Approximation Techniques

Supplemental Reading
(papers posted on Piazza)

Query Approximation

- Summarize the data with a **Sketching** Algorithm.
  - Bloom Filters (Set Containment)
  - Flajolet & Martin Count-Distinct Sketch
  - Count Sketch (Frequent Items/Top-K)
- Sample the data and estimate the error
  - … or better yet, keep generating samples!
    - Online Aggregation & Ripple Joins

Example: F&M Count Sketch

$hash(O_1)$ = 01011011

$hash(O_2)$ = 00110111

$hash(O_3)$ = 00111000

$hash(O_4)$ = 10010010

Example: F&M Count Sketch

$hash(O_1) = 01011011$ $\rho(O_1) = 0$

$hash(O_2) = 00110111$ $\rho(O_2) = 0$

$hash(O_3) = 00111000$ $\rho(O_3) = 3$

$hash(O_4) = 10010010$ $\rho(O_4) = 1$

Example: F&M Count Sketch

$hash(O_1)$ = 01011011 $\rho(O_1)$ = 0 : 0001

$hash(O_2)$ = 00110111 $\rho(O_2)$ = 0 : 0001

$hash(O_3)$ = 00111000 $\rho(O_3)$ = 3 : 1000

$hash(O_4)$ = 10010010 $\rho(O_4)$ = 1 : 0010

1011

R = 210

Example: F&M Count Sketch

$hash(O_1) = 01011011$ $\rho(O_1) = 0 : 0001$

$hash(O_2) = 00110111$ $\rho(O_2) = 0 : 0001$

$hash(O_3) = 00111000$ $\rho(O_3) = 3 : 1000$

$hash(O_4) = 10010010$ $\rho(O_4) = 1 : 0010$

$2^R/\varphi = 4/0.77351 = 5.2$

1011

$R = 2 1 0$ ←

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

```
hash(O1) = 01011011
hash(O2) = 00110111
hash(O3) = 00111000
hash(O4) = 10010010
```

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

$hash(O_1) = 1$

$hash(O_2) = 1$

$hash(O_3) = 0$

$hash(O_4) = 0$

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

-| +| -| +| -| +| -| -| +|

hash(O_1) = 1 +|

hash(O_2) = 1 +|

hash(O_3) = 0 -|

hash(O_4) = 0 -|

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

-1 +1 -1 +1 -1 +1 -1 -1 +1 = -1

hash(O_1) =	1	+1	E[Count(O_1)] = -1
hash(O_2) =	1	+1	E[Count(O_2)] = -1
hash(O_3) =	0	-1	E[Count(O_3)] = 1
hash(O_4) =	0	-1	E[Count(O_4)] = 1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

```
hash(O1) = 01011011
hash(O2) = 00110111
hash(O3) = 00111000
hash(O4) = 10010010
```

0	0	0	0
0	0	0	0

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

Each Pair: <Top/Bottom, +1/-1>

$hash(O_1) = 01|01|10|11$

$hash(O_2) = 00|11|01|11$

$hash(O_3) = 00|11|10|00$

$hash(O_4) = 10|01|00|10$

0	0	0	0
0	0	0	0

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

Each Pair: <Top/Bottom, +1/-1>

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

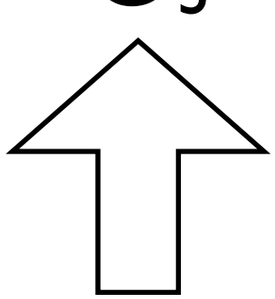
0	0	0	0
0	0	0	0

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

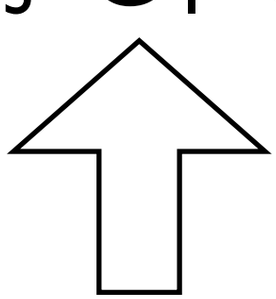
0	+1	−1	0
−1	0	0	−1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

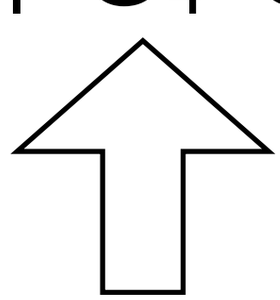
0	+1	−2	+1
0	+1	0	−1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

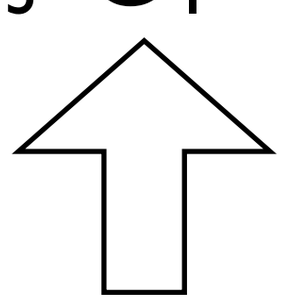
−1	+1	−2	0
0	+2	−1	−1

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

−2	+4	−6	+4
−1	+5	−1	+3

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

-2	+4	-6	+4
-1	+5	-1	+3

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

hash(O_1) <B,+1>, <B,+1>, <T,-1>, <T,+1>

hash(O_2) <B,-1>, <T,+1>, <B,+1>, <T,+1>

hash(O_3) <B,-1>, <T,+1>, <T,-1>, <B,-1>

hash(O_4) <T,-1>, <B,+1>, <B,-1>, <T,-1>

-2	+4	-6	+4
-1	+5	-1	+3

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

`hash(`O_1`)` <B,+1>, <B,+1>, <T,-1>, <T,+1>

`hash(`O_2`)` <B,-1>, <T,+1>, <B,+1>, <T,+1>

`hash(`O_3`)` <B,-1>, <T,+1>, <T,-1>, <B,-1>

`hash(`O_4`)` <T,-1>, <B,+1>, <B,-1>, <T,-1>

-2	+4	-6	+4
-1	+5	-1	+3
-1	+4	+5	+6

1.5

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

-2	+4	-6	+4
-1	+5	-1	+3

Example: CountSketch

Estimate the Count of Each Object

~~Find the 2 Most Frequent Objects~~

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

-2	+4	-6	+4
-1	+5	-1	+3
-3	+1	+4	+6

1.5

Wiggle Room

More Hashes

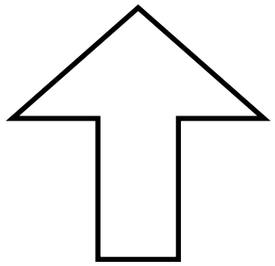
=

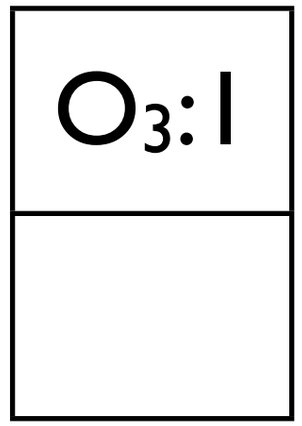
More Accurate Estimate

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1





`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

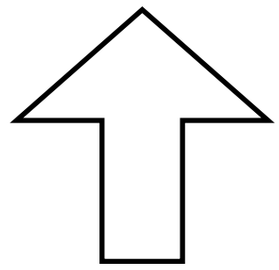
`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

0	+1	−1	0
−1	0	0	−1

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_3:1

O_1:1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

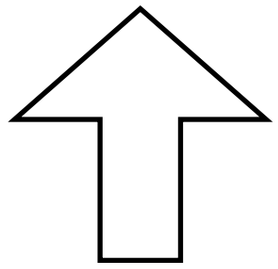
`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

0	+1	-2	1
0	+1	0	-1

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_3:1
O_1:1

$hash(O_1)$ <B, +1>, <B,+1>, <T,-1>,<T,+1>

$hash(O_2)$ <B, -1>, <T,+1>, <B,+1>,<T,+1>

$hash(O_3)$ <B, -1>, <T,+1>, <T,-1>,<B,-1>

$hash(O_4)$ <T, -1>, <B,+1>, <B,-1>,<T,-1>

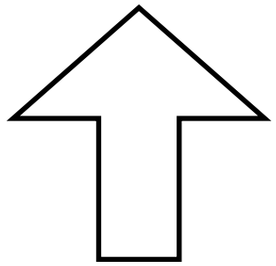
−1	+1	−2	0
0	+2	−1	−1
0	+1	+1	+2

+1

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_3:1

O_1:1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

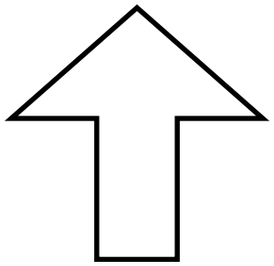
−1	+2	−2	+1
−1	+2	0	−1
0	+1	+1	+2

+1

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_3:1

O_1:1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

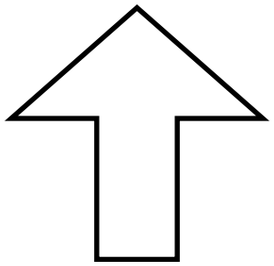
−2	+2	−2	0
−1	+3	−1	−1
0	+1	+2	+3

+1.5

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_4:1.5

O_3:1

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

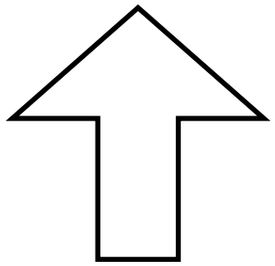
−2	+2	−2	0
−1	+3	−1	−1
0	+1	+2	+3

+1.5

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_4:1.5

O_3:1

$hash(O_1)$ <B, +1>, <B,+1>, <T,-1>,<T,+1>

$hash(O_2)$ <B, -1>, <T,+1>, <B,+1>,<T,+1>

$hash(O_3)$ <B, -1>, <T,+1>, <T,-1>,<B,-1>

$hash(O_4)$ <T, -1>, <B,+1>, <B,-1>,<T,-1>

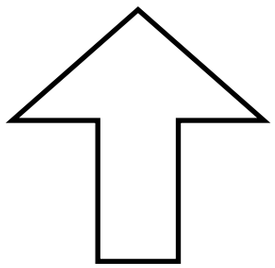
−2	+2	−3	+1
0	+4	−1	−1
0	+1	+3	+4

+2

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_1:2

O_4:1.5

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

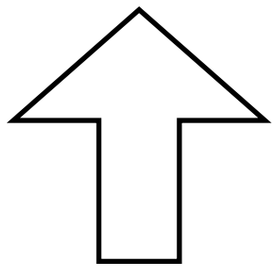
−2	+2	−3	+1
0	+4	−1	−1
0	+1	+3	+4

+2

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_1:2
O_4:1.5

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

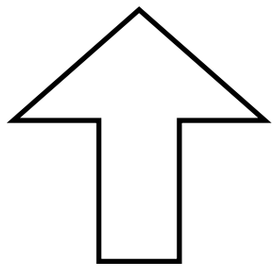
−2	+3	−4	+1
−1	+4	−1	−2
+1	+2	+3	+4

+2.5

Example: CountSketch

Find the 2 Most Frequent Objects

O_3 O_1 O_4 O_2 O_4 O_1 O_3 O_3 O_1



O_3:2.5
O_1:2

`hash(`O_1`)` <B, +1>, <B,+1>, <T,-1>,<T,+1>

`hash(`O_2`)` <B, -1>, <T,+1>, <B,+1>,<T,+1>

`hash(`O_3`)` <B, -1>, <T,+1>, <T,-1>,<B,-1>

`hash(`O_4`)` <T, -1>, <B,+1>, <B,-1>,<T,-1>

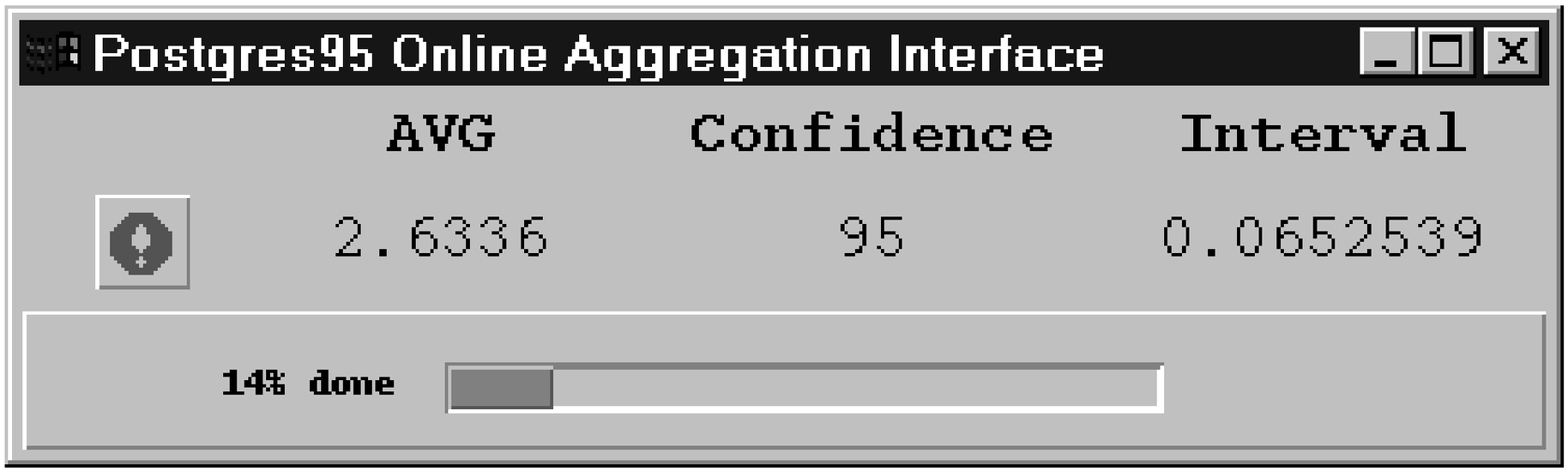
−2	+3	−4	+1
−1	+4	−1	−2
+1	+2	+3	+4

+2.5

Online Aggregation

- Give (some) results immediately.
  - (Imprecise) results obtained by sampling.
- More time, more samples, more accuracy.
  - Query takes longer to complete.
  - … but result estimate available sooner.

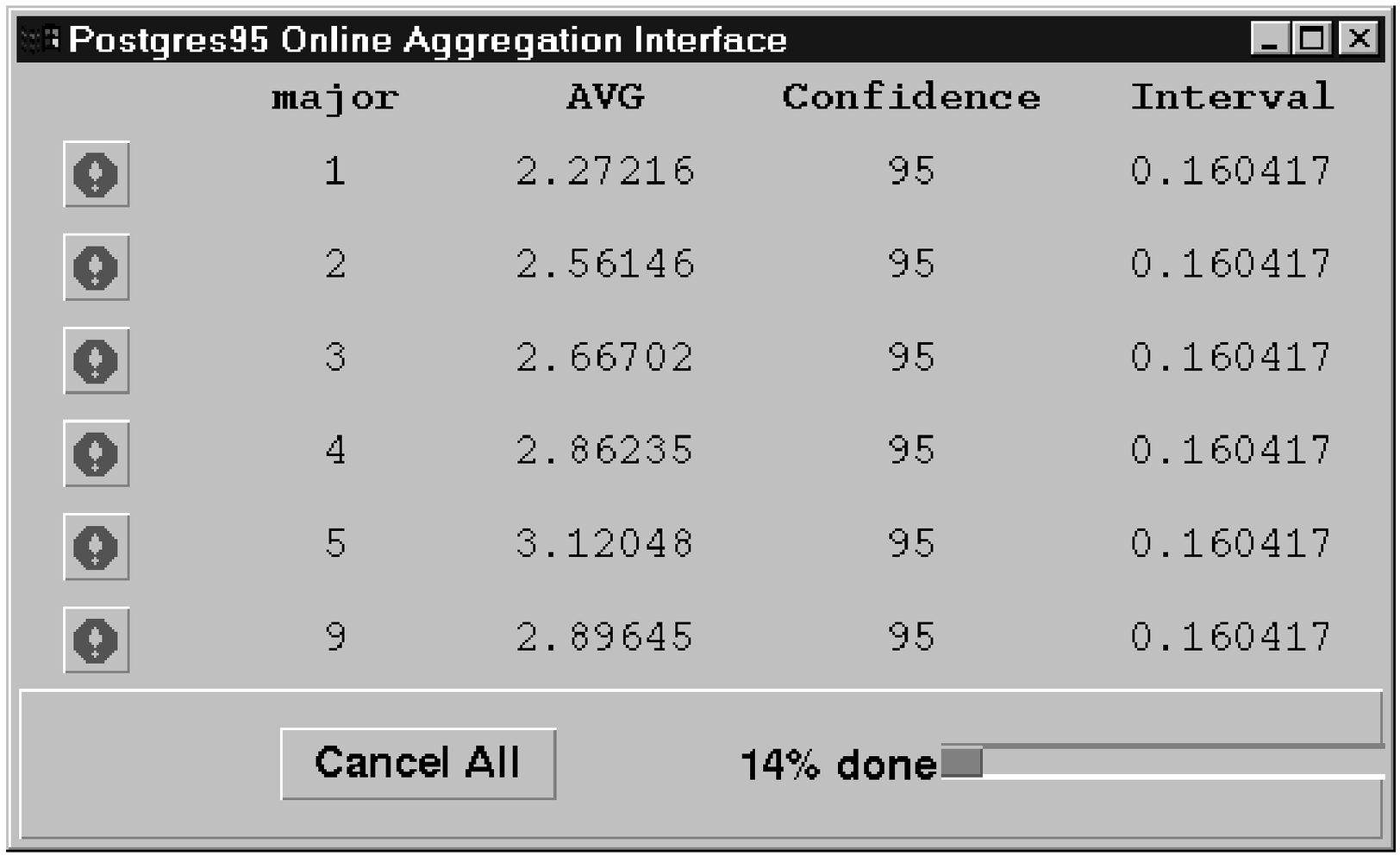
Online Aggregation



OA Challenges

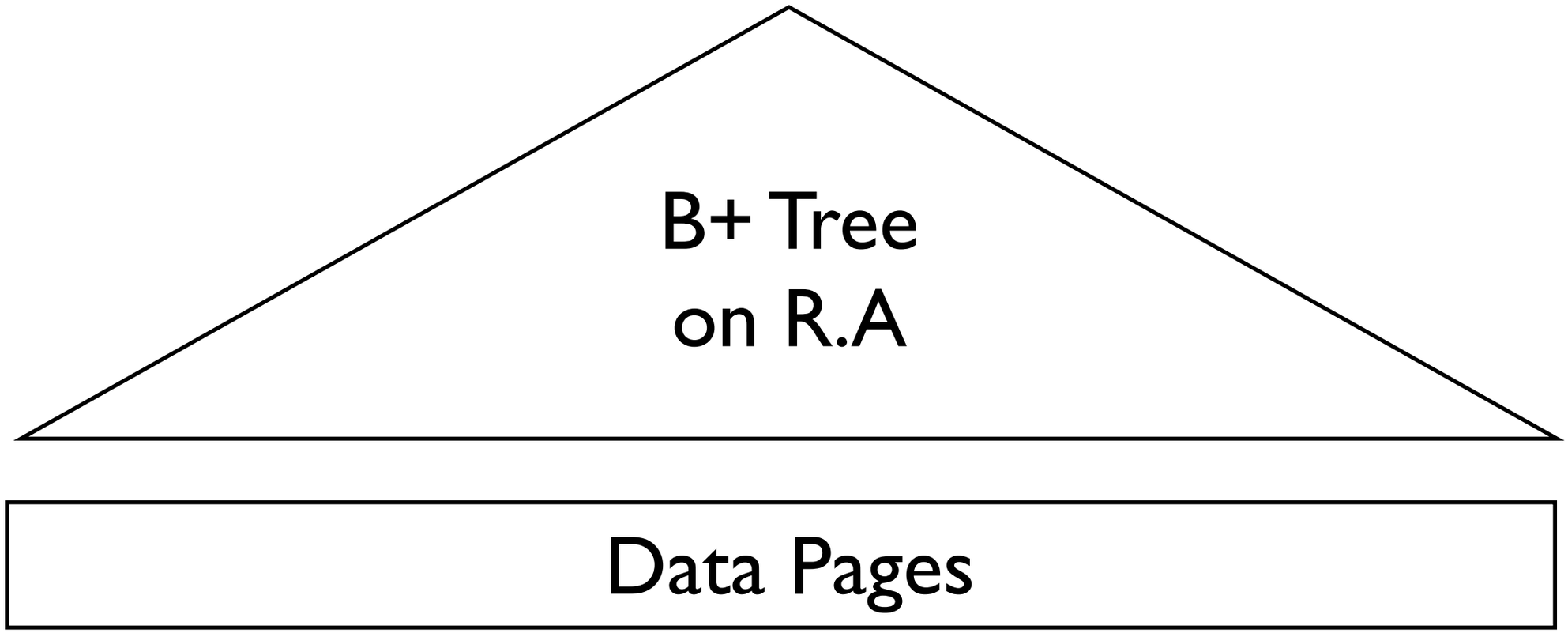
- Sampling: Need Random Access to Data.
  - Heap (Unsorted) Files, Flash Drives
- Fairness: Sampling For “Rare” Group-By Columns.
  - Index Striding
- Blocking: Joins Must Be Streamed.
  - Ripple Join

Fairness

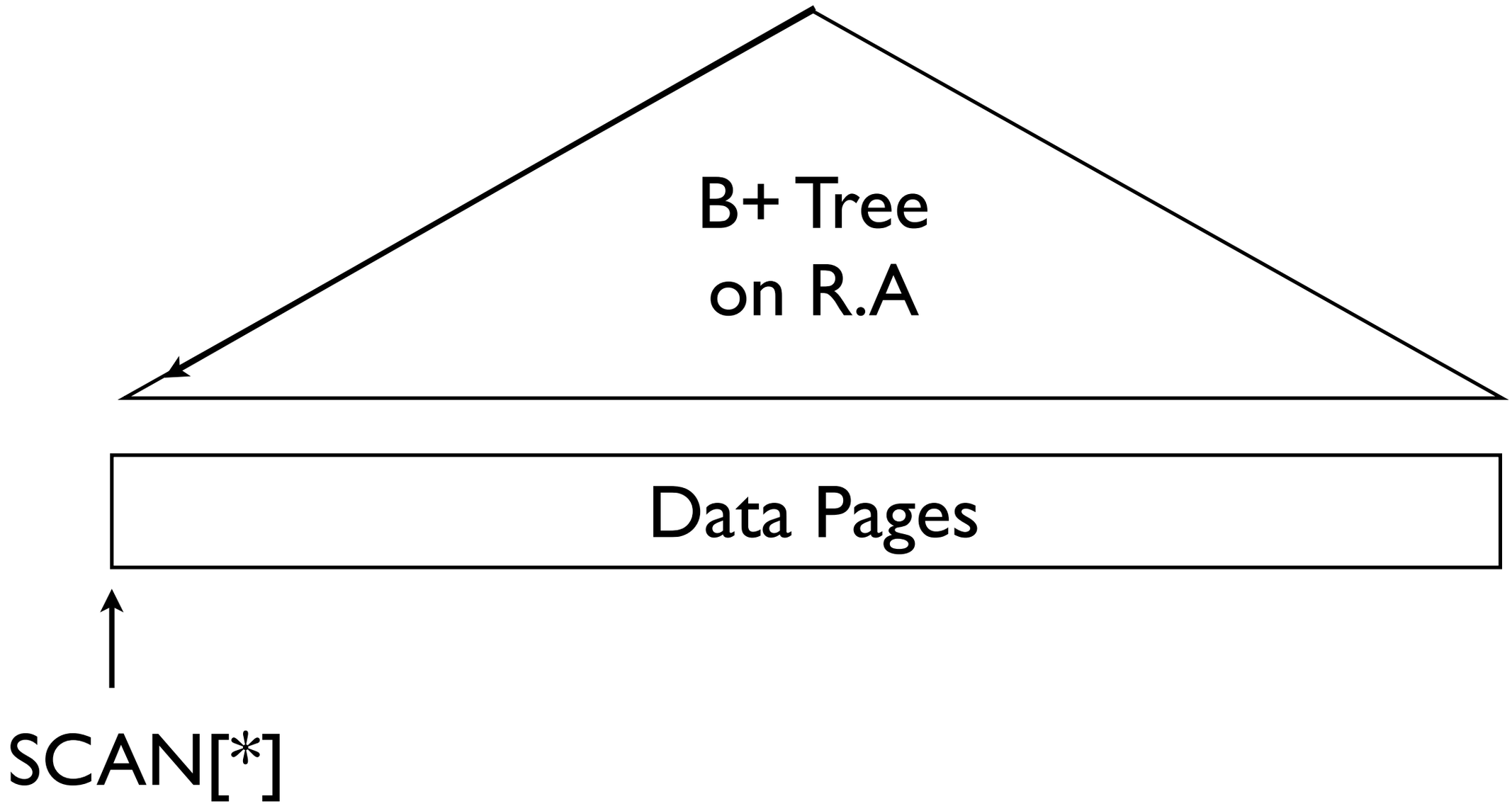


Goal: Keep all Confidences / Intervals in Synch.

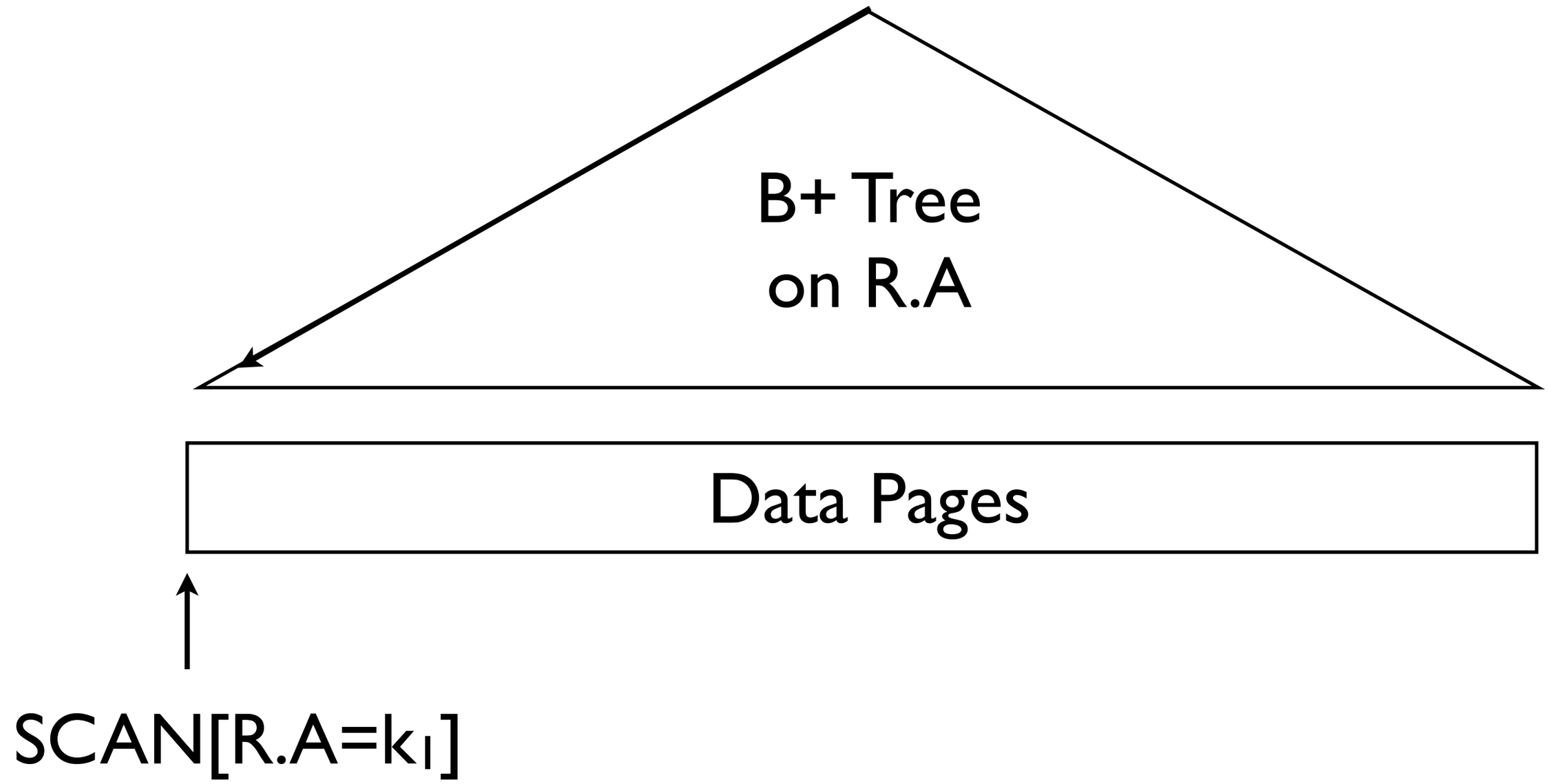
Index Striding



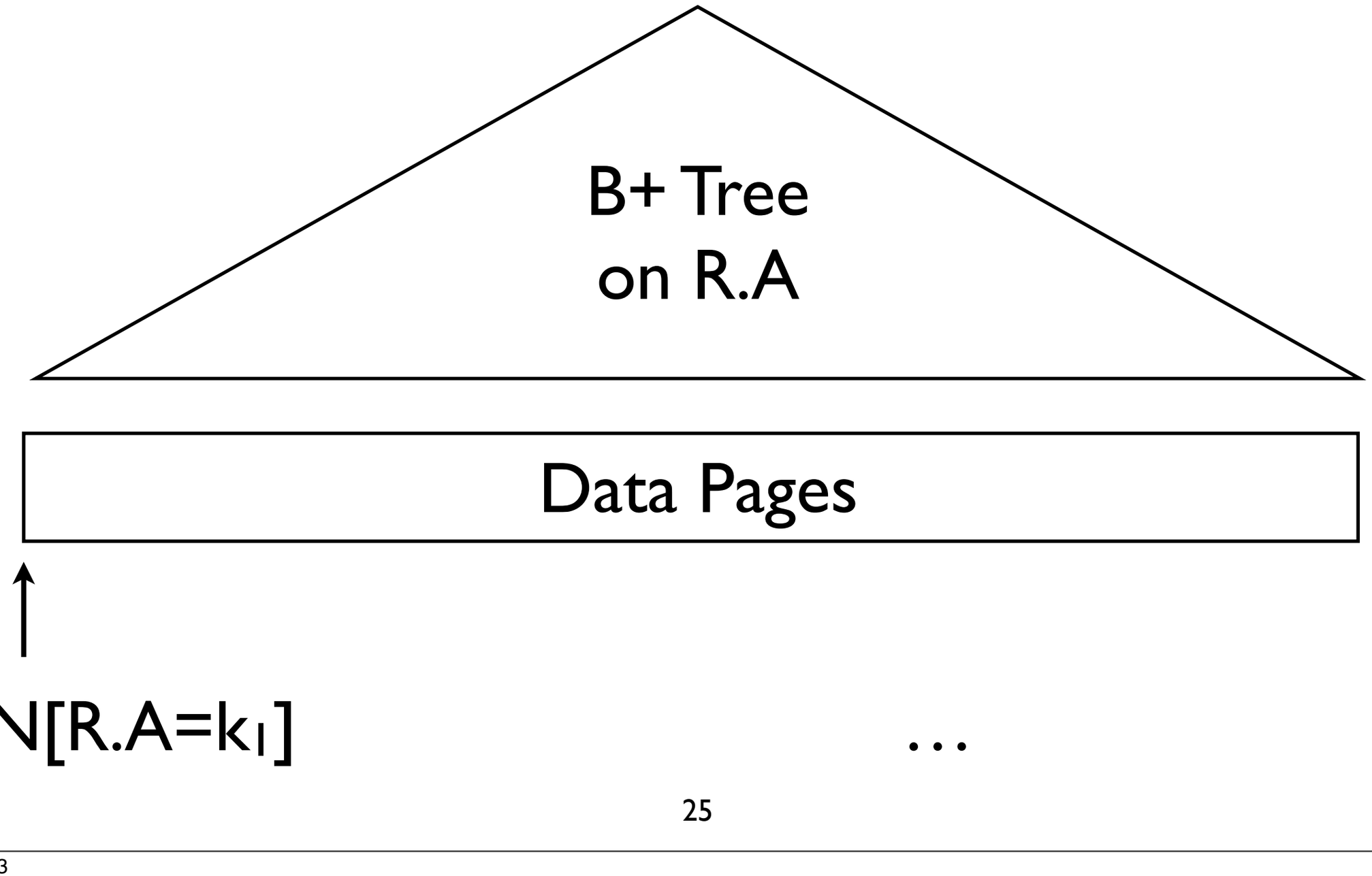
Index Striding



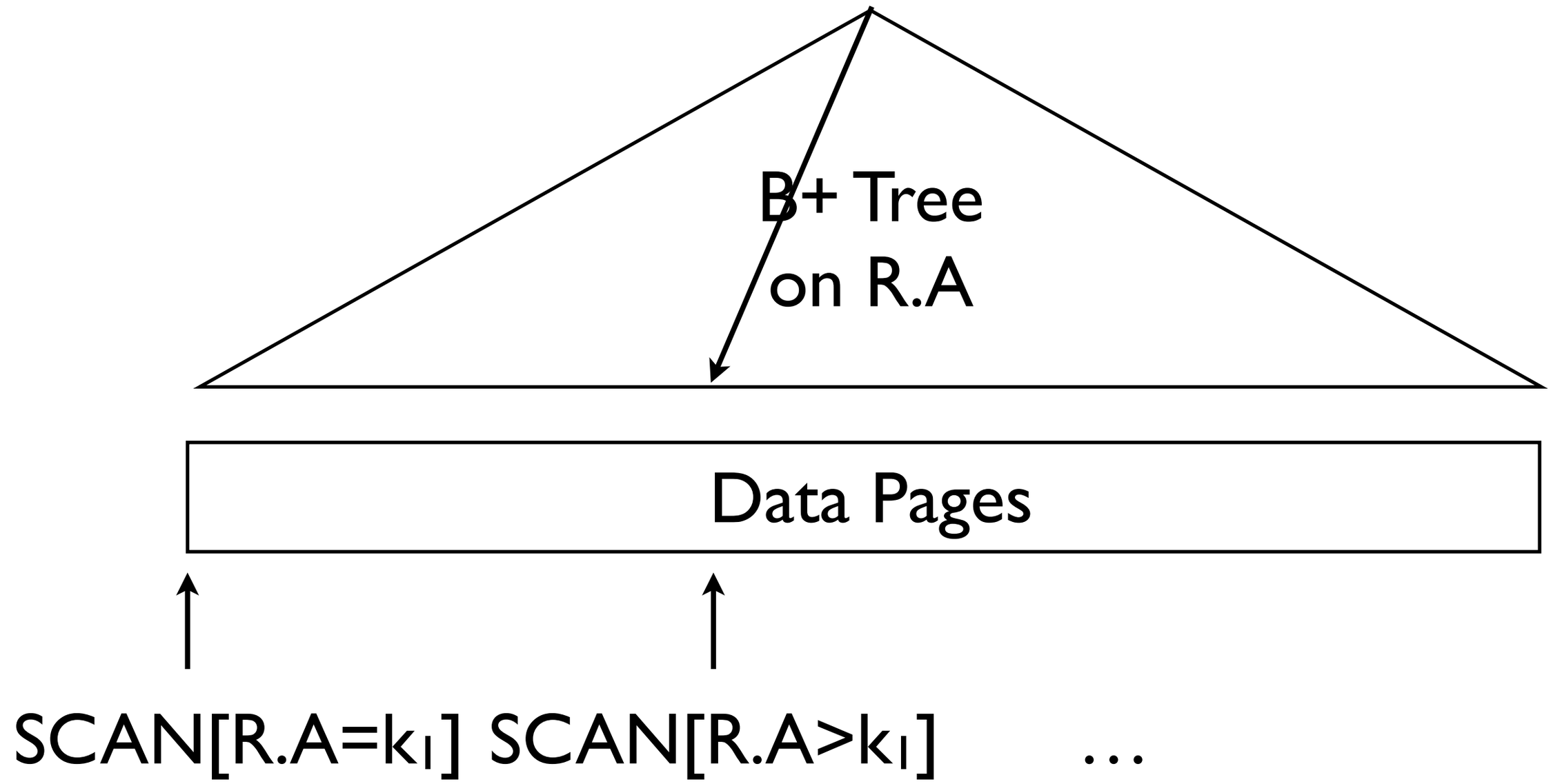
Index Striding



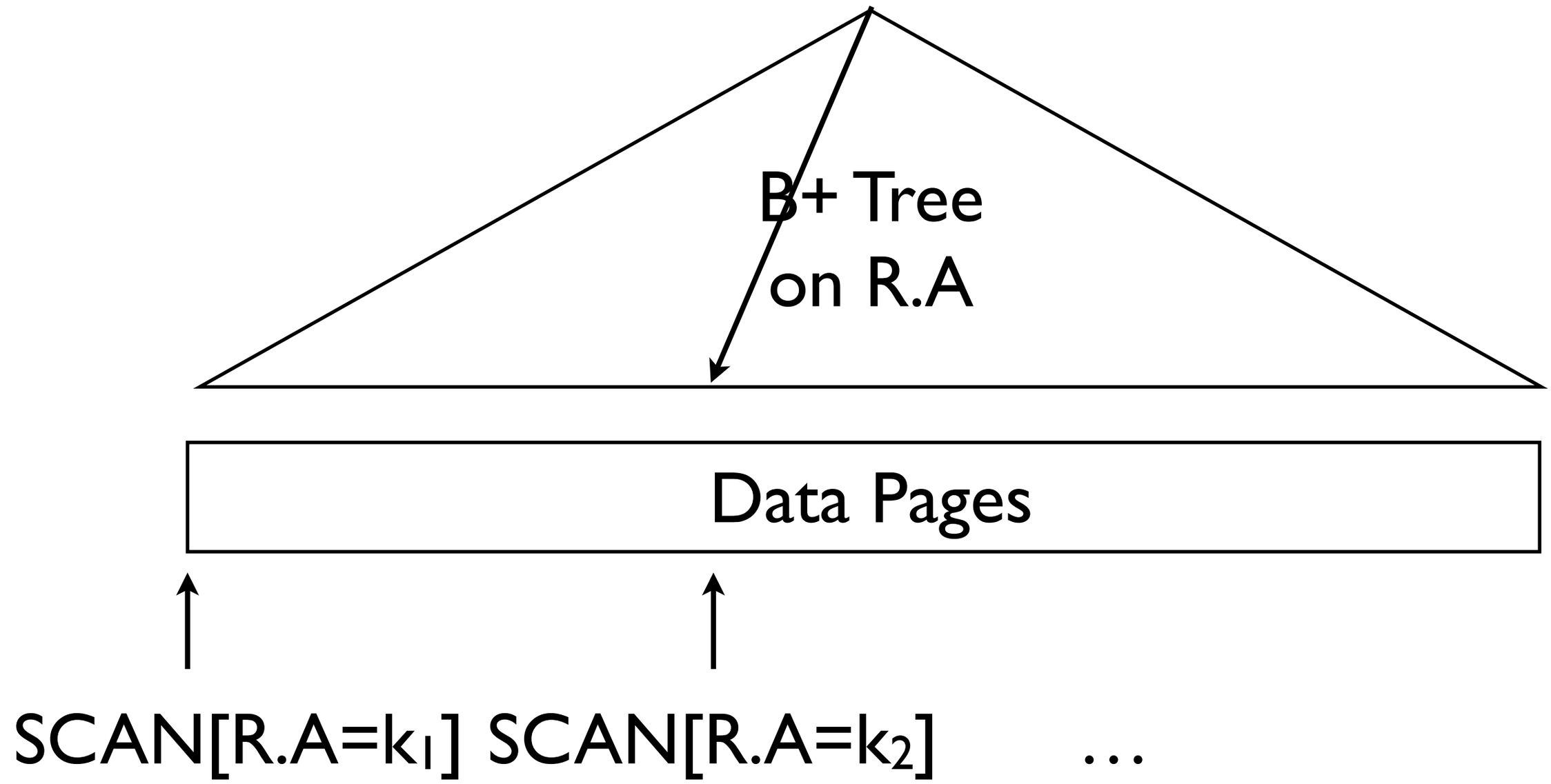
Index Striding



Index Striding



Index Striding



Index Striding

SCAN[R.A=k_1]

SCAN[R.A=k_2]

SCAN[R.A=k_3]

SCAN[R.A=k_4]

SCAN[R.A=k_5]

…

One Scan for Each GB Key

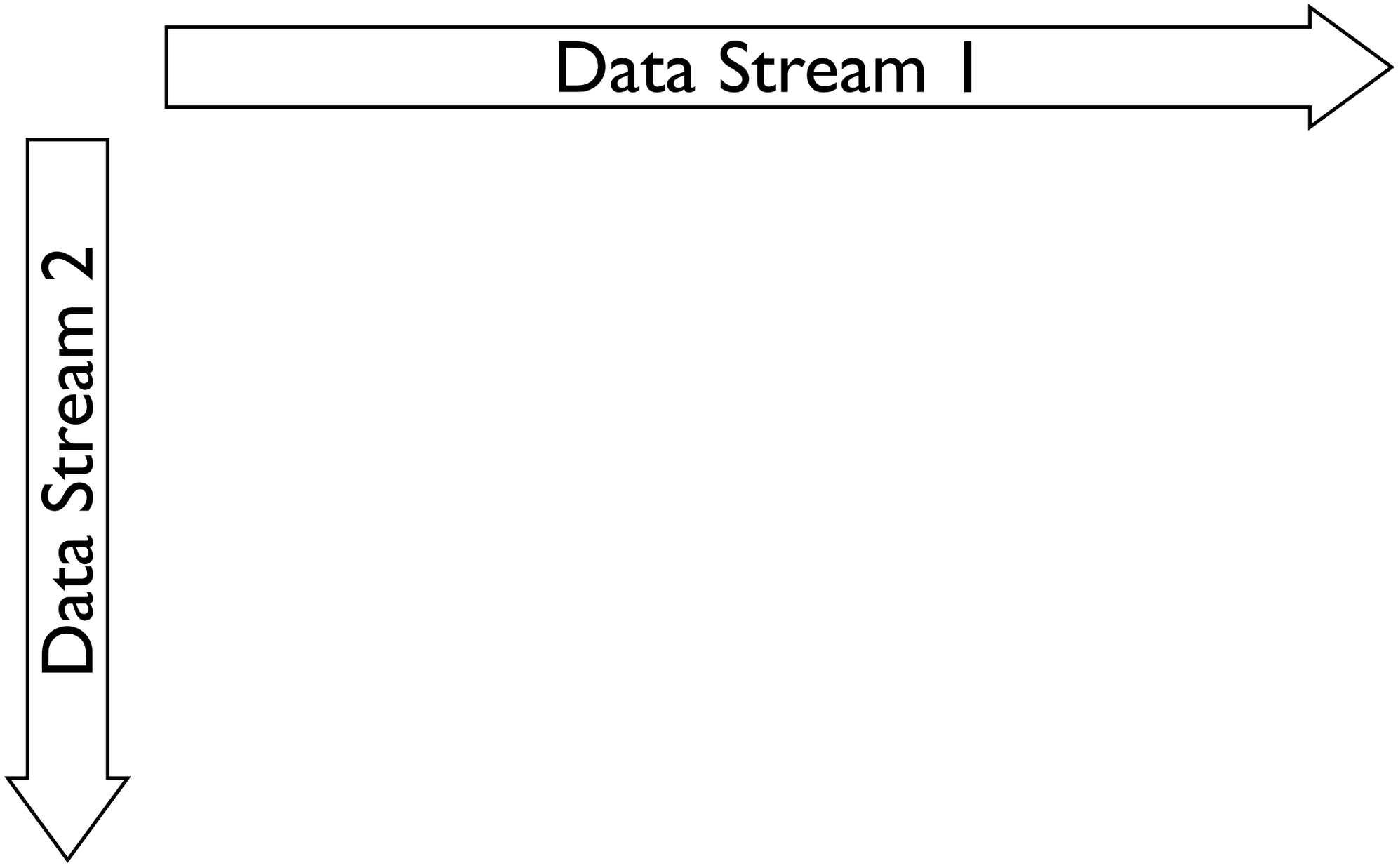
Each Scan is now a Heap Scan

Split resources evenly between each scan created.

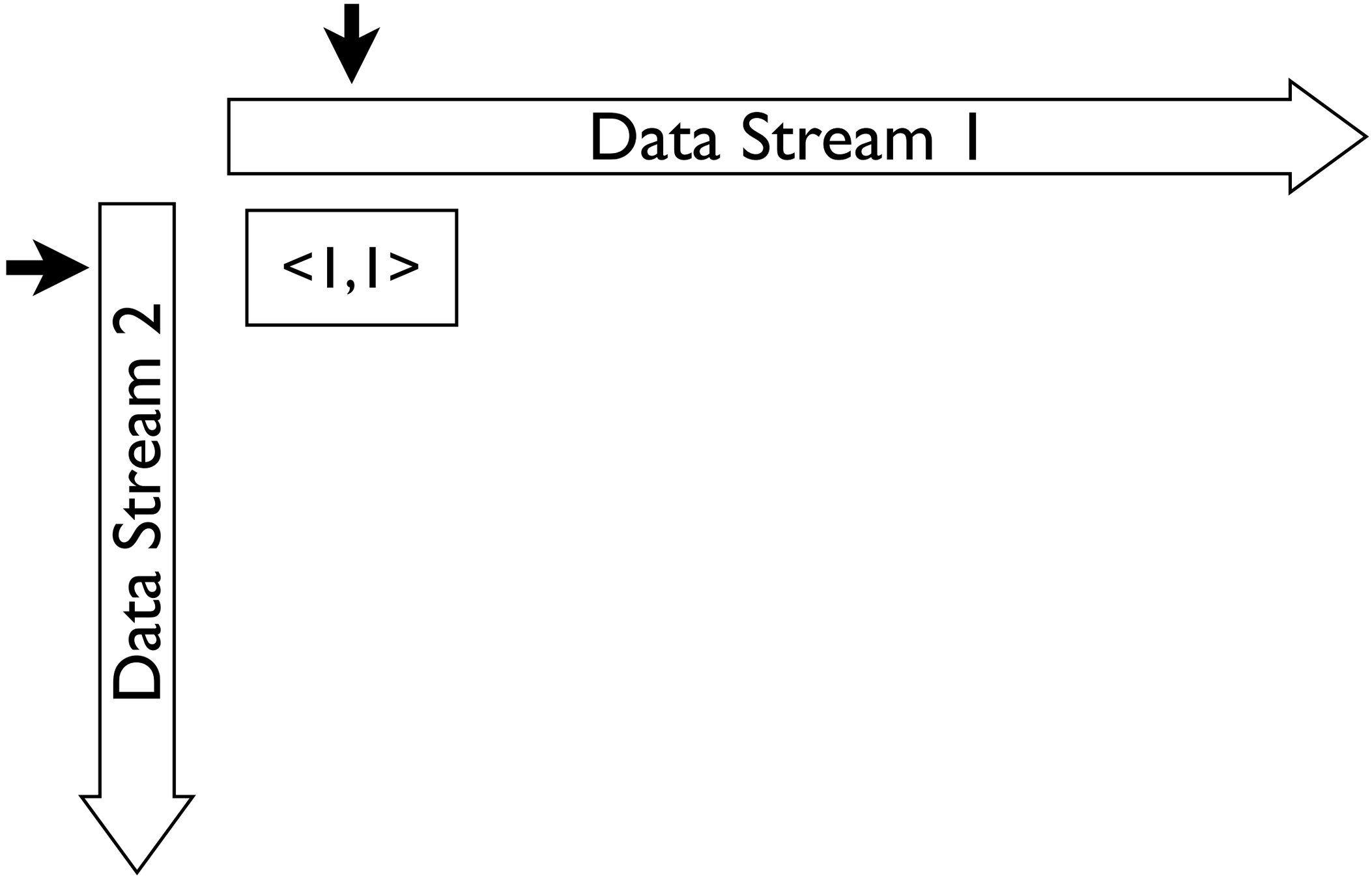
Non-Blocking Joins

- Sort/Merge Join
  - We want the data unsorted
- Index-Nested Loop Join
  - Could work if only few tuples matched.
- Hybrid Hash Join
  - Could work if one table is small.

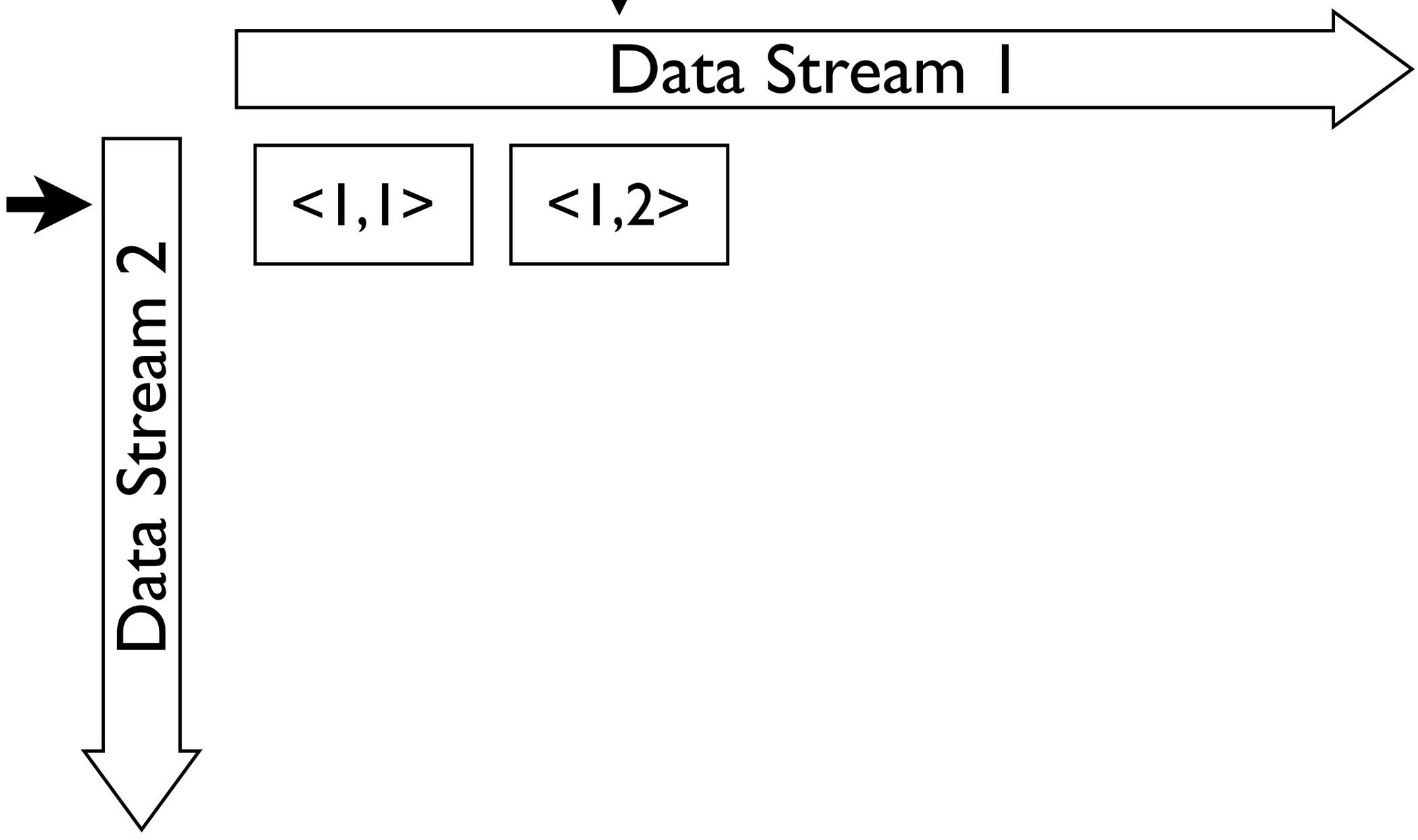
Ripple Join



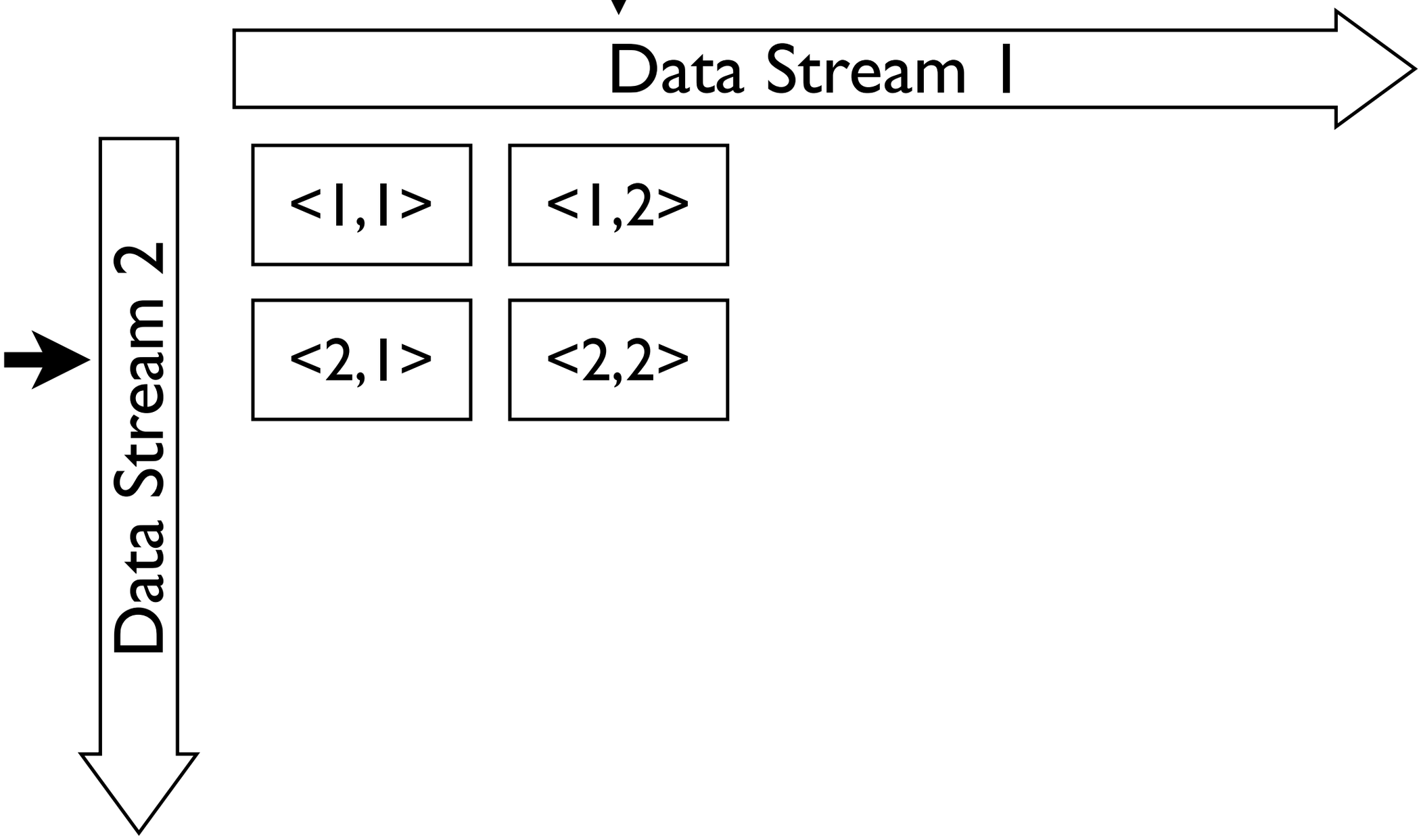
Ripple Join



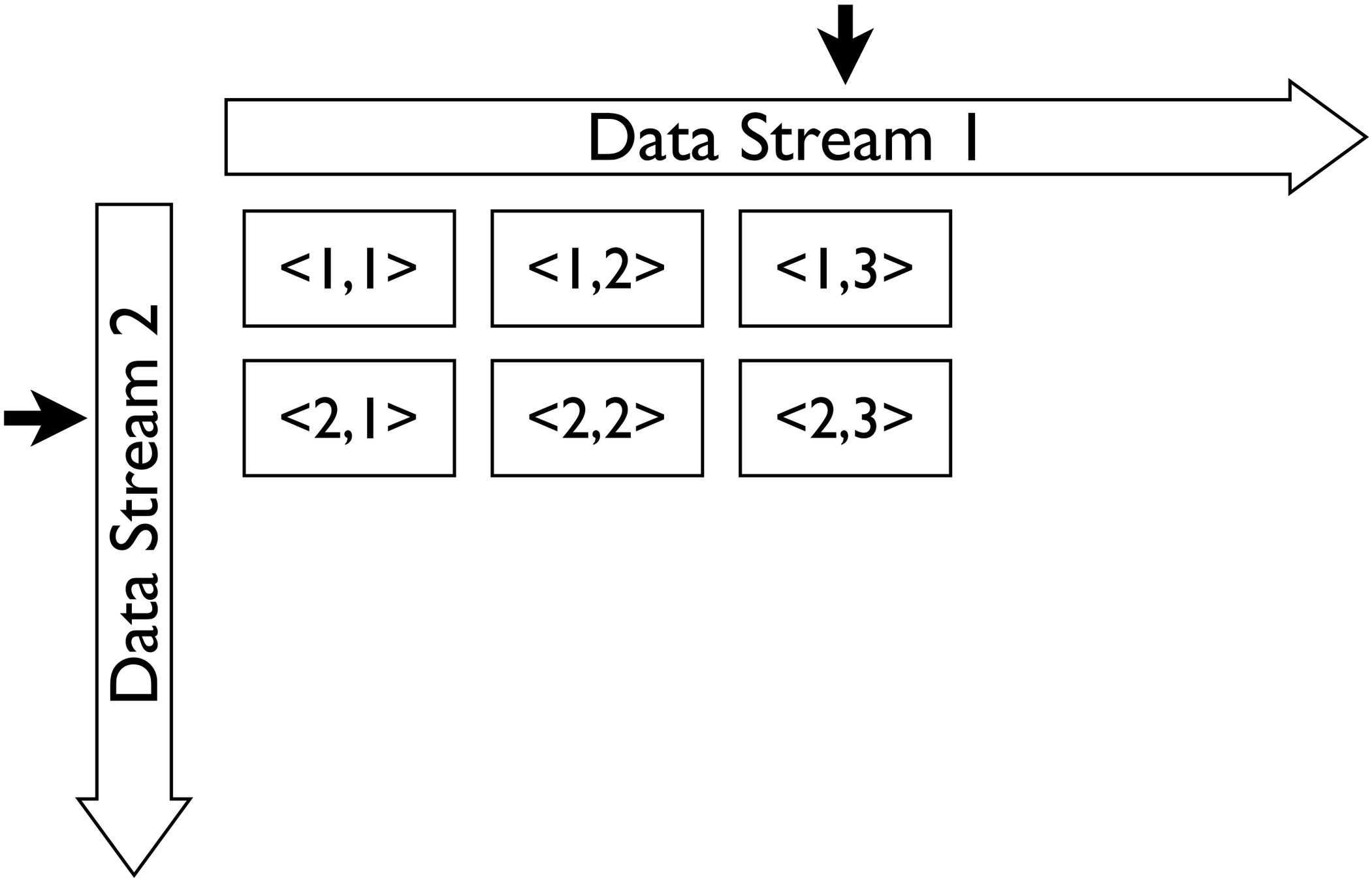
Ripple Join



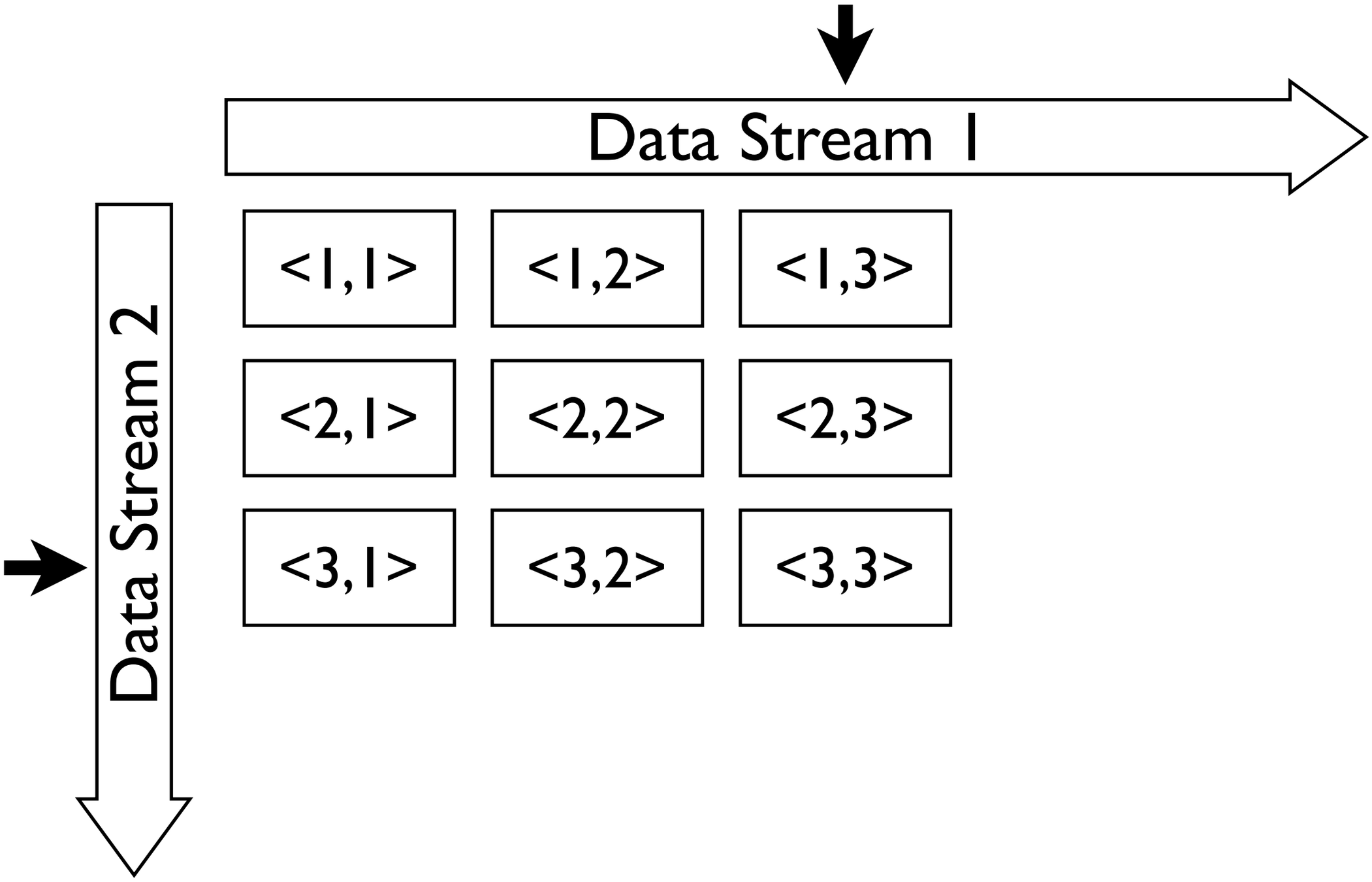
Ripple Join



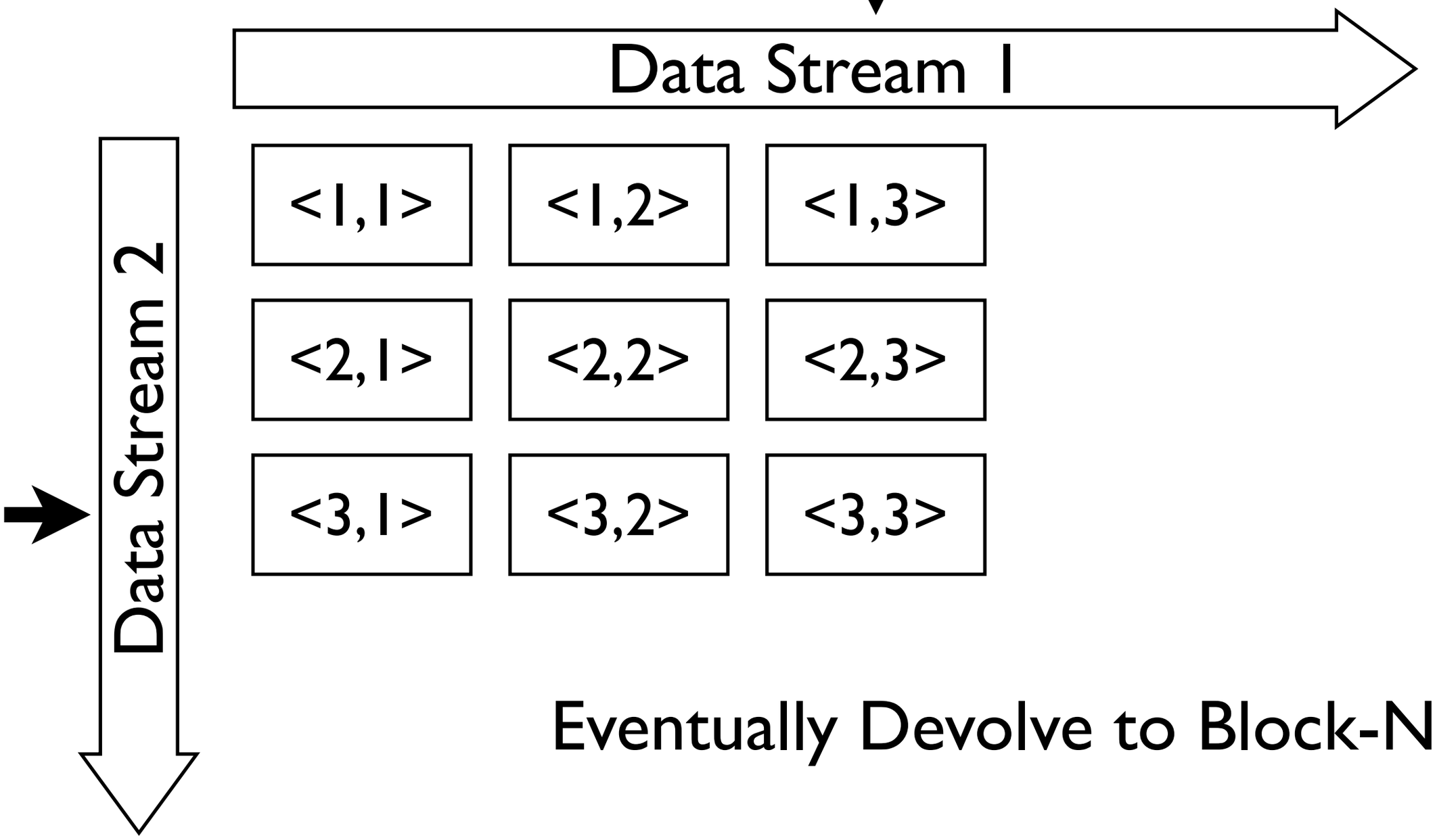
Ripple Join



Ripple Join



Ripple Join



Eventually Devolve to Block-NLJ