object := {};
        variable $user-doc as object := ...;
        variable $user-entry := {
                "access-attempt": fn:current-time(),
                "email_verified": false,
                "failed": false
        };
        variable $changed := false;
        if ($username eq $user-doc[[3]].name) then {
                replace json value of $user-entry.email_verified with true;
                replace json value of $user-entry.access-attempt with "2024-11-08T16:00:47.203Z";
                $changed := true;
        } else {
                replace json value of $user-entry.failed with true;
        }
        if ($changed eq true) then {
                insert json $user-entry into $log;
        } else {
                exit returning {"failure": true};
        }
        $log
};
local:validate-and-log("aaa@aaa.com")
```

DINFK

Systems@ETH Zürich

Example usage - validation

```
variable $usernames := ["test@test.com", "test1@test.com", "aaa@aaa.com", "a@a.com", "test9999@test.com"];
variable $data := ...
variable $counter := 1;
variable $res := ();
while ($counter lt (size($usernames) + 1)) {
        variable $updated_entry := local:validate-and-return($data, $usernames[[$counter]]);
        if ($updated_entry["status"] eq "failure") then {
                exit returning "failure";
        } else $res := ($res, $updated_entry);
        $counter := $counter + 1;
}
$res
```

- Test data content and updates
- Mutability
- In the future: store updates to a datalake!

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scripting achievements

- Enable imperative coding
- Give more flexibility to users
- Allow for complex algorithms
- Backwards compatible
- Non-scripting code remains the same
- Support libraries such as numpy and pandas!

D INFK

Systems@ETH Zürich

Numpy and pandas

- Provide analytical methods for
  - Multi-dimensional arrays
  - DataFrames
- Optimized for performance and large amounts of data
- Used in especially *AI* and *ML* applications
- Only available in Python

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Numpy and pandas within RumbleDB

- Researched the most popular and used methods
- Adapted over 20 functions
- Documented supported features
- Handled optional arguments
- Tested against the Python version
- Used scripting

DINFK

Systems@ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Numpy binsearch used by *digitize*

```
declare %an:sequential function jsoniq_numpy:binsearch($arr as array, $searched_element) {
        variable $low := 1;
        variable $high := size($arr) + 1;
        while ($low lt $high) {
                variable $mid_index := integer($low + ($high - $low) div 2);
                if ($arr[[$mid_index]] eq $searched_element) then exit returning $mid_index;
                else if ($searched_element le $arr[[$mid_index]]) then $high := $mid_index;
                else $low := $mid_index + 1;
        }
        exit returning if ($low eq 1) then 0 else $low;
};
```

DINFK

Systems@ETH Zürich

Implemented methods

Numpy	Pandas
linspace/logspace	describe
arange	sample
random + random.uniform	isnull
mean/max/min/median	fillna
unique	dropna
…	

Experiments

Carried out using a local machine

M1 PRO chip

32 GB LPDDR5 RAM

1TB of SSD storage

No cache reliance

Datasets used:

GitHub Archive (and subsets)

Generated XML

Cold runs

Repeated 3 to 5 times for each data split

Results

RumbleDB performance

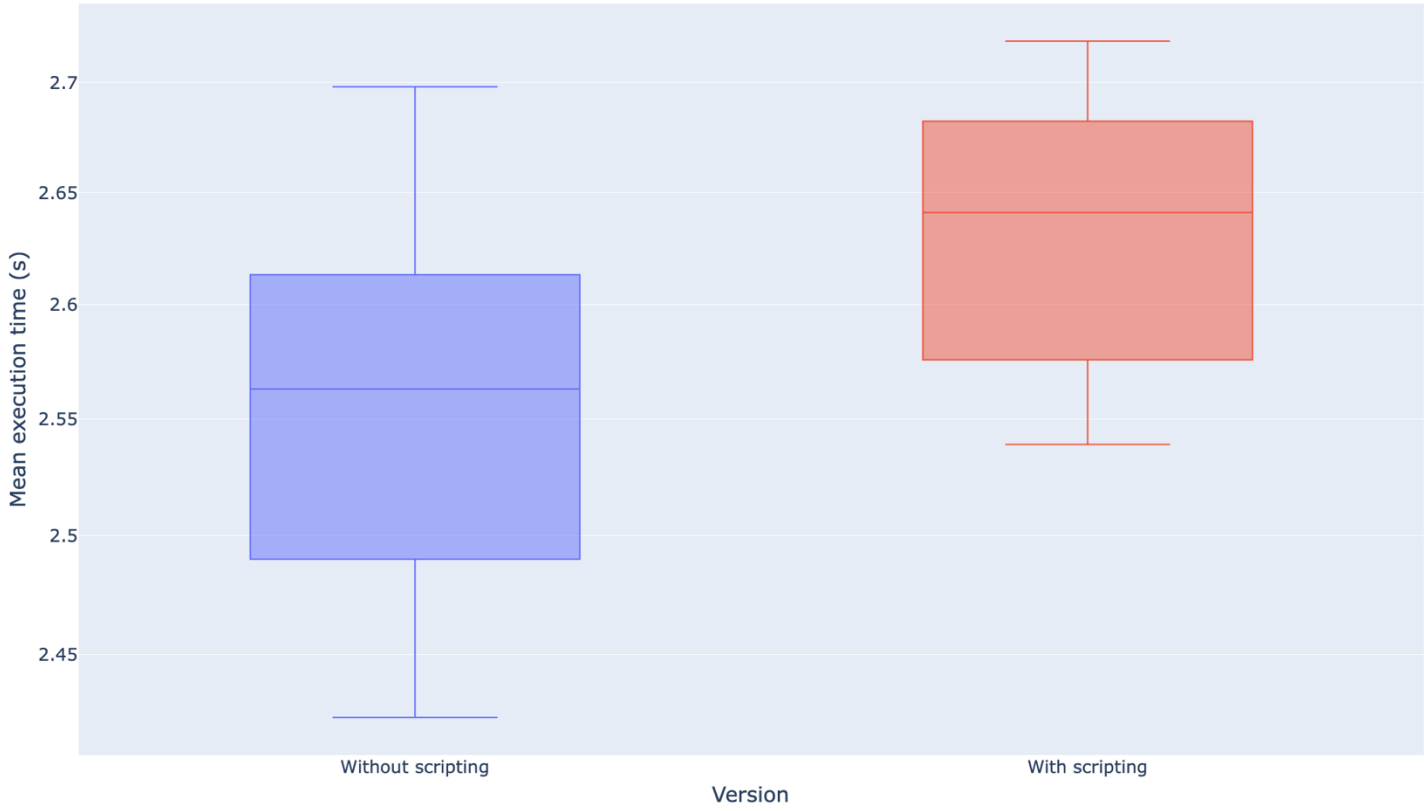
Query:

find all distinct logins among **each** event's actor **and** return their **count**

```
let $path := "..git-archive-big.json"
let $events := json-file($path)
let $actors := $events.actor
let $logins := $actors.login
let $distinct-logins := distinct-values($logins)
return count($distinct-logins)
```

Results

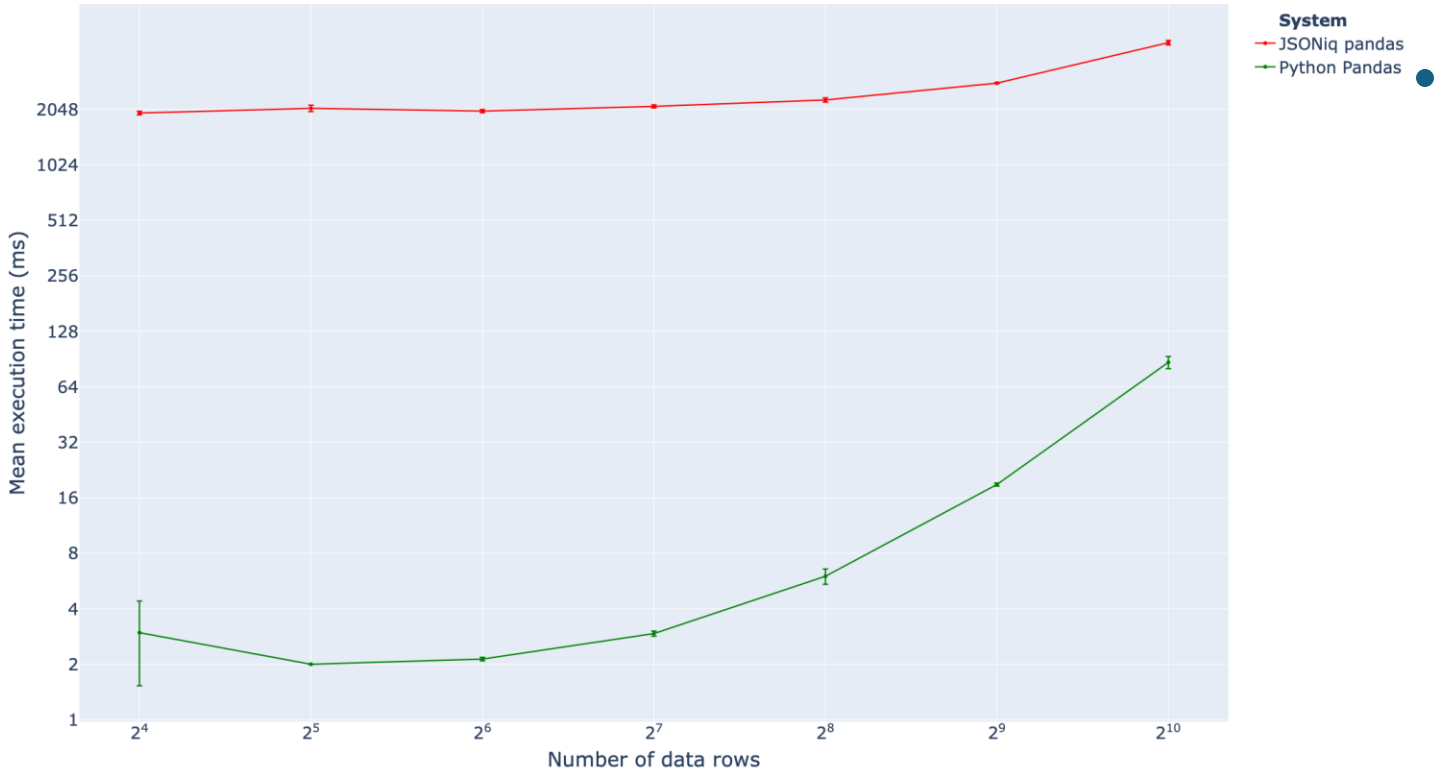
RumbleDB performance



Results

Pandas ‘describe’ performance

- Use ‘event’ data from GitHub Archive
- Calling describe on datasets of sizes of power of 2.



Future work

- Interaction with external systems
- Supporting more methods from numpy and pandas
- Supporting more optional arguments
- Improve performance

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Ready for your questions!

D INFK

Systems@ETH Zürich

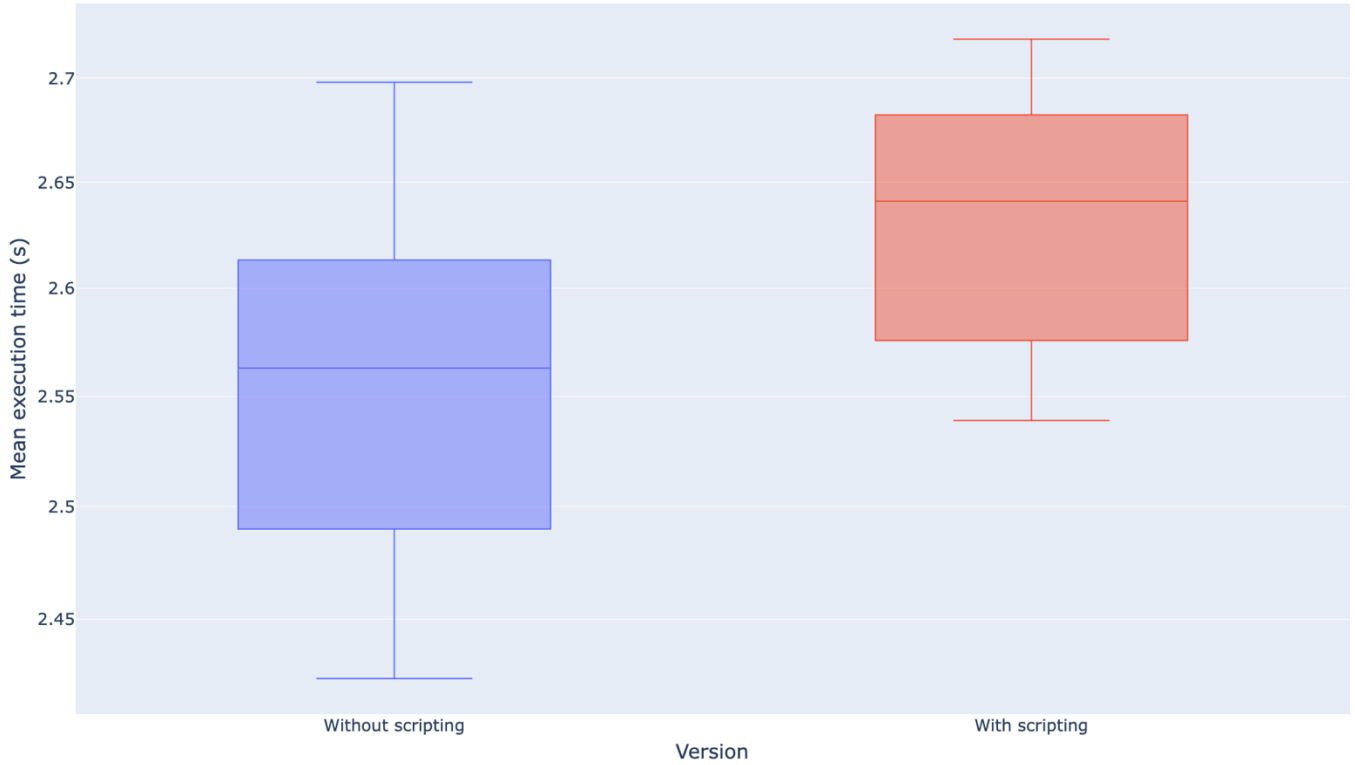
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Backup slides

DINFK

Systems@ETH Zürich

Results previous version



Case Study: Axis part 1

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

```
(: Method expects array to be of the same size. :)
declare function jsoniq_numpy:compute_min_values_of_arrays($array1, $array2) {
      typeswitch($array1)
      case array
            return
                  let $accumulated_sub_dimensions :=
                        for $i in 1 to size($array1)
                        return jsoniq_numpy:compute_min_values_of_arrays($array1[[$i]], $array2[[$i]])
                  return [$accumulated_sub_dimensions]
      Default
            return if ($array1 lt $array2) then $array1
                  else $array2
};
```

DINFK

Systems@ETH Zürich

Case Study: Axis part 2

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

```
if ($current_dimension eq $axis) then {
        (: Take the first array as minimum :)
        variable $mini := $array[[1]];
        variable $i := 2;
        while ($i le size($array)) {
                $mini := jsoniq_numpy:compute_min_values_of_arrays($mini, $array[[$i]]);
                $i := $i + 1;
        }
        exit returning $mini;
} else {
        let $accumulated_sub_dimensions :=
                let $size := size($array)
                for $i in 1 to $size
                return jsoniq_numpy:compute_min_along_axis($array[[$i]], $axis, $current_dimension + 1, $max_dim)
        return exit returning [$accumulated_sub_dimensions];
}
```

DINFK

Systems@ETH Zürich