The 15th International Semantic Web Conference, Kobe, Japan

Journal Article

Operator-aware Approach for Boosting Performance in Processing RDF streams

Special Issue on Stream Processing, Journal of Web Semantics

Danh Le Phuoc | Open Distributed Systems,
Technical University of Berlin, Germany

SMARTER Project – Grant No. 661180 (H2020-MSCA-IF-2014)

Agenda

- Overview of RDF Processing
- On Boosting The Processing Throughput
- Challenges in Incremental Evaluation
- Operator-Aware Approach
- Evaluation
- Summary

SMARTER Project – Grant No. 661180 (H2020-MSCA-IF-2014)

Overview of RDF Stream Processing(RSP)

$$S_{pickup} = \left\{ \begin{array}{l} :ride_1\ :taxi\ \ :89...CF4 \\ :ride_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48".} \end{array} \right\}.$$

$$S_{dropoff} = \left\{ \begin{array}{l} :ride_1\ :dropoffTime\ \texttt{"2013-01-01 15:18:10".} \\ :ride_1\ :triptime\ \texttt{382.} \end{array} \right\}.$$

$$S_{fare} = \left\{ \begin{array}{l} :trans_1\ :fare\ \texttt{7.} \\ :trans_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48".} \end{array} \right\}.$$

Stream data in RDF

Overview of RDF Stream Processing(RSP)

The query for continuous computation:

*"**hourly** riding rate of **active taxies** of **last 1000** payment transactions"*

$$S_{pickup} = \left\{ \begin{array}{l} :ride_1\ :taxi\ :89...CF4 \\ :ride_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48"}. \end{array} \right\}.$$

$$S_{dropoff} = \left\{ \begin{array}{l} :ride_1\ :dropoffTime\ \texttt{"2013-01-01 15:18:10"}. \\ :ride_1\ :triptime\ \texttt{382}. \end{array} \right\}.$$

$$S_{fare} = \left\{ \begin{array}{l} :trans_1\ :fare\ \texttt{7}. \\ :trans_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48"}. \end{array} \right\}.$$

Stream data in RDF

Overview of RDF Stream Processing(RSP)

*"**hourly** riding rate of **active taxies** of **last 1000** payment transactions"*

```
SELECT ?taxi  (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM   NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM   NAMED WINDOW :W2 ON  nyctaxi:pickup
                           [RANGE PT2H @:pickupTime]
FROM   NAMED WINDOW :W3 ON  nyctaxi:dropoff
                           [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```

Continuous Query in SPARQL-like language

$$S_{pickup} = \left\{ \begin{array}{l} :ride_1\ :taxi\ \ :89...CF4 \\ :ride_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48".} \end{array} \right\}.$$

$$S_{dropoff} = \left\{ \begin{array}{l} :ride_1\ :dropoffTime\ \texttt{"2013-01-01 15:18:10".} \\ :ride_1\ :triptime\ \texttt{382.} \end{array} \right\}.$$

$$S_{fare} = \left\{ \begin{array}{l} :trans_1\ :fare\ \texttt{7.} \\ :trans_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48".} \end{array} \right\}.$$

Stream data in RDF

European Commission

Overview of RDF Stream Processing(RSP)

Technische Universität Berlin

*"**hourly** riding rate of **active taxies** of **last 1000** payment transactions"*

```
SELECT ?taxi  (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM   NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM   NAMED WINDOW :W2 ON  nyctaxi:pickup
                           [RANGE PT2H @:pickupTime]
FROM   NAMED WINDOW :W3 ON  nyctaxi:dropoff
                           [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```

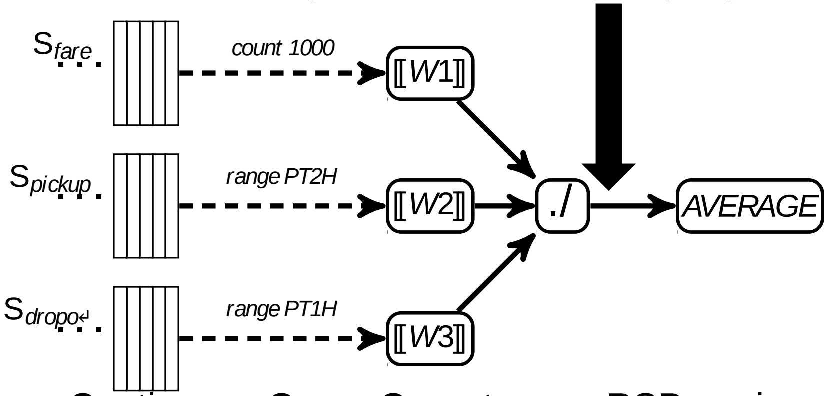
Continuous Query in SPARQL-like language

$$S_{pickup} = \left\{ \begin{array}{l} :ride_1\ :taxi\ :89...CF4 \\ :ride_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48"}. \end{array} \right\}.$$

$$S_{dropoff} = \left\{ \begin{array}{l} :ride_1\ :dropoffTime\ \texttt{"2013-01-01 15:18:10"}. \\ :ride_1\ :triptime\ 382. \end{array} \right\}.$$

$$S_{fare} = \left\{ \begin{array}{l} :trans_1\ :fare\ 7. \\ :trans_1\ :pickupTime\ \texttt{"2013-01-01 15:11:48"}. \end{array} \right\}.$$

Stream data in RDF



Continuous Query Operators on RSP engines

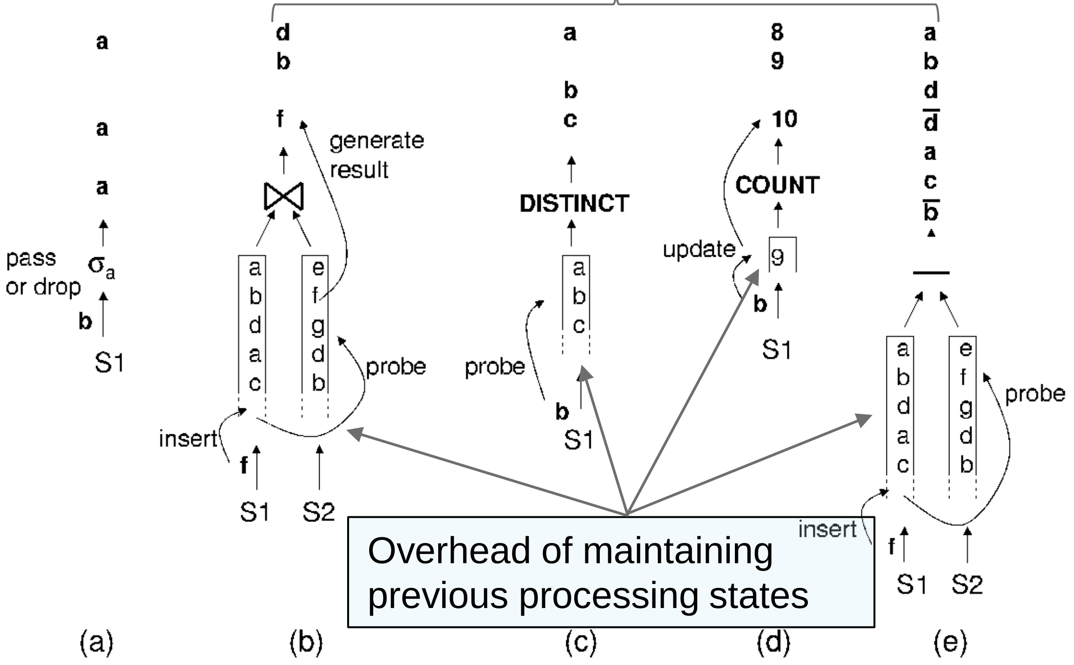
Open Distributed Sytems ODS

On boosting the processing throughput of RSP engines

- Experience from Implementation CQELS Execution Framework: **Using off-the-shelf data structures and algorithms are not enough!!??**
  - Hardly can reach 10000 operator executions/second on large windows(100k-1M entries)
  - Big overhead of using row-based data structures
- **Bottom-up perspective:** investigating closely to data structures and algorithms
  - ***Highly efficient data structures*** for maintaining processing states
  - Sophisticate ***incremental evaluation algorithms*** of query operators

Incremental Evaluation for continuous operators in a Nutshell

stateful sliding window operators:
reuse previous computing effort



Overhead of maintaining previous processing states

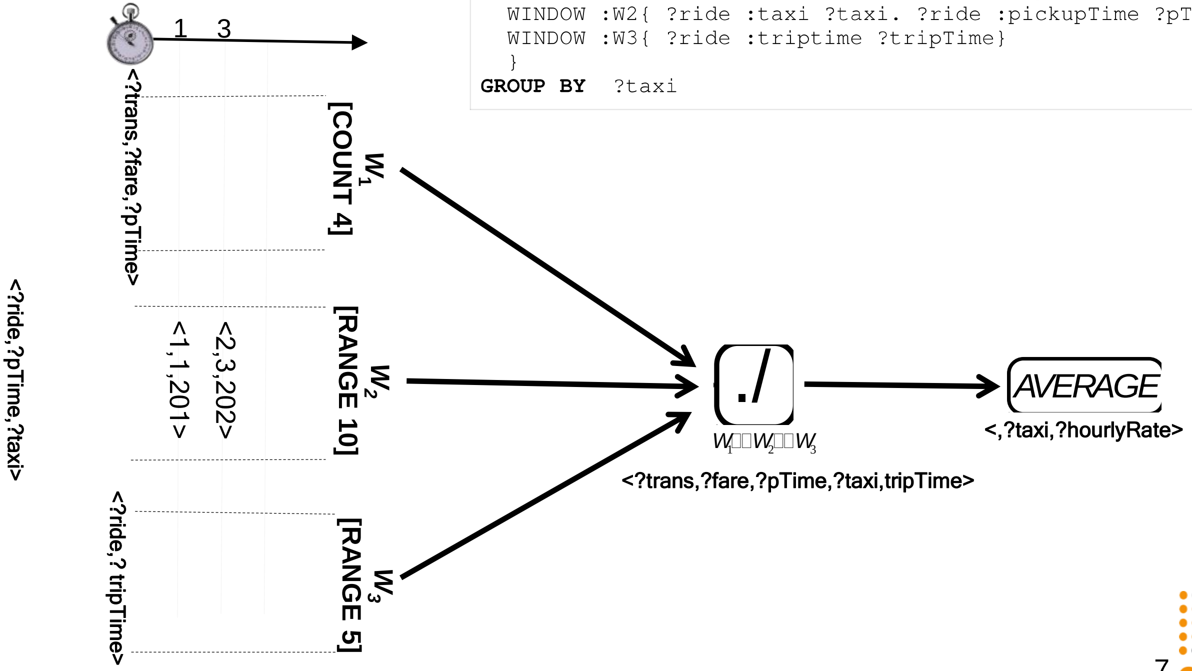
L. Golab, M. T. Özsu, Data stream management, Synthesis Lectures on Data Management (2010) 1–73.

Incremental Evaluation of Continuous Queries over RDF Stream: Issues and Challenges

- **Row-based data structure** is not suitable for :
  - very small RDF data elements (encoded as fixed-size integers)
  - unusually large number individual data points (millions of mappings/RDF nodes are generated/evicted per second)
- **Timestamping or negative-tuple solutions** for incremental computation of RDF data elements and mappings have technical issues:
  - Auxiliary data **(extra timestamps or negative tuples)** might be bigger than original data
  - Other limitations of state-of-art techniques (**double computation in evicting expired computing state**)

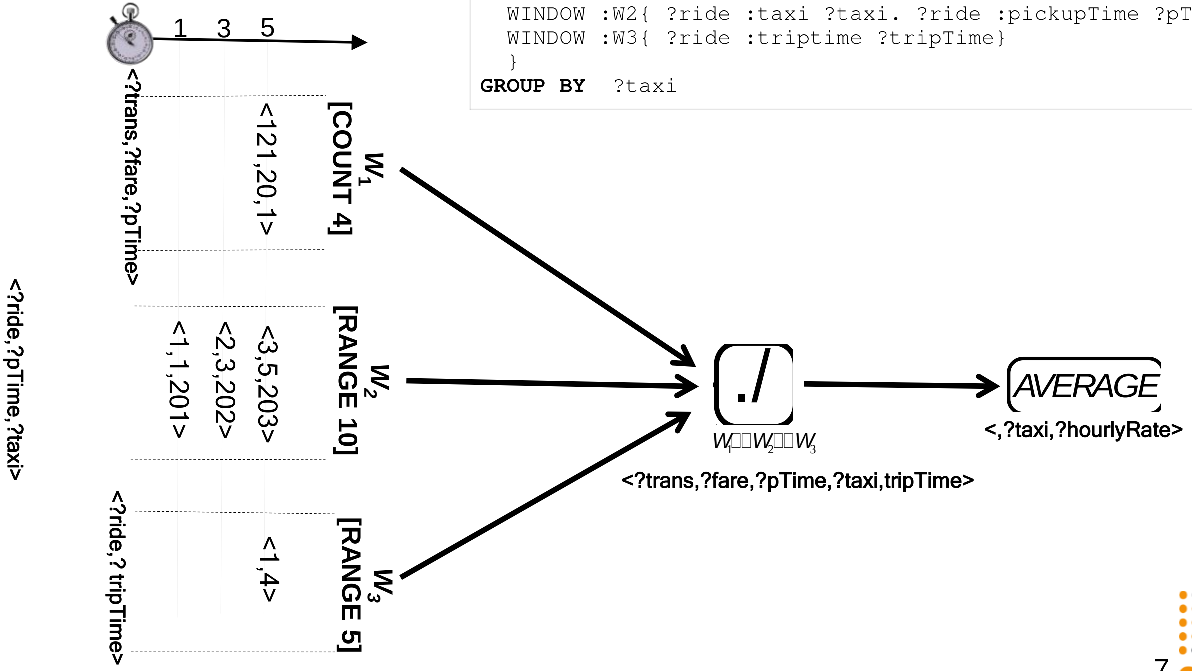
Incremental Evaluation of Continuous Query: sliding data flow

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



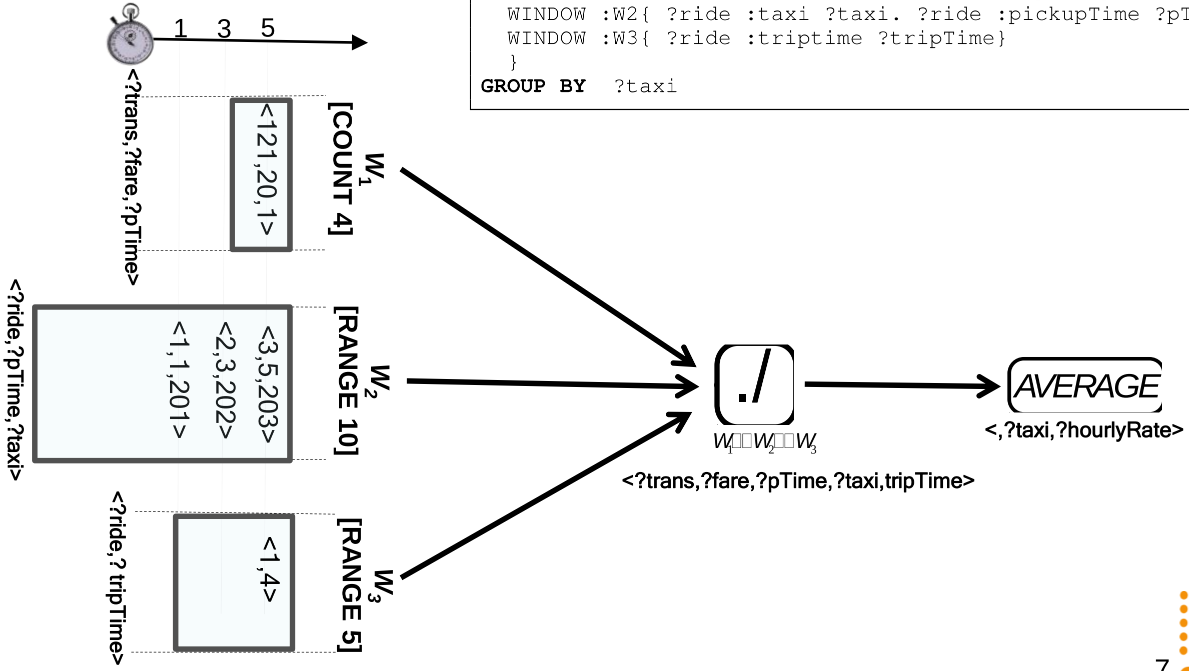
Incremental Evaluation of Continuous Query: sliding data flow

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



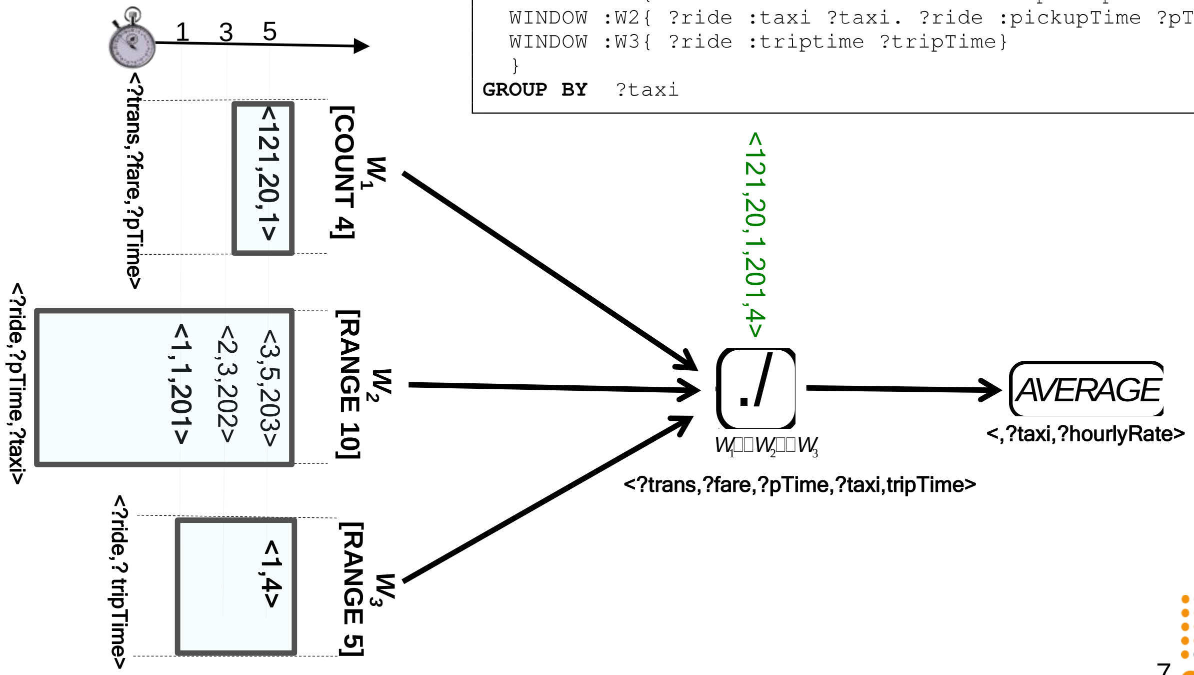
Incremental Evaluation of Continuous Query: sliding data flow

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



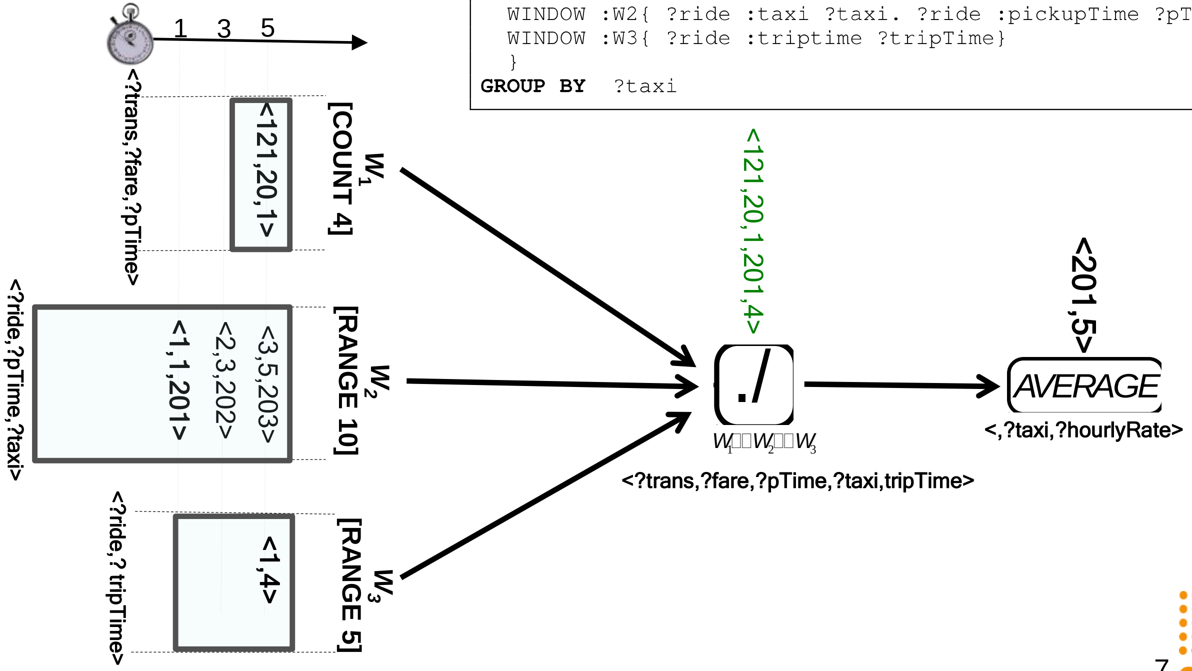
Incremental Evaluation of Continuous Query: sliding data flow

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```

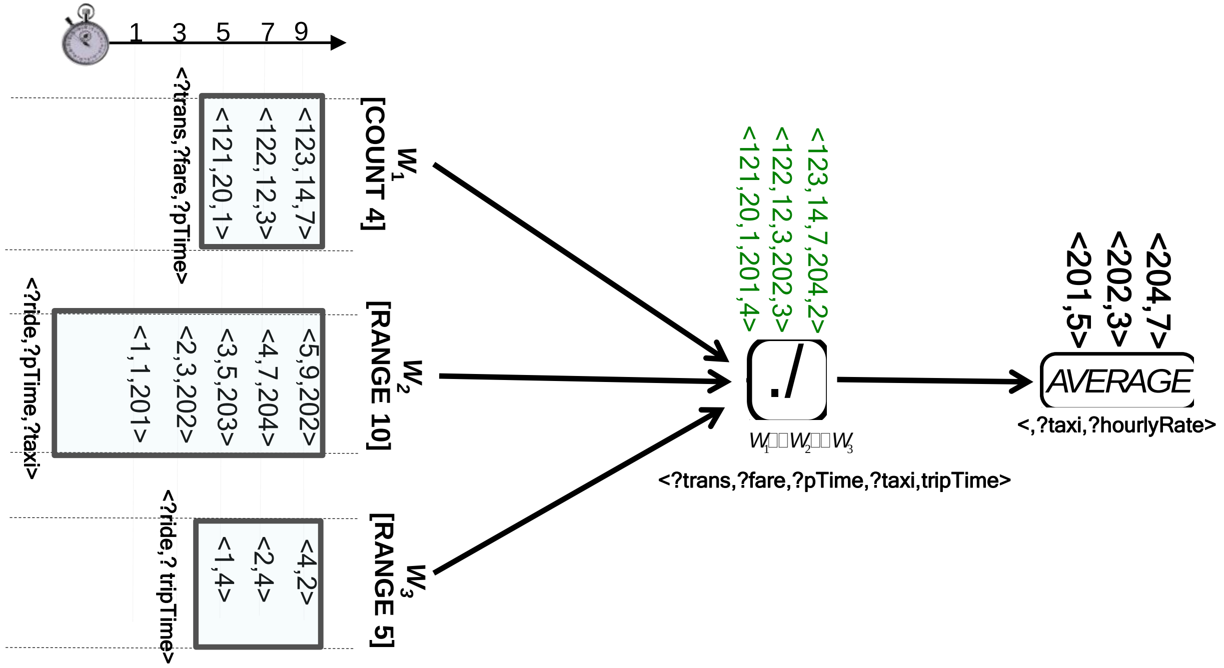


Incremental Evaluation of Continuous Query: sliding data flow

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```

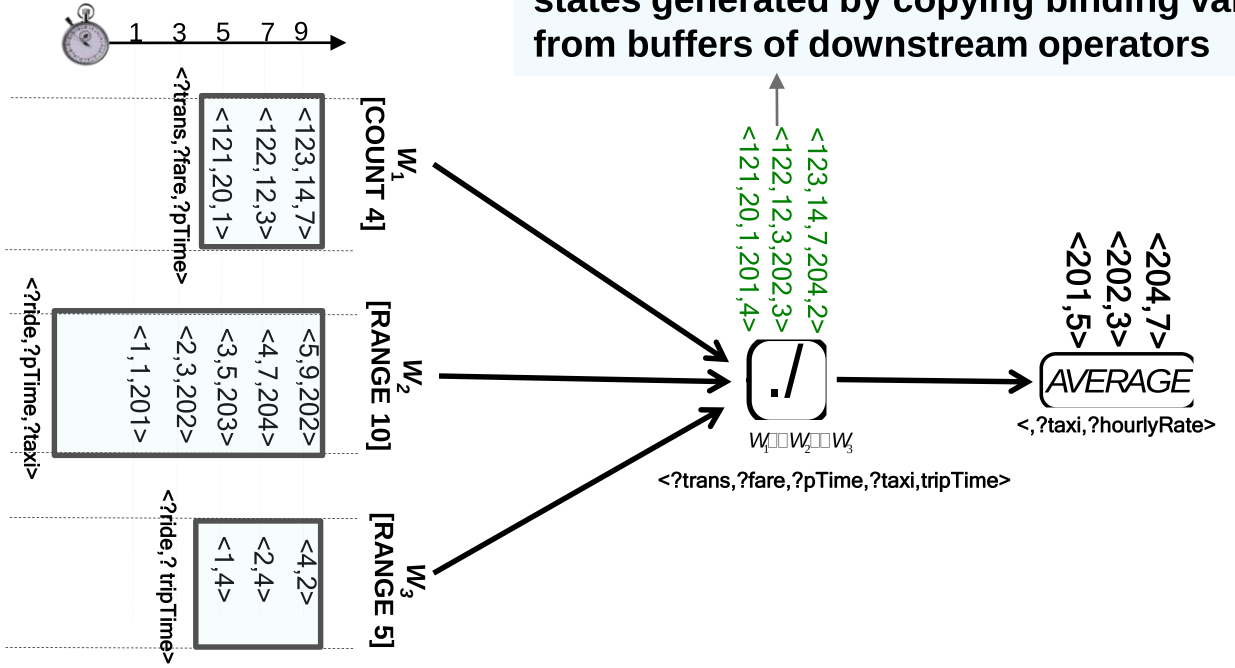


Incremental Evaluation of Continuous Query: generating processing state



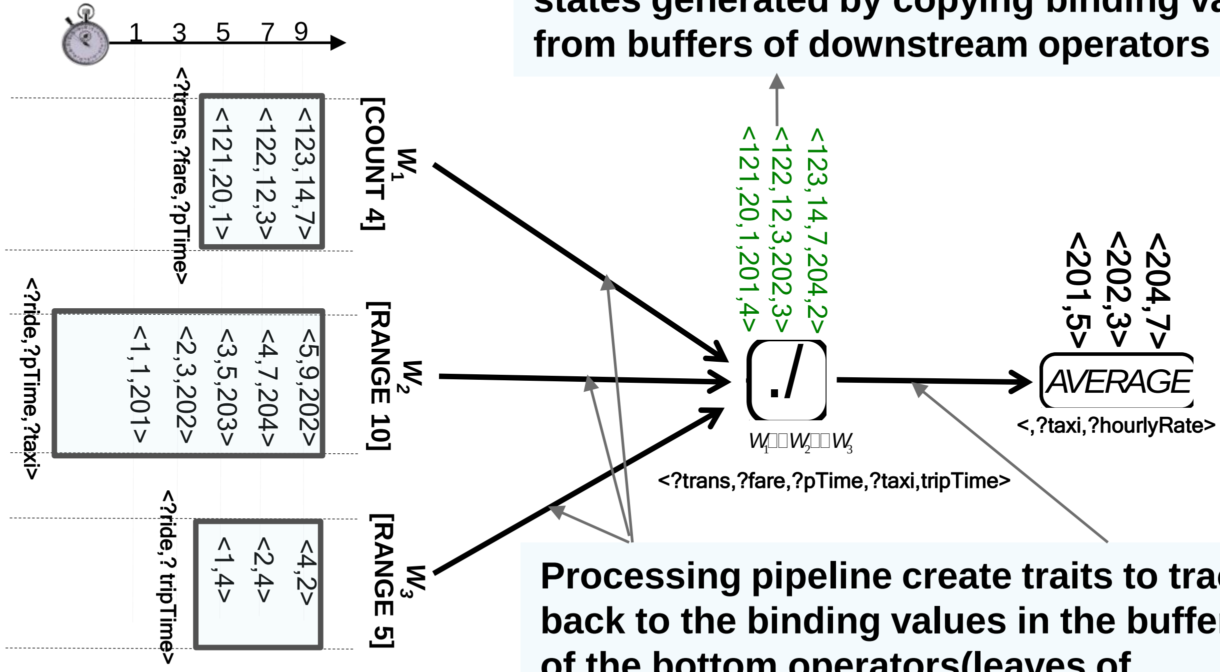
Incremental Evaluation of Continuous Query: generating processing state

Large number of intermediate processing states generated by copying binding values from buffers of downstream operators



Incremental Evaluation of Continuous Query: generating processing state

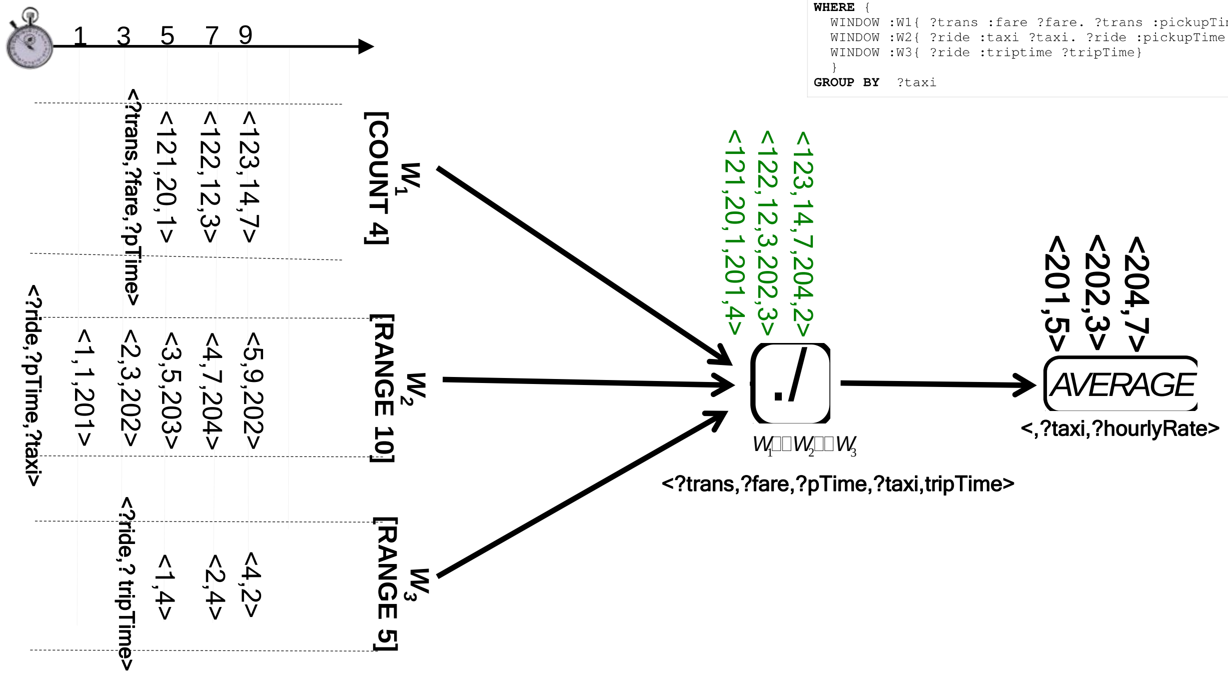
Large number of intermediate processing states generated by copying binding values from buffers of downstream operators



Processing pipeline create traits to trace back to the binding values in the buffers of the bottom operators(leaves of operator tree)

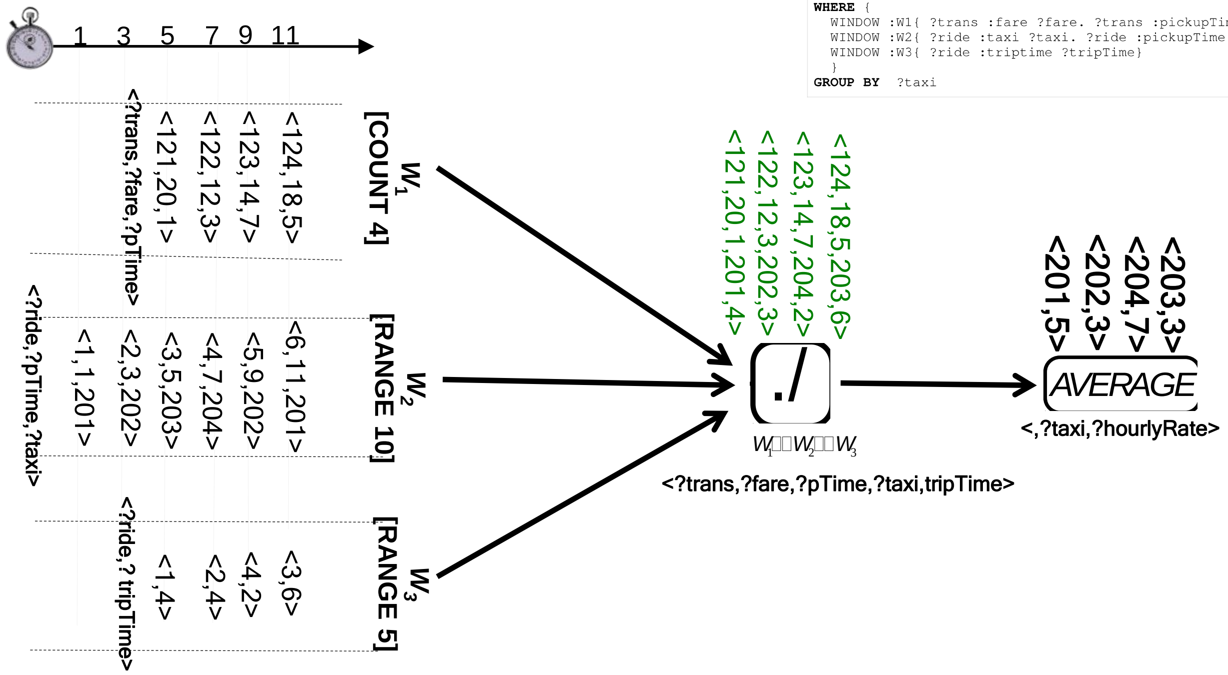
Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                             [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                             [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



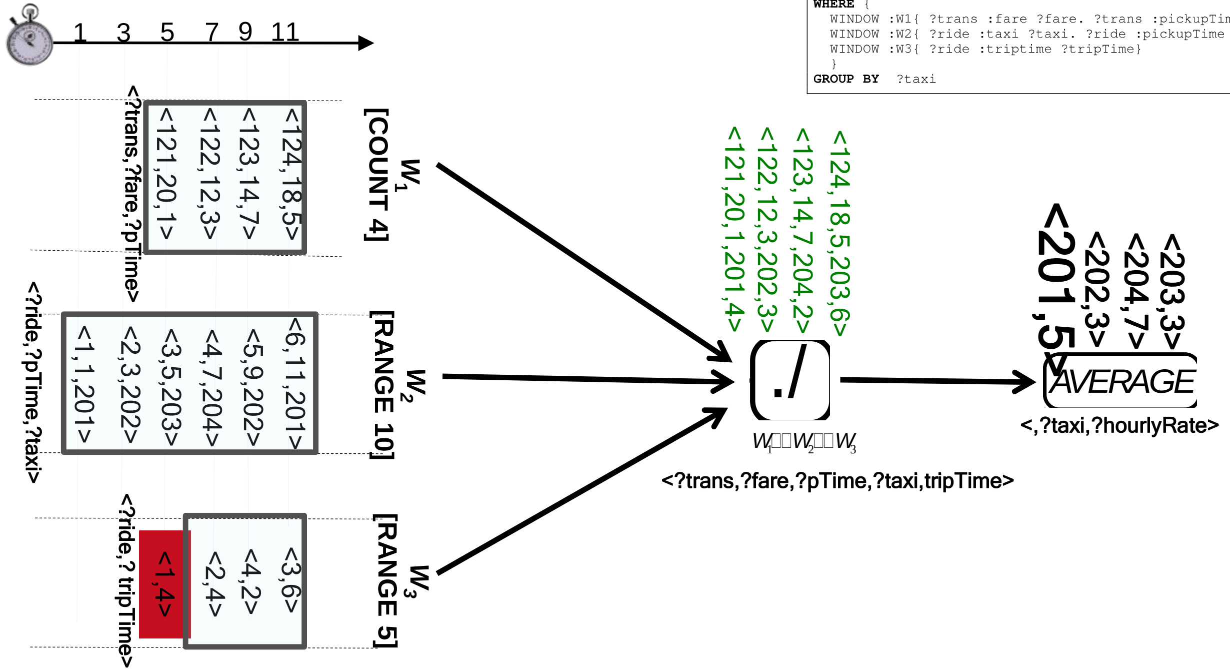
Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi  (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                          [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                          [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



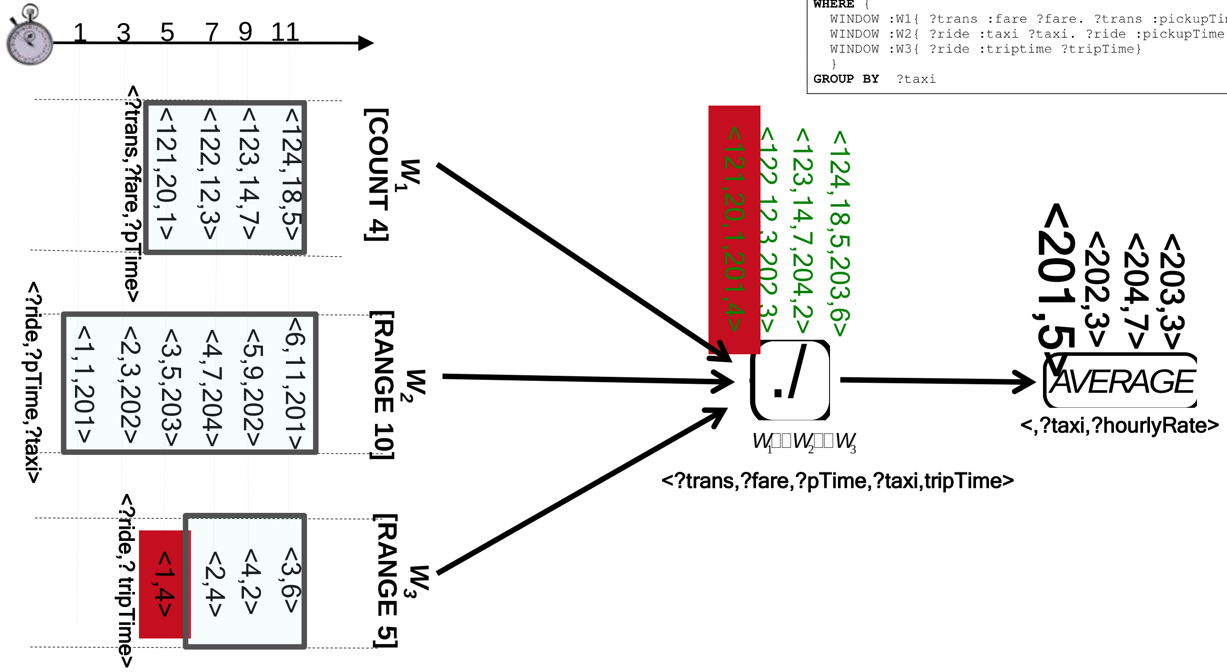
Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                             [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                             [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



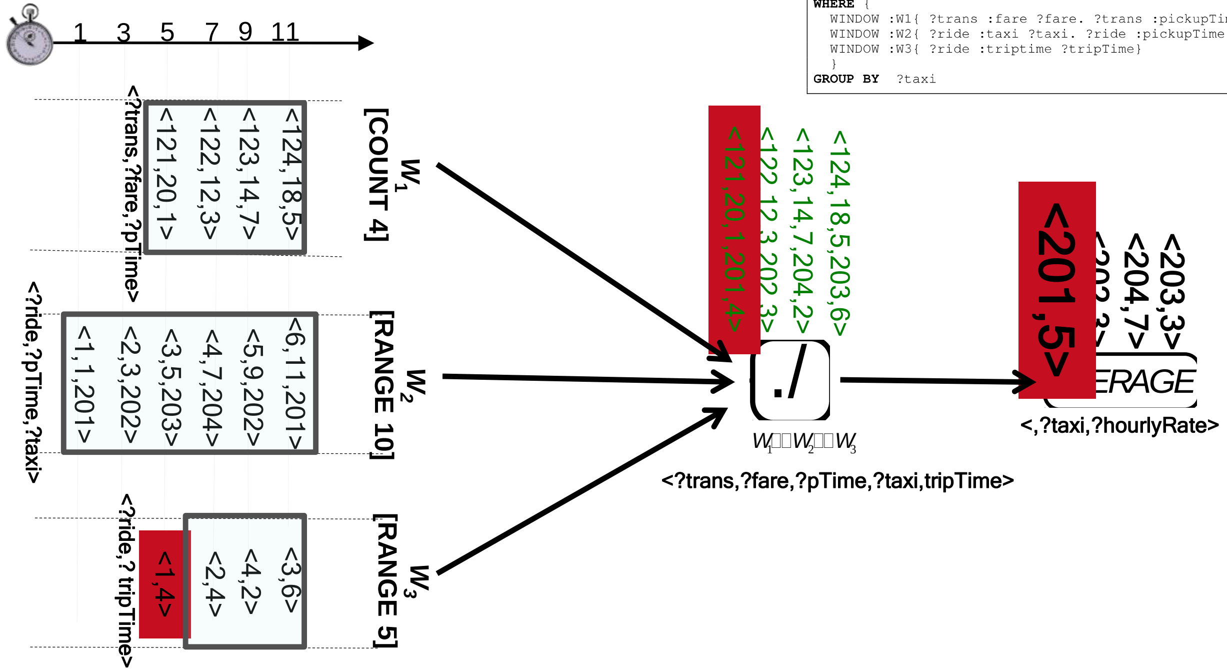
Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi  (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                              [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                              [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



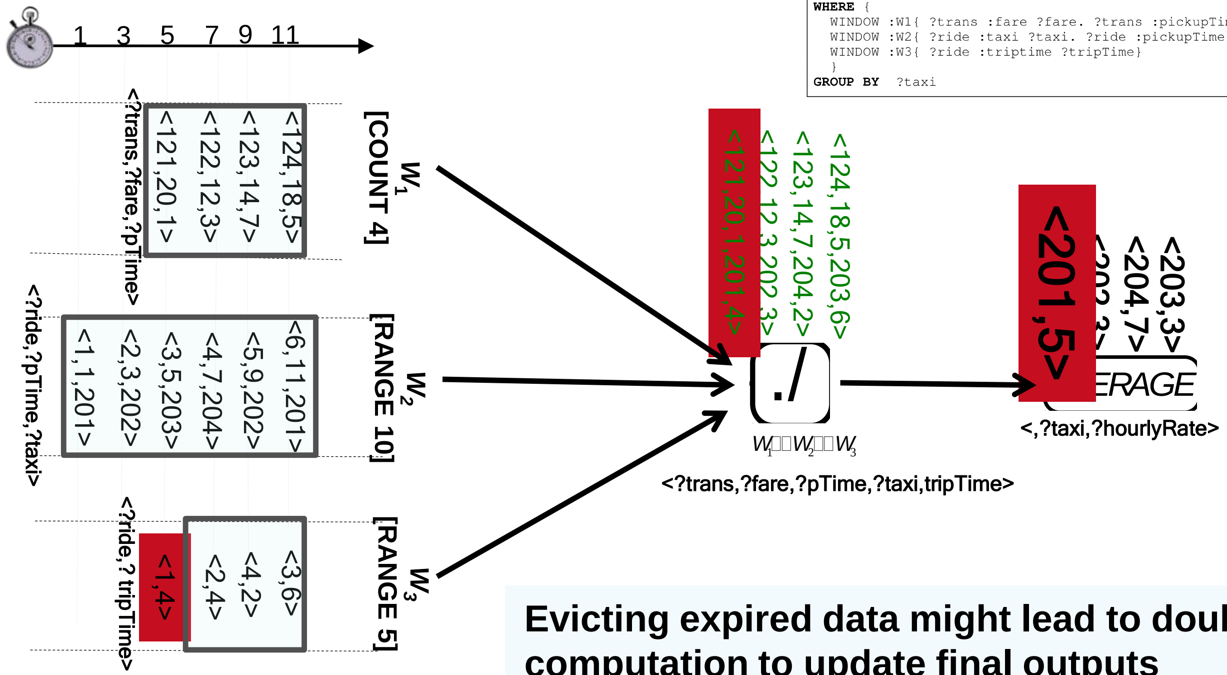
Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi  (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                               [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                               [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```



Incremental Evaluation of Continuous Query: evicting processing state

```
SELECT ?taxi (AVERAGE(?fare/(?tripTime/3600)) AS ?hourlyRate)
FROM  NAMED WINDOW :W1 ON  nyctaxi:fare [COUNT 1000]
FROM  NAMED WINDOW :W2 ON  nyctaxi:pickup
                            [RANGE PT2H @:pickupTime]
FROM  NAMED WINDOW :W3 ON  nyctaxi:dropoff
                            [RANGE PT1H @:dropoffTime]
WHERE {
  WINDOW :W1{ ?trans :fare ?fare. ?trans :pickupTime ?pTime}
  WINDOW :W2{ ?ride :taxi ?taxi. ?ride :pickupTime ?pTime}
  WINDOW :W3{ ?ride :triptime ?tripTime}
  }
GROUP BY  ?taxi
```

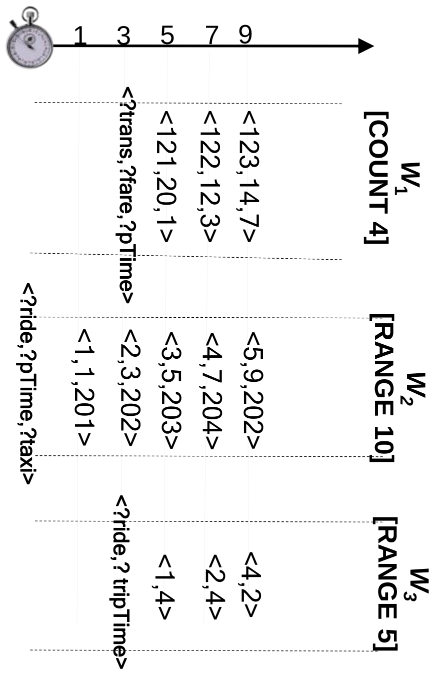


Evicting expired data might lead to double computation to update final outputs

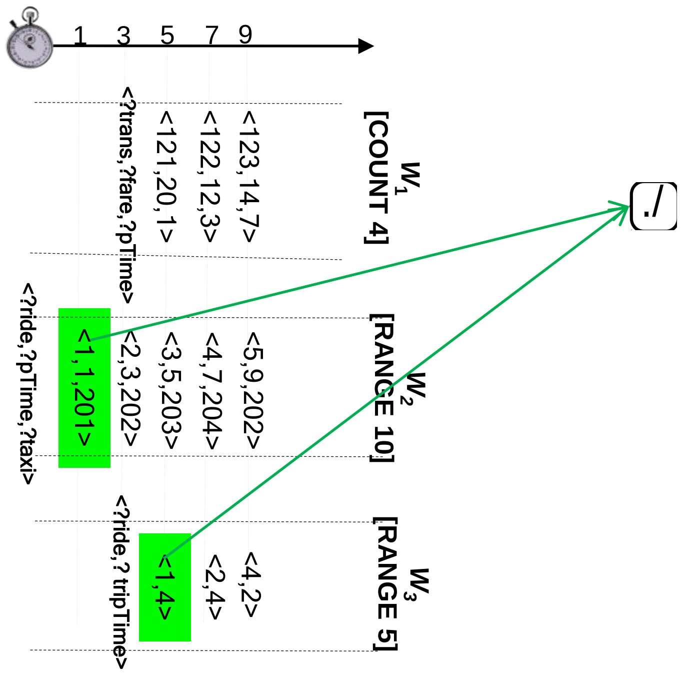
Operator-aware Approach

- Operator-aware data structures designed for:
  - Bookkeeping how the processing states were generated by the the query operators
  - Indexing windowing buffers tailored for query operators' behaviors
- Algorithms for incremental evaluations driven by operator-aware data structures

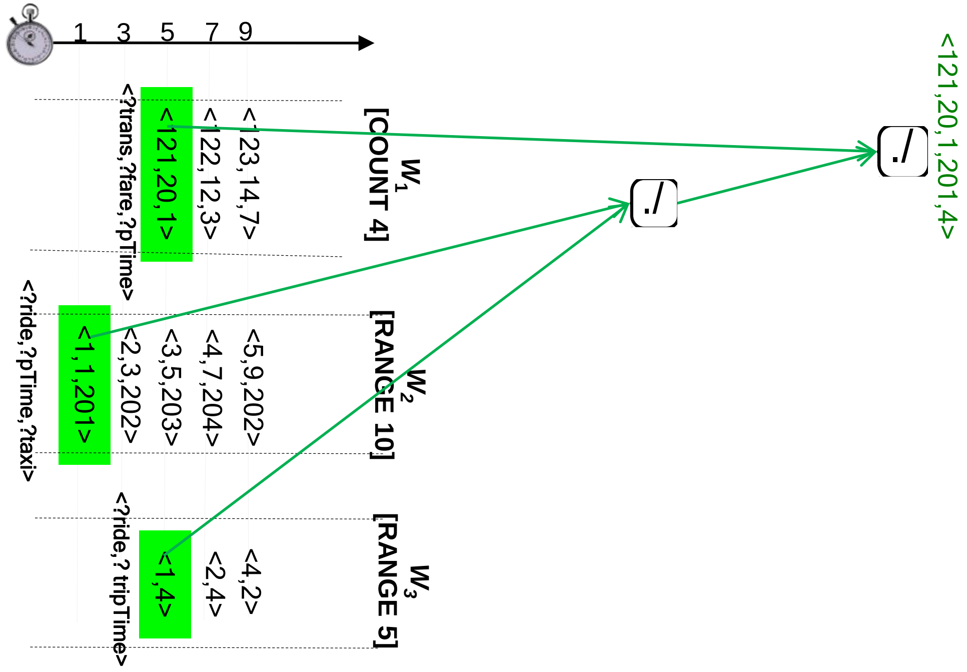
Tree-based data structure for solution mappings



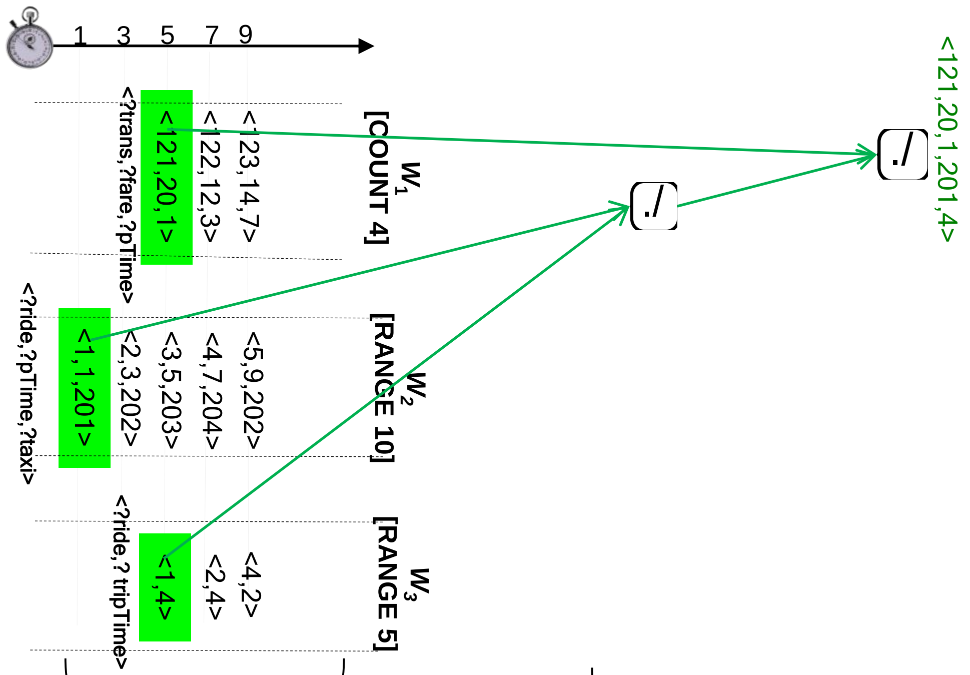
Tree-based data structure for solution mappings



Tree-based data structure for solution mappings



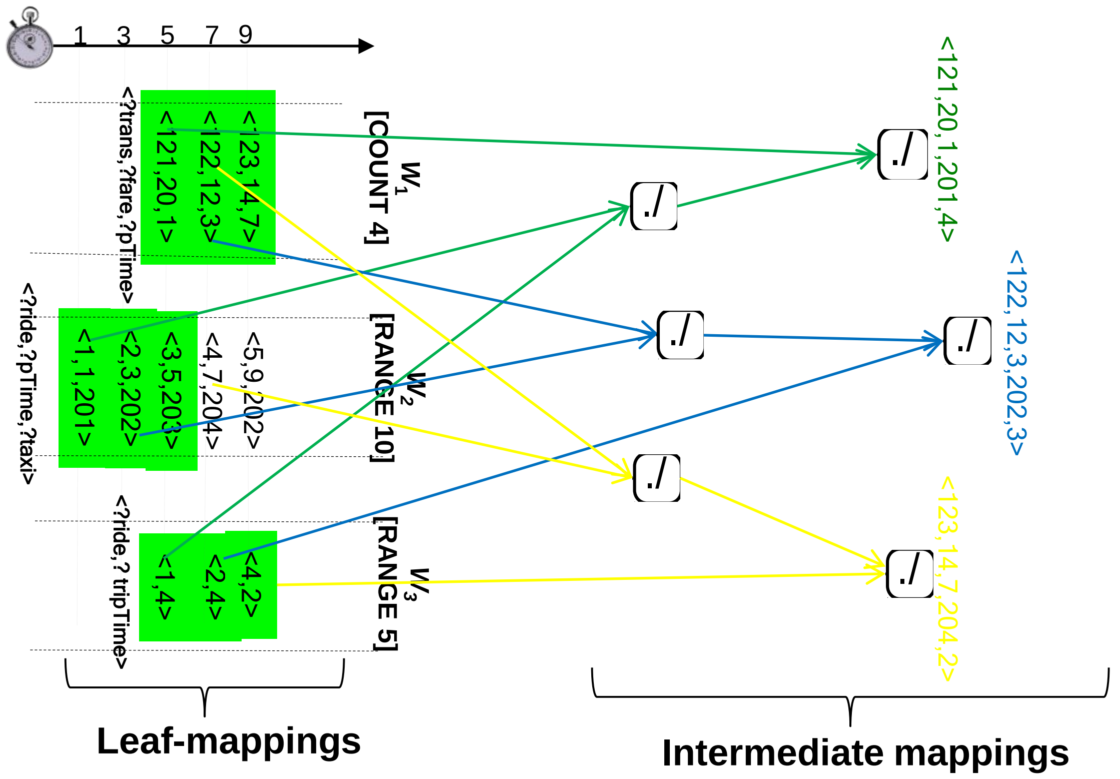
Tree-based data structure for solution mappings



Leaf-mappings

Intermediate mappings

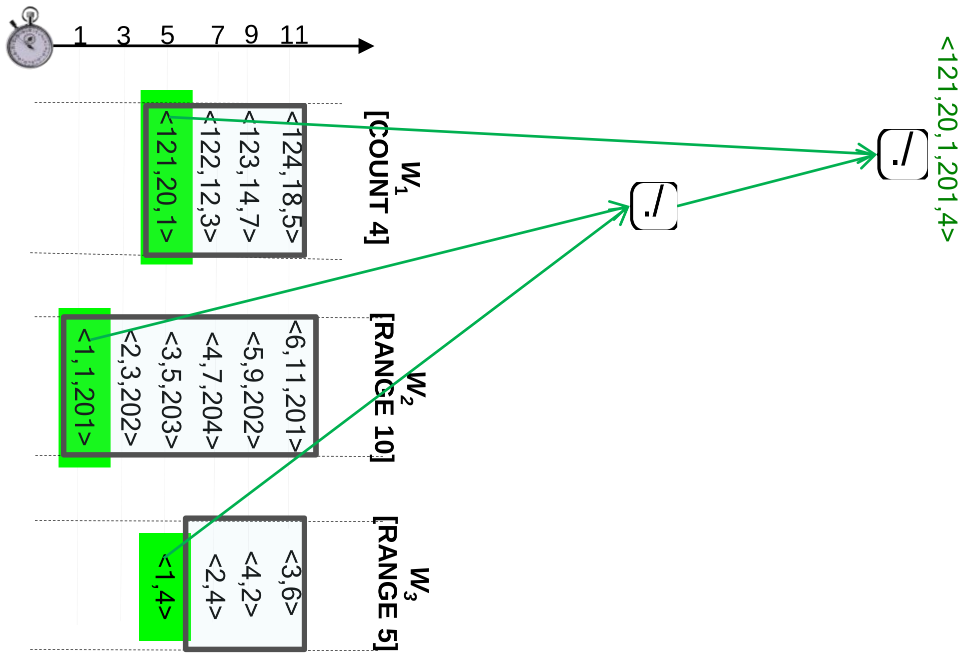
Tree-based data structure for solution mappings



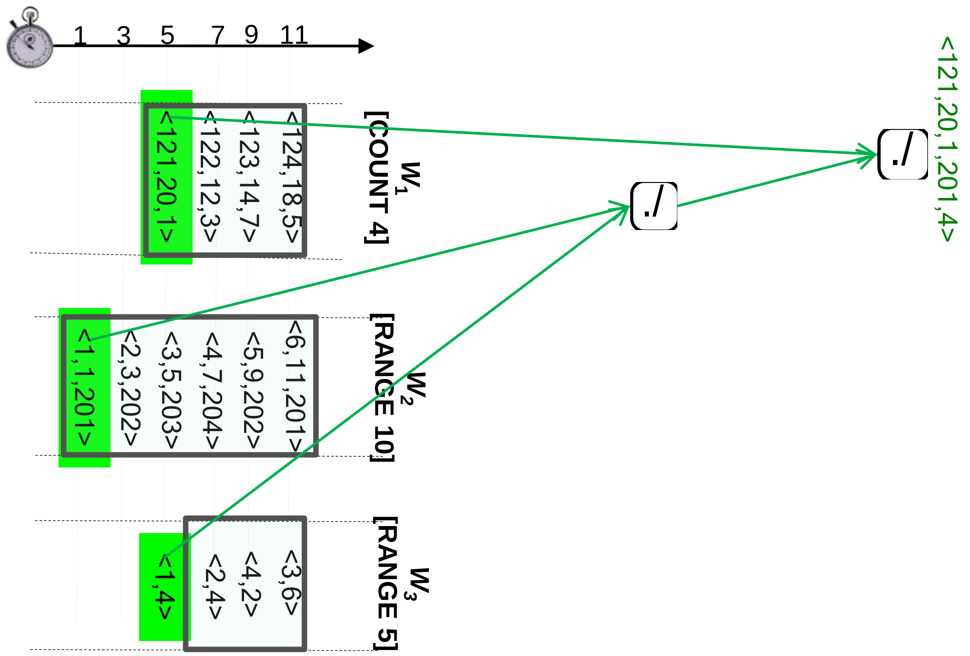
Leaf-mappings

Intermediate mappings

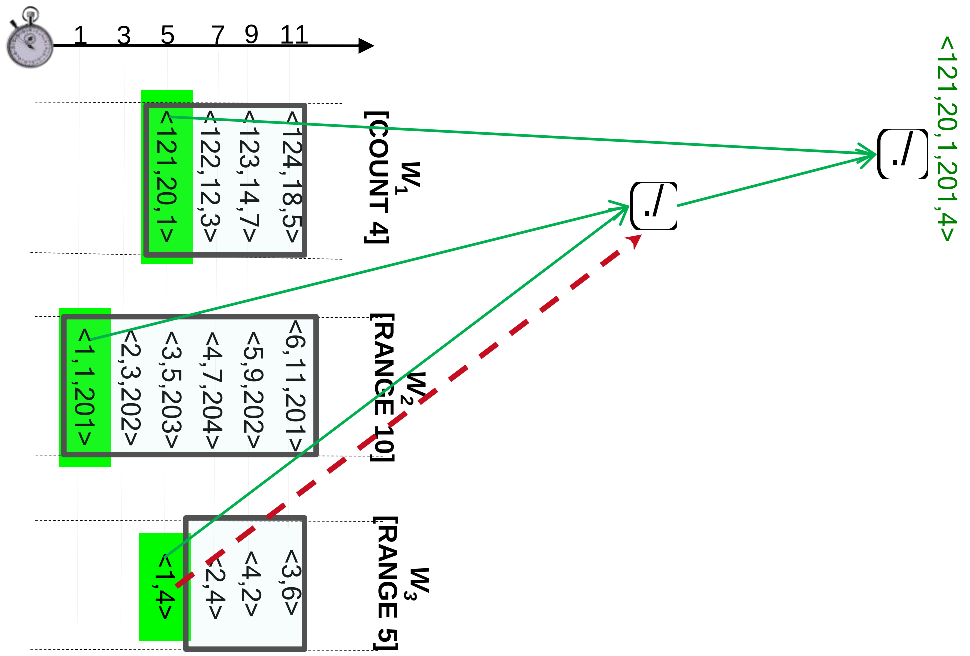
Evicting expired mappings



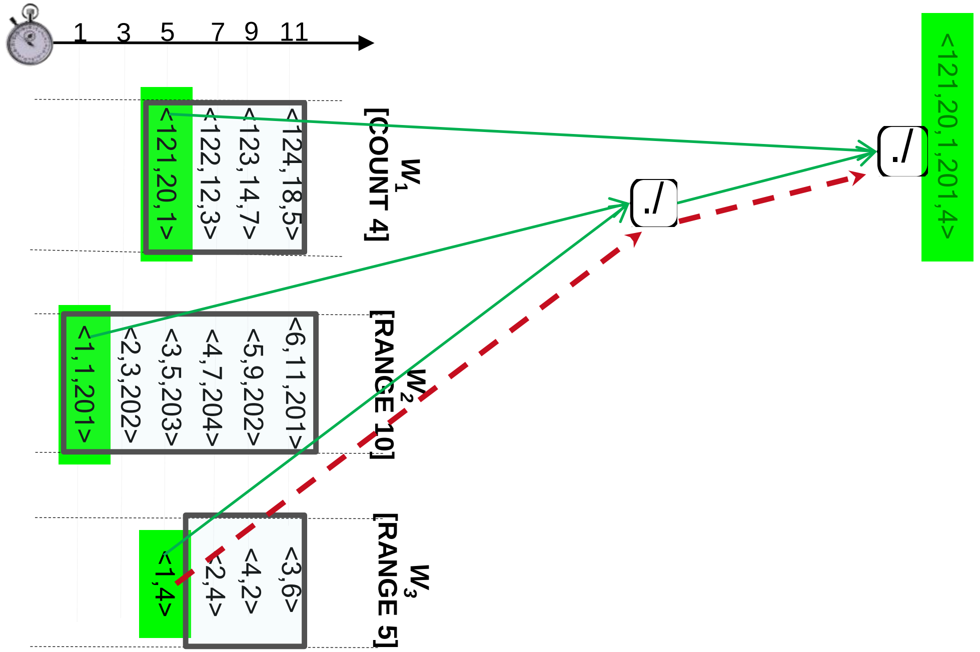
Evicting expired mappings



Evicting expired mappings

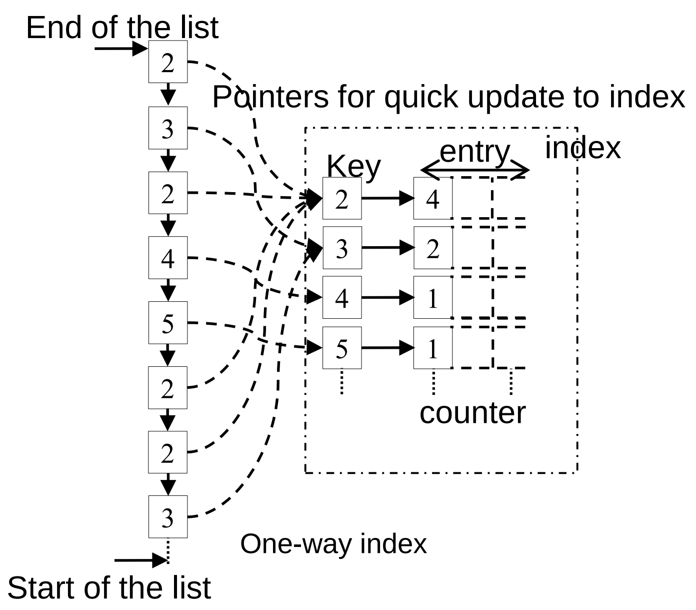


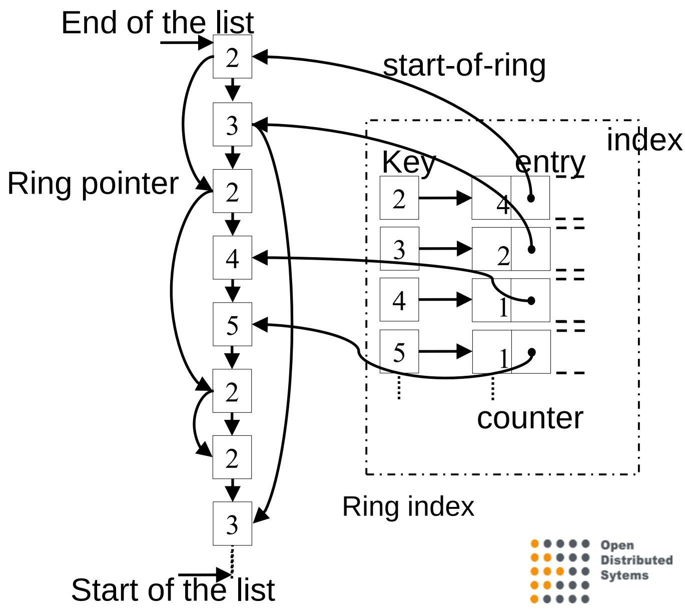
Evicting expired mappings



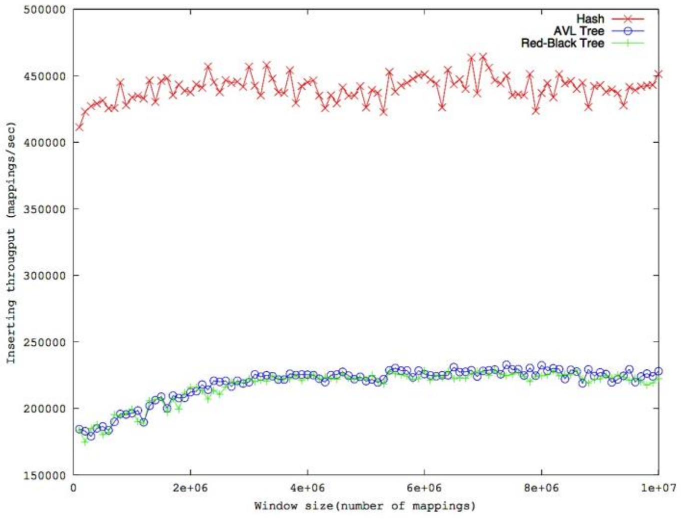
Ring Indexes on bags of mappings

- Operator-aware indexes for quick lookup operations
- Low maintenance cost for fast insert/delete operations

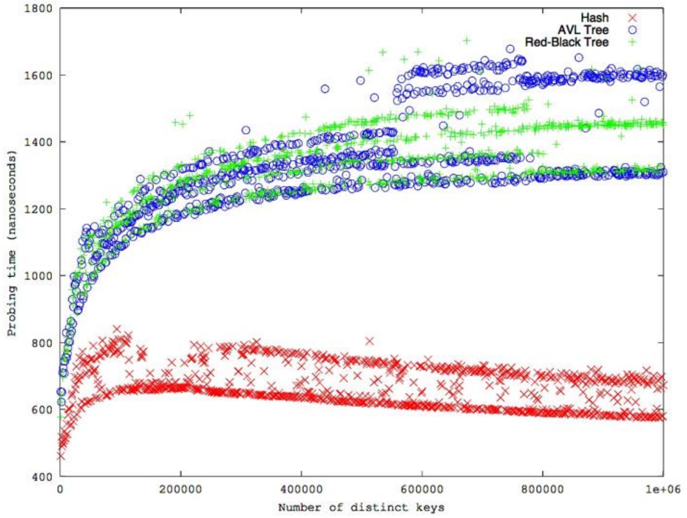




Throughputs of Ring indexes



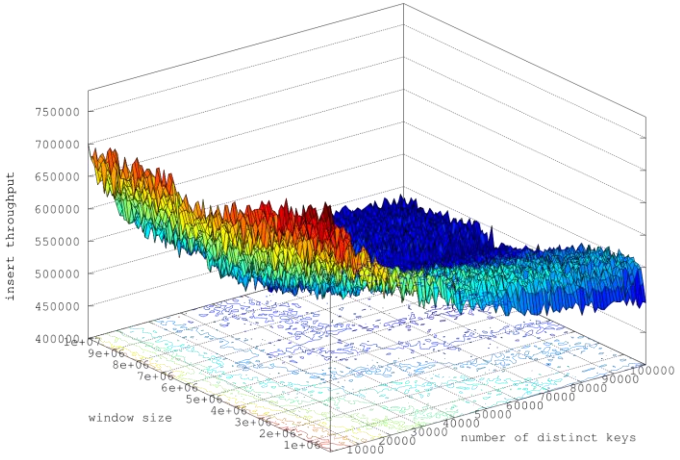
Insert throughput for 1M keys

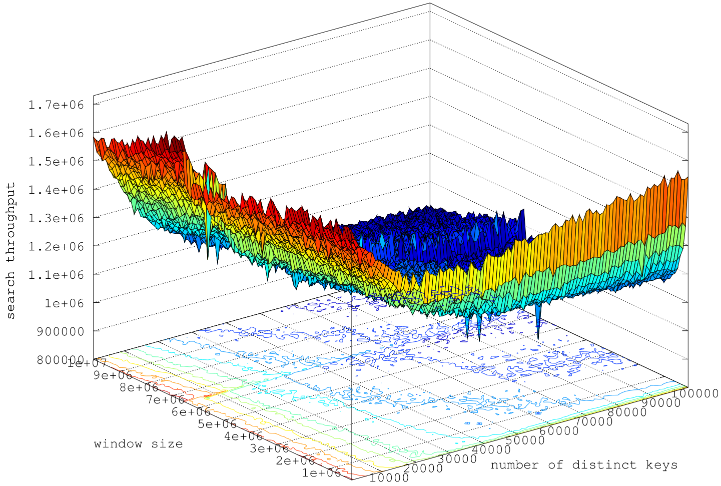


Probing time for 1M-mapping windows

Hash outperforms over AVL Tree and Red-Black Tree

Throughputs of Ring indexes (Cont.)

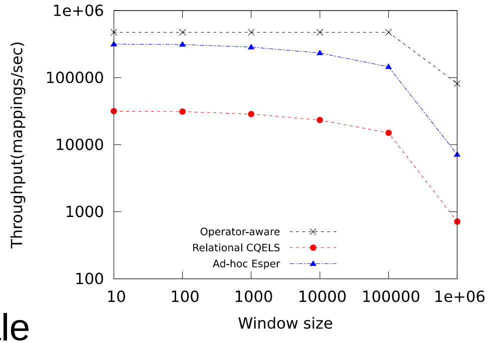




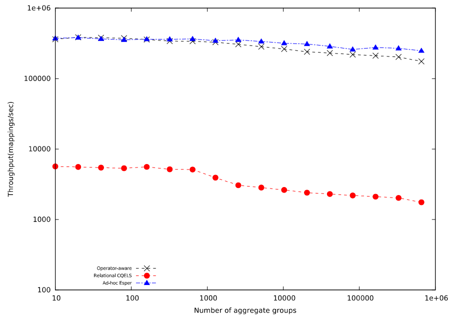
Inserting throughput : 500-900k

Probing/searching throughput: 1M-1.6M

Throughputs of Query Operators



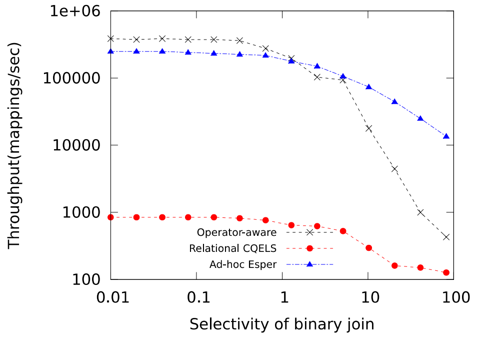
(a) 3-way join



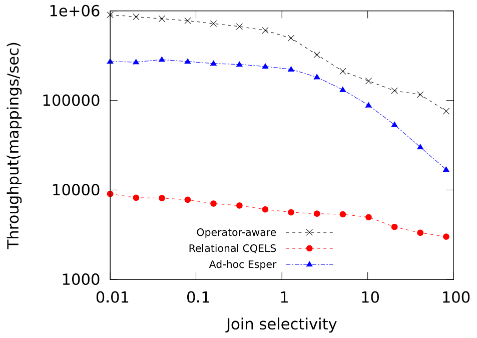
(b) AVERAGE

Log scale

Log scale



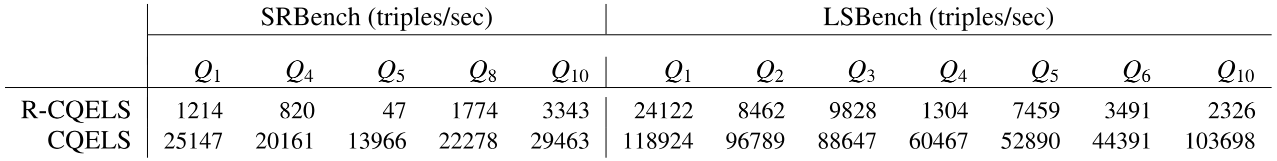
(c) MIN with Join



(d) DISTINCT with Join

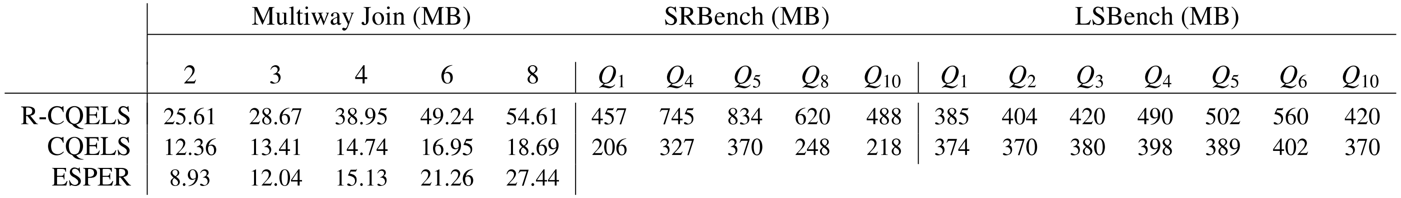
- Operator-aware implementations outperform to relational implementations
- ... are marginally faster than ad-hoc implementations of ESPER in most cases

Throughputs & Memory footprint



	SRBench (triples/sec)					LSBench (triples/sec)						
	Q_1	Q_4	Q_5	Q_8	Q_{10}	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_{10}
R-CQELS	1214	820	47	1774	3343	24122	8462	9828	1304	7459	3491	2326
CQELS	25147	20161	13966	22278	29463	118924	96789	88647	60467	52890	44391	103698

Processing Throughputs: 5-12 times more than relational



	Multiway Join (MB)					SRBench (MB)					LSBench (MB)						
	2	3	4	6	8	Q_1	Q_4	Q_5	Q_8	Q_{10}	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_{10}
R-CQELS	25.61	28.67	38.95	49.24	54.61	457	745	834	620	488	385	404	420	490	502	560	420
CQELS	12.36	13.41	14.74	16.95	18.69	206	327	370	248	218	374	370	380	398	389	402	370
ESPER	8.93	12.04	15.13	21.26	27.44												

Memory Footprint: twice less memory than relational and 20-50% less than ESPER

Summary

- Incremental evaluation algorithms based on operator-aware data structures:
  - Overcome technical issues on traditional incremental evaluation techniques/algorithms
  - Perform several orders of magnitude faster than relation-based implementations
- Throughputs on operator-aware operations on processing state:
  - Up to 1 million of updates/sec vs. 10k of relation-based one
  - Up to 1.6 million lookup operations/second
  - Outperform over relational operations by order of magnitutes
  - Consume twice less memory than relation-based implementations
- The implementation will be open sourced in the next release of CQELS(cqels.org)