Wisent: An In-Memory Serialization Format for Leafy Trees

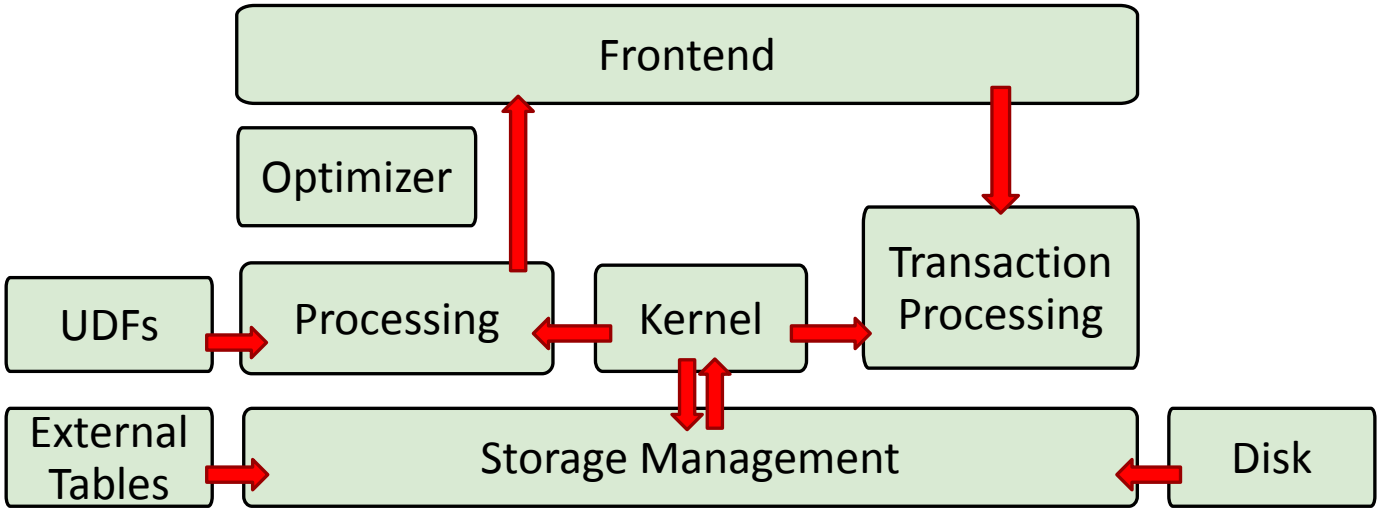
HUBERT MOHR-DAURAT

HOLGER PIRK

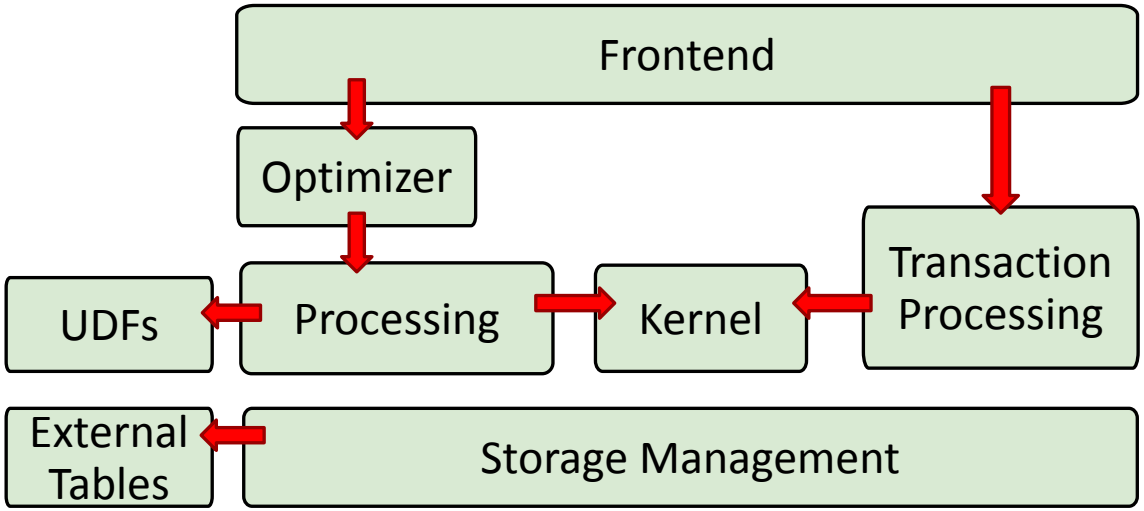
IMPERIAL COLLEGE LONDON

LARGE-SCALE DATA & SYSTEMS GROUP

Diverse Data Exchange in DBMS



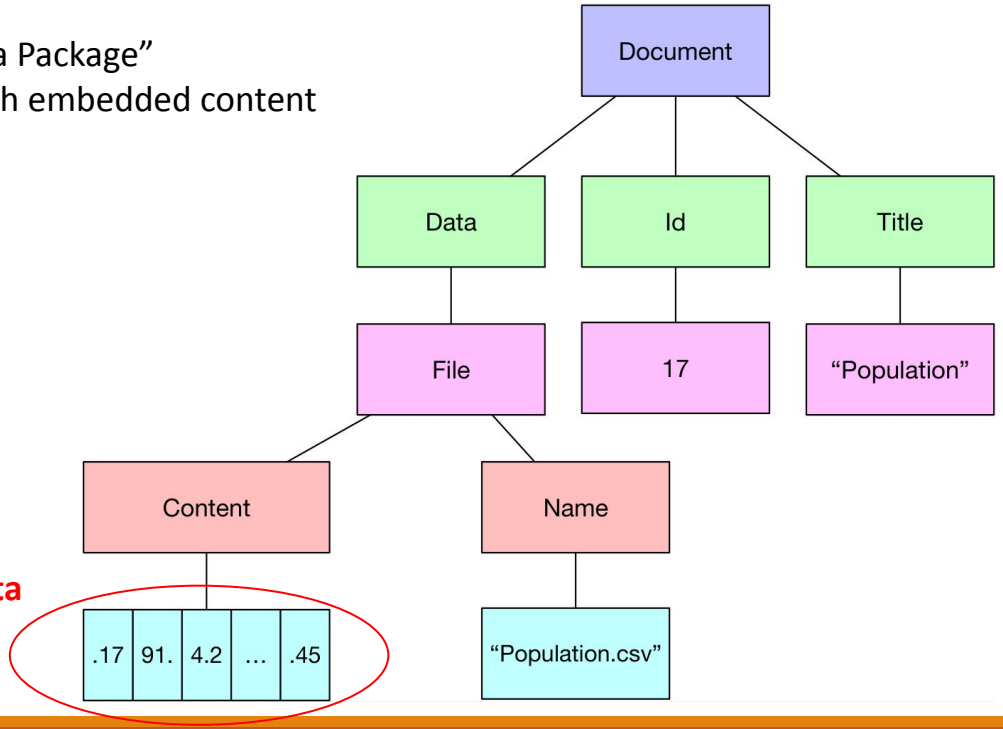
Diverse Code Exchange in DBMS



Could we have ***one*** exchange format to unify all this?

Performance issue with JSON/BSON

E.g., “Data Package”
With embedded content



Performance issue with JSON/BSON

```
{
  "Data": {
    "File" : {
      "Content": [
            0.17
            91.0
            4.20
            0.45
          ],
      "Name": "Population.csv"
    }
  },
  "Id": 17,
  "Title": "Population"
}
```

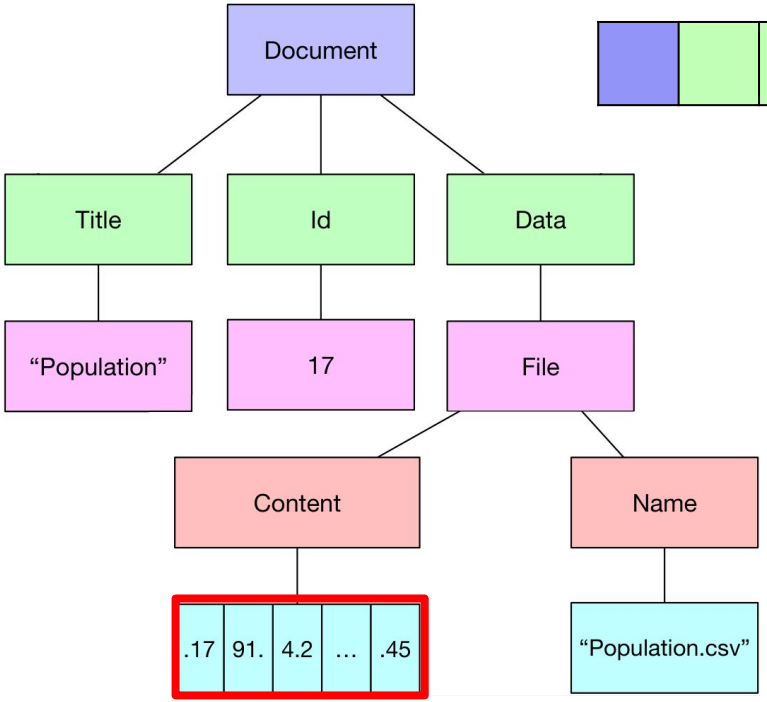
object start	name [1, \0]	value type	value [0.17]	end of object
object start	name [2, \0]	value type	value [91.0]	end of object
object start	name [3, \0]	value type	value [4.20]	end of object
object start	name [4, \0]	value type	value [0.45]	end of object

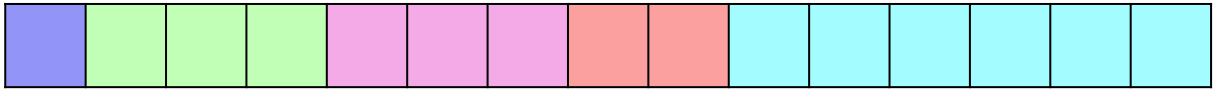
Depth-First Tree is the Wrong Default Choice

e.g., Table of Content

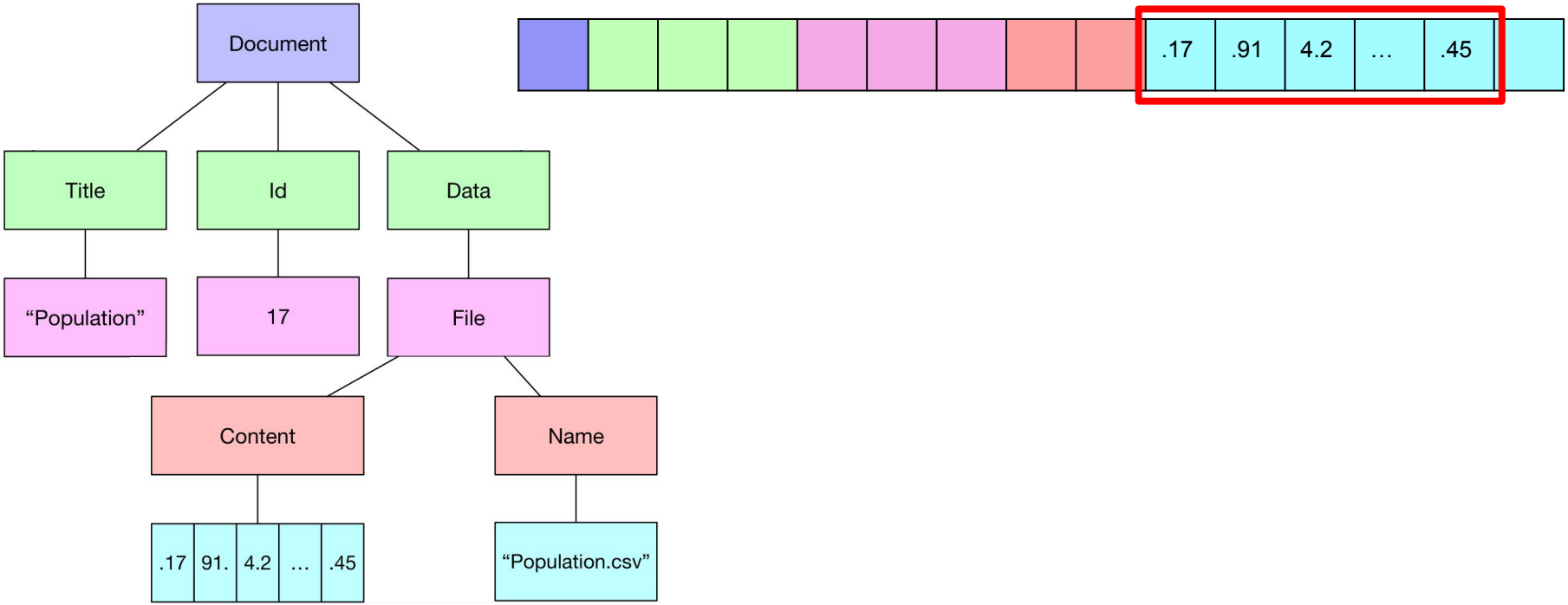
Our Approach

Breadth-First Representation





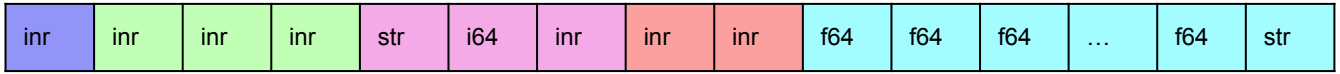
Breadth-First Representation



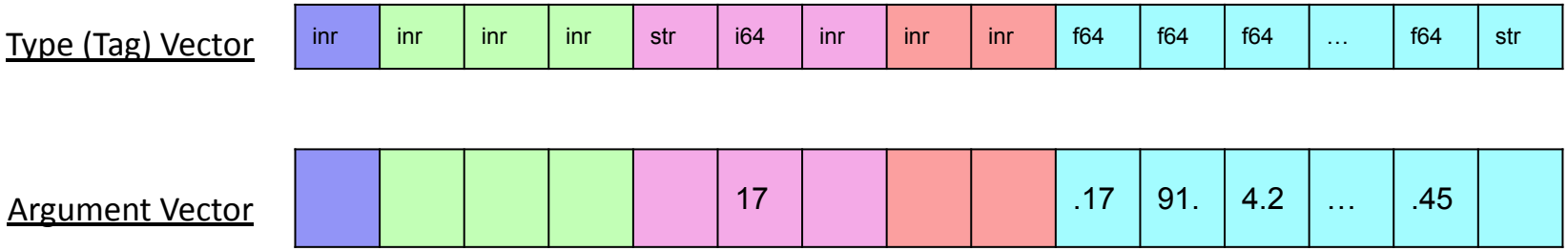
Wisent Format

Wisent Format

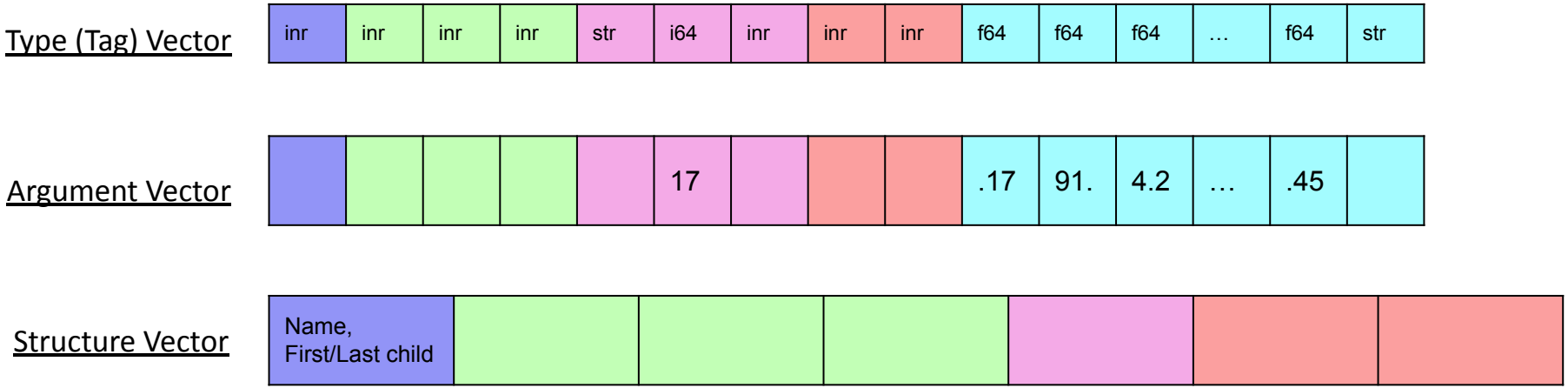
Type (Tag) Vector



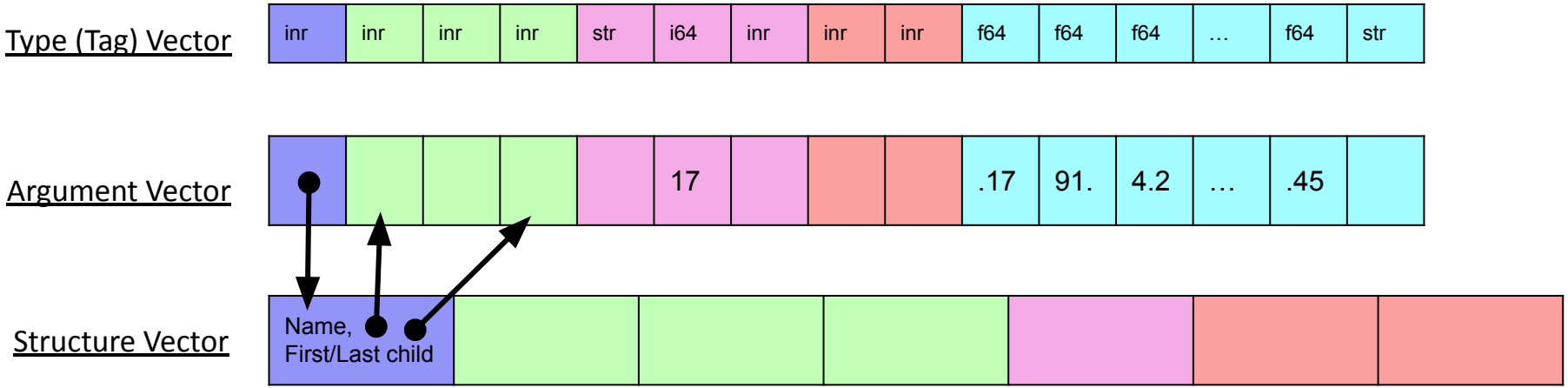
Wisent Format



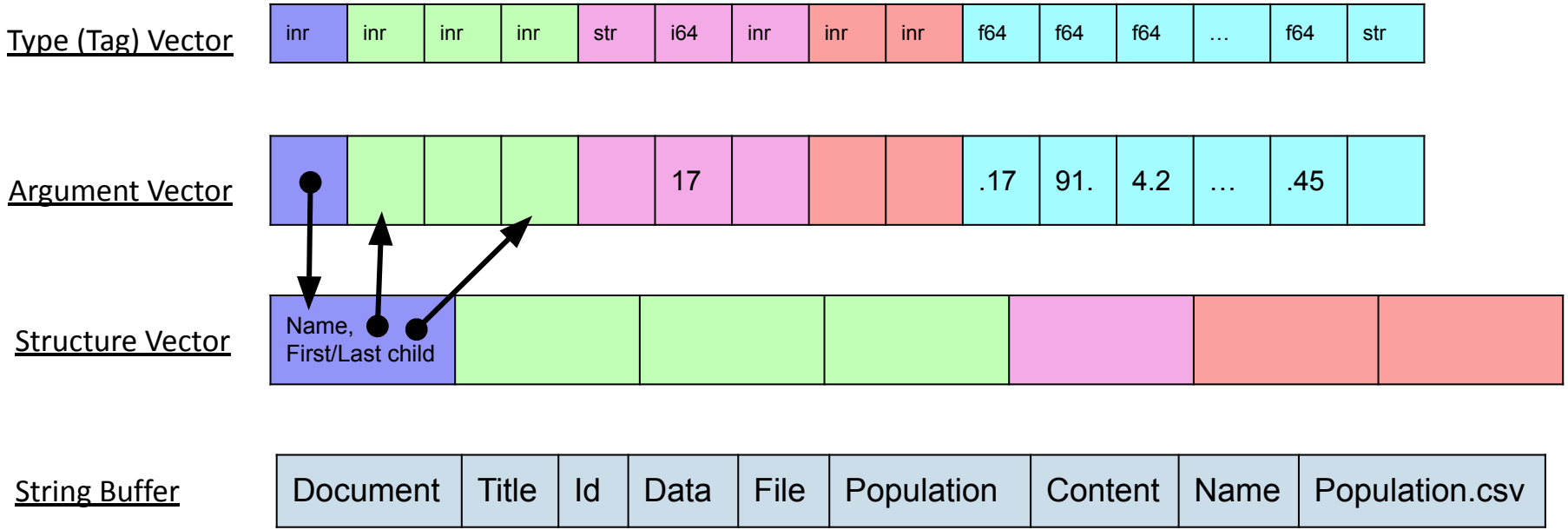
Wisent Format



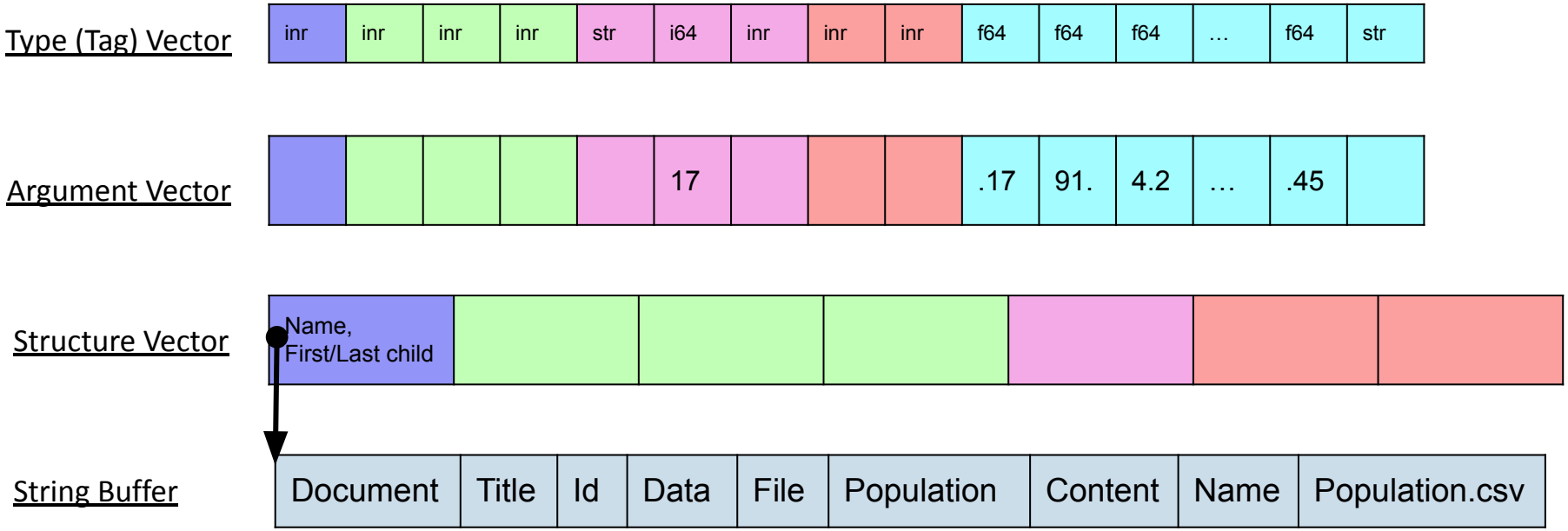
Wisent Format



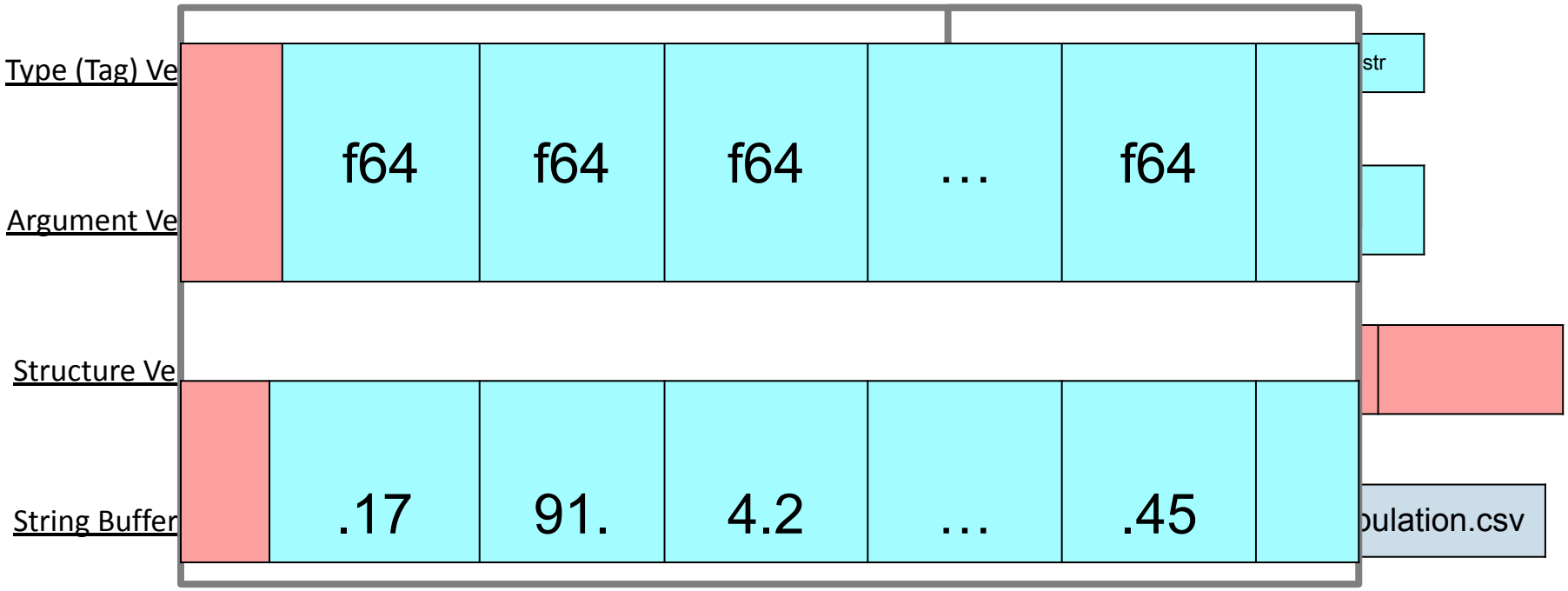
Wisent Format



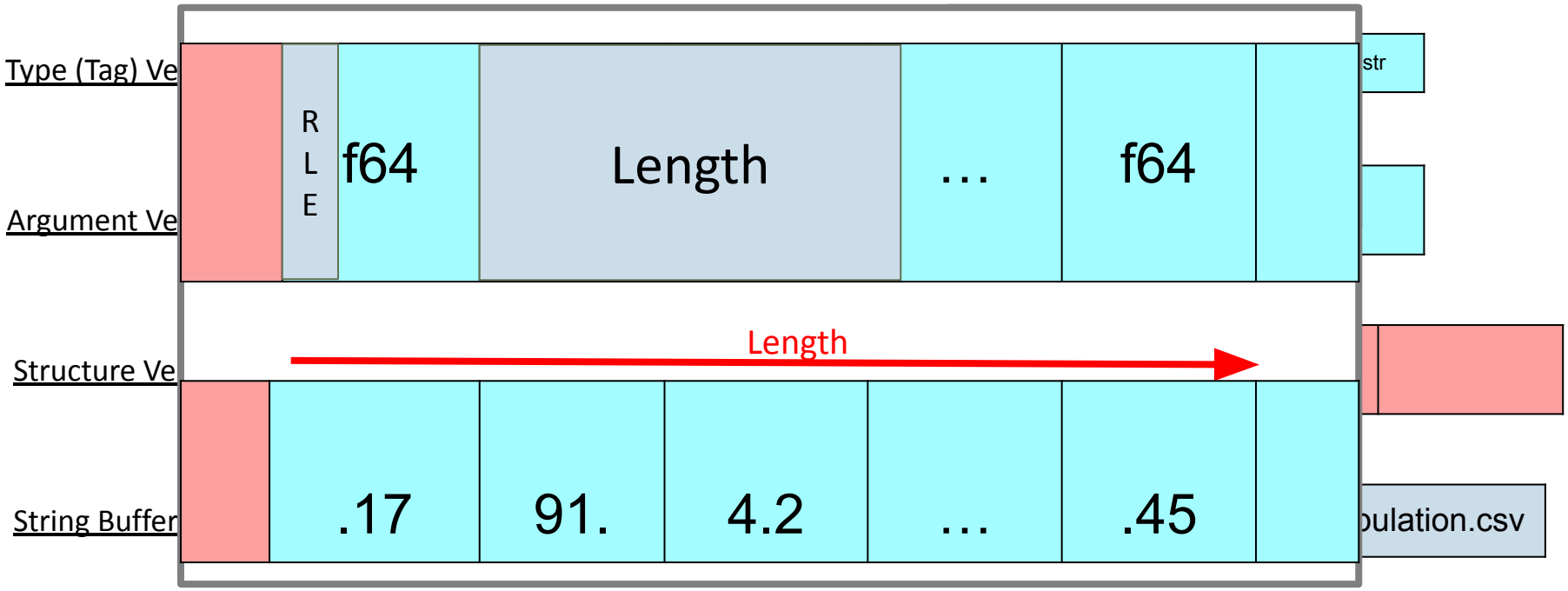
Wisent Format



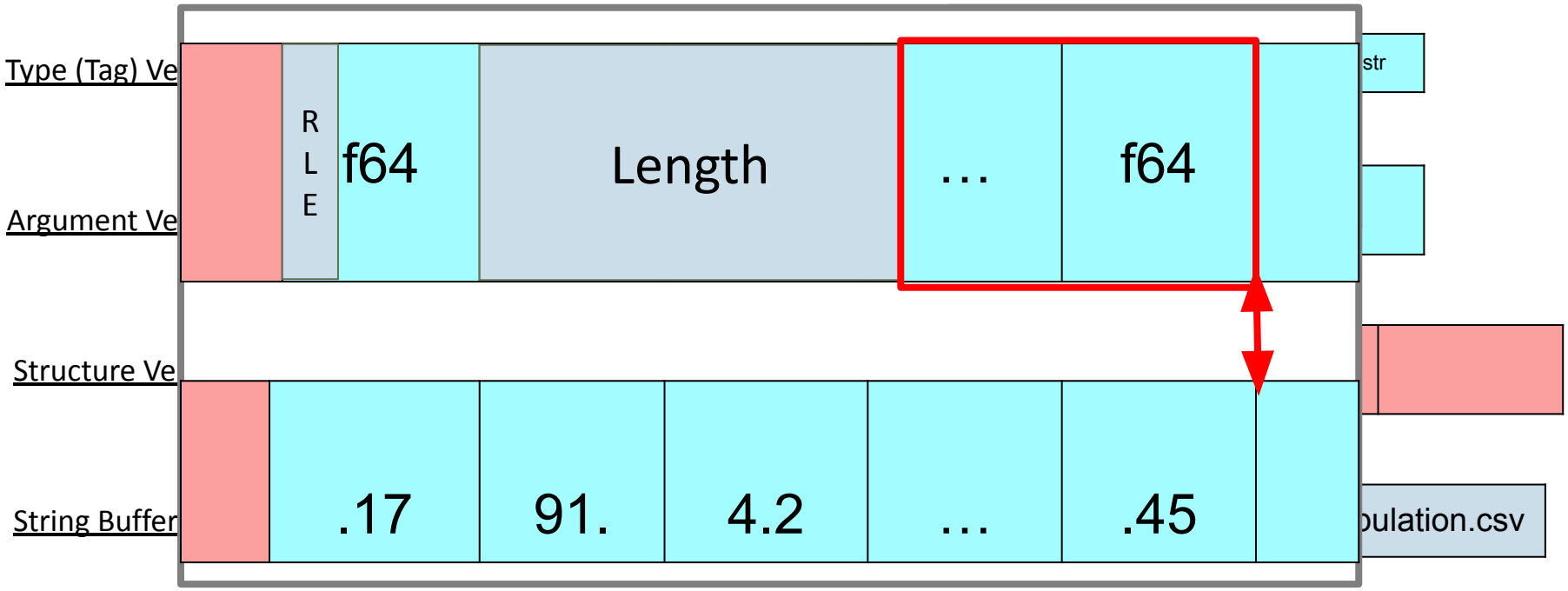
Run-Length Encoding for the Data Types



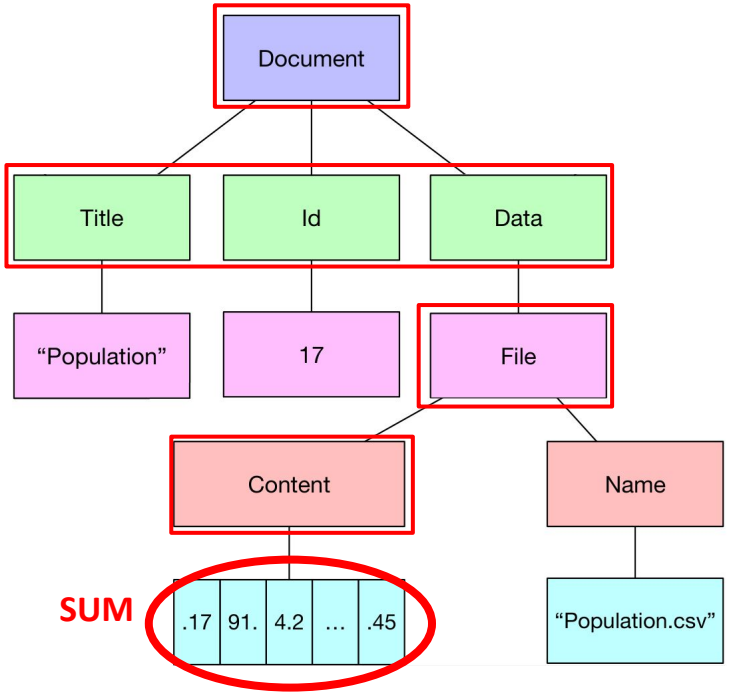
Run-Length Encoding for the Data Types

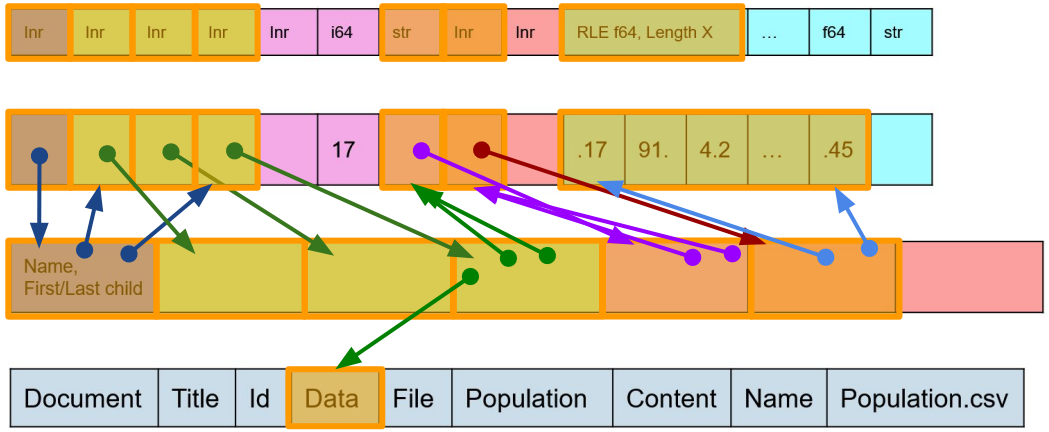


Run-Length Encoding for the Data Types

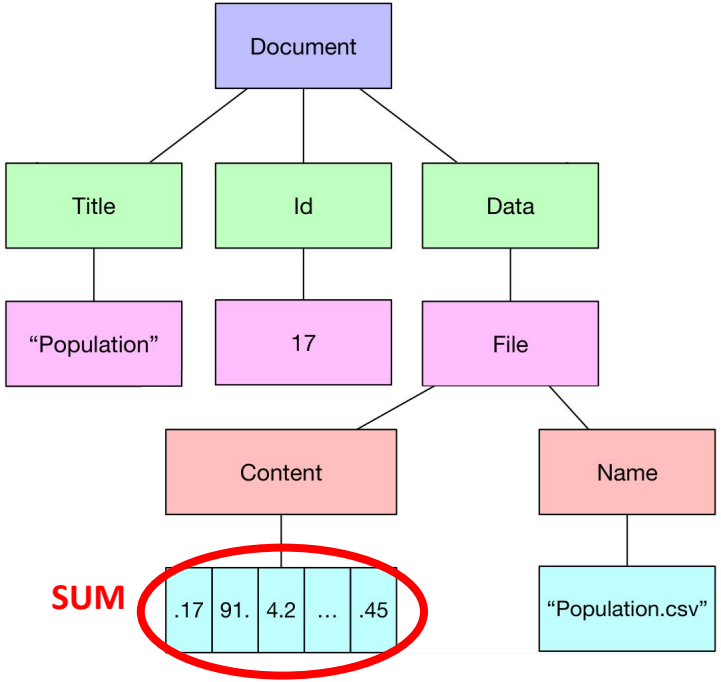


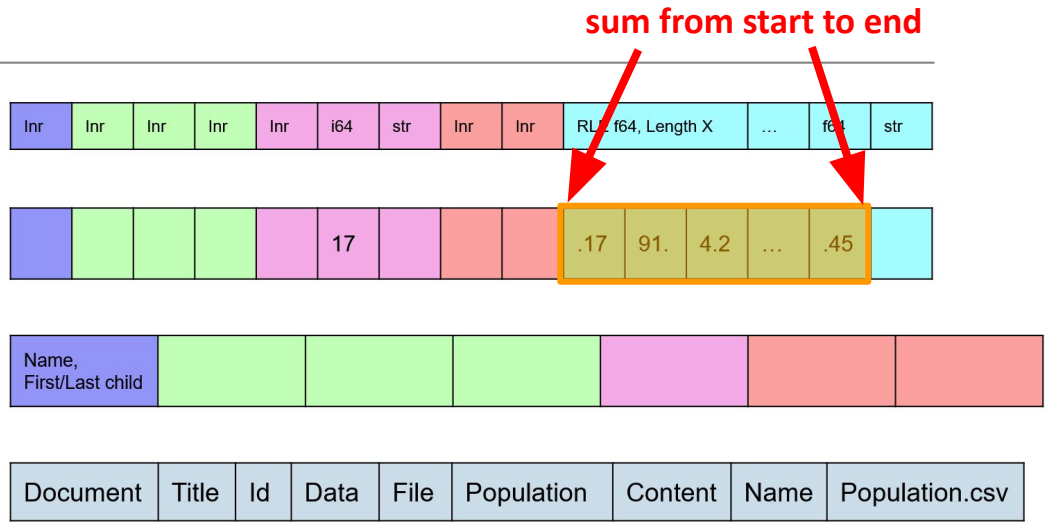
Deserialization





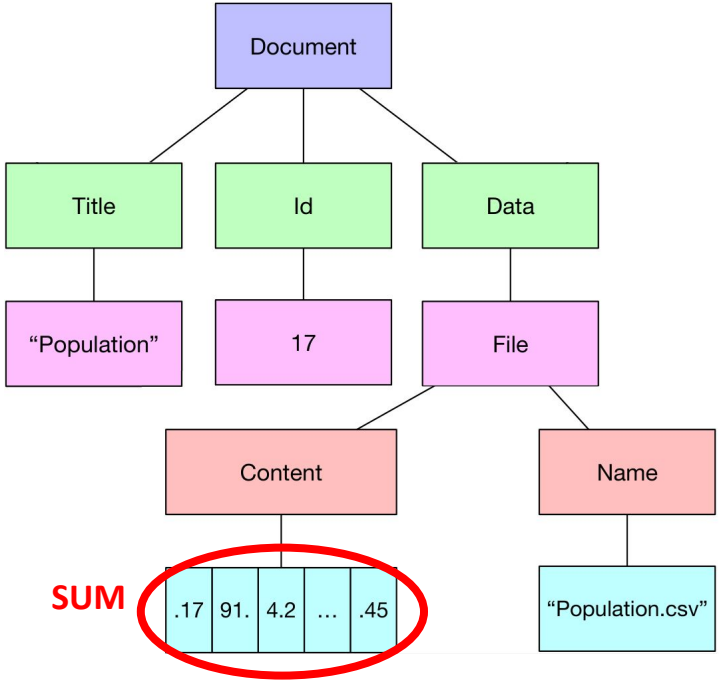
Deserialization

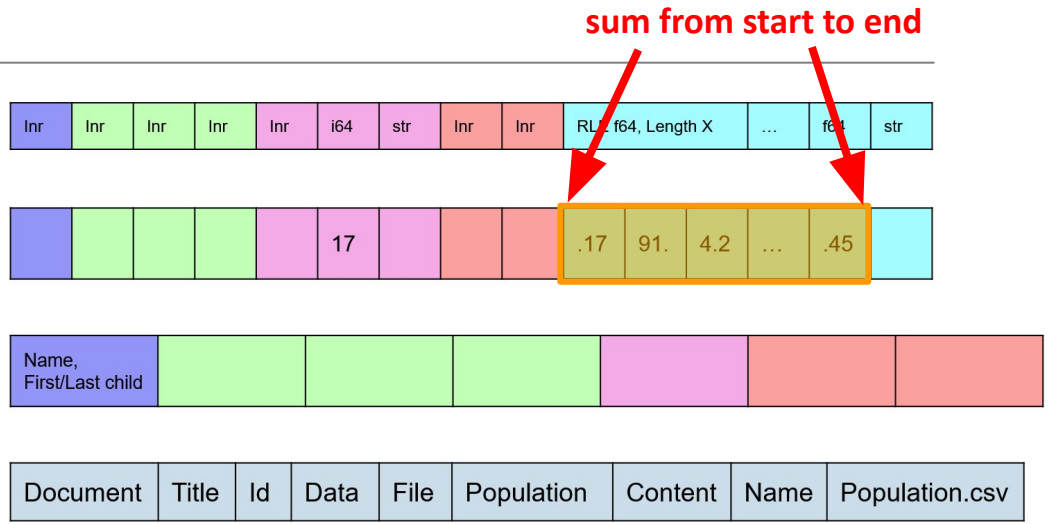




- RLE to accelerate sibling processing

Deserialization



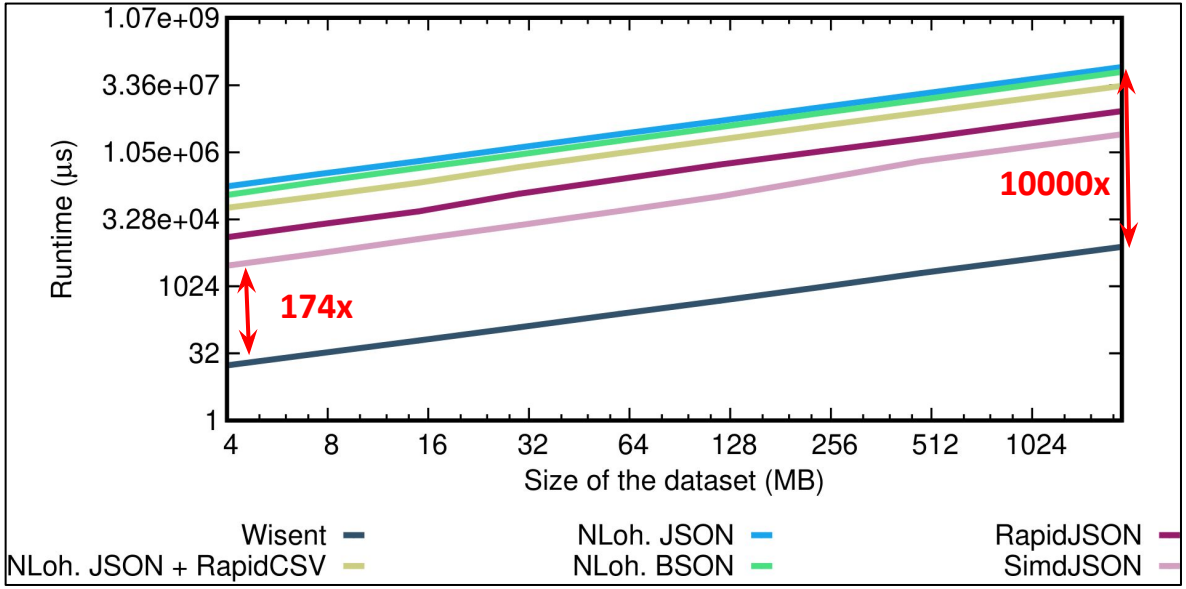


- RLE to accelerate sibling processing
- Lazy loading to avoid reading unnecessary data

Quantitative Evaluation

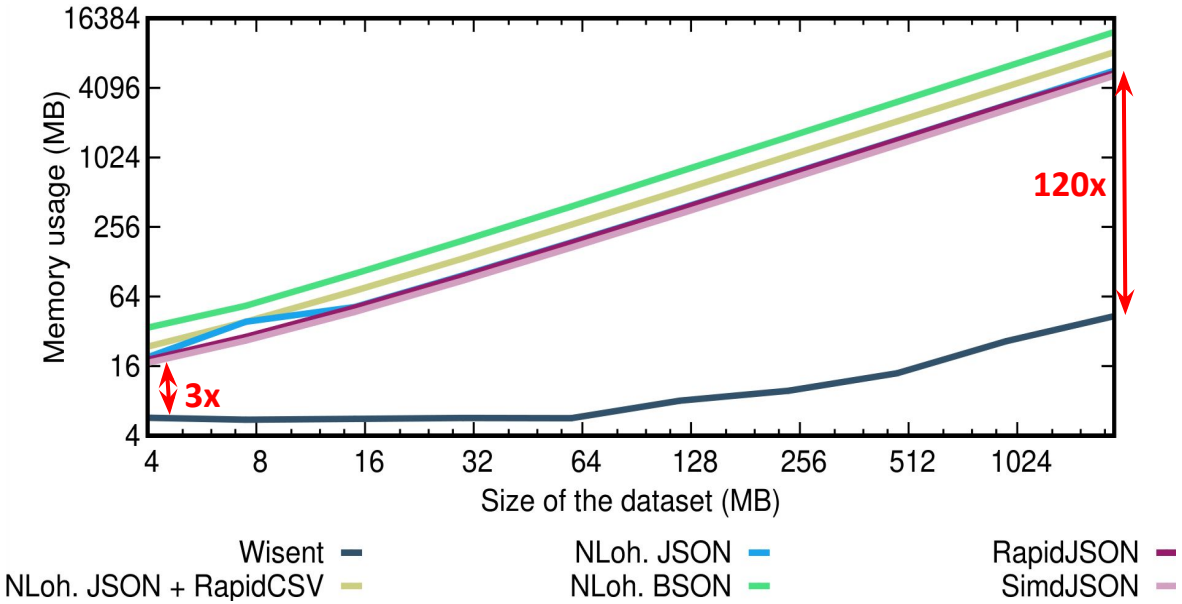
Deserialization : Runtime Performance

- Embeds CSV data + JSON metadata into Wisent
- Exchanges as shared memory
- Aggregates (Sum) on one column (out of 100)
- Dataset up to 2GB



Deserialization : Memory Usage

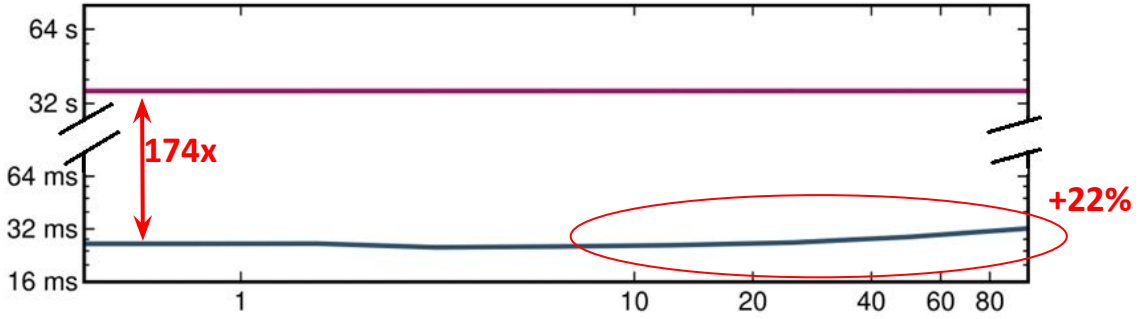
- Embeds CSV data + JSON metadata into Wisent
- Exchanges as shared memory
- Aggregates (Sum) on one column (out of 100)
- Dataset up to 2GB



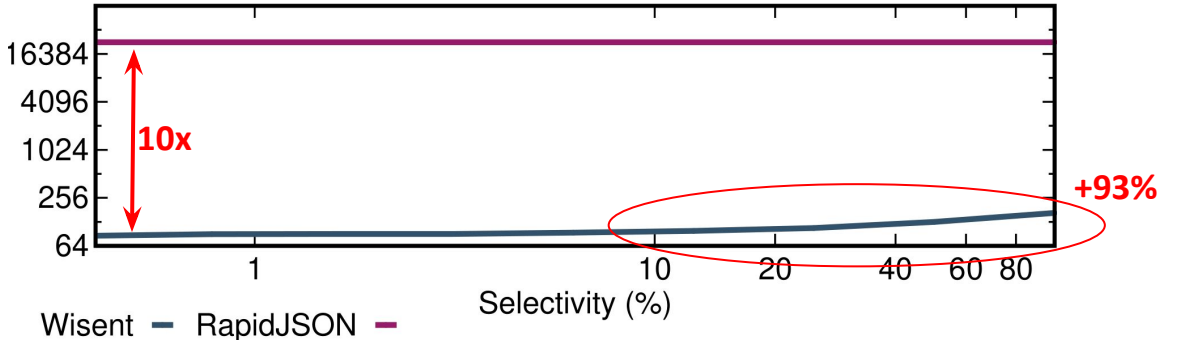
Deserialization with Variable Selectivity

- Aggregate (1 column) + Filtering (1 column)
- Dataset size: 8GB

Runtime

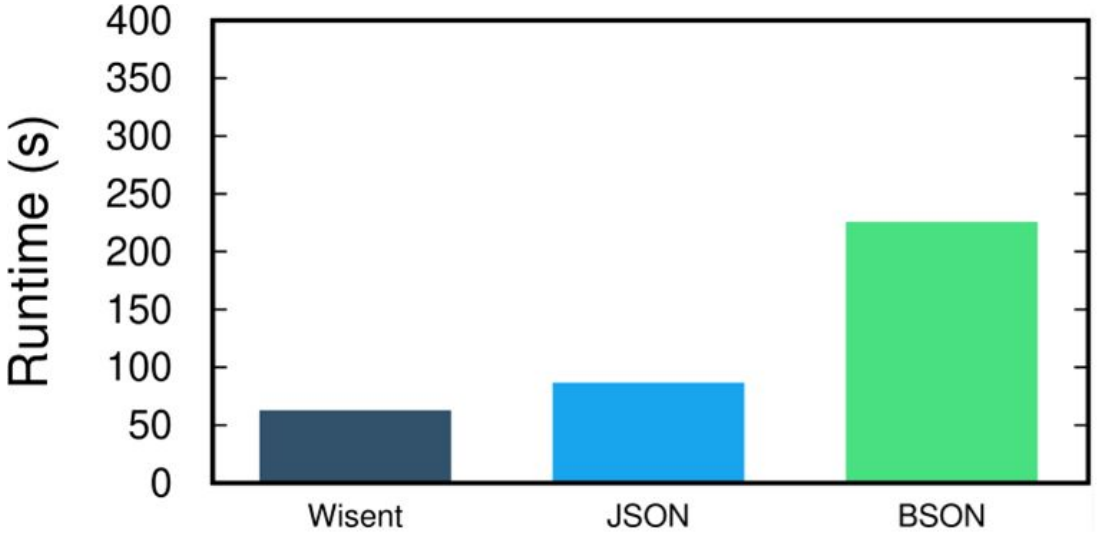


Memory (MB)



(All other baselines fail to deserialize this size of JSON document)

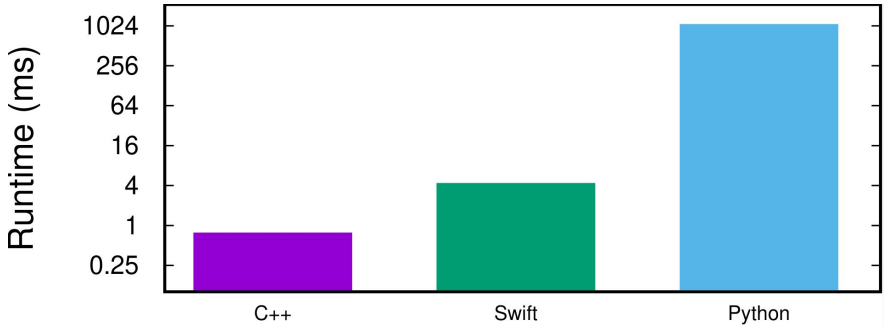
Serialization performance



Qualitative Evaluation

Deserializers implemented in C++/Swift/Python

	C++	Swift	Python
Lines of code	153	93	135



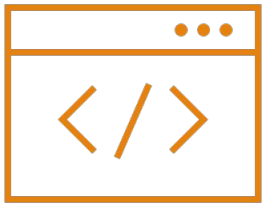
Conclusion

Advantages over other formats:

	Fast decoding	Memory efficient	Simple decoding	Runtime-nested data
JSON/BSON	x	x	x	✓
Thrift/Protobuf	✓	✓	x	x
Wisent	✓	✓	✓	✓

+ Benefits all our research work for kernel composition.
  - Kernel composability paper in revision (VLDB 2024)

Availability



Source at boss.lsds.uk/wisent

Feel free to check/use/contribute!