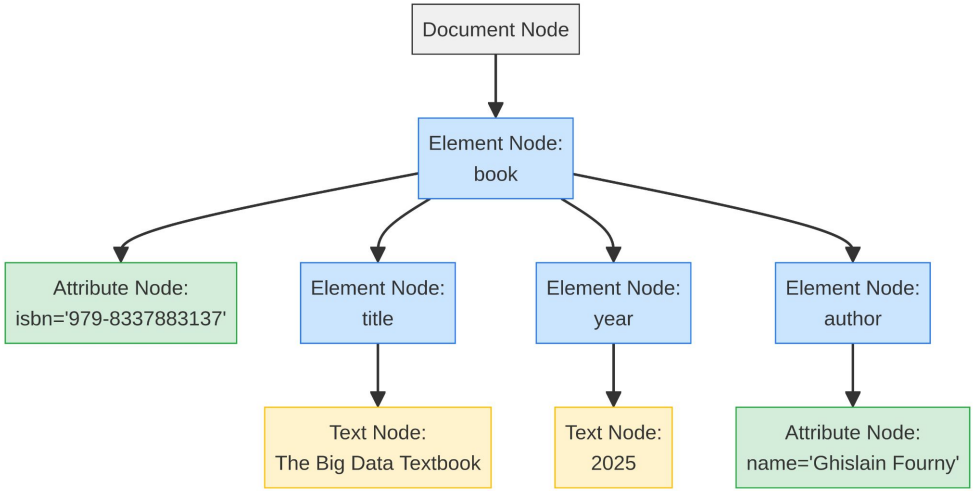
XML processing on the cluster: Bringing XQuery 3.1 to RumbleDB

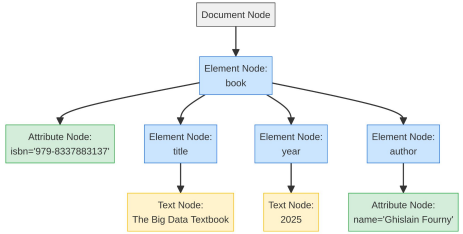
Matteo Agnoletto (ETH Zurich), Ghislain Fourny (ETH Zurich)
Declarative Amsterdam 2025
Nov 7th, 2025

```
<book isbn=”979-8337883137”>
  <title>The Big Data Textbook</title>
  <year>2025</year>
  <author name=”Ghislain Fourny”/>
</book>
```

XML

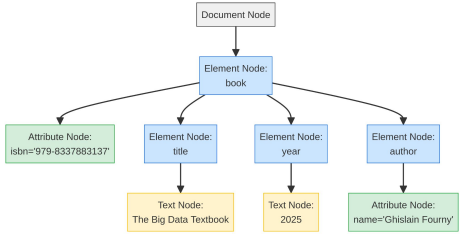


```
<book isbn=”979-8337883137”>
  <title>The Big Data Textbook</title>
  <year>2025</year>
  <author name=”Ghislain Fourny”/>
</book>
```



• • •

```
<book isbn=”979-8337883137”>
  <title>The Big Data Textbook</title>
  <year>2025</year>
  <author name=”Ghislain Fourny”/>
</book>
```

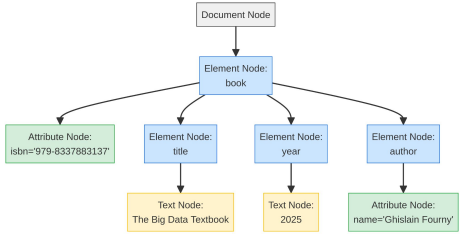


...

XQuery

```
for $book in xml-files("books/*.xml")
where data($book/@isbn) = "979-8337883137"
return
  <book>
       <isbn>{ $book/@isbn }</isbn>
       <title>{ $book/title/text() }</title>
  </book>
```

```
<book isbn=”979-8337883137”>
  <title>The Big Data Textbook</title>
  <year>2025</year>
  <author name=”Ghislain Fourny”/>
</book>
```

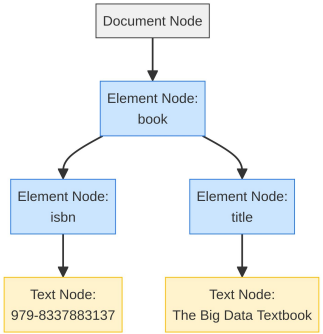


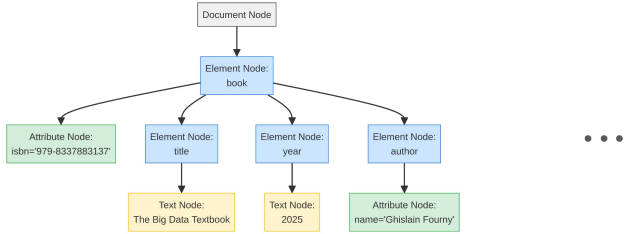
...

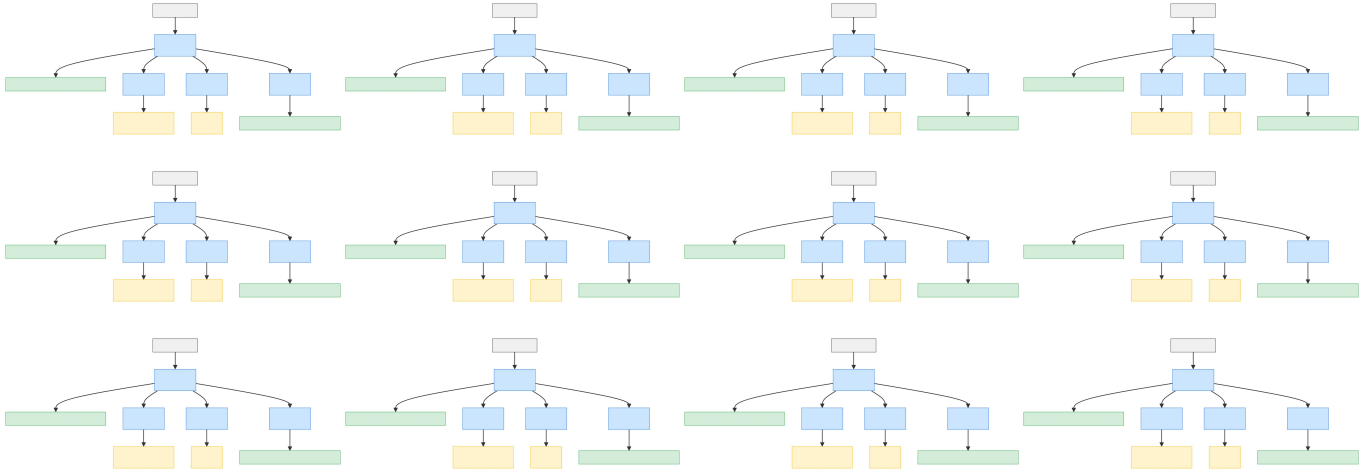
XQuery

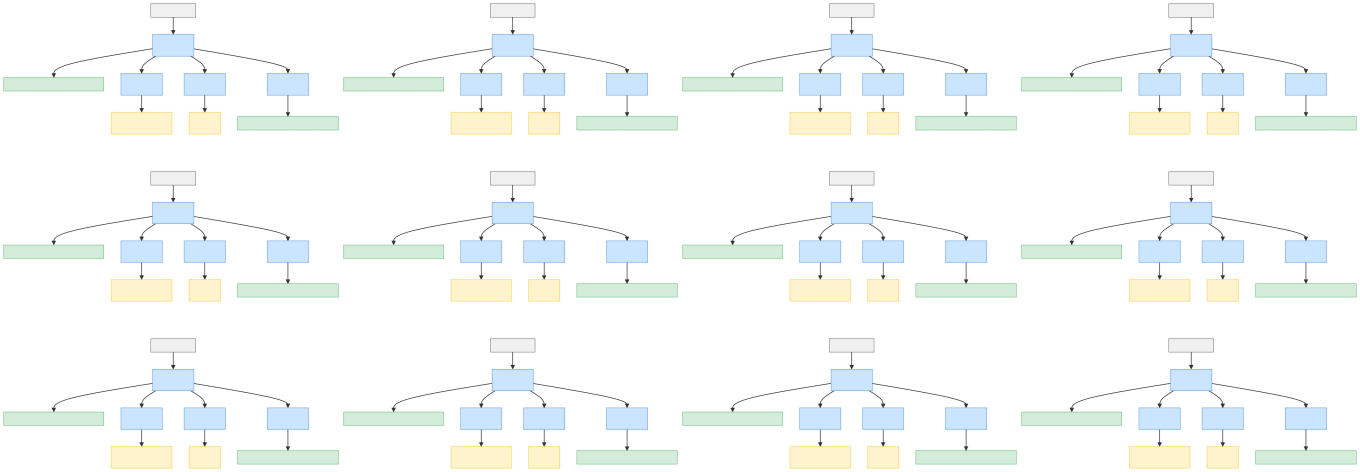
```
for $book in xml-files("books/*.xml")
where data($book/@isbn) = "979-8337883137"
return
  <book>
       <isbn>{ $book/@isbn }</isbn>
       <title>{ $book/title/text() }</title>
  </book>
```

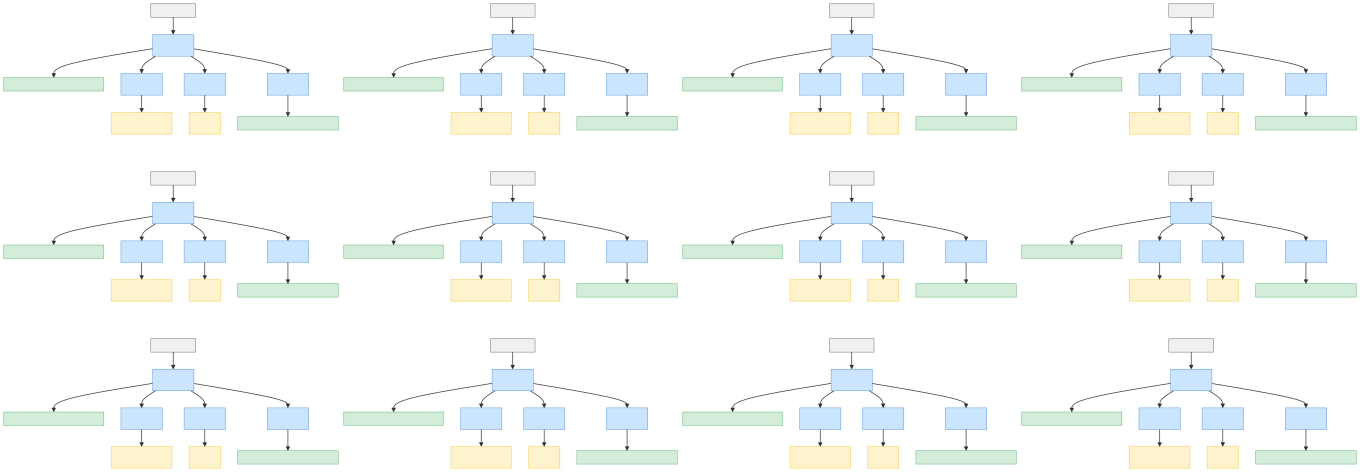
```
<book>
  <isbn>979-8337883137</isbn>
  <title>The Big Data Textbook</title>
</book>
```



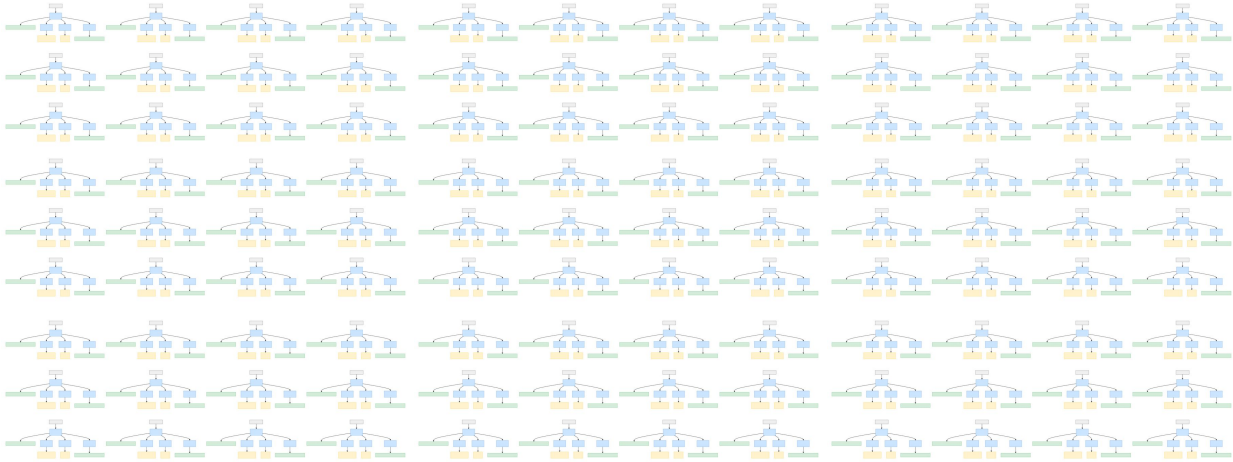

Document Node
Element Node: book
Attribute Node: isbn='979-8337883137'
Element Node: title
Element Node: year
Element Node: author
Text Node: The Big Data Textbook
Text Node: 2025
Attribute Node: name='Ghislain Fourny'



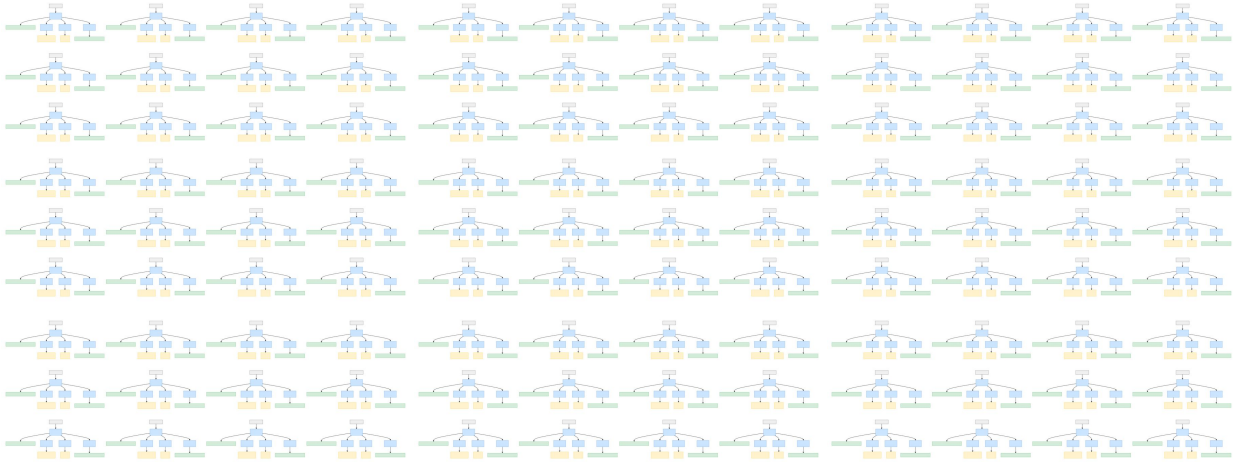


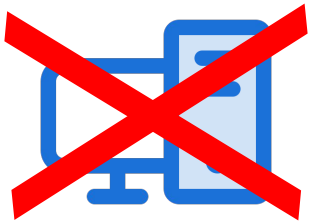


M+ files
PB-size

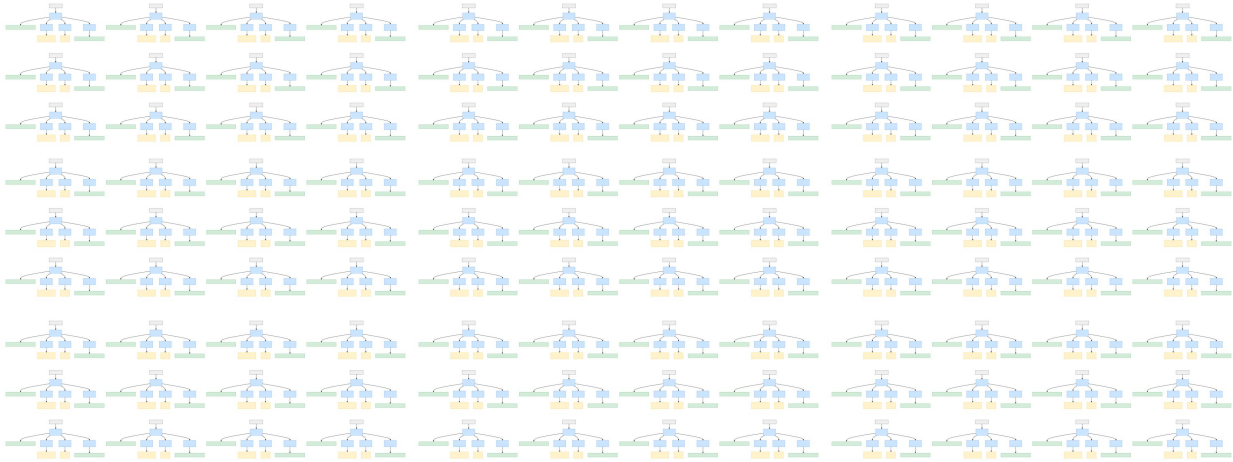


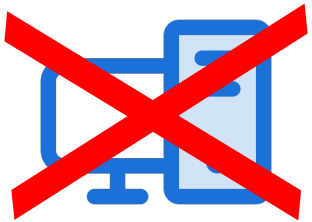
M+ files
PB-size



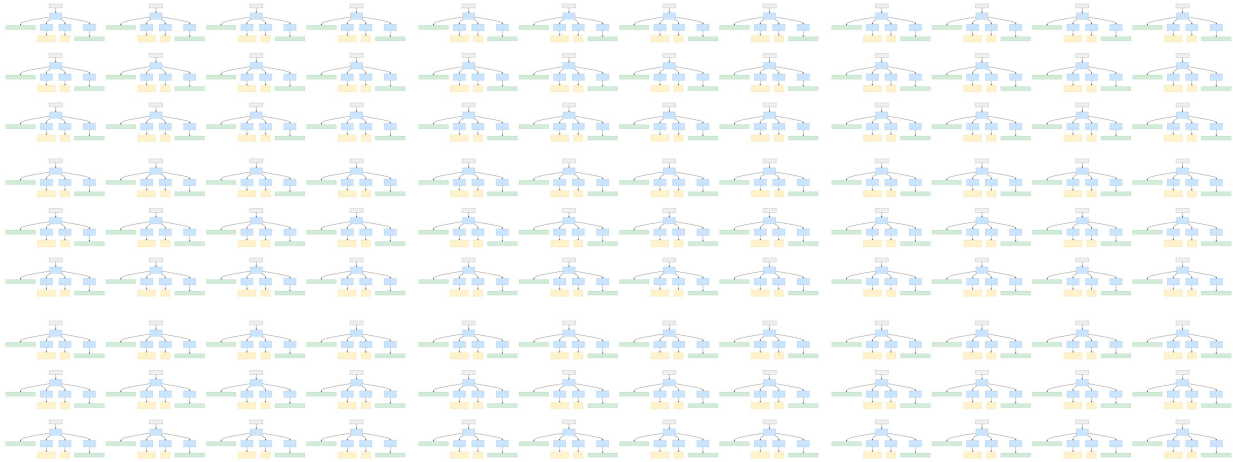


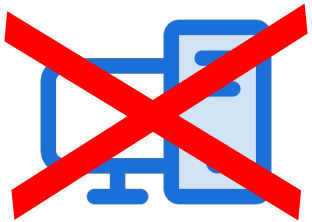
M+ files
PB-size





M+ files
PB-size





How to efficiently query a large collection of XML data with millions of items, distributed on a cluster?

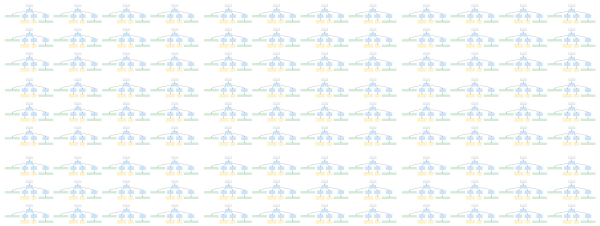
How to efficiently query a large collection of XML data with millions of items, distributed on a cluster?

RumbleDB is a query execution engine
for massive, semi-structured,
heterogeneous datasets,
with native JSONiq and XQuery 3.1 support
running locally, or distributed on the cluster.

RumbleDB is a query execution engine for massive , semi-structured, heterogeneous datasets, with native JSONiq and XQuery 3.1 support running locally, or distributed on the cluster.

RumbleDB is a query execution engine for massive , semi-structured, heterogeneous datasets, with native JSONiq and XQuery 3.1 support running locally, or distributed on the cluster.

M+ files, PB-size



RUMBLE

RumbleDB is a query execution engine for massive, semi-structured, heterogeneous datasets, with native JSONiq and XQuery 3.1 support running locally, or distributed on the cluster.

RumbleDB is a query execution engine for massive, semi-structured, heterogeneous datasets,
with native JSONiq and XQuery 3.1 support
running locally, or distributed on the cluster.

Formats: JSON, XML, CSV, …

RumbleDB is a query execution engine for massive, semi-structured, heterogeneous datasets,

with native JSONiq and XQuery 3.1 support

running locally, or distributed on the cluster.

Formats: JSON, XML, CSV, …
Storage Mediums: HDFS, S3, Local, …

RUMBLE

RumbleDB is a query execution engine
for massive, semi-structured,
heterogeneous datasets,
with native JSONiq and XQuery 3.1 support
running locally, or distributed on the cluster.

RumbleDB supports XQuery 3.1 & XML

RumbleDB supports XQuery 3.1 & XML Natively

RumbleDB supports XQuery 3.1 & XML Natively

Query

RumbleDB supports XQuery 3.1 & XML Natively

Query → Dedicated ANTLR4 XQuery Parser

RumbleDB supports XQuery 3.1 & XML Natively

Query → Dedicated ANTLR4 XQuery Parser

Data Retrieval

RumbleDB supports XQuery 3.1 & XML Natively

Query → Dedicated ANTLR4 XQuery Parser

Data Retrieval → Direct access and parse of XML Data

RumbleDB supports a wide range of XQuery 3.1 expressions…

RumbleDB supports a wide range of XQuery 3.1 expressions…

- Arithmetic Operations & Comparisons

RumbleDB supports a wide range of XQuery 3.1 expressions…

- Arithmetic Operations & Comparisons
- FLOWR & Control Expressions

RumbleDB supports a wide range of XQuery 3.1 expressions…

- Arithmetic Operations & Comparisons
- FLOWR & Control Expressions
- Node Constructors & Comparisons

RumbleDB supports a wide range of XQuery 3.1 expressions…

- Arithmetic Operations & Comparisons
- FLOWR & Control Expressions
- Node Constructors & Comparisons
- Path Expressions

RumbleDB supports a wide range of XQuery 3.1 expressions…

- Arithmetic Operations & Comparisons
- FLOWR & Control Expressions
- Node Constructors & Comparisons
- Path Expressions
- XPath & XQuery Functions

...and more

…and more

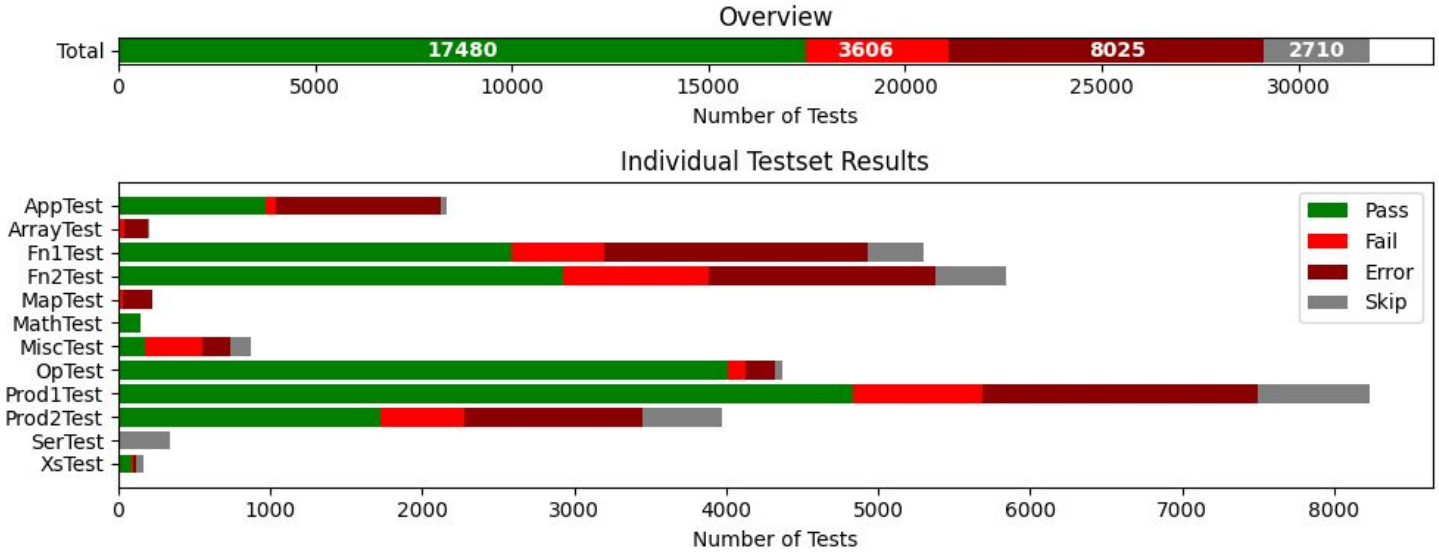
- XQuery Scripting Extension Proposal

…and more

- XQuery Scripting Extension Proposal
- JSONiq Update Facility

Rumble Test Suite (qt3tests) Results

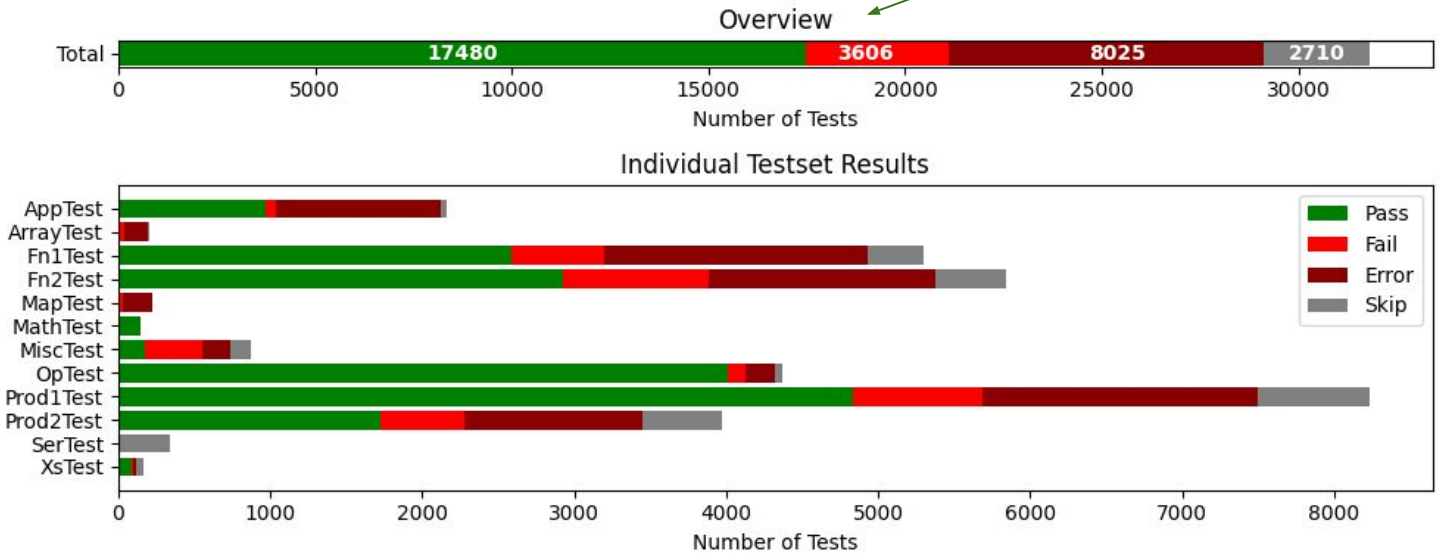
Rumble DB 2.0.0 “Lemon Ironwood”



Rumble Test Suite (qt3tests) Results

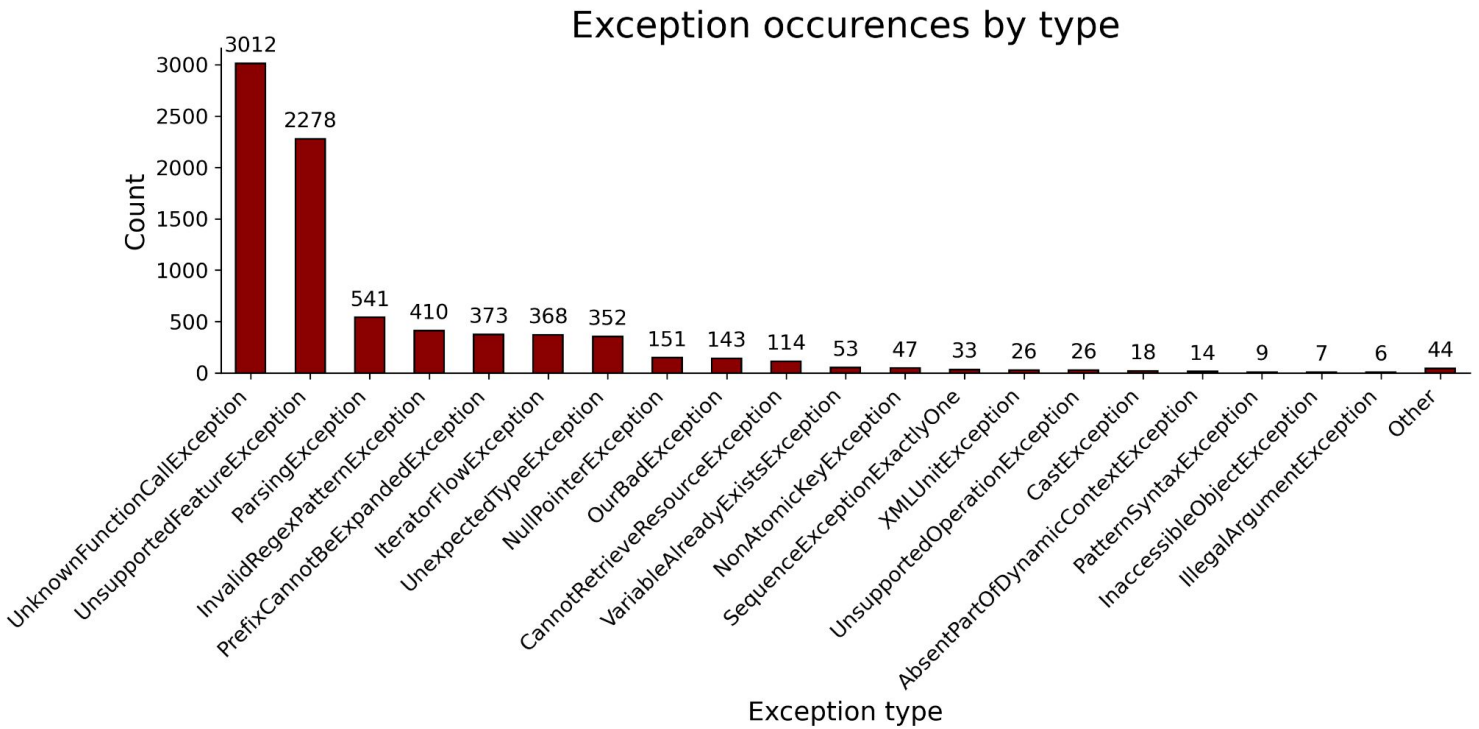
Rumble DB 2.0.0 “Lemon Ironwood”

54%+

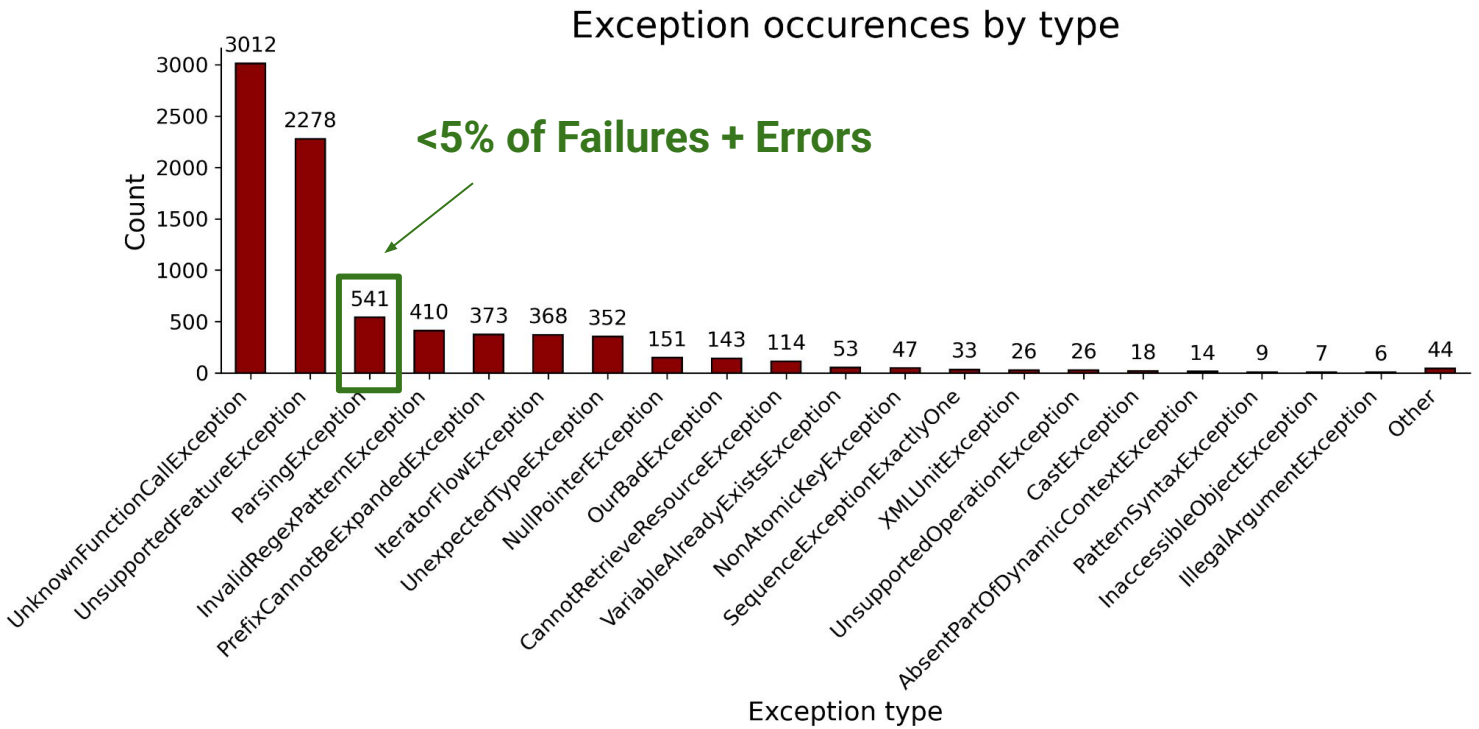


Rumble Test Suite (qt3tests) Results

Rumble DB 2.0.0 “Lemon Ironwood”



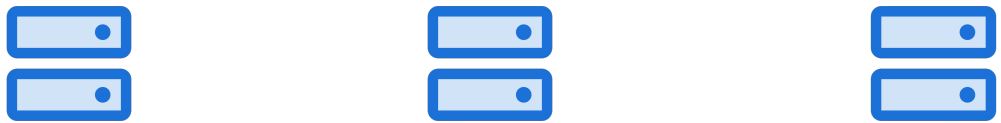
Rumble Test Suite (qt3tests) Results
Rumble DB 2.0.0 “Lemon Ironwood”



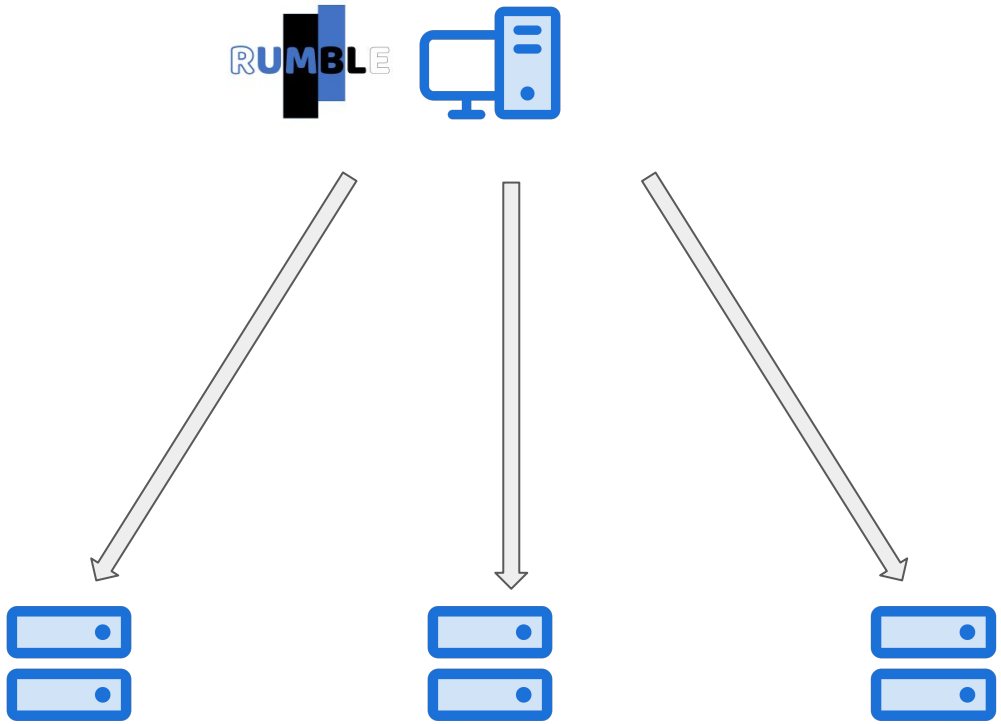
RumbleDB is a query execution engine
for massive, semi-structured,
heterogeneous datasets
with native JSONiq and XQuery 3.1 support
running locally, or distributed on the cluster.

RumbleDB is a query execution engine
for massive, semi-structured,
heterogeneous datasets
with native JSONiq and XQuery 3.1 support
running locally, or distributed on the cluster.

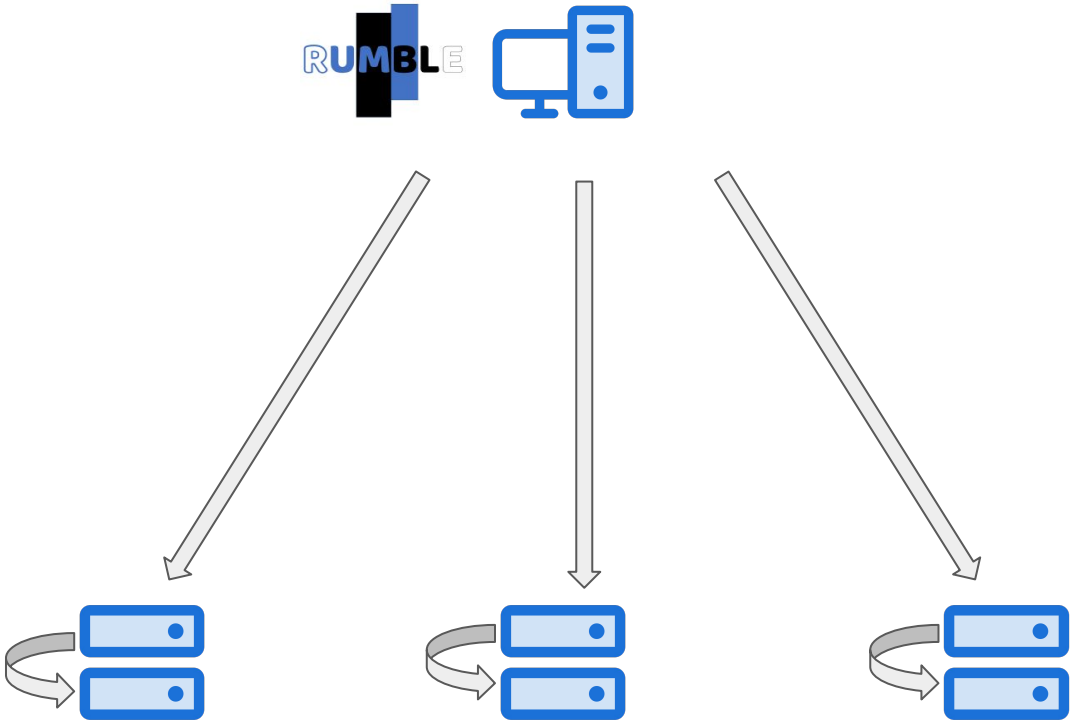
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



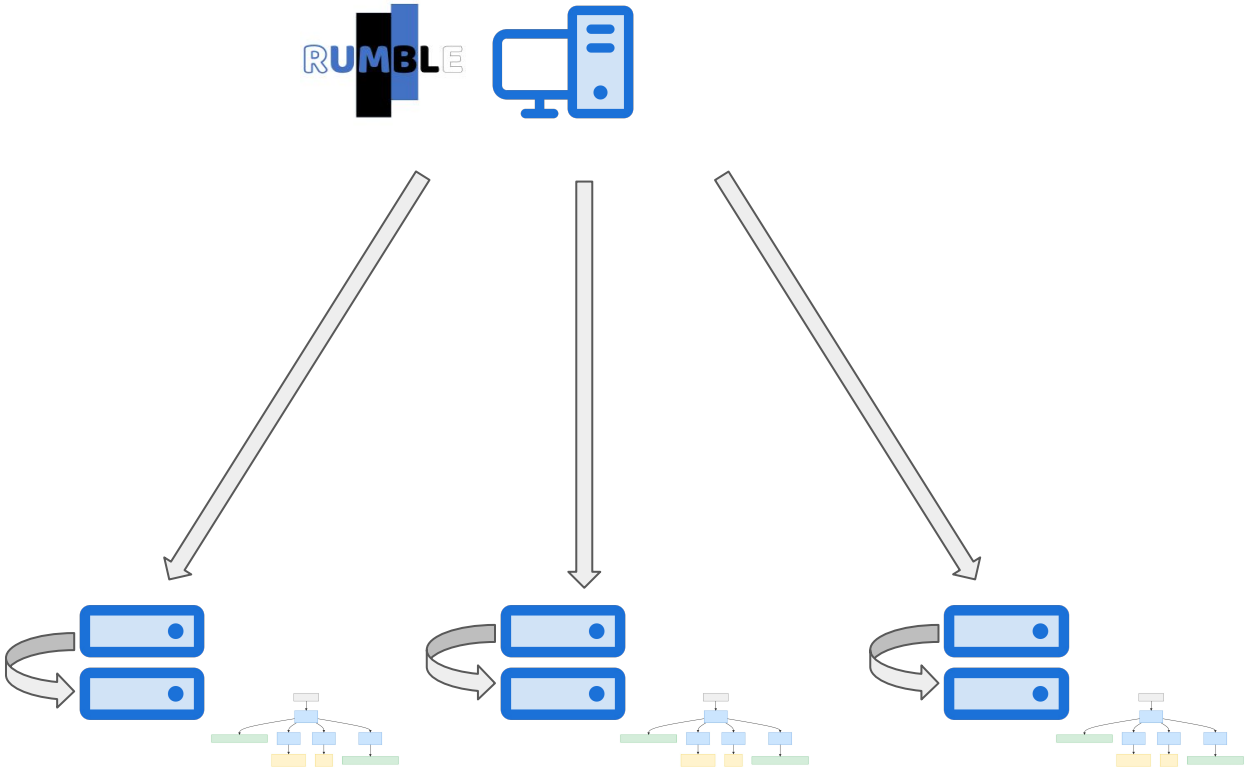
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



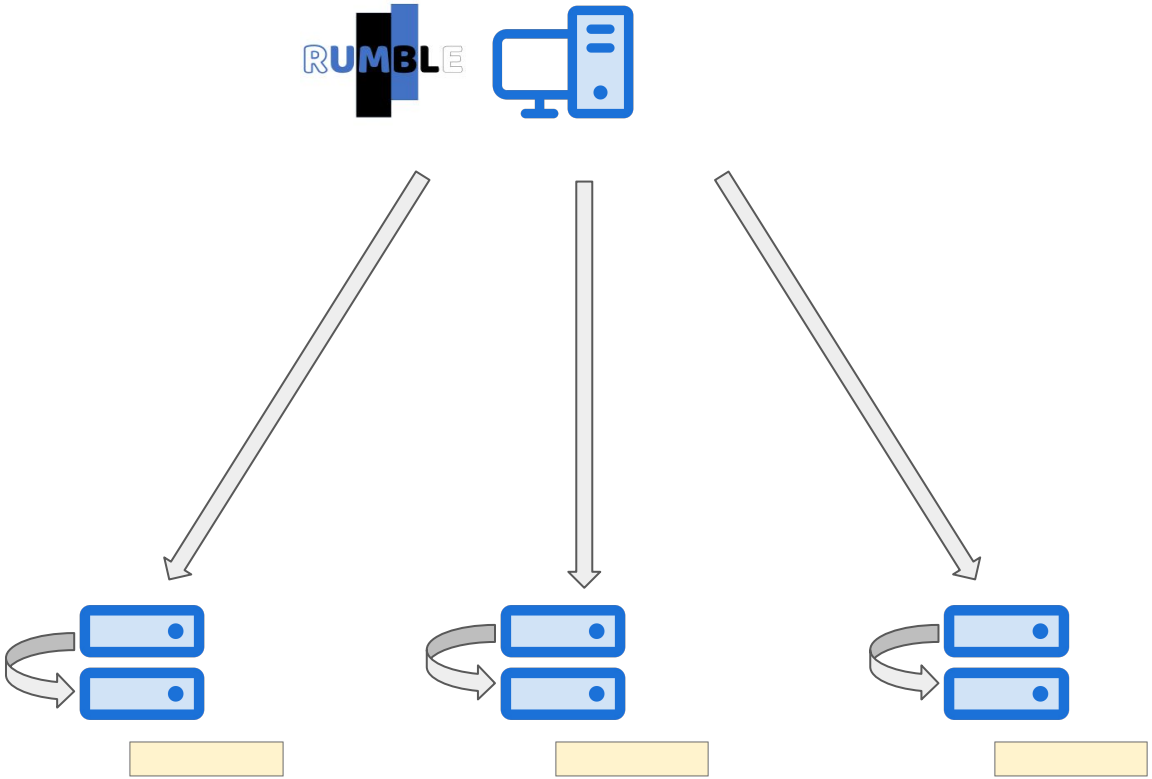
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



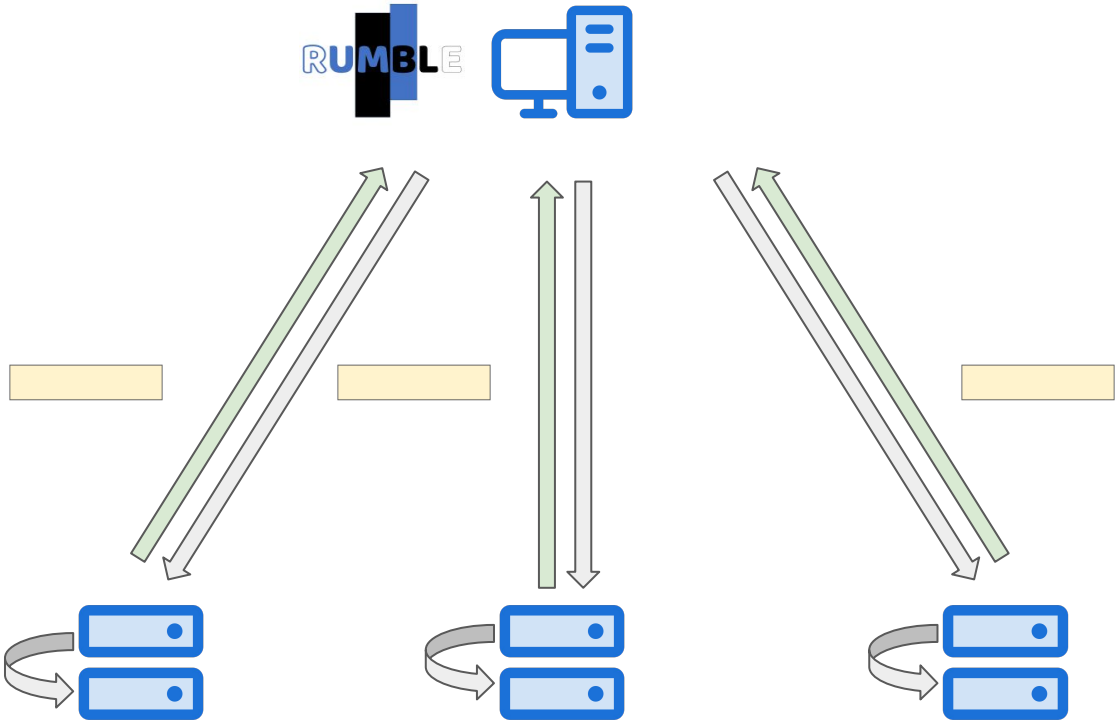
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



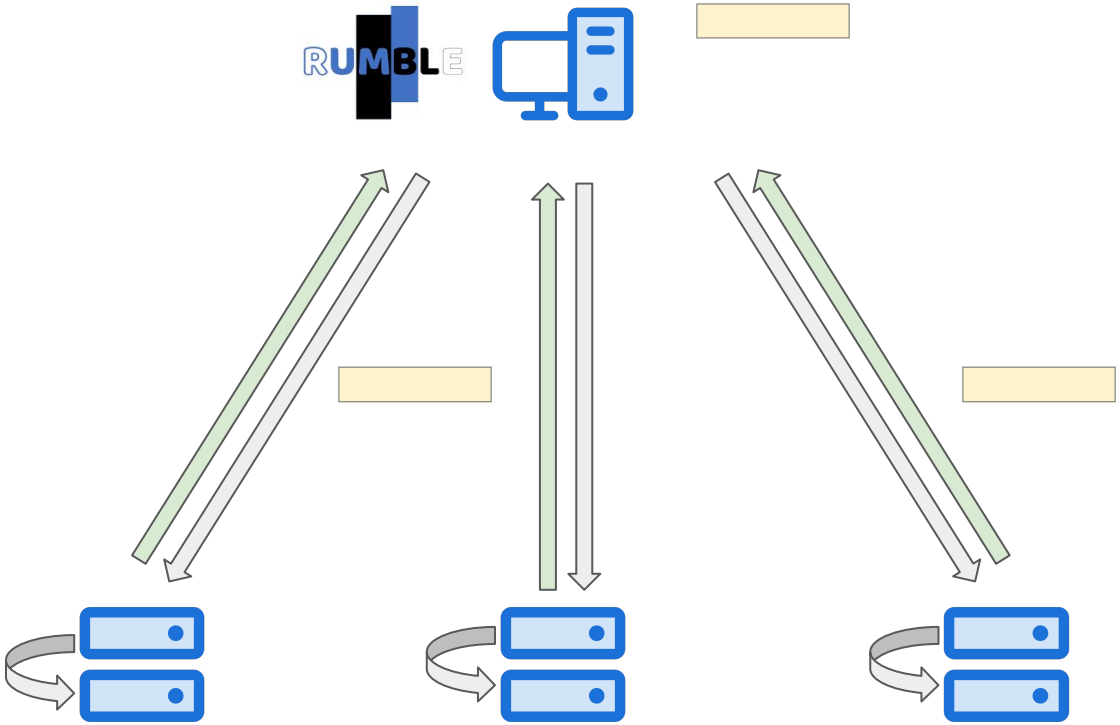
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



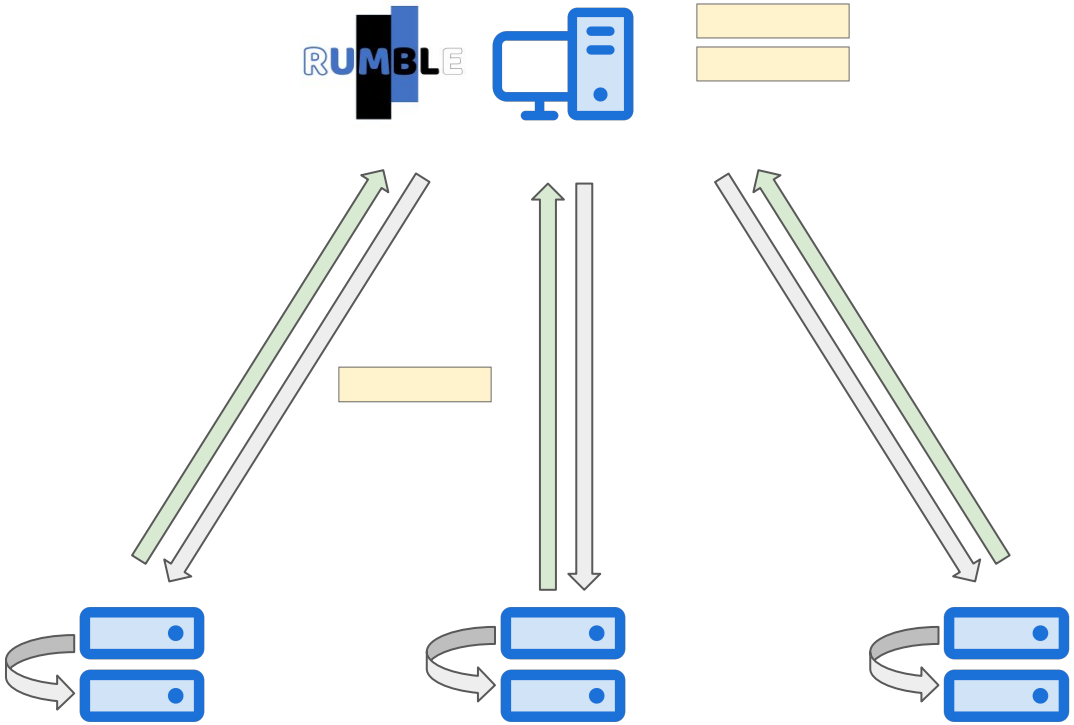
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



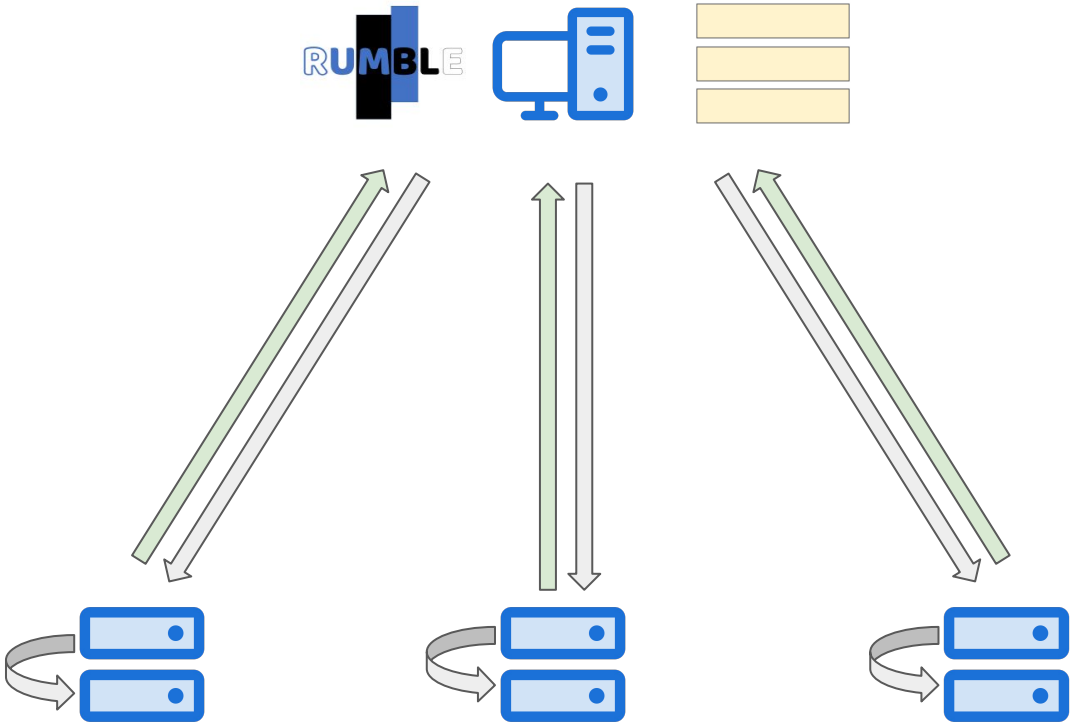
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



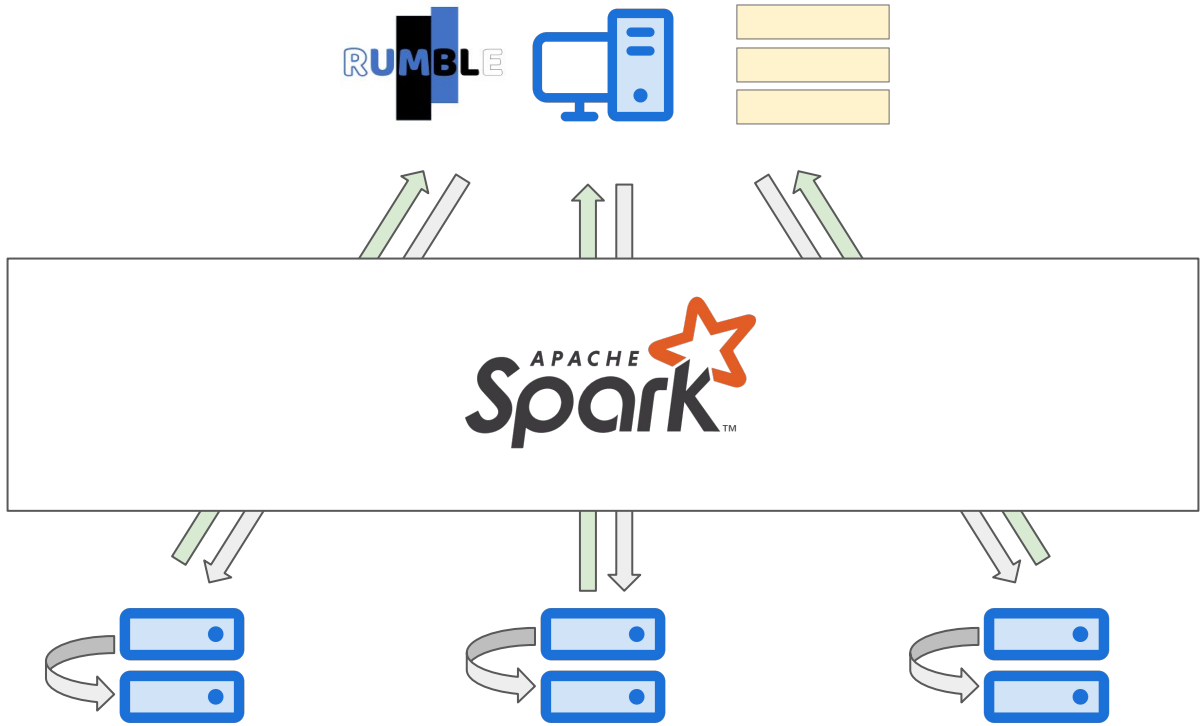
```
for $book in xml-files("books/*.xml")
return $book/isbn
```



```
for $book in xml-files("books/*.xml")
return $book/isbn
```



```
for $book in xml-files("books/*.xml")
return $book/isbn
```

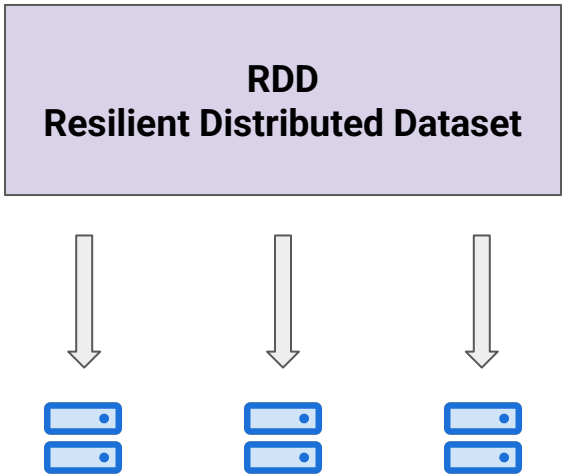



APACHE
Spark

RDD
Resilient Distributed Dataset

RDD
Resilient Distributed Dataset

- Big
- Partitioned
- Distributed

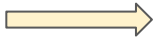


- Big
- Partitioned
- Distributed


10
01


hadoop

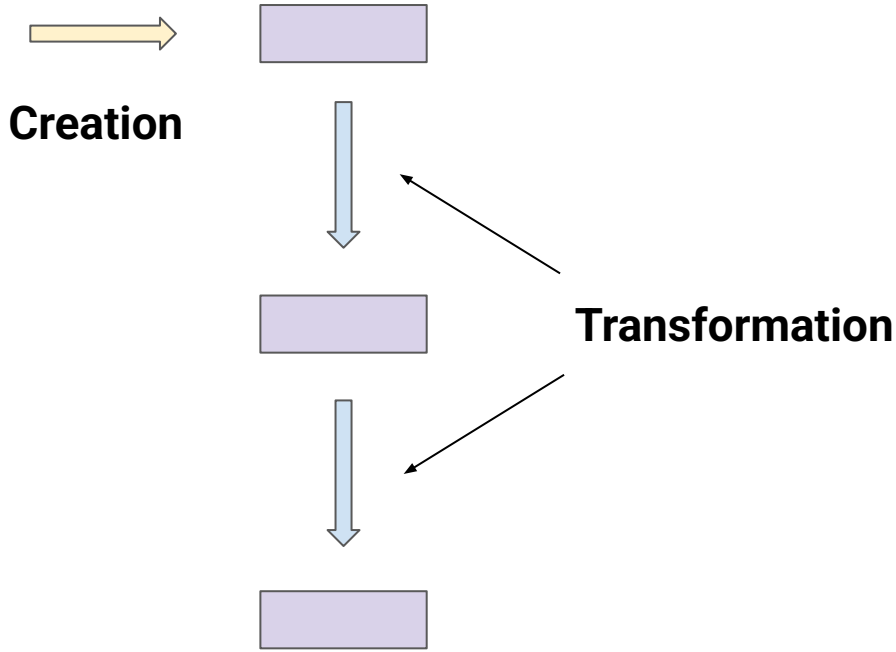

APACHE
Spark™

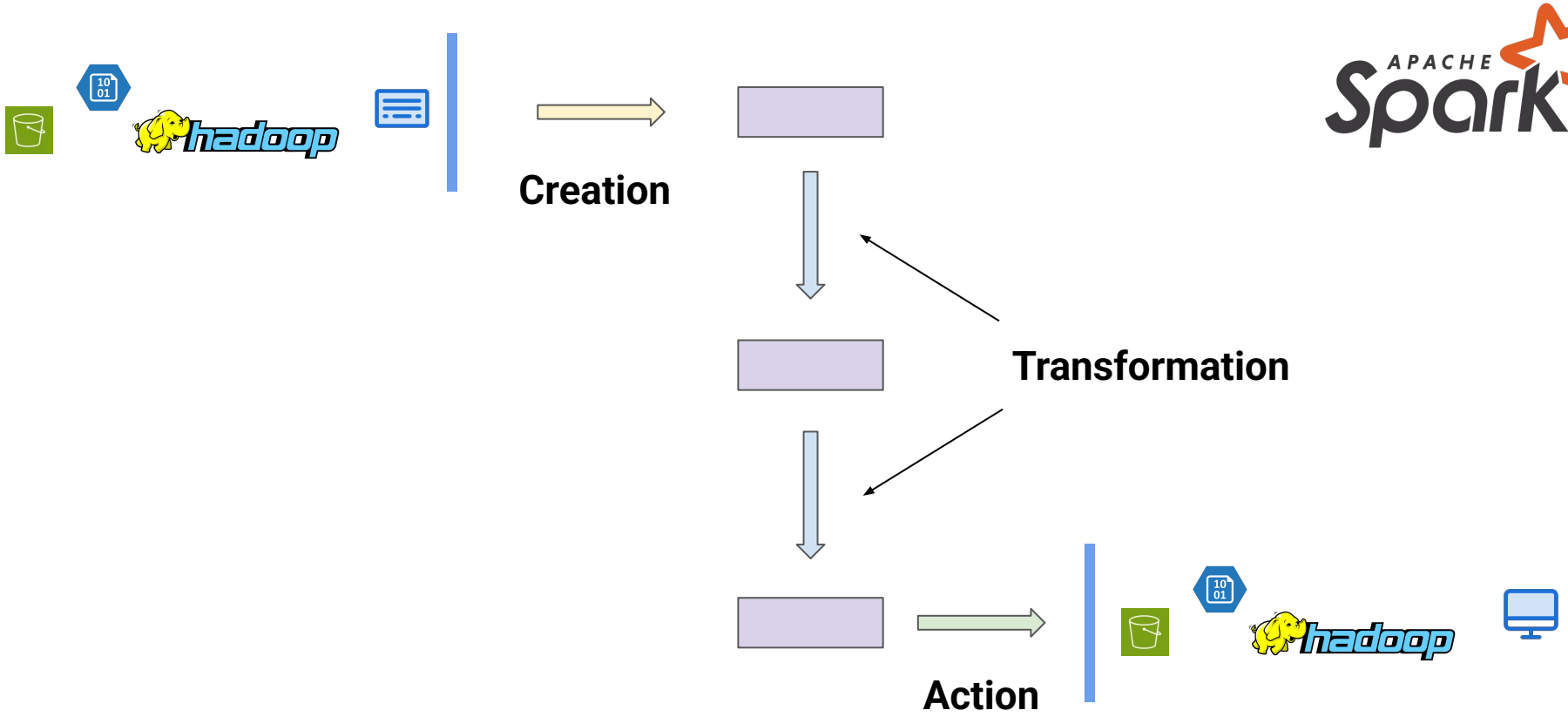


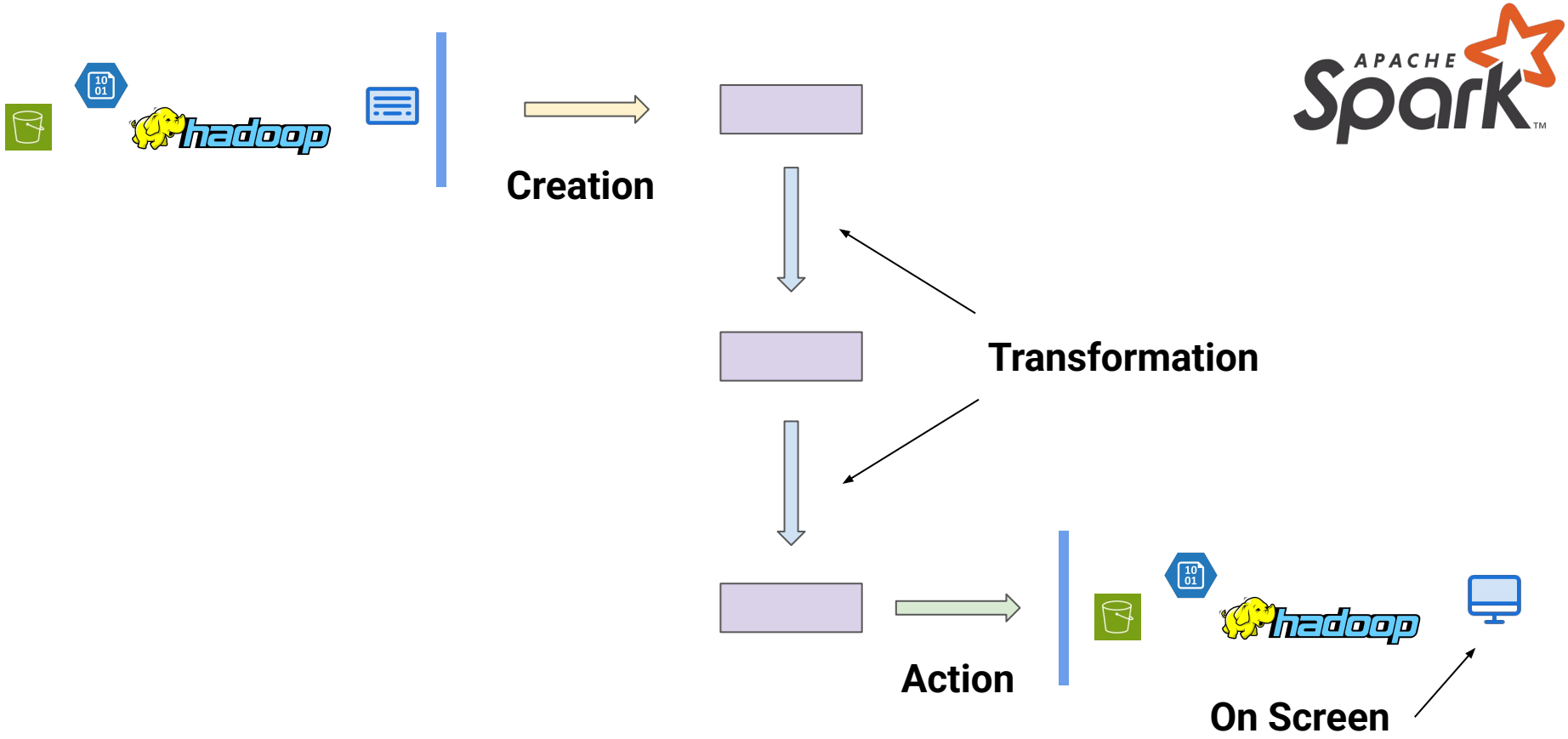
Creation

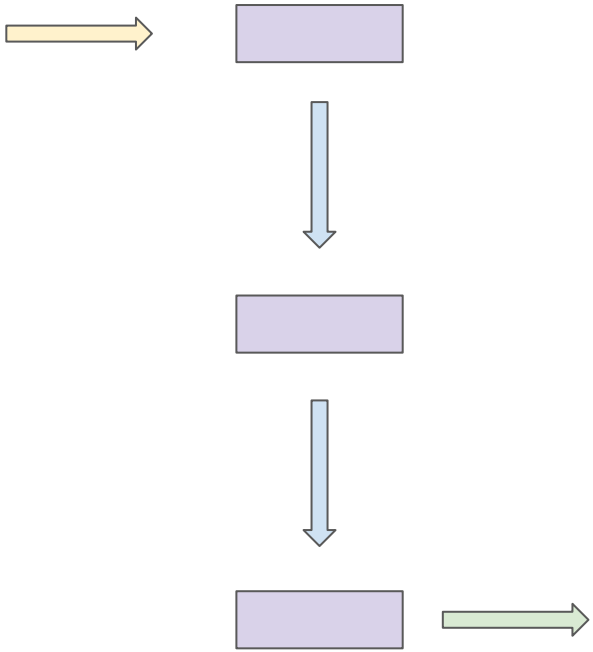
On the fly

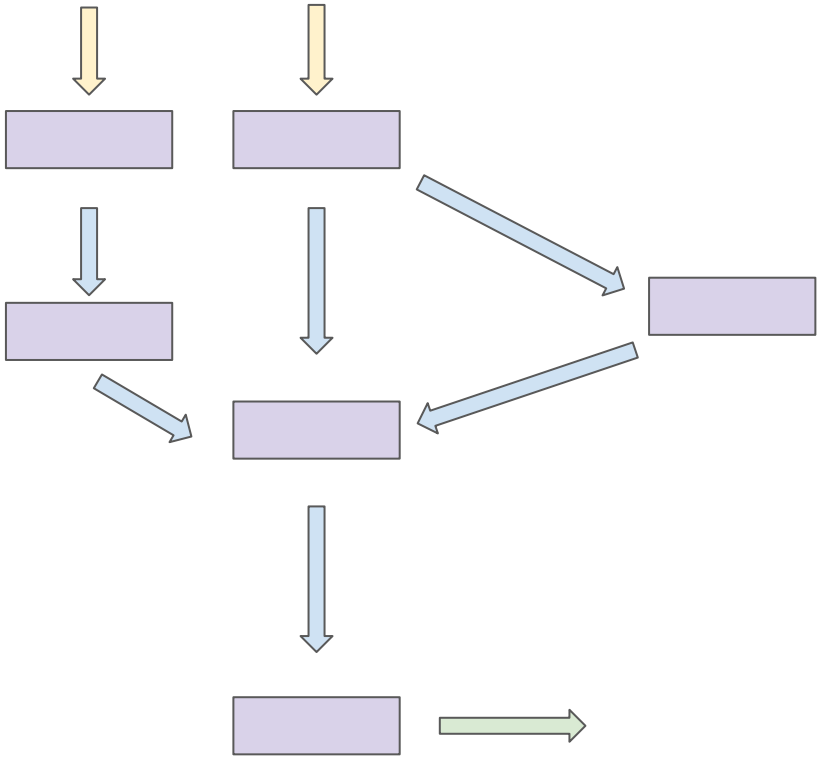
Creation


Creation
Transformation

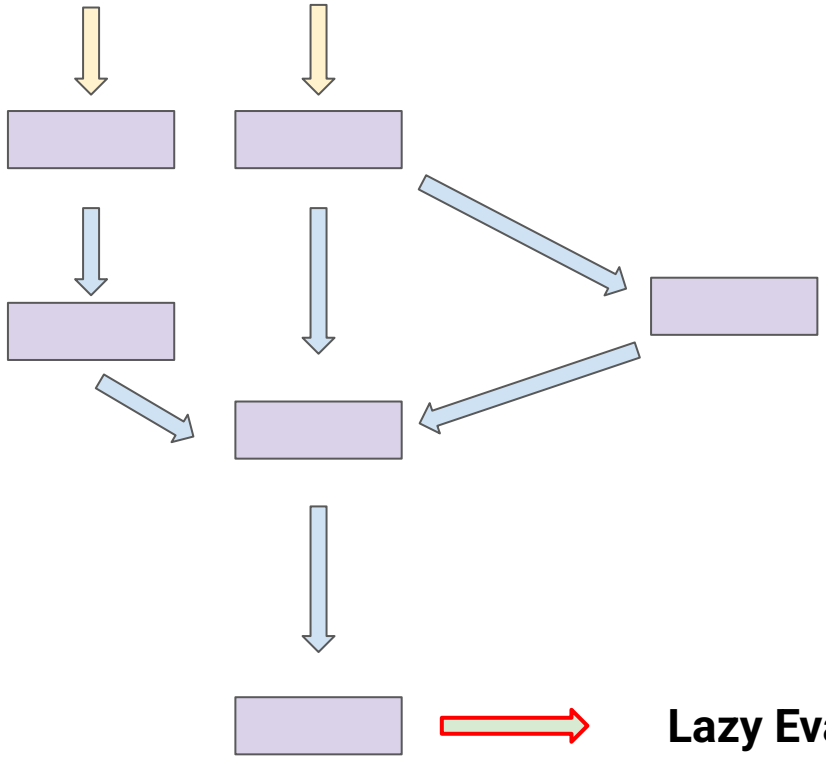

Creation
Transformation
Action


Creation
Transformation
Action
On Screen

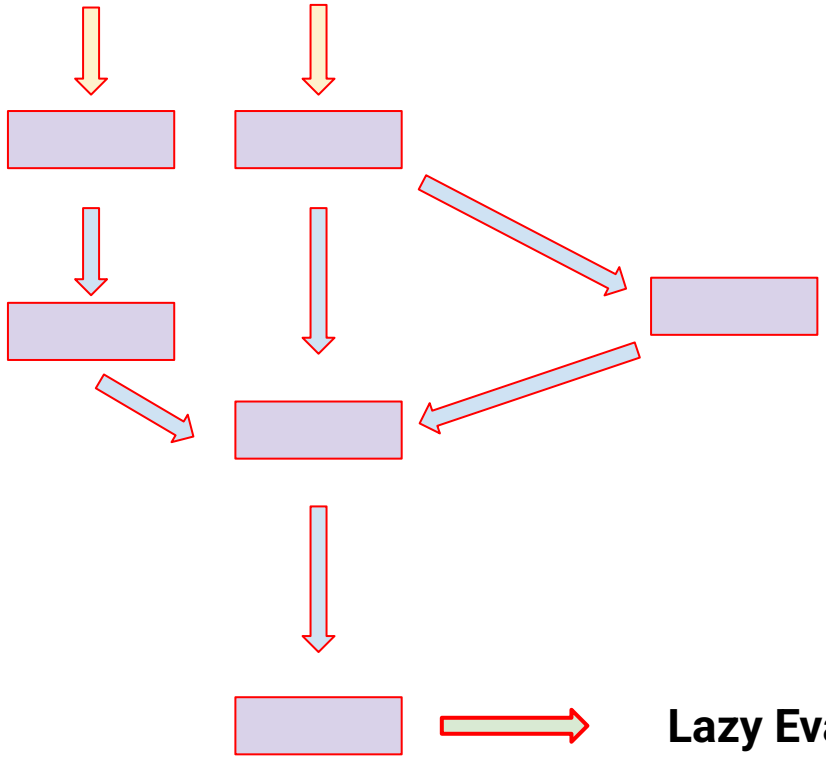





APACHE
Spark™



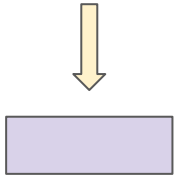
Lazy Evaluation



Lazy Evaluation

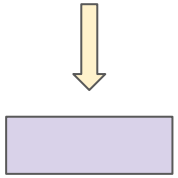

```
for $book in xml-files("books/*.xml")
return $book/isbn
```

spark.read.text(...)

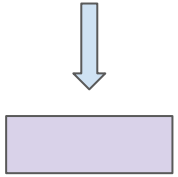



```
for $book in xml-files("books/*.xml")
return $book/isbn
```

spark.read.text(...)

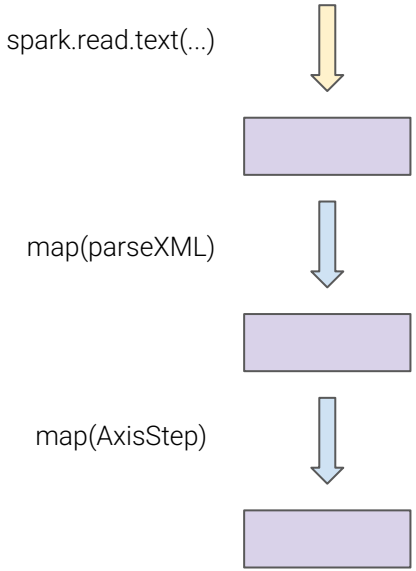


map(parseXML)

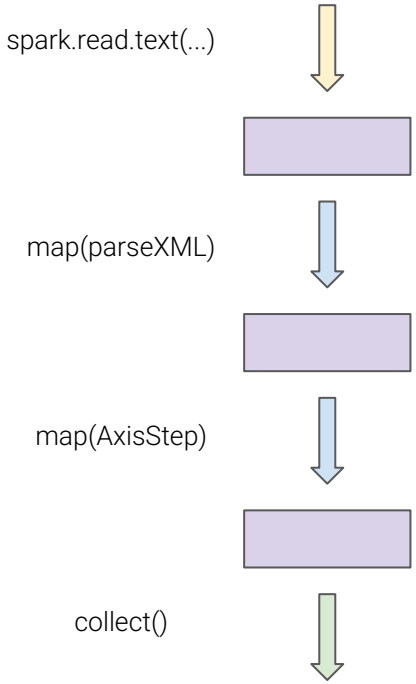


```
for $book in xml-files("books/*.xml")
return $book/isbn
```

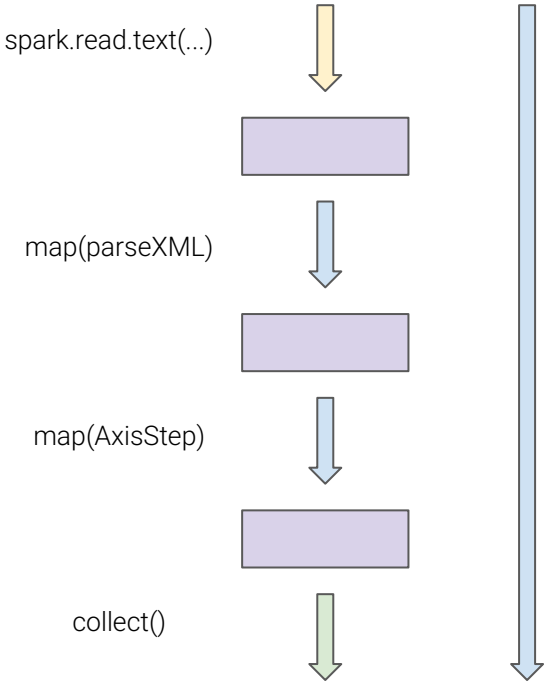

```
for $book in xml-files("books/*.xml")
return $book/isbn
```




```
for $book in xml-files("books/*.xml")
return $book/isbn
```

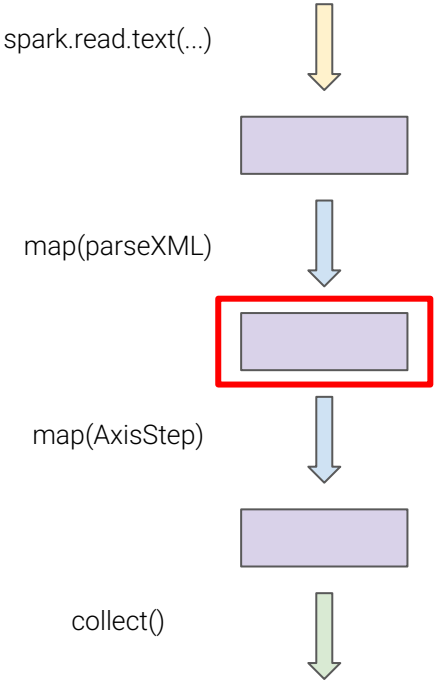



```
for $book in xml-files("books/*.xml")
return $book/isbn
```

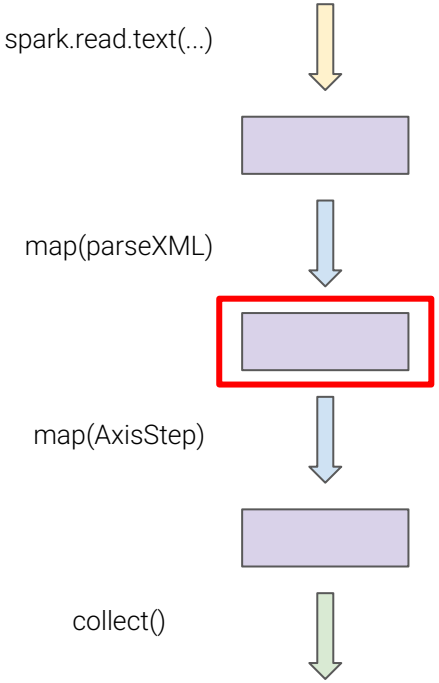


Flow of tuples

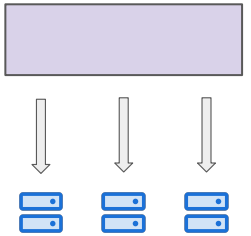

```
for $book in xml-files("books/*.xml")
return $book/isbn
```



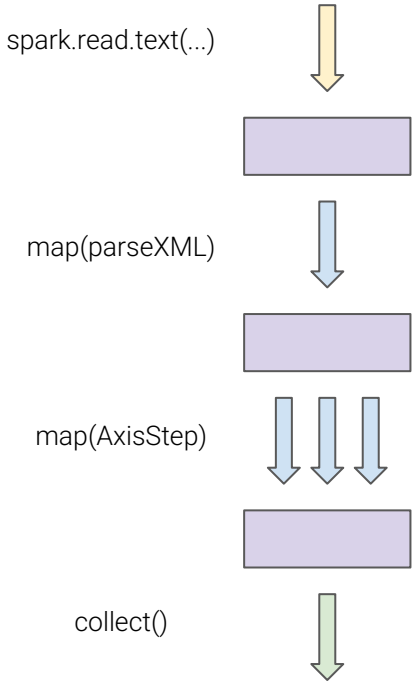

```
for $book in xml-files("books/*.xml")
return $book/isbn
```



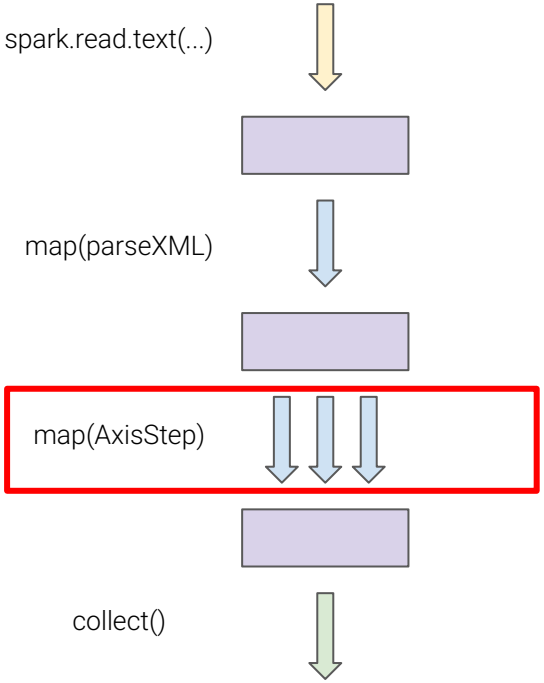
Distributed



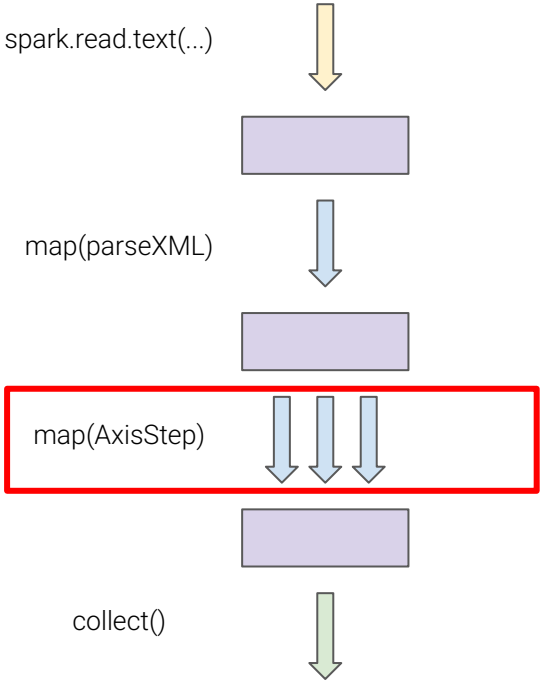

```
for $book in xml-files("books/*.xml")
return $book/isbn
```




```
for $book in xml-files("books/*.xml")
return $book/isbn
```




```
for $book in xml-files("books/*.xml")
return $book/isbn
```

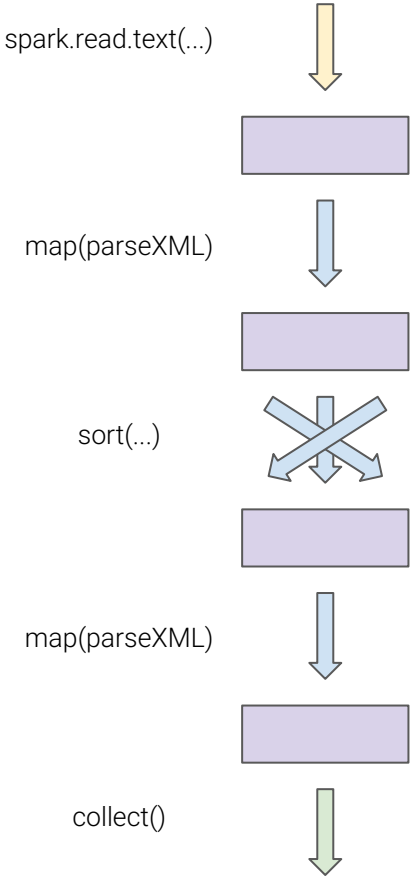


No ordering constraint

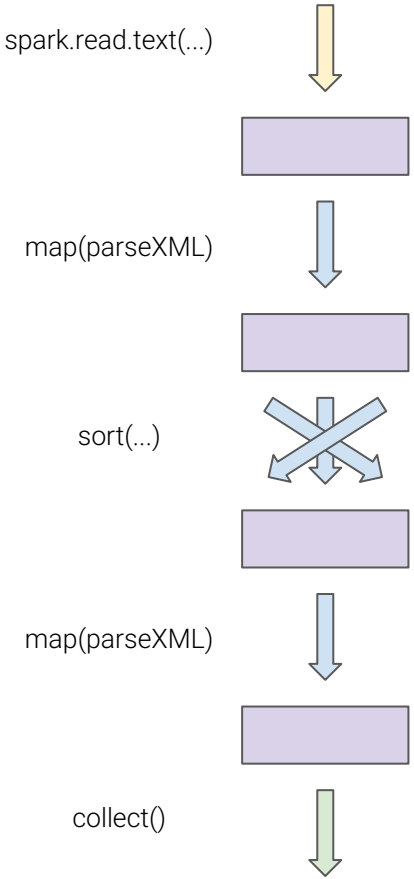
=

Full Parallelism

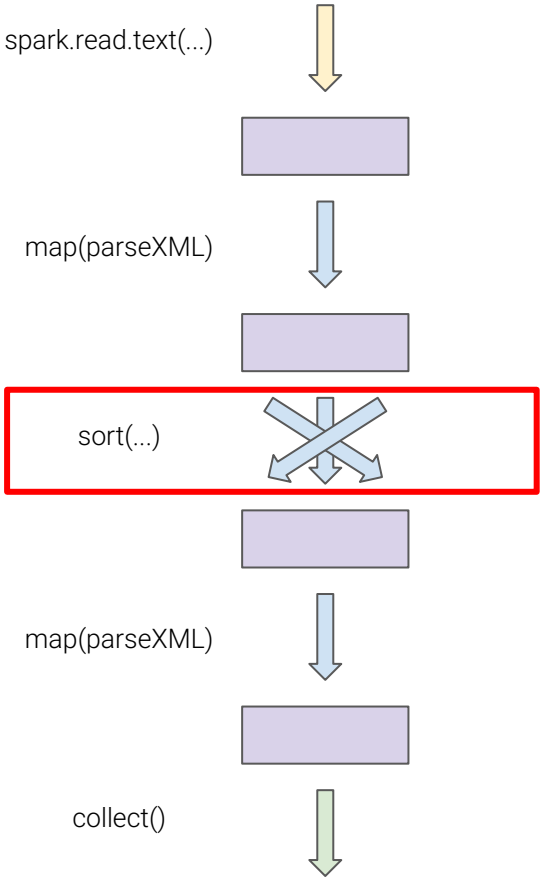

```
for $book in xml-files("books/*.xml")
order by $book/year ascending
return $book/isbn
```



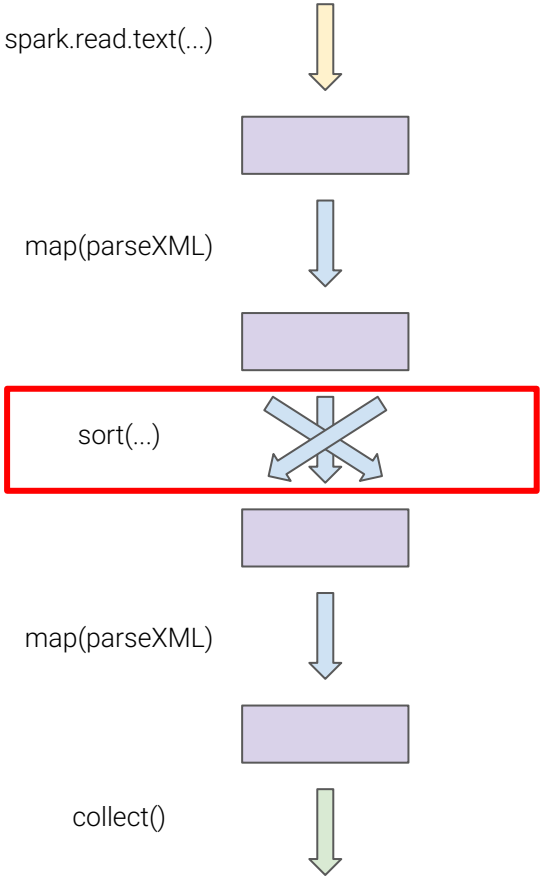

```
for $book in xml-files("books/*.xml")
order by $book/year ascending
return $book/isbn
```




```
for $book in xml-files("books/*.xml")
order by $book/year ascending
return $book/isbn
```

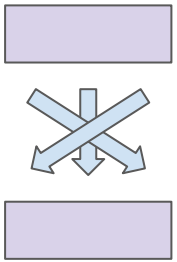



```
for $book in xml-files("books/*.xml")
order by $book/year ascending
return $book/isbn
```



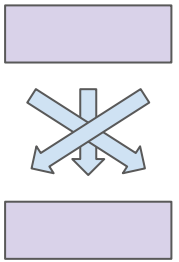
Ordering constraint
=
Shuffling

Shuffling is expensive



Shuffling is expensive

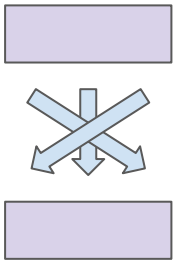
Shuffling = Moving partitions around the cluster



Shuffling is expensive

Shuffling = Moving partitions around the cluster

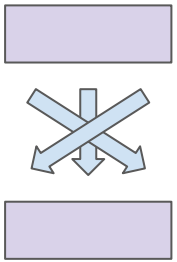
- Network operations



Shuffling is expensive

Shuffling = Moving partitions around the cluster

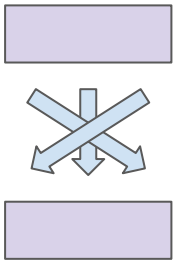
- Network operations
- Serialization + Deserialization
    - Memory Representation → Bits for transmission



Shuffling is expensive

Shuffling = Moving partitions around the cluster

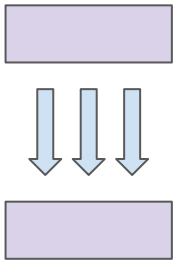
- Network operations
- Serialization + Deserialization
    - Memory Representation → Bits for transmission



O(N^2) Complexity! (N = Nodes)

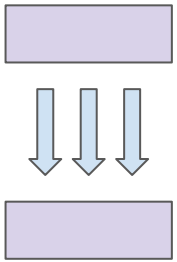
Shuffling vs non-shuffling expressions

Shuffling vs non-shuffling expressions

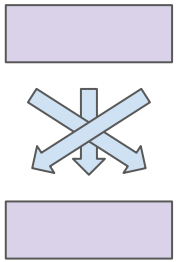


- For
- Let
- Where
- Arithmetic
- Conditionals
- …

Shuffling vs non-shuffling expressions

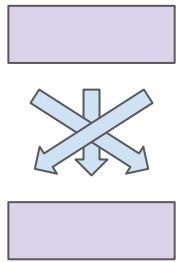


- For
- Let
- Where
- Arithmetic
- Conditionals
- …



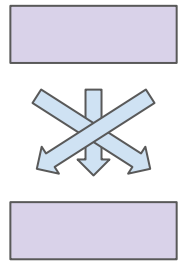
- Group by
- Order by
- distinct-values()
- Path Expressions
- …

Reducing shuffles in RumbleDB: the starting point



Reducing shuffles in RumbleDB: the starting point

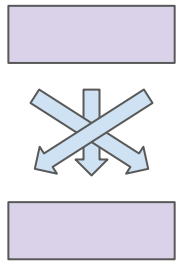
At every step:



Reducing shuffles in RumbleDB: the starting point

At every step:

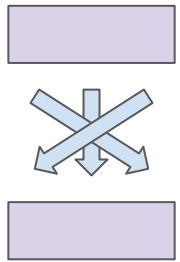
- Uniqueness-check → distinct()



Reducing shuffles in RumbleDB: the starting point

At every step:

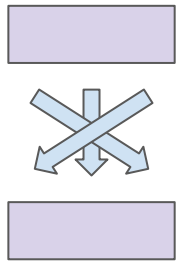
- Uniqueness-check → distinct()
- Path-based sorting → sort()



Reducing shuffles in RumbleDB: the starting point

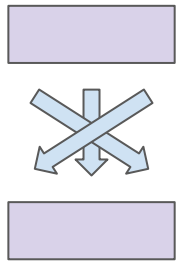
At every step:

- Uniqueness-check → distinct()
- Path-based sorting → sort()



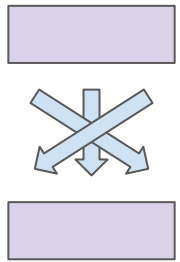
2 Shuffles per step!

Reducing shuffles in RumbleDB: optimizations



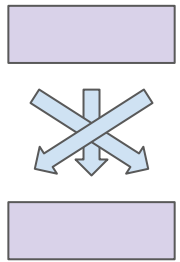
Reducing shuffles in RumbleDB: optimizations

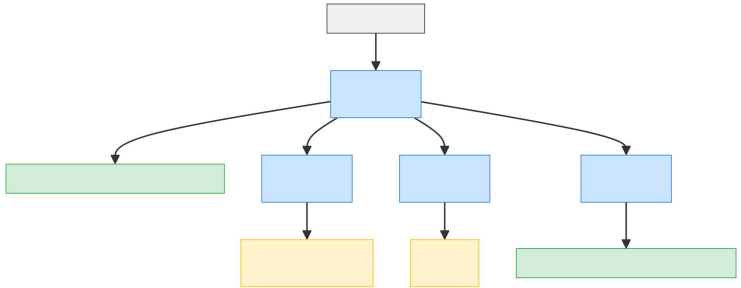
- **Optimization #1**:



Reducing shuffles in RumbleDB: optimizations

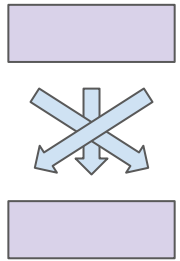
- **Optimization #1**:
  - Only forward steps

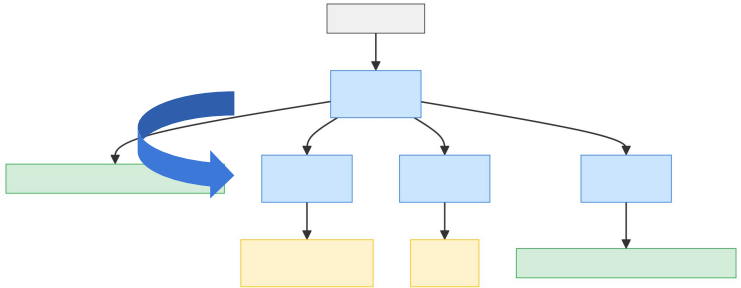




Reducing shuffles in RumbleDB: optimizations

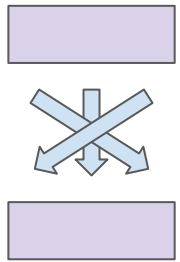
- **Optimization #1**:
  - Only forward steps

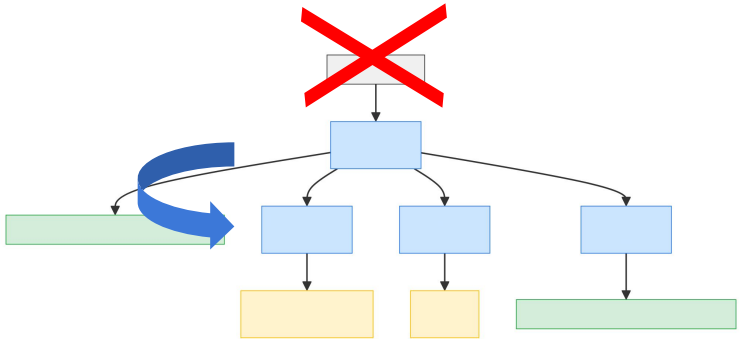




Reducing shuffles in RumbleDB: optimizations

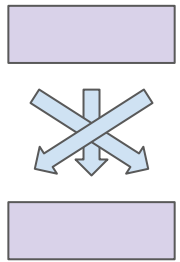
- **Optimization #1**:
  - Only forward steps → skip parent serialization

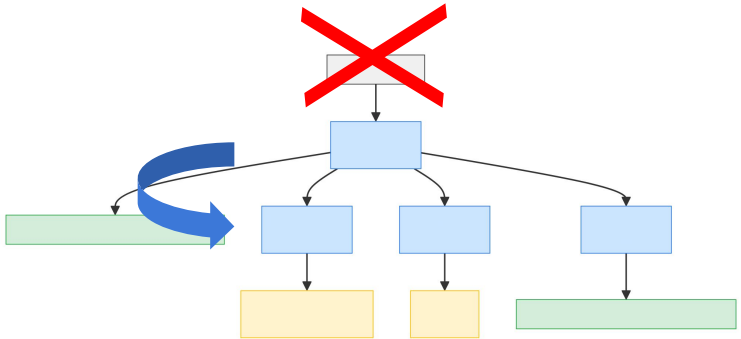




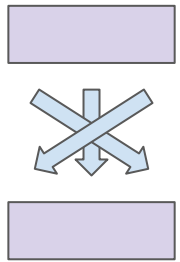
Reducing shuffles in RumbleDB: optimizations

- **Optimization #1**:
  - Only forward steps → skip parent serialization
  - Static analysis!



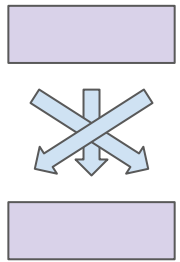


Reducing shuffles in RumbleDB: optimizations



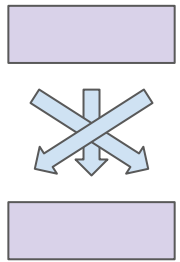
Reducing shuffles in RumbleDB: optimizations

- **Optimization # 2**: rework uniqueness + sorting



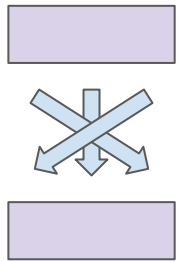
Reducing shuffles in RumbleDB: optimizations

- **Optimization # 2**: rework uniqueness + sorting
  - Group by document → groupBy()



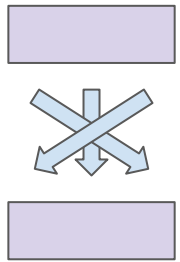
Reducing shuffles in RumbleDB: optimizations

- **Optimization # 2**: rework uniqueness + sorting
  - Group by document → groupBy()
  - Uniqueness and sorting checks → **locally**



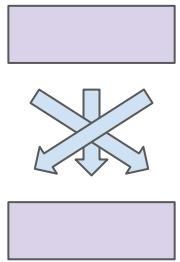
Reducing shuffles in RumbleDB: optimizations

- **Optimization # 2**: rework uniqueness + sorting
  - Group by document → groupBy()
  - Uniqueness and sorting checks → **locally**



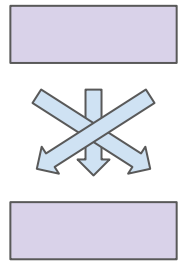
Only 1 Shuffle!

Reducing shuffles in RumbleDB: optimizations



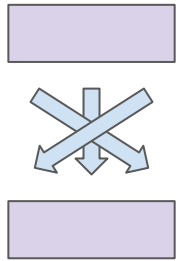
Reducing shuffles in RumbleDB: optimizations

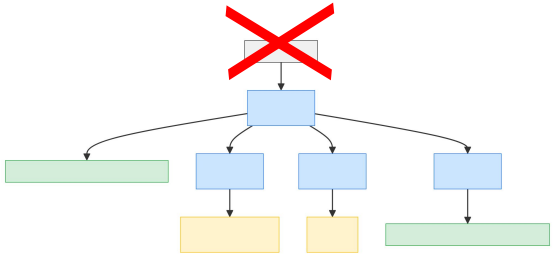
- **Experimental**: if



Reducing shuffles in RumbleDB: optimizations

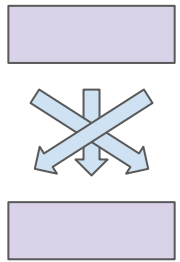
- **Experimental**: if
  - Parent pointers are removed (Optimization #1)

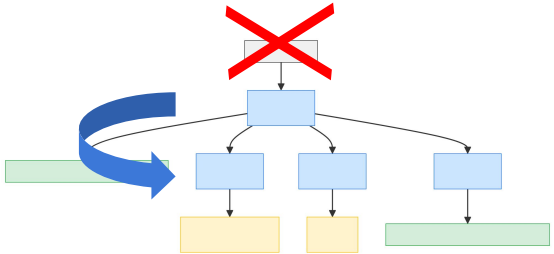




Reducing shuffles in RumbleDB: optimizations

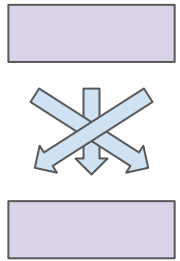
- **Experimental**: if
  - Parent pointers are removed (Optimization #1)
  - RHS is a forward step operation (e.g. $book/isbn)



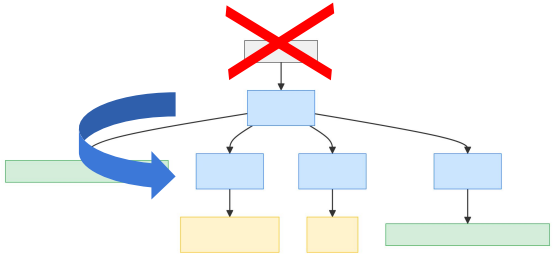


Reducing shuffles in RumbleDB: optimizations

- **Experimental**: if
  - Parent pointers are removed (Optimization #1)
  - RHS is a forward step operation (e.g. $book/isbn)

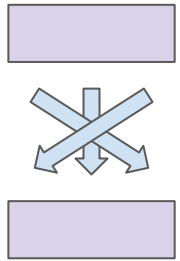


→ **skip uniqueness check + sorting**

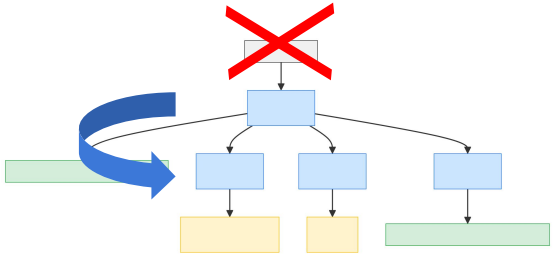


Reducing shuffles in RumbleDB: optimizations

- **Experimental**: if
  - Parent pointers are removed (Optimization #1)
  - RHS is a forward step operation (e.g. $book/isbn)



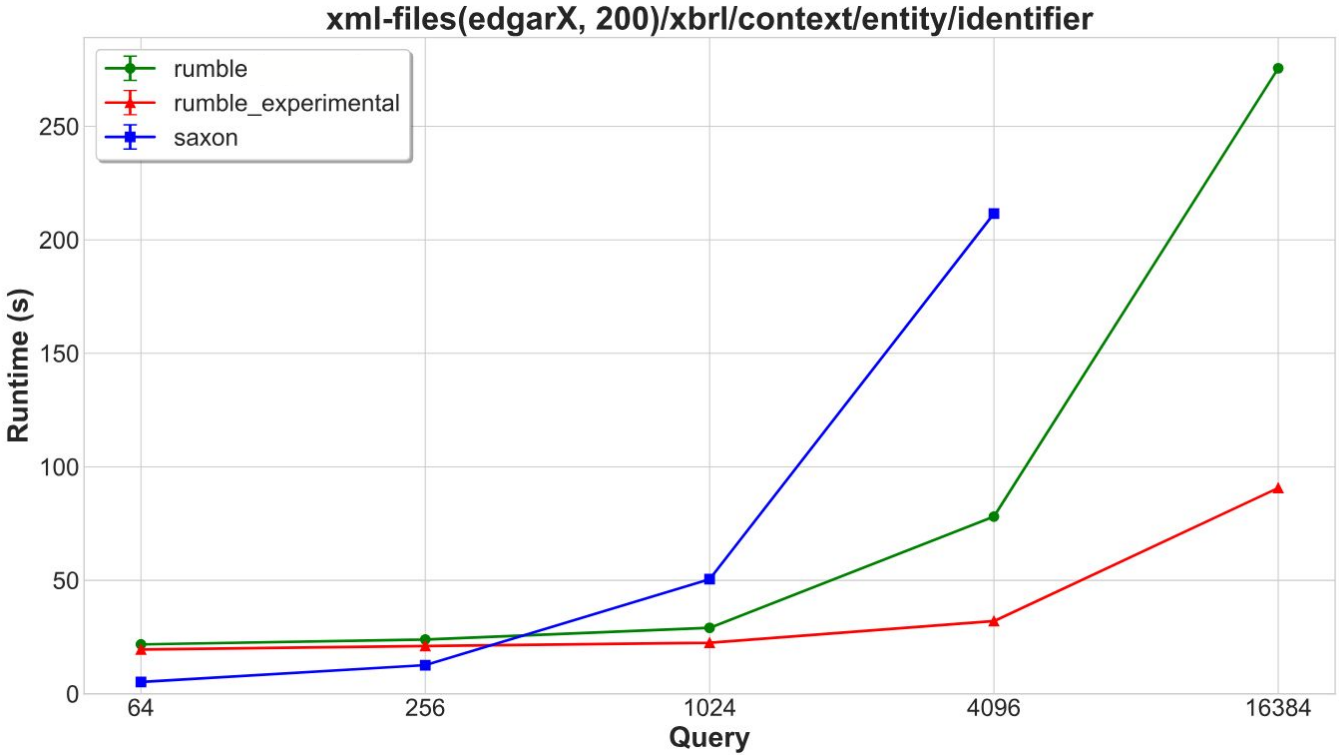
→ **skip uniqueness check + sorting**



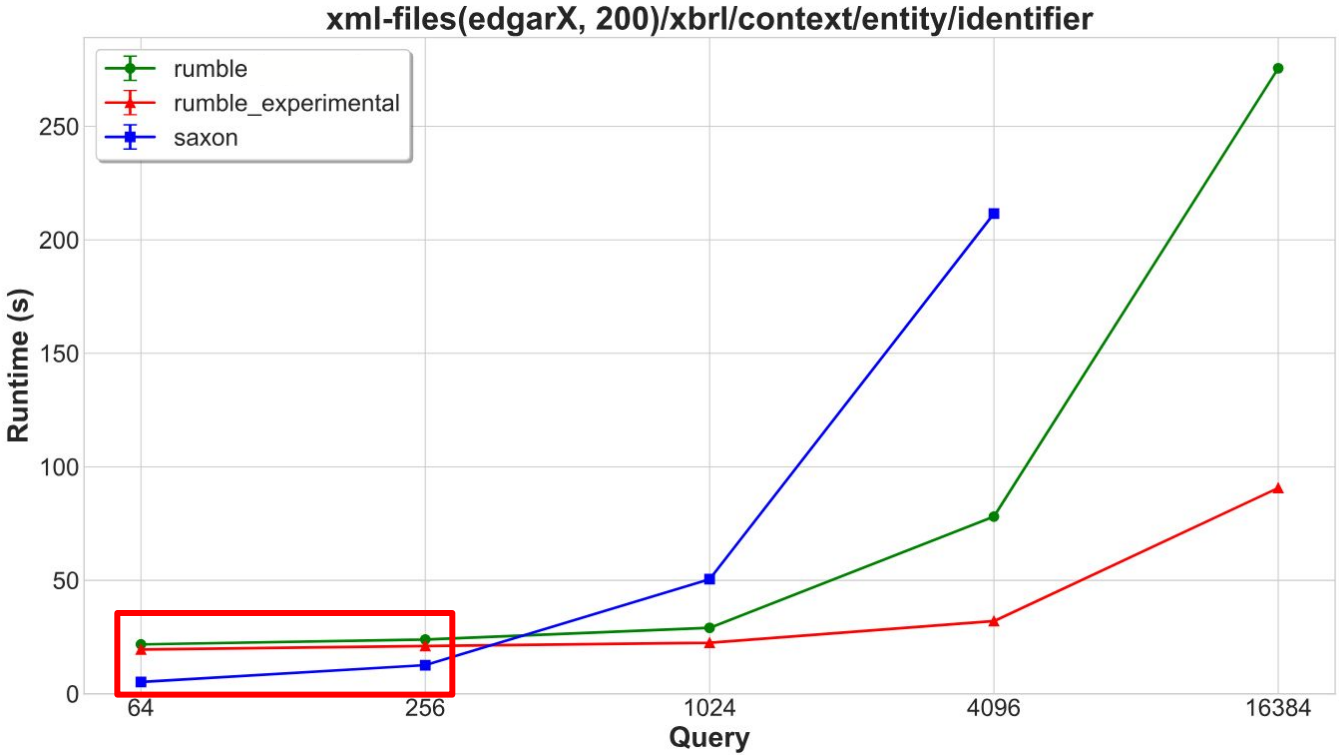
Disabled by default

Does parallel querying pay off?

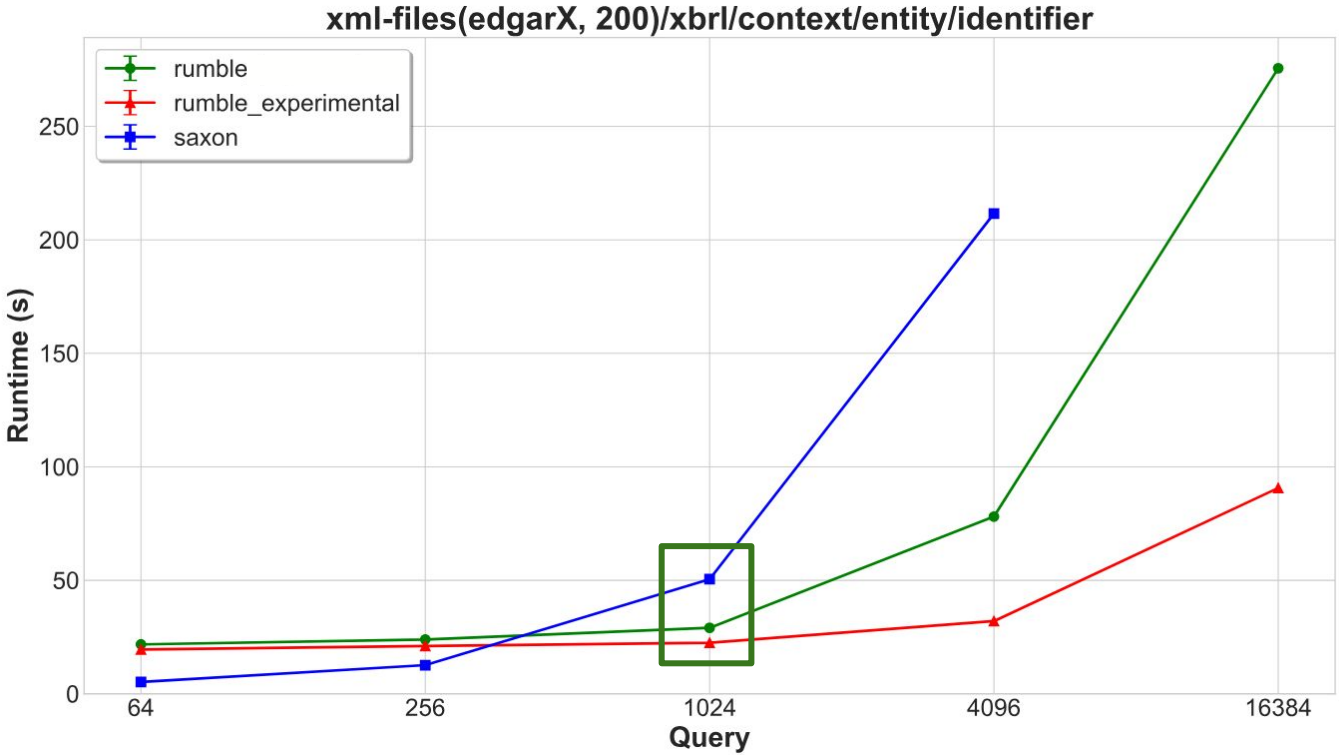
Performance of RumbleDB 1.23 and Saxon
AWS Cluster, 4x m5.24xlarge (96 vCPU, 384GB RAM)



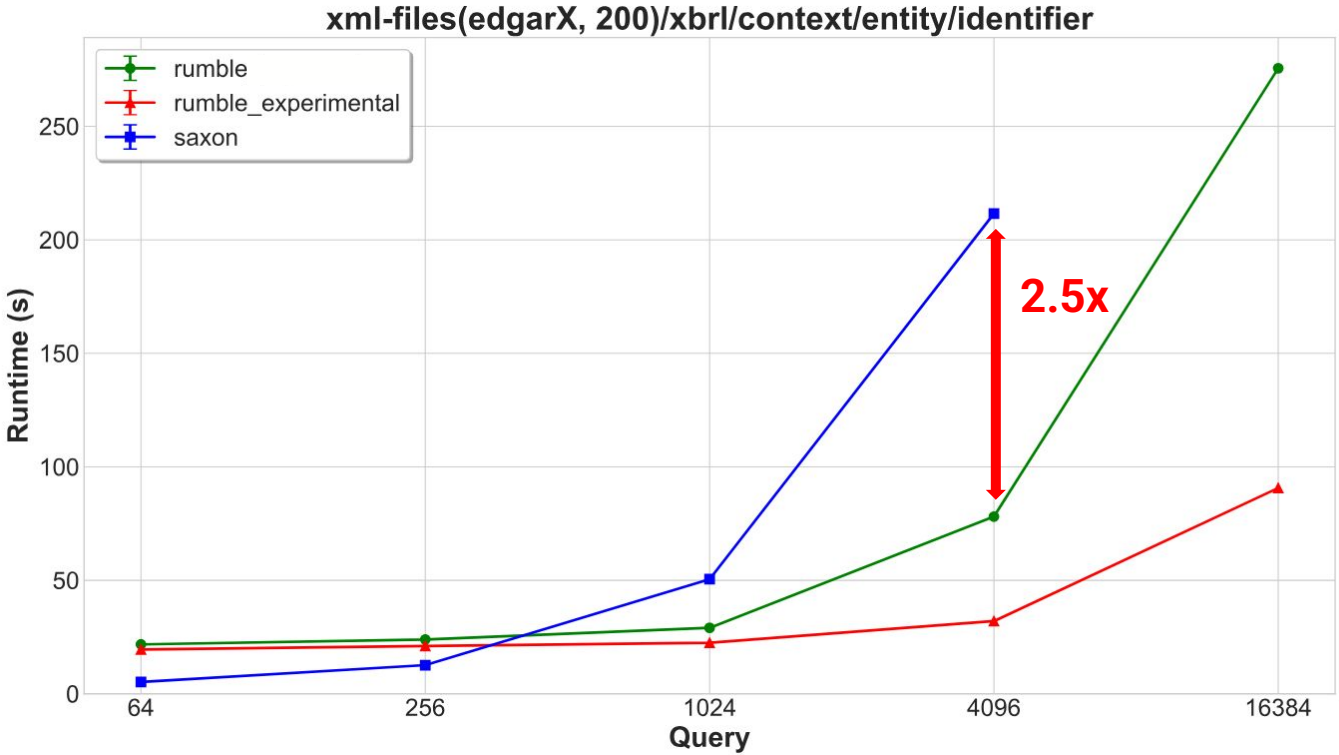
Performance of RumbleDB 1.23 and Saxon
AWS Cluster, 4x m5.24xlarge (96 vCPU, 384GB RAM)



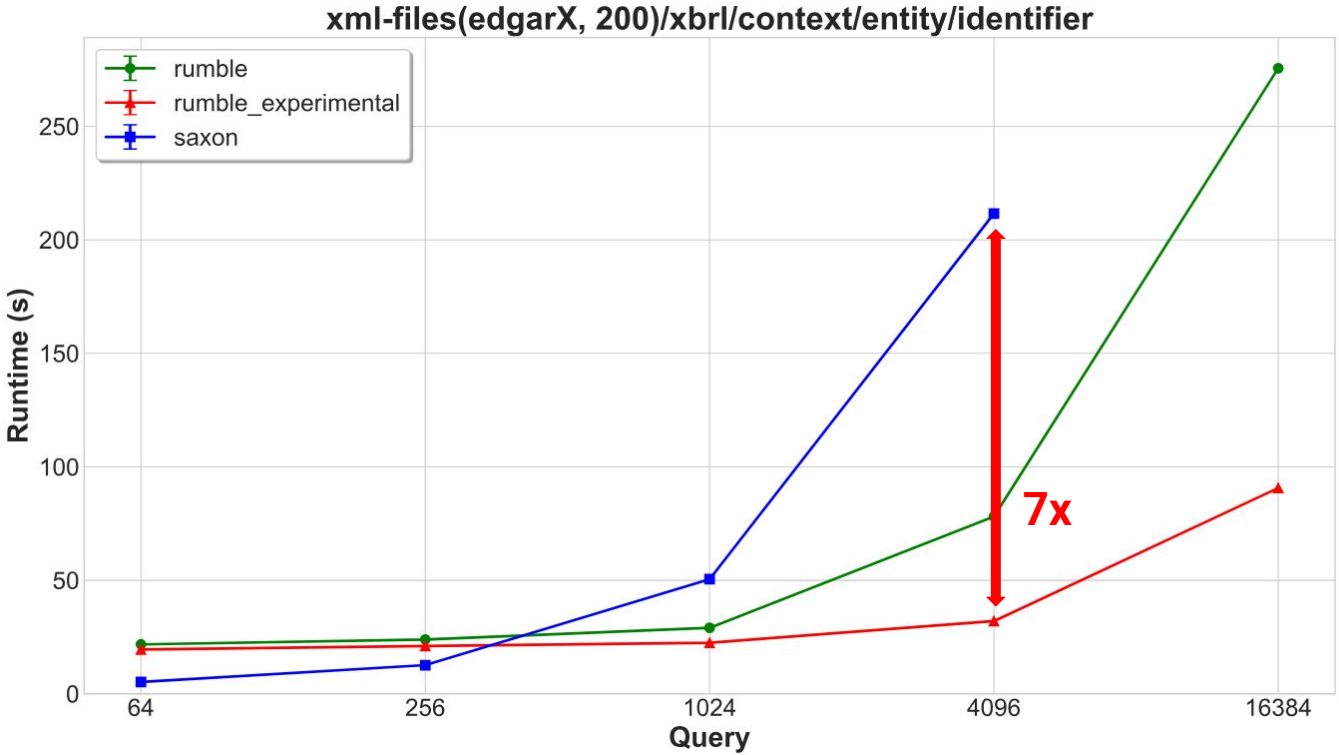
Performance of RumbleDB 1.23 and Saxon
AWS Cluster, 4x m5.24xlarge (96 vCPU, 384GB RAM)



Performance of RumbleDB 1.23 and Saxon
AWS Cluster, 4x m5.24xlarge (96 vCPU, 384GB RAM)



Performance of RumbleDB 1.23 and Saxon
AWS Cluster, 4x m5.24xlarge (96 vCPU, 384GB RAM)

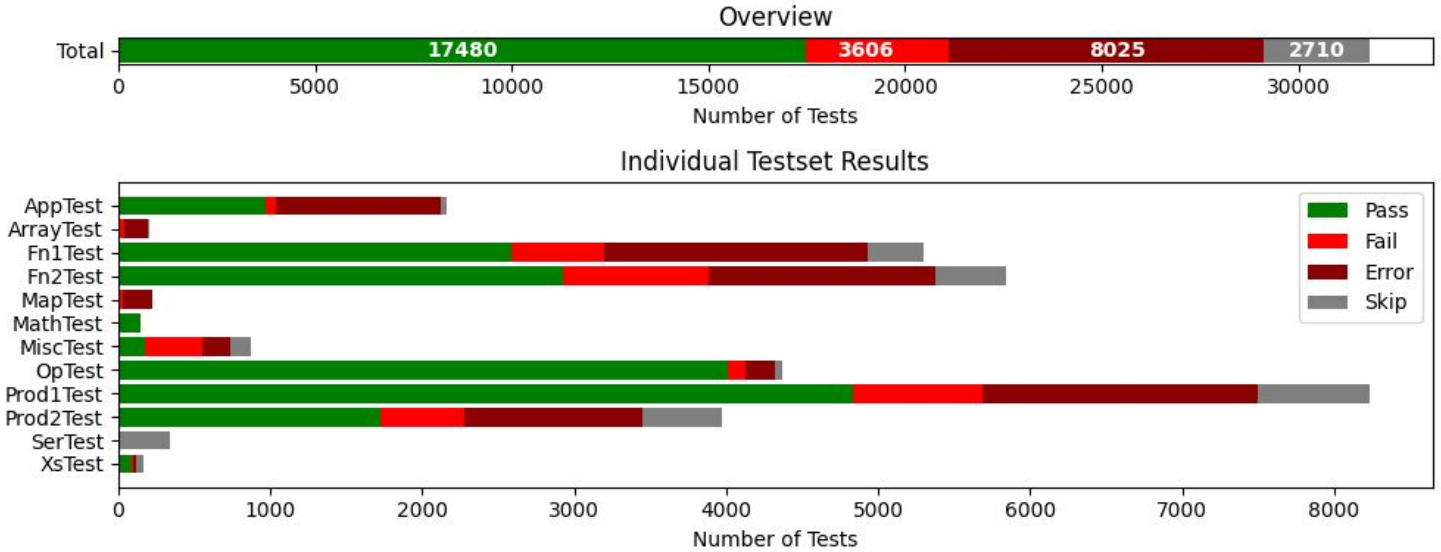


Querying massive datasets on a cluster is possible.

Querying massive datasets on a cluster is possible.
What’s next?

Rumble Test Suite Results

Rumble DB 2.0.0 “Lemon Ironwood”



Future Work

Future Work

- **Implement more XQuery features**

Future Work

- **Implement more XQuery features**
  - XDM features, XML-related functions, more constructors

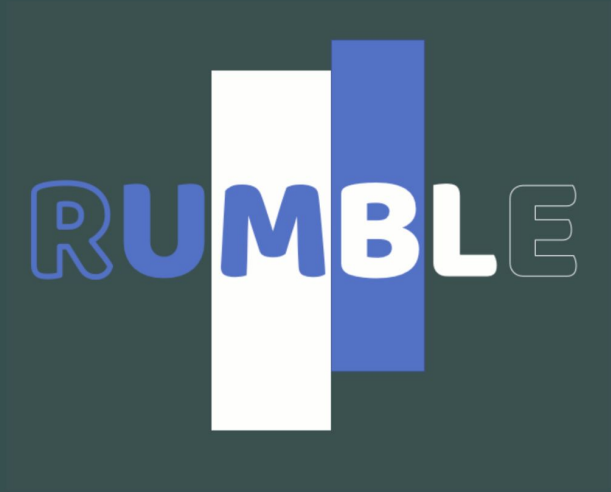
Future Work

- **Implement more XQuery features**
  - XDM features, XML-related functions, more constructors
- **Optimize, optimize, optimize**

Future Work

- **Implement more XQuery features**
  - XDM features, XML-related functions, more constructors
- **Optimize, optimize, optimize**
  - Close gap between RumbleDB and existing engines on a single node.

IT'S DATAFRAMES. JUST WITHOUT THE FRAMES.



2.0.0 "LEMON IRONWOOD" BETA

POWERED BY APACHE SPARK

Get started with RumbleDB

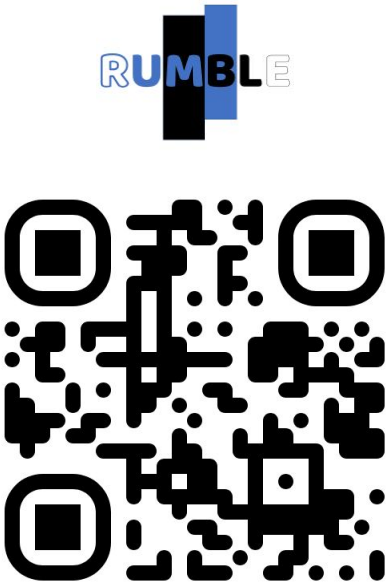
Get started with RumbleDB

```
pip install jsoniq
```

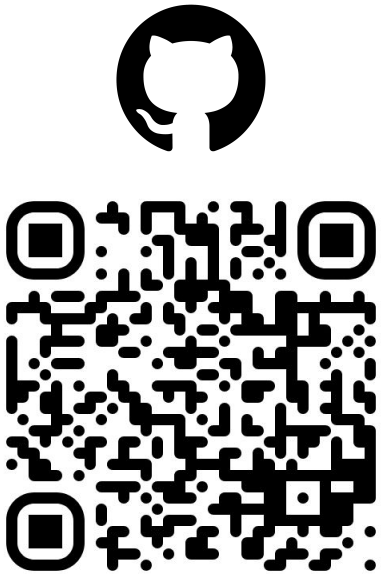
Get started with RumbleDB

colab.research.google.com/github/RumbleDB
/rumble/blob/master/RumbleSandbox.ipynb

Get started with RumbleDB



rumbledb.org



github.com/RumbleDB/rumble