Approximation Techniques

Supplemental Reading
(papers posted on Piazza)

- Project 2 Graded.
  - Results will be posted tonight.
  - If anyone has something to say...

The Leader Board

1. <for rent>
2. <for rent>
3. <for rent>
4. <for rent>
5. <for rent>

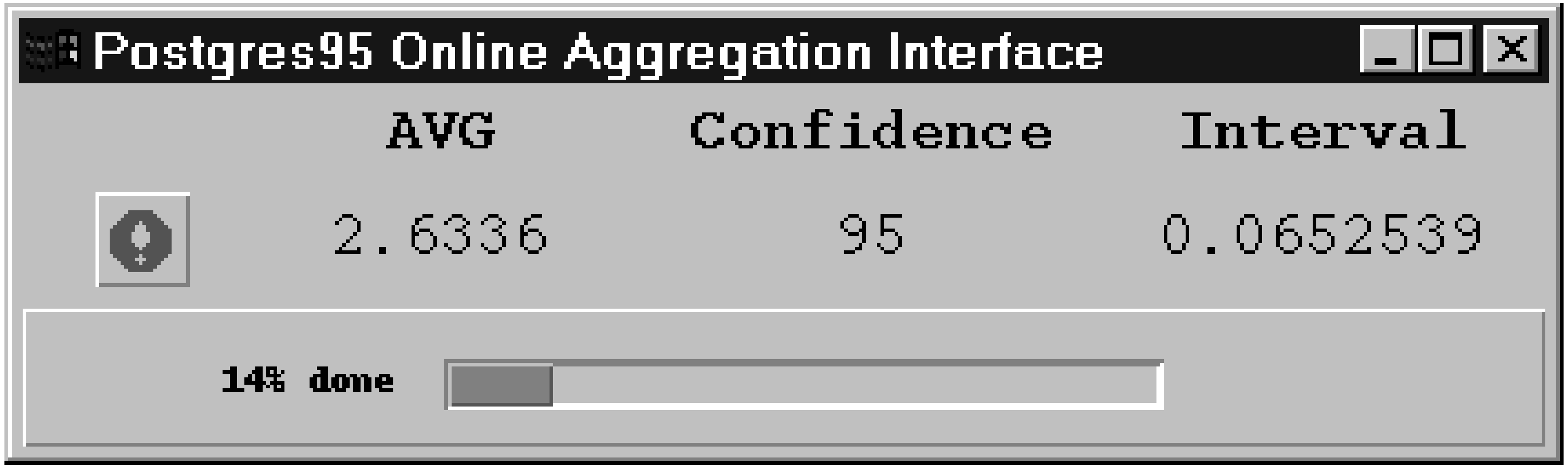
Query Approximation

- 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

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.

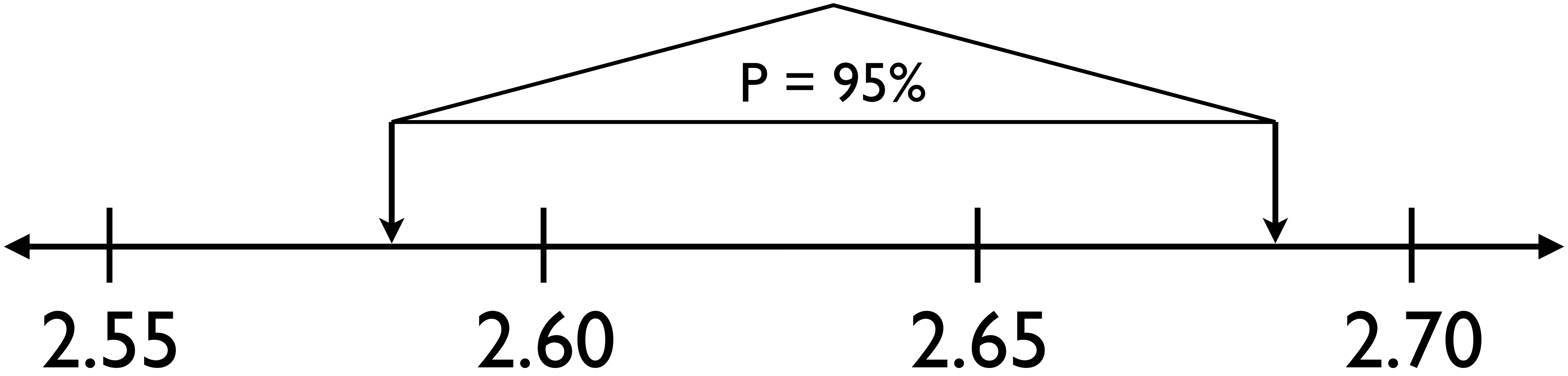
Online Aggregation



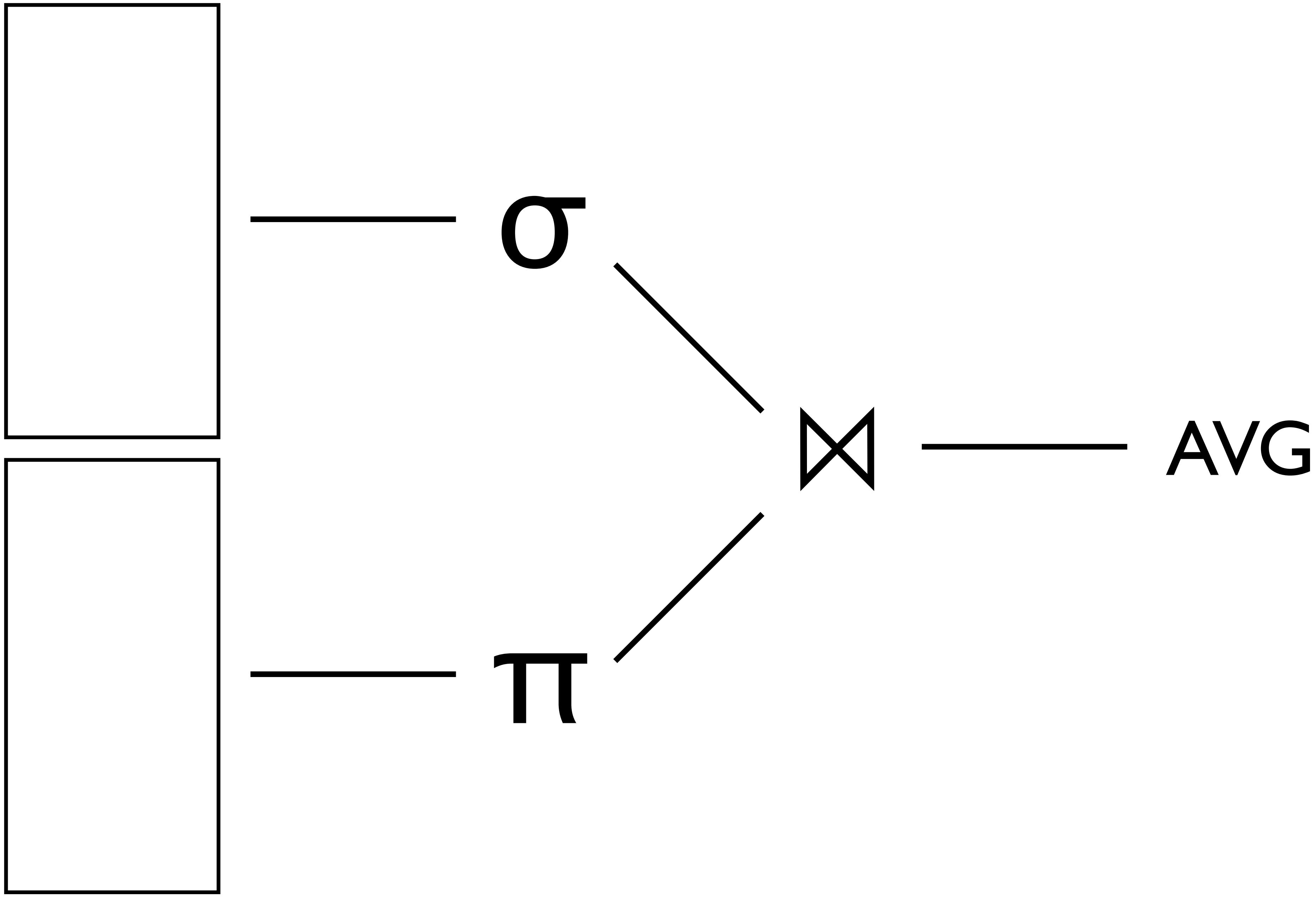
Postgres95 Online Aggregation Interface

AVG	Confidence	Interval
2.6336	95	0.0652539

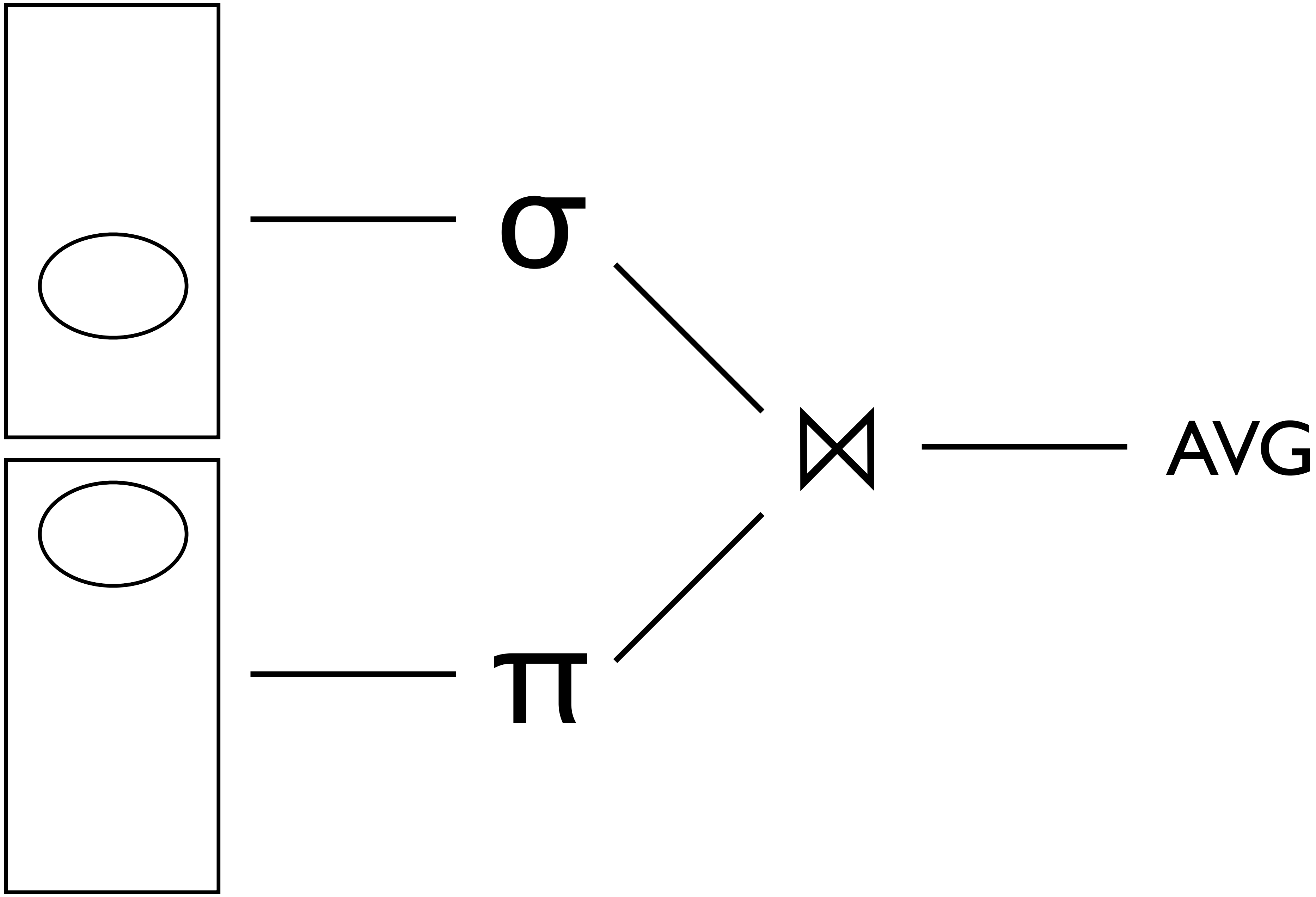
14% done



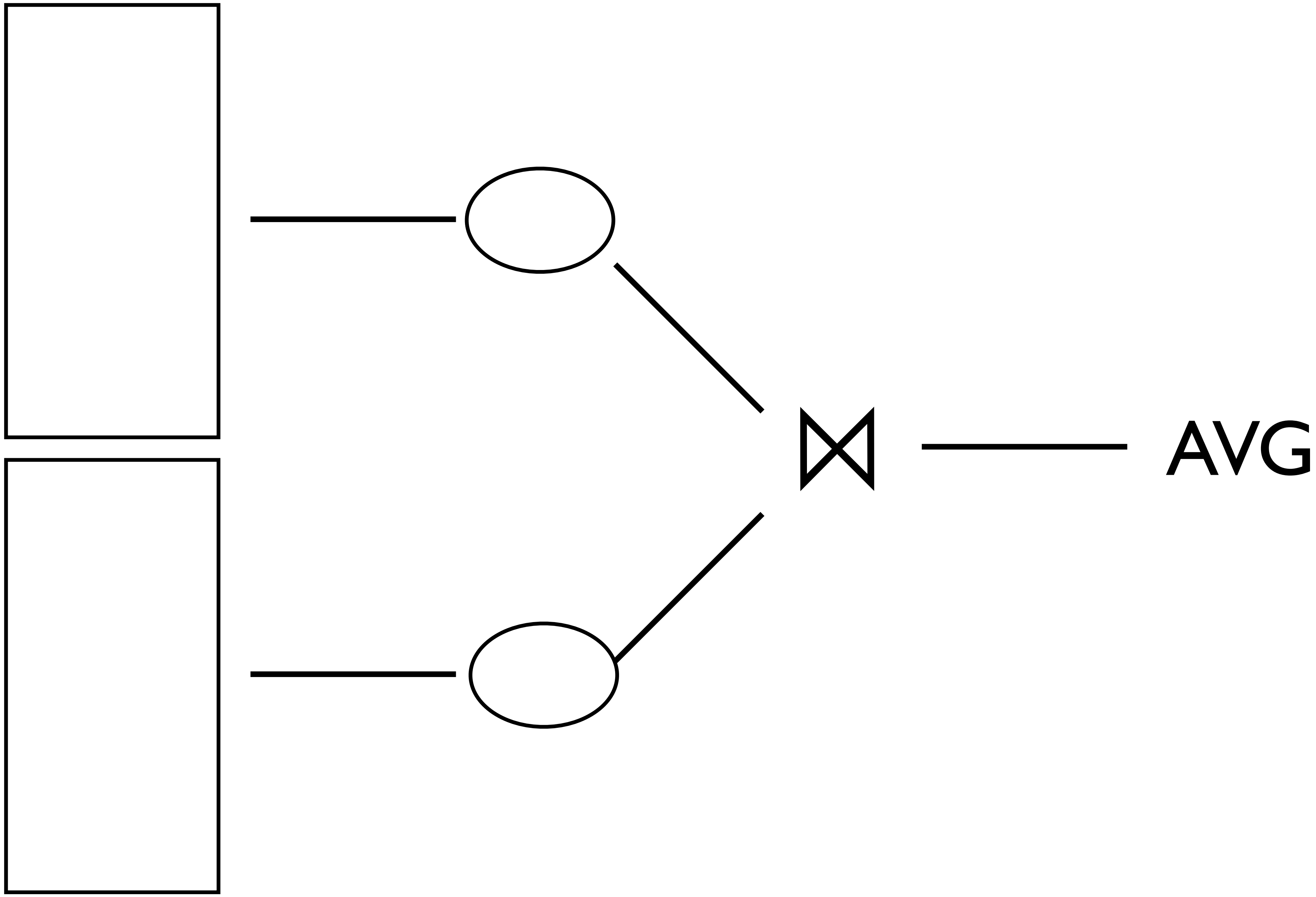
Online Aggregation



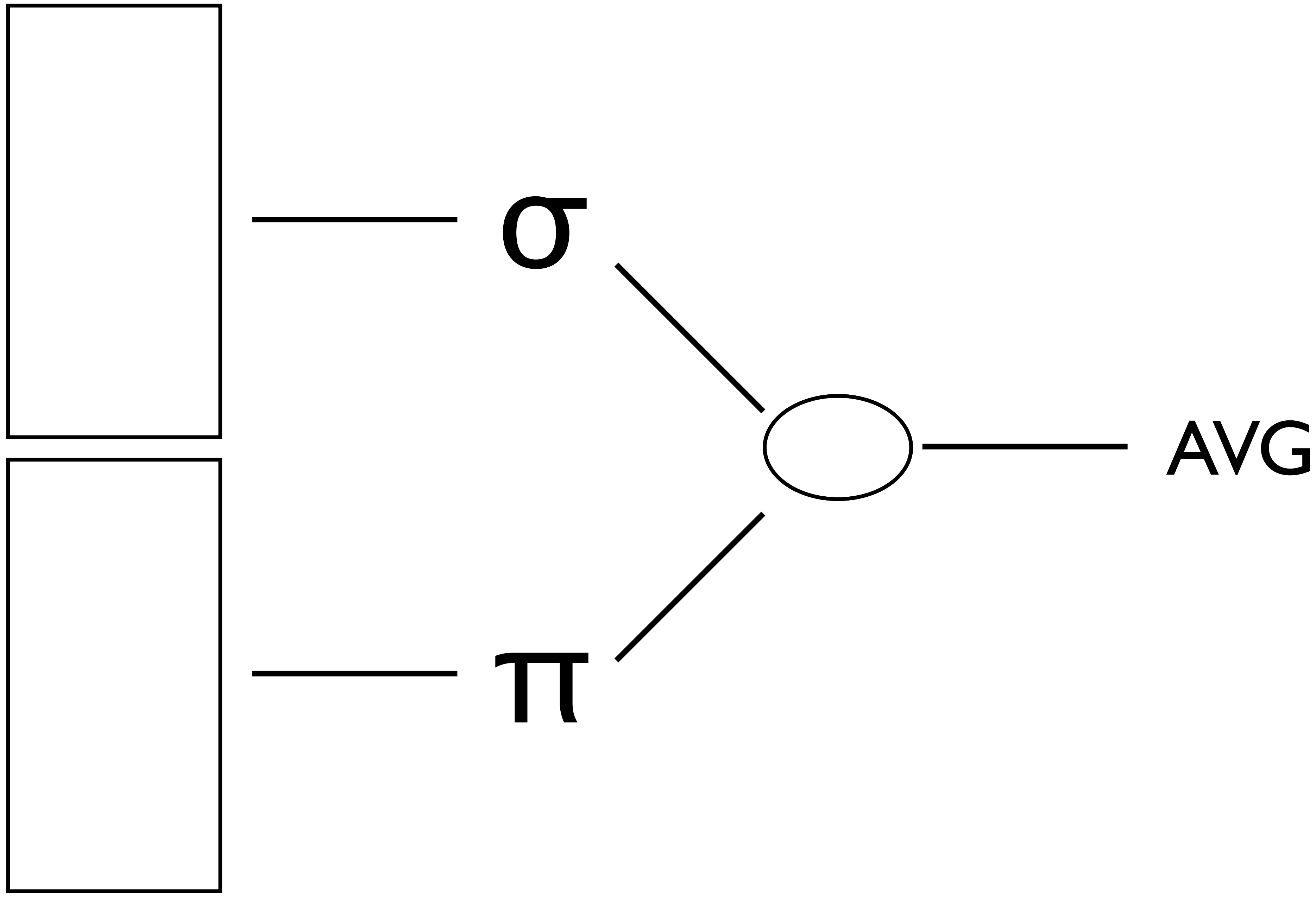
Online Aggregation



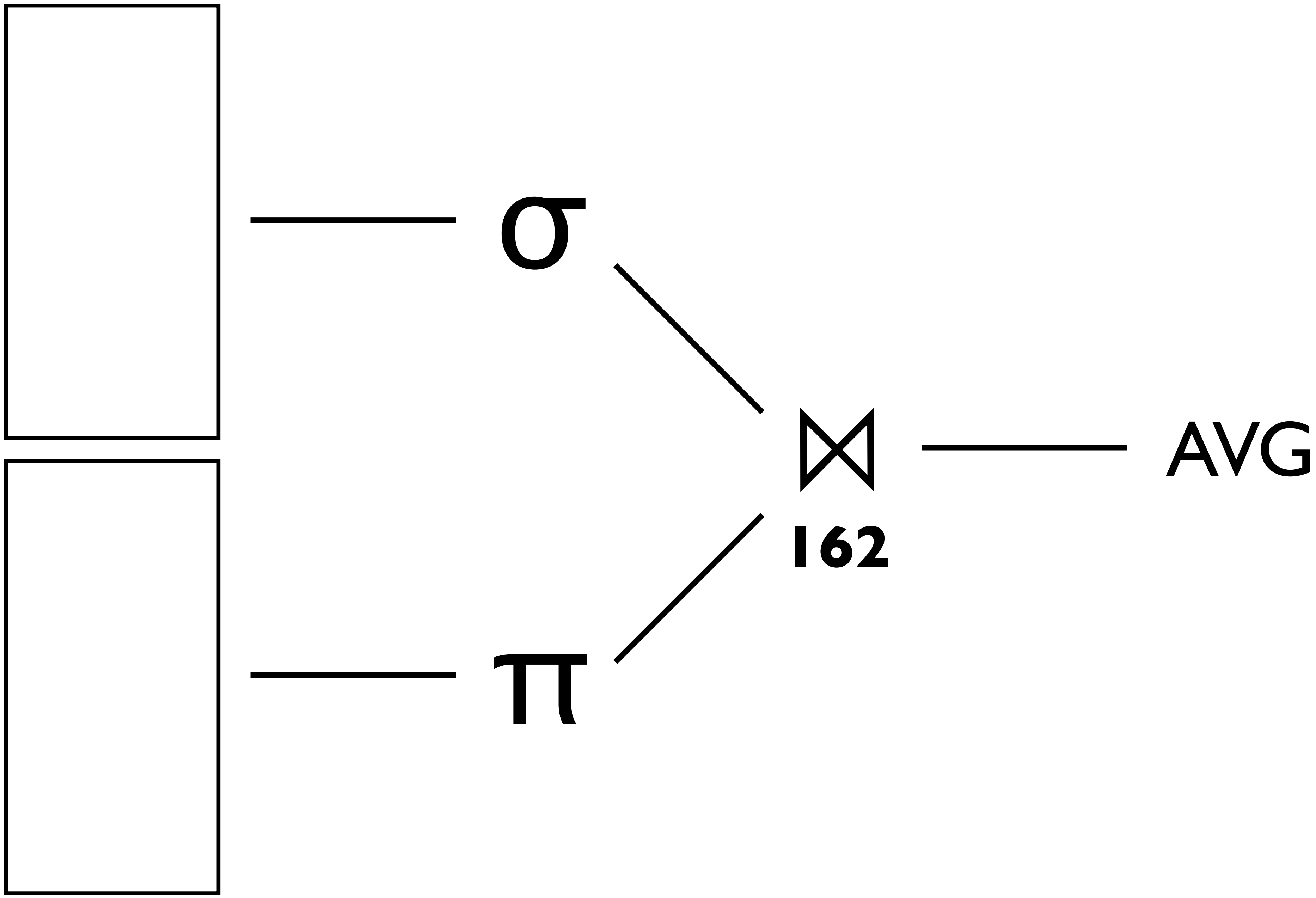
Online Aggregation



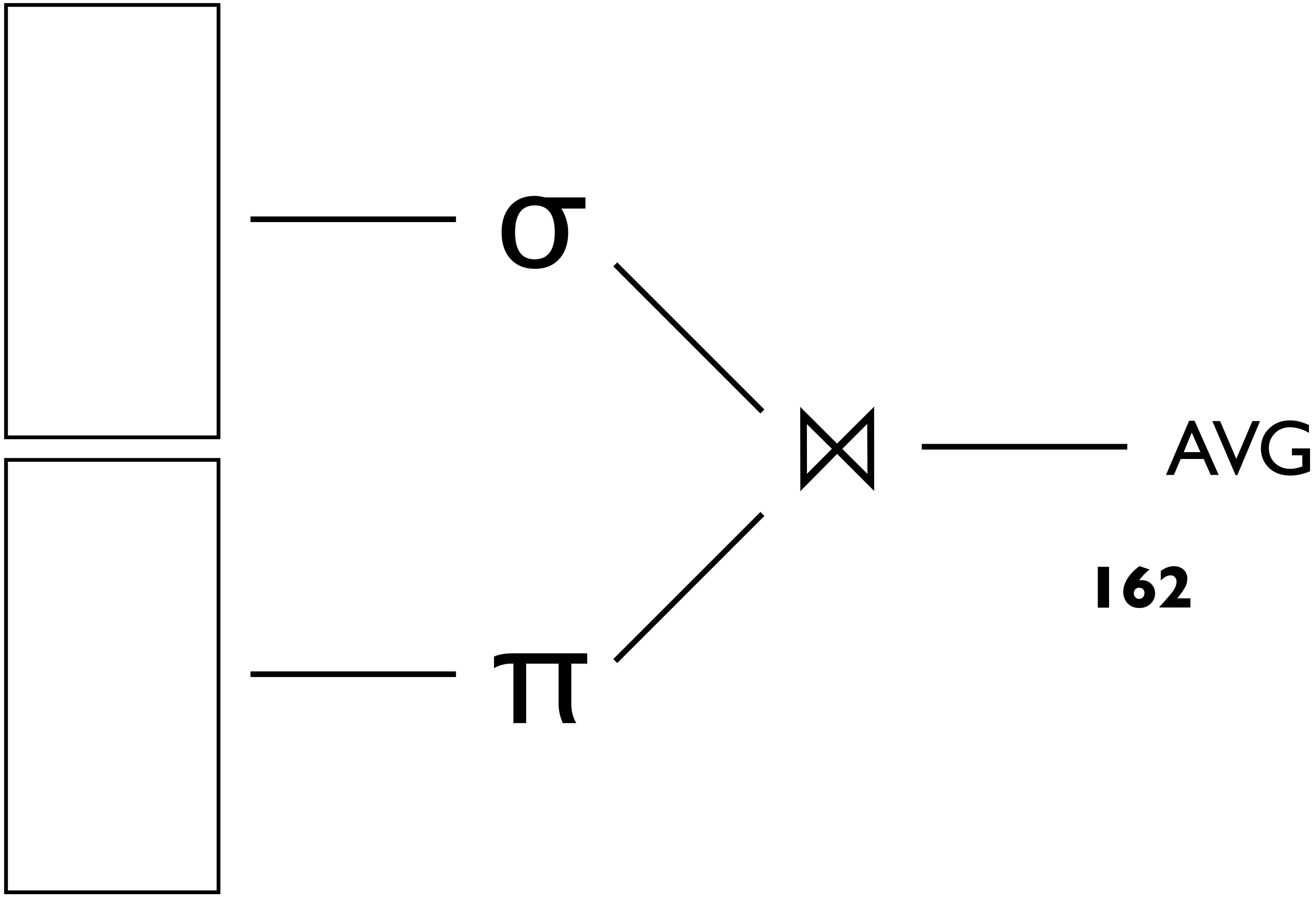
Online Aggregation



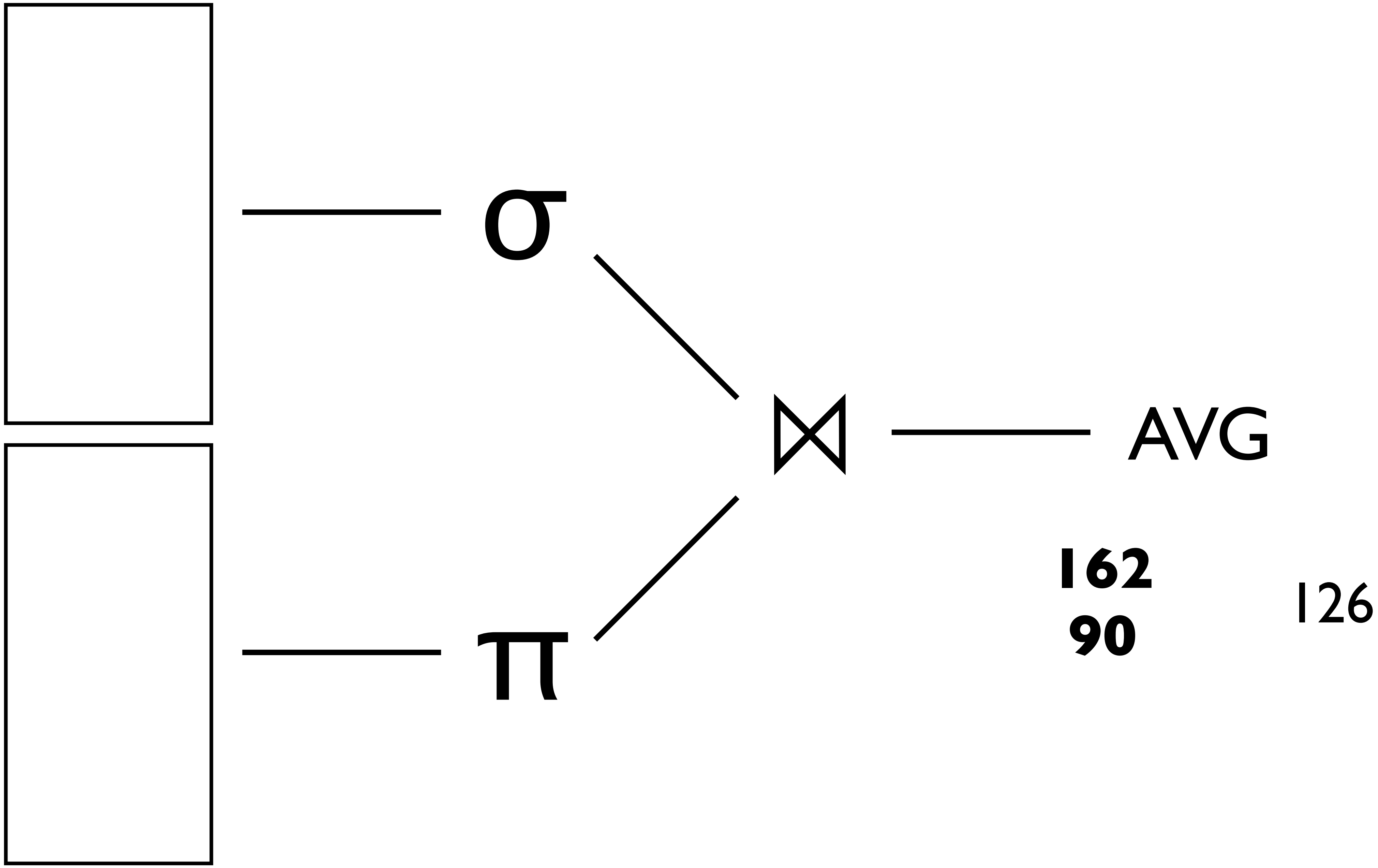
Online Aggregation



Online Aggregation



Online Aggregation



σ

π

⋈

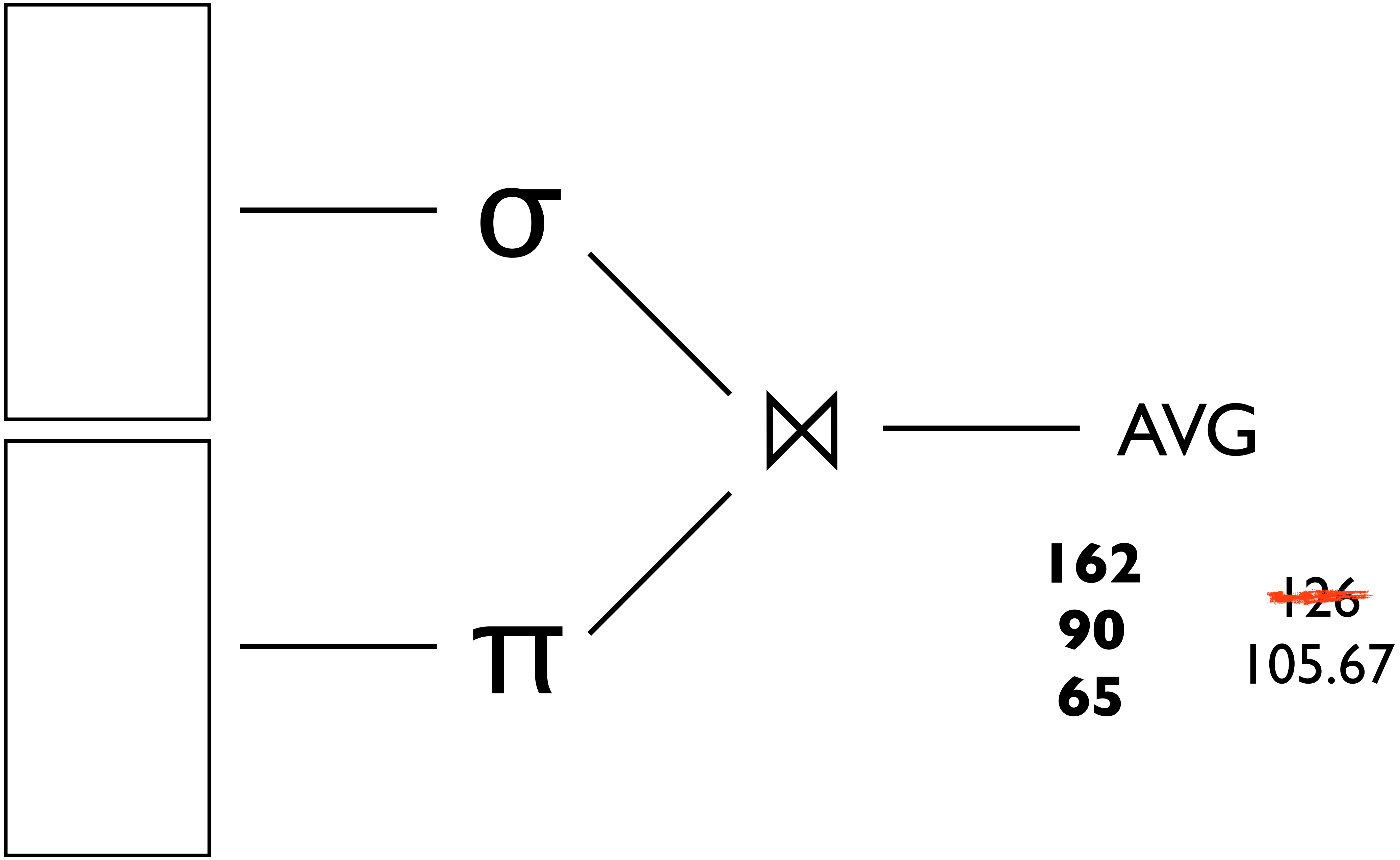
AVG

162

90

126

Online Aggregation



σ

π

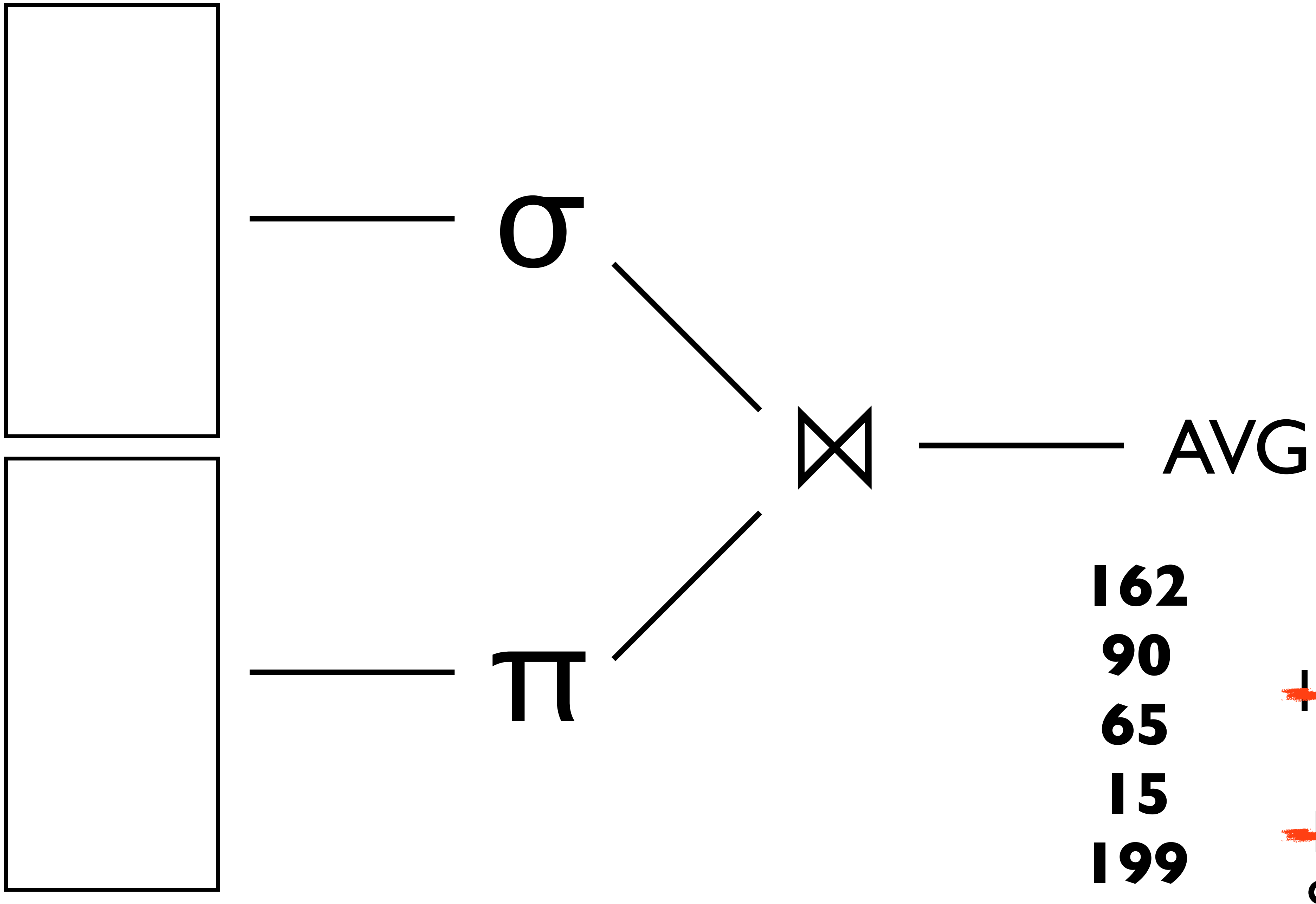
$\bowtie$

AVG

162
90
65

~~126~~
105.67

Online Aggregation



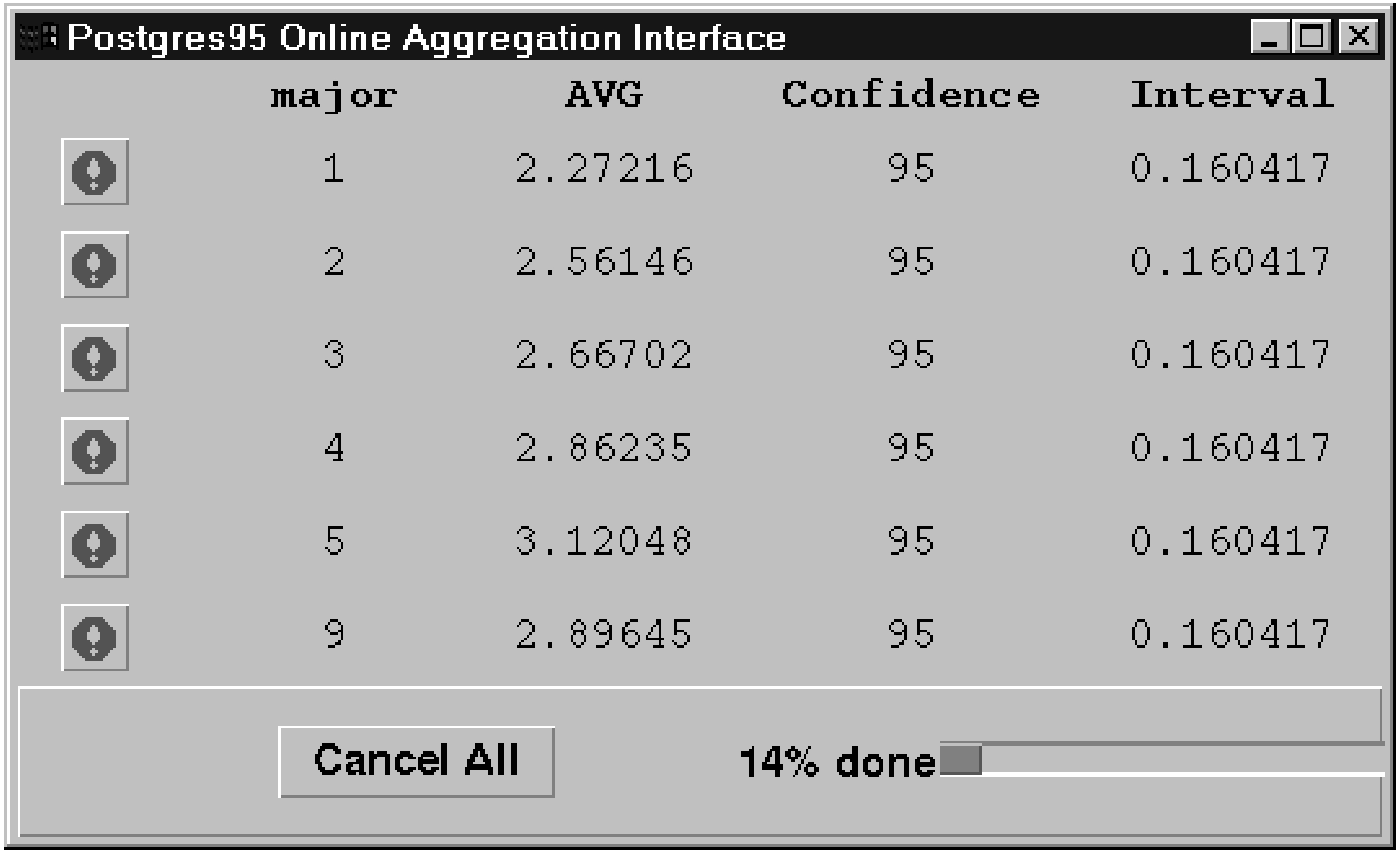
162
90
65
15
199
58

~~126~~
~~105.67~~
~~83~~
~~106.2~~
98.17

Online 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

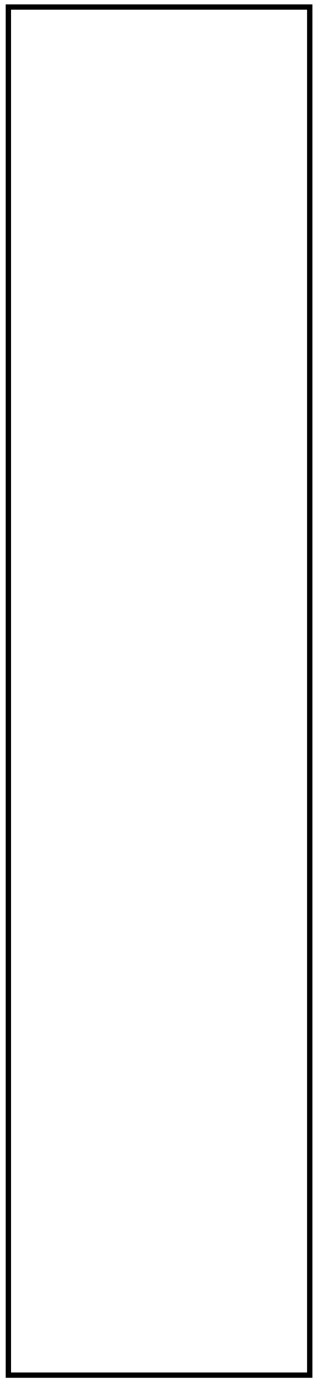


major	AVG	Confidence	Interval
1	2.27216	95	0.160417
2	2.56146	95	0.160417
3	2.66702	95	0.160417
4	2.86235	95	0.160417
5	3.12048	95	0.160417
9	2.89645	95	0.160417

Group by aggregates produce <u>many</u> results.

Each sample contributes to <u>one</u> result.

Fairness



GB Key 1

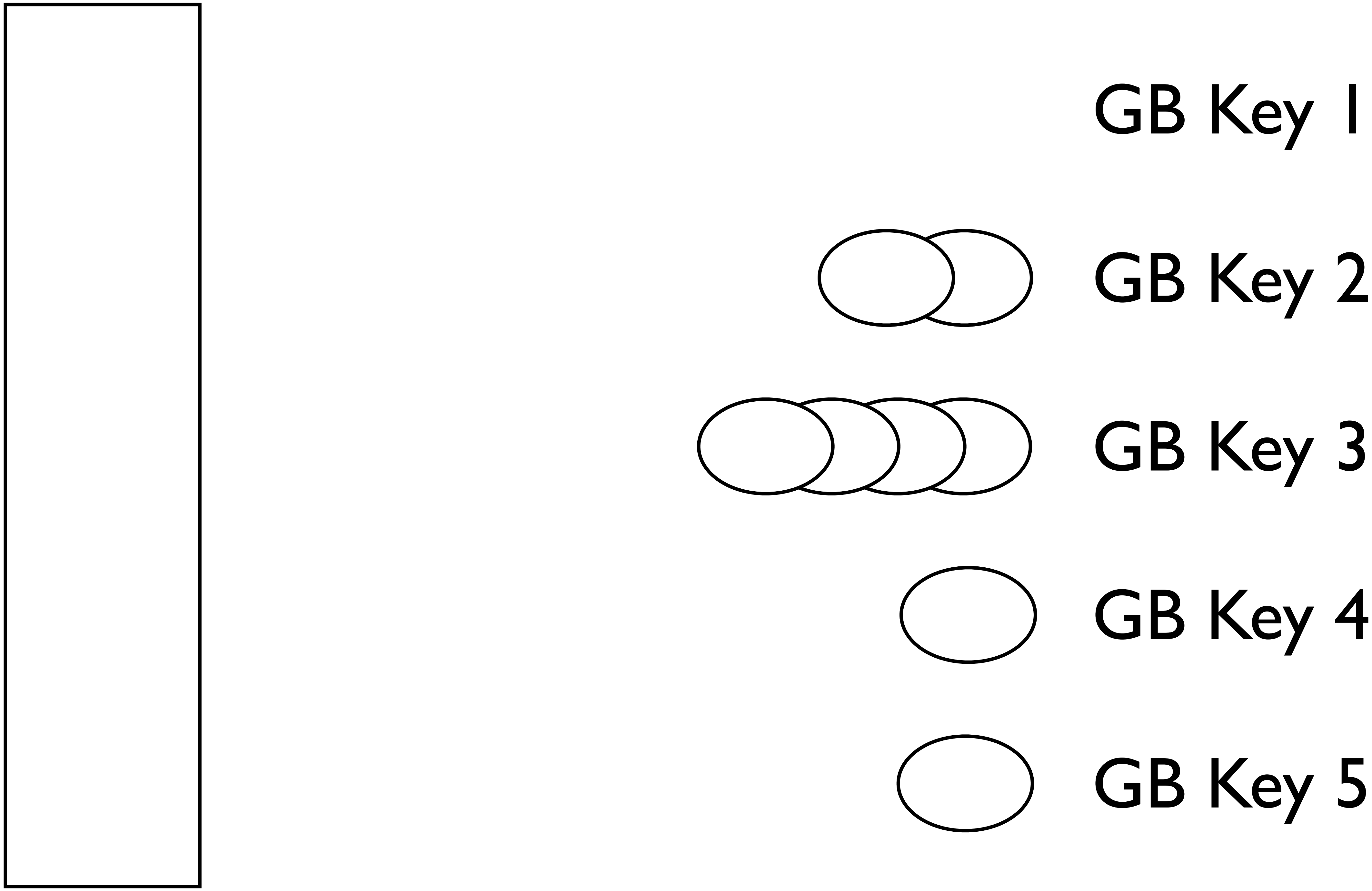
GB Key 2

GB Key 3

GB Key 4

GB Key 5

Fairness



GB Key 1

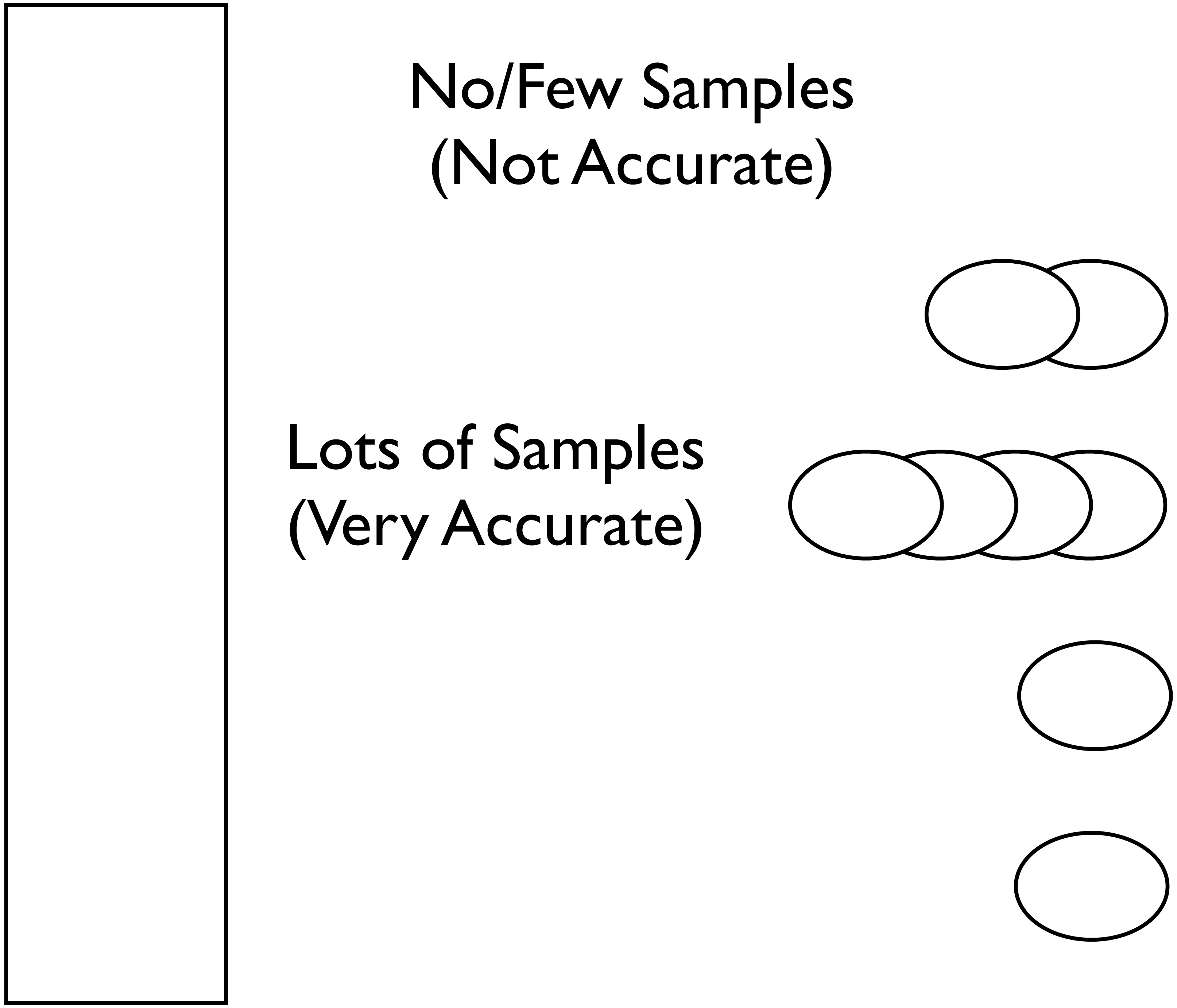
GB Key 2

GB Key 3

GB Key 4

GB Key 5

Fairness

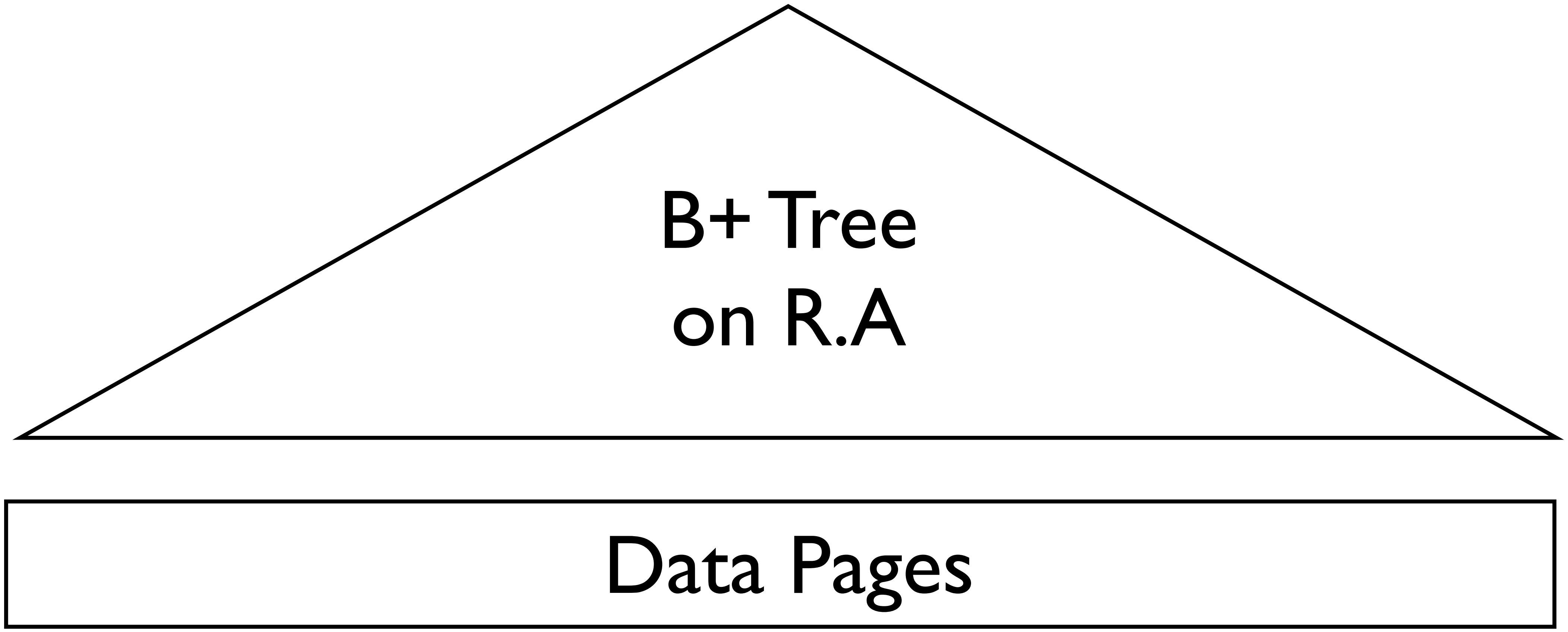

No/Few Samples
(Not Accurate)
Lots of Samples
(Very Accurate)
GB Key 1
GB Key 2
GB Key 3
GB Key 4
GB Key 5

Fairness

Problem: Group-By Key Distribution in Data May Be Skewed.

Solution: Scan All Group-By Keys In Parallel

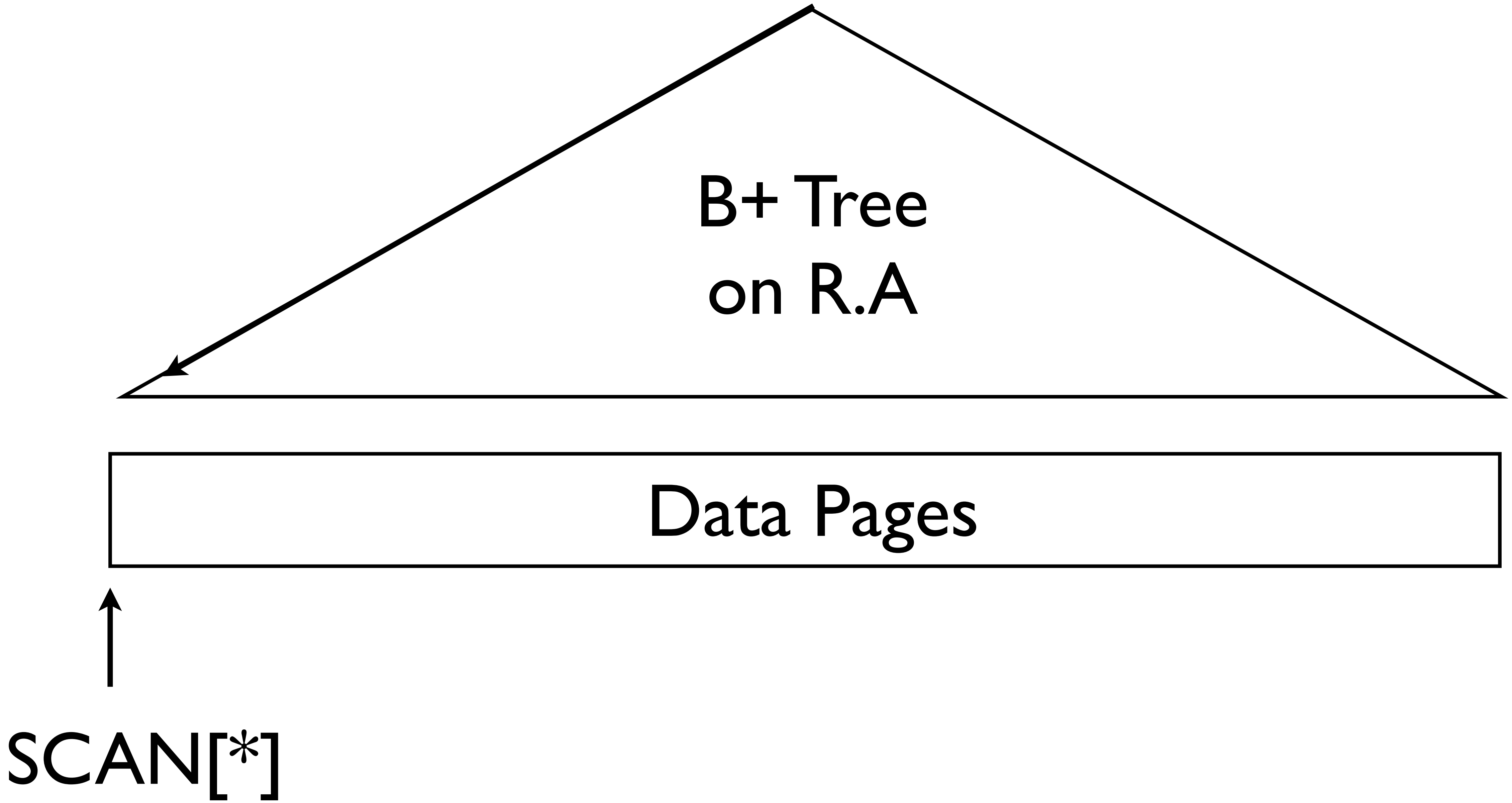
Index Striding



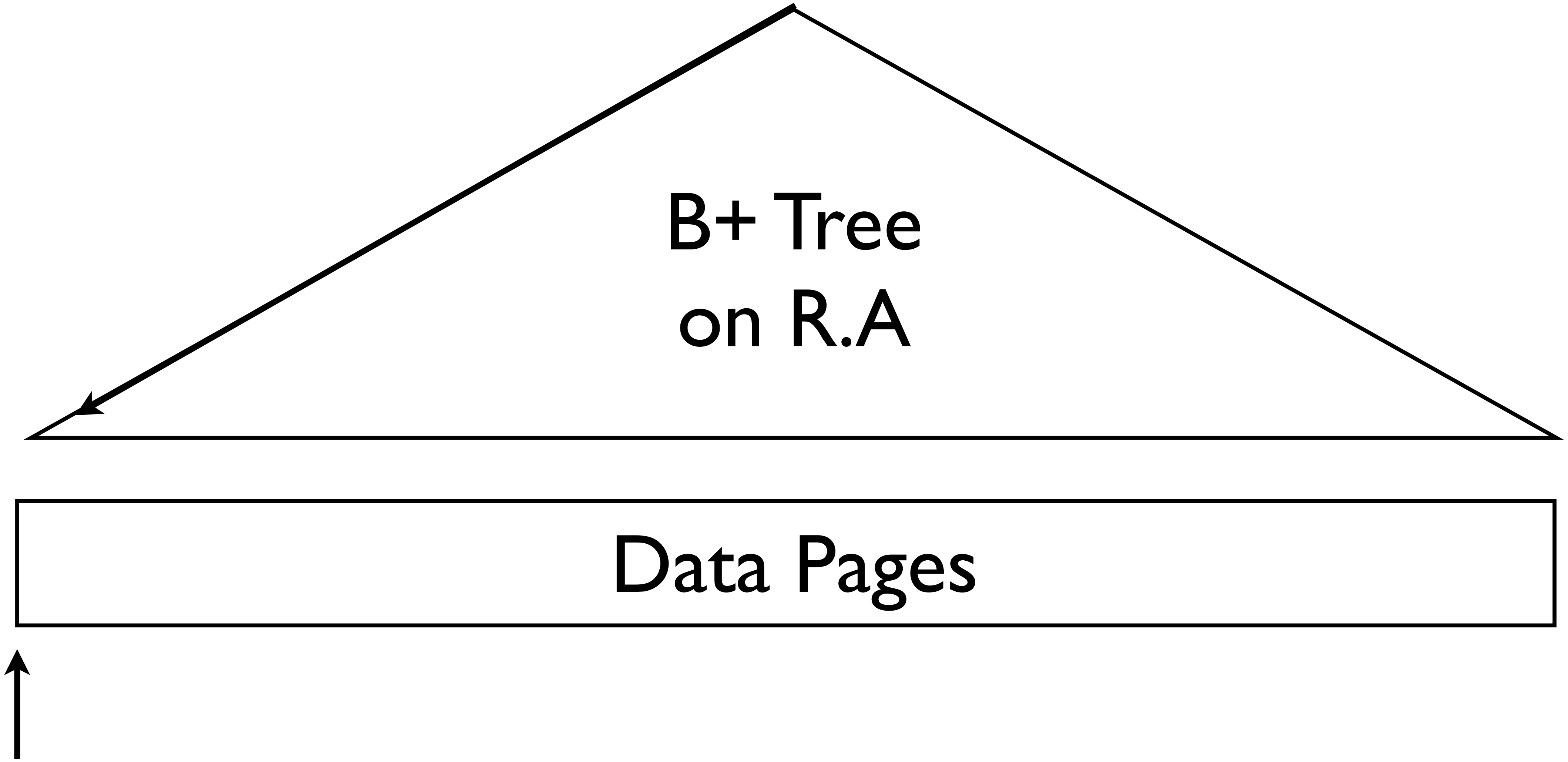
B+ Tree
on R.A

Data Pages

Index Striding

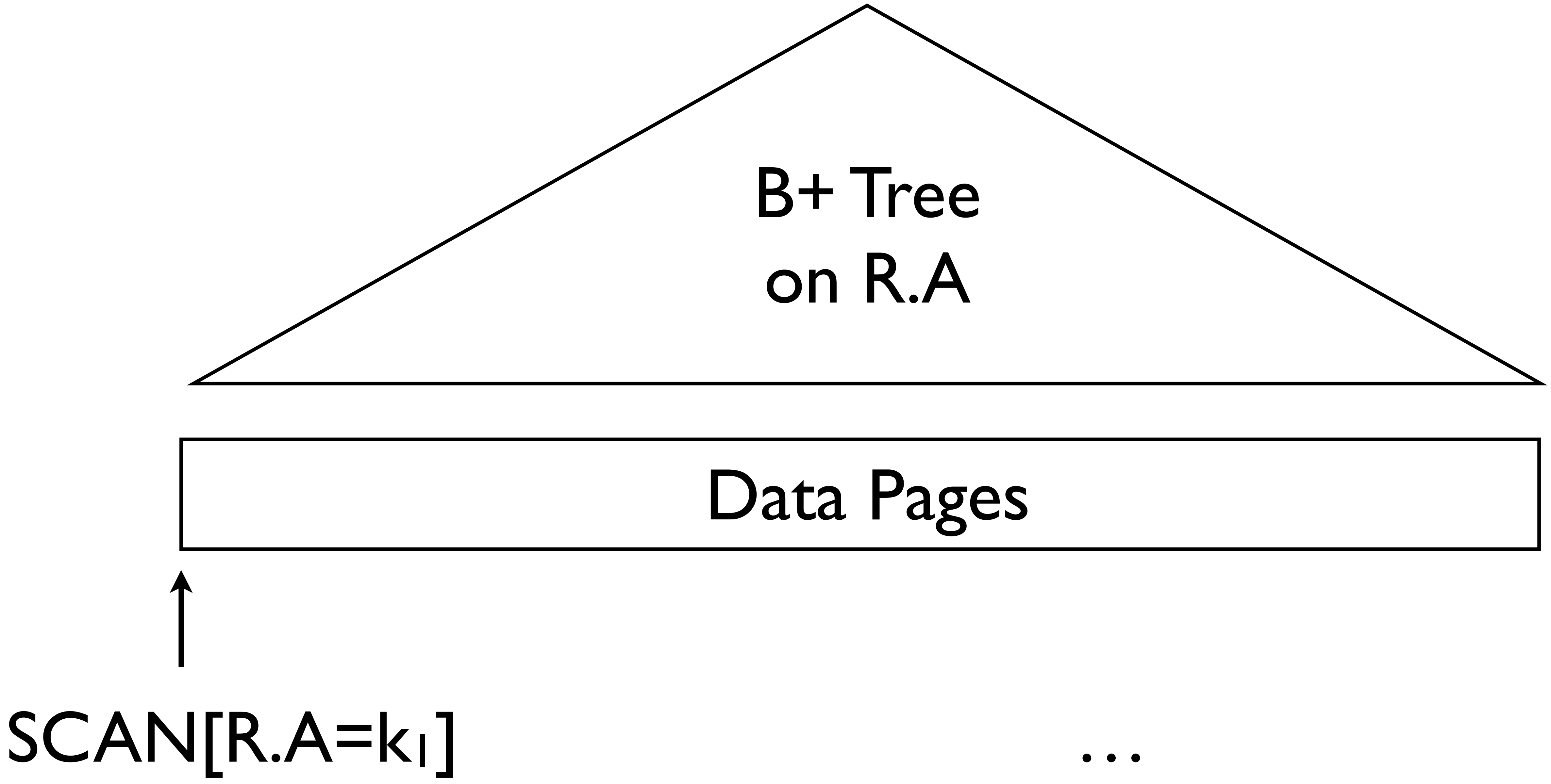


Index Striding



SCAN[R.A=k_1]

Index Striding



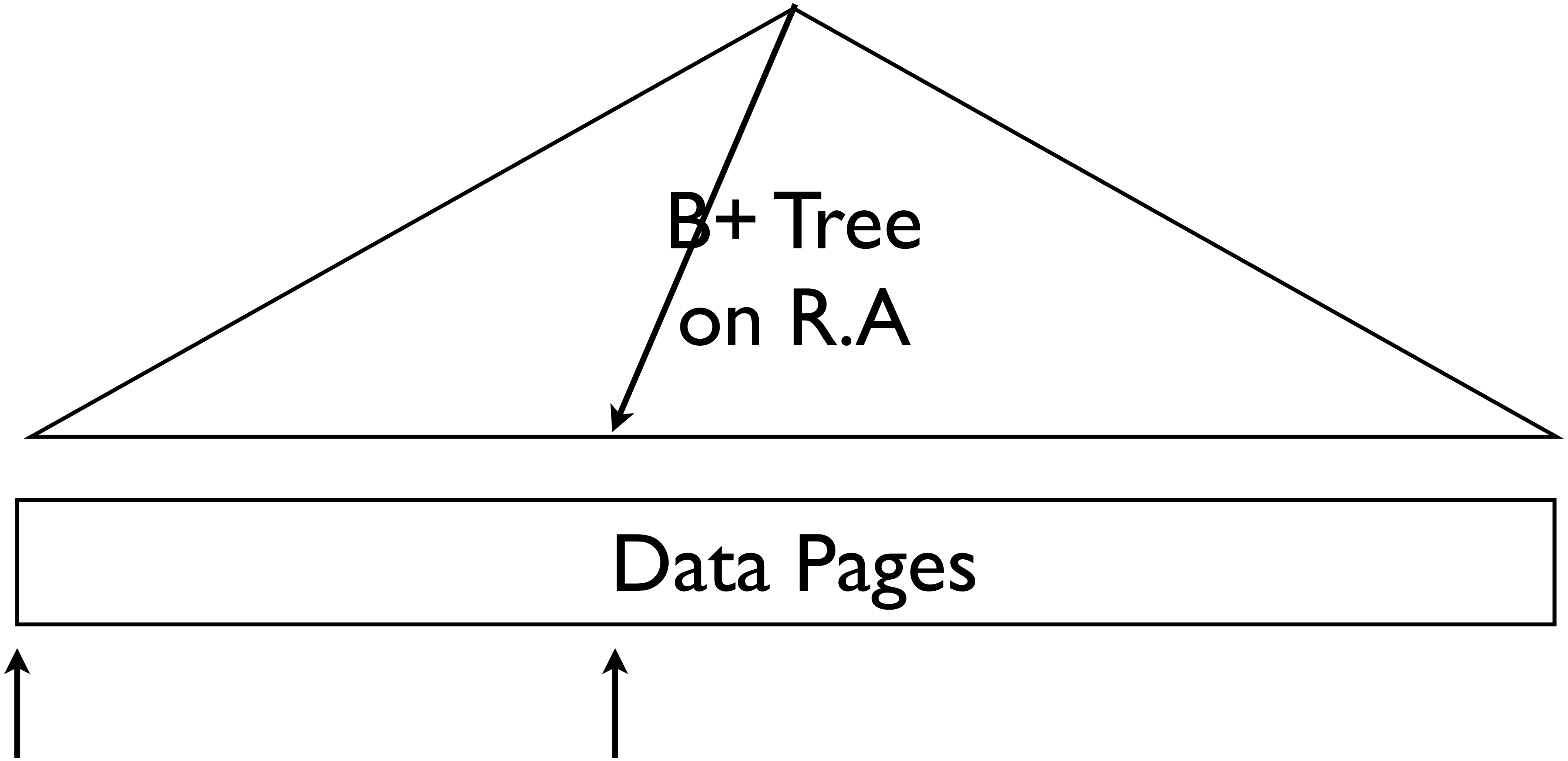
B+ Tree
on R.A

Data Pages

SCAN[R.A=k_1]

…

Index Striding

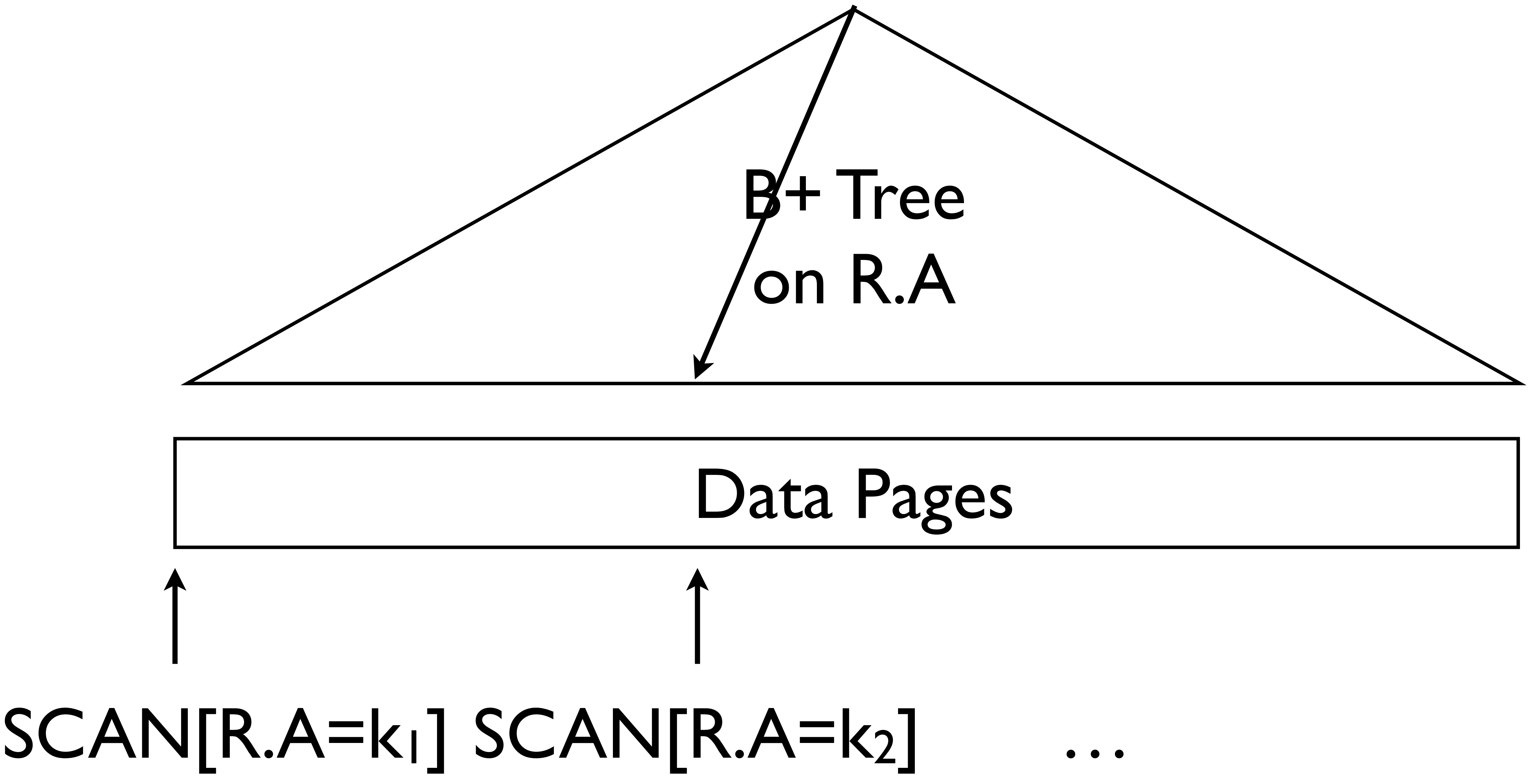


B+ Tree
on R.A

Data Pages

SCAN[R.A=k_1] SCAN[R.A>k_1] …

Index Striding



B+ Tree
on R.A

Data Pages

SCAN[R.A=k_1] SCAN[R.A=k_2] …

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.

Blocking

Problem: Can’t get “Online” results with blocking operations

Solution: Only required blocking op: Join. Use nonblocking joins.

Non-Blocking Joins

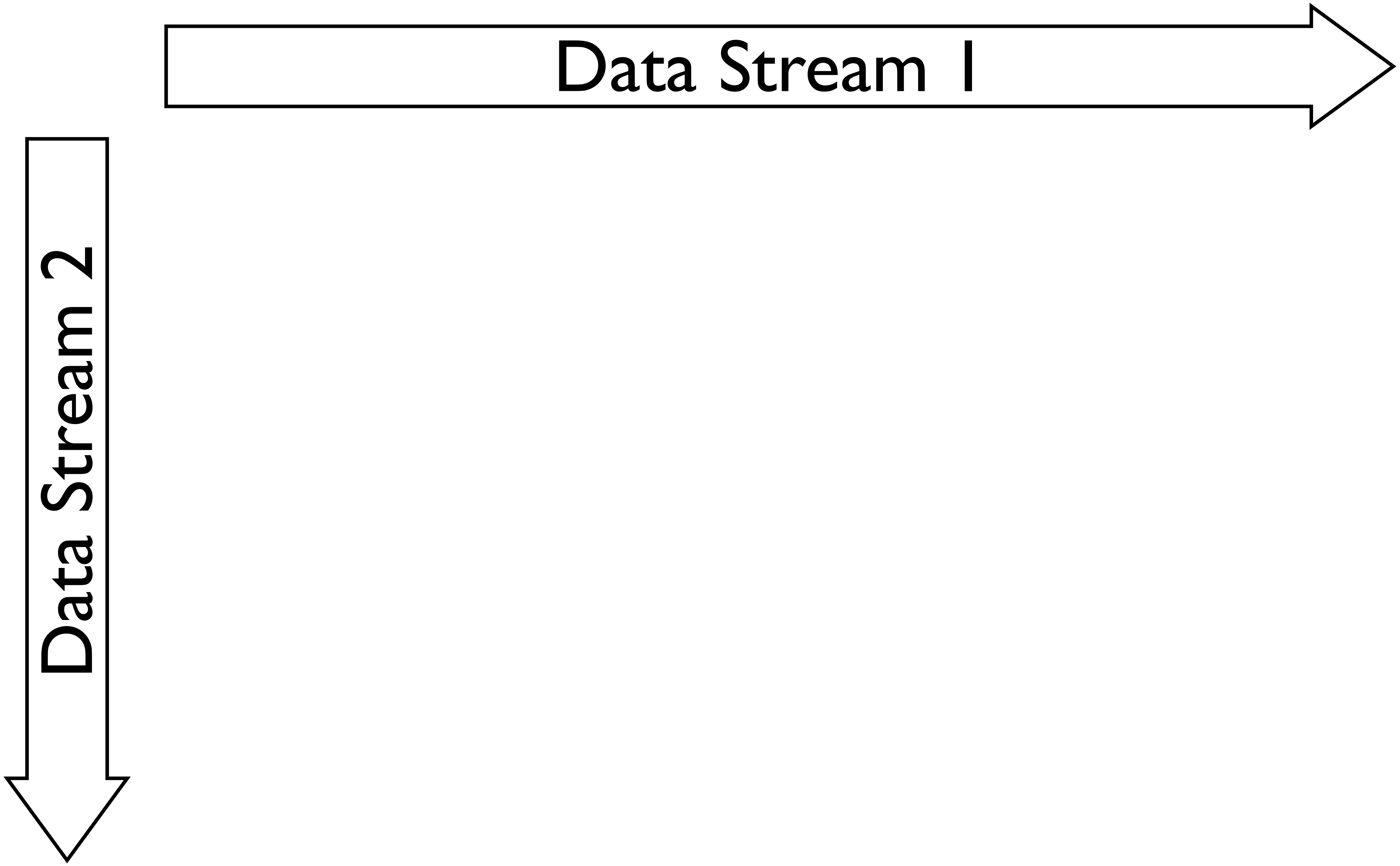
- Sort/Merge Join
    - We want the data unsorted
- Index-Nested Loop Join
    - Could work if only few tuples matched.
- Hybrid Hash Join
    - Could work if one table is small.

Blocking

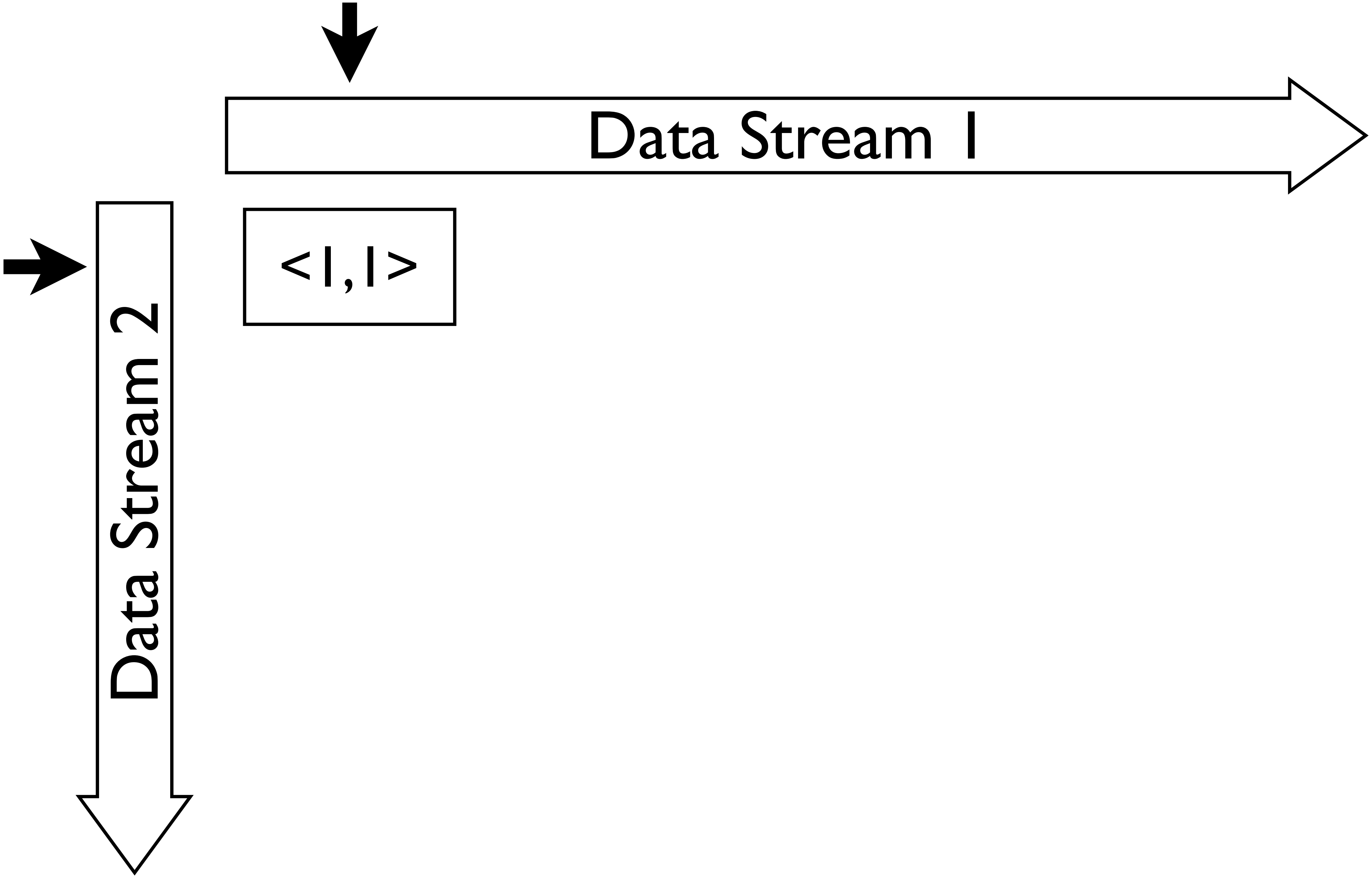
Problem: Join Algos are Blocking

Solution: Ripple Joins

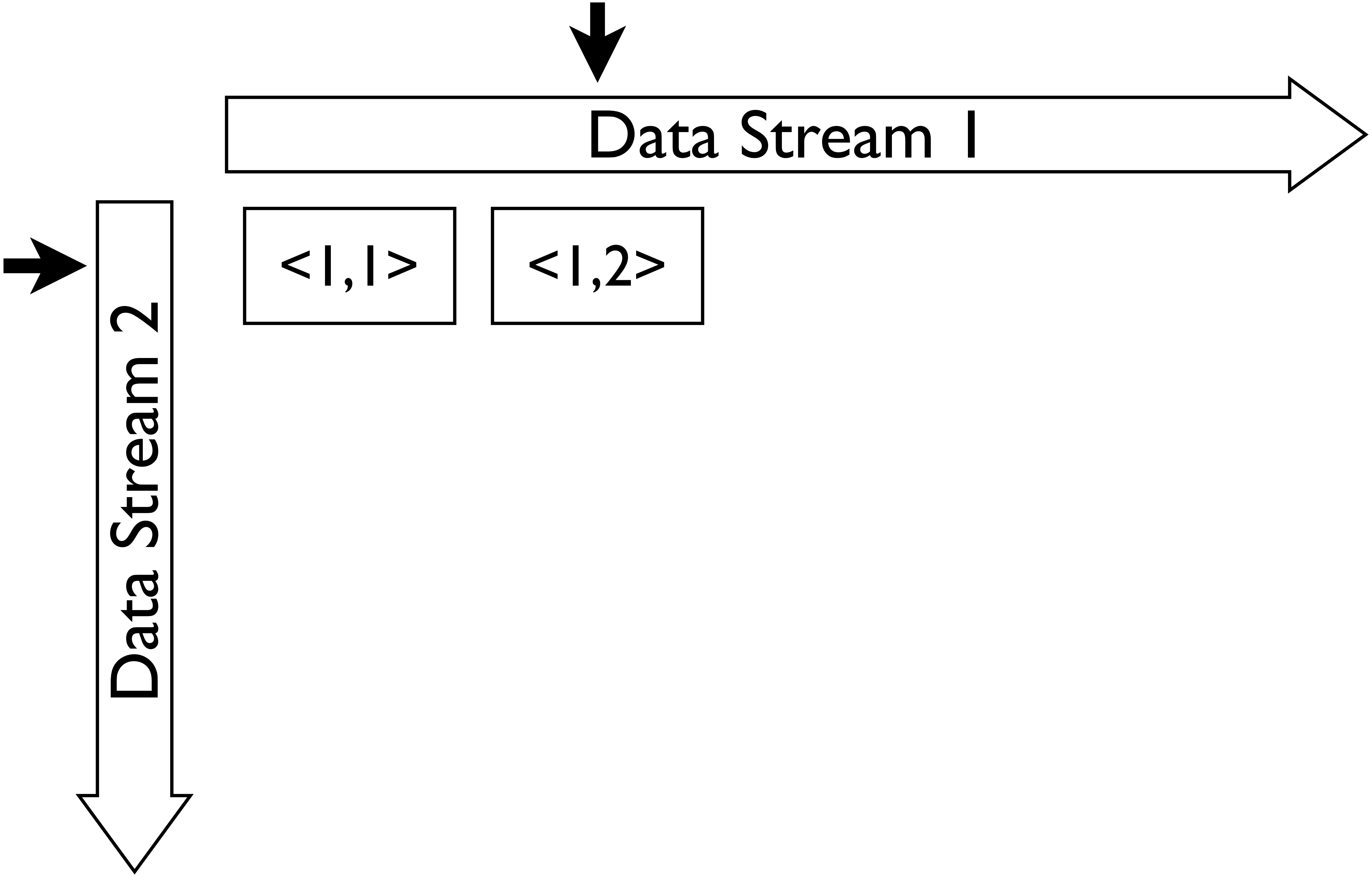
Ripple Join



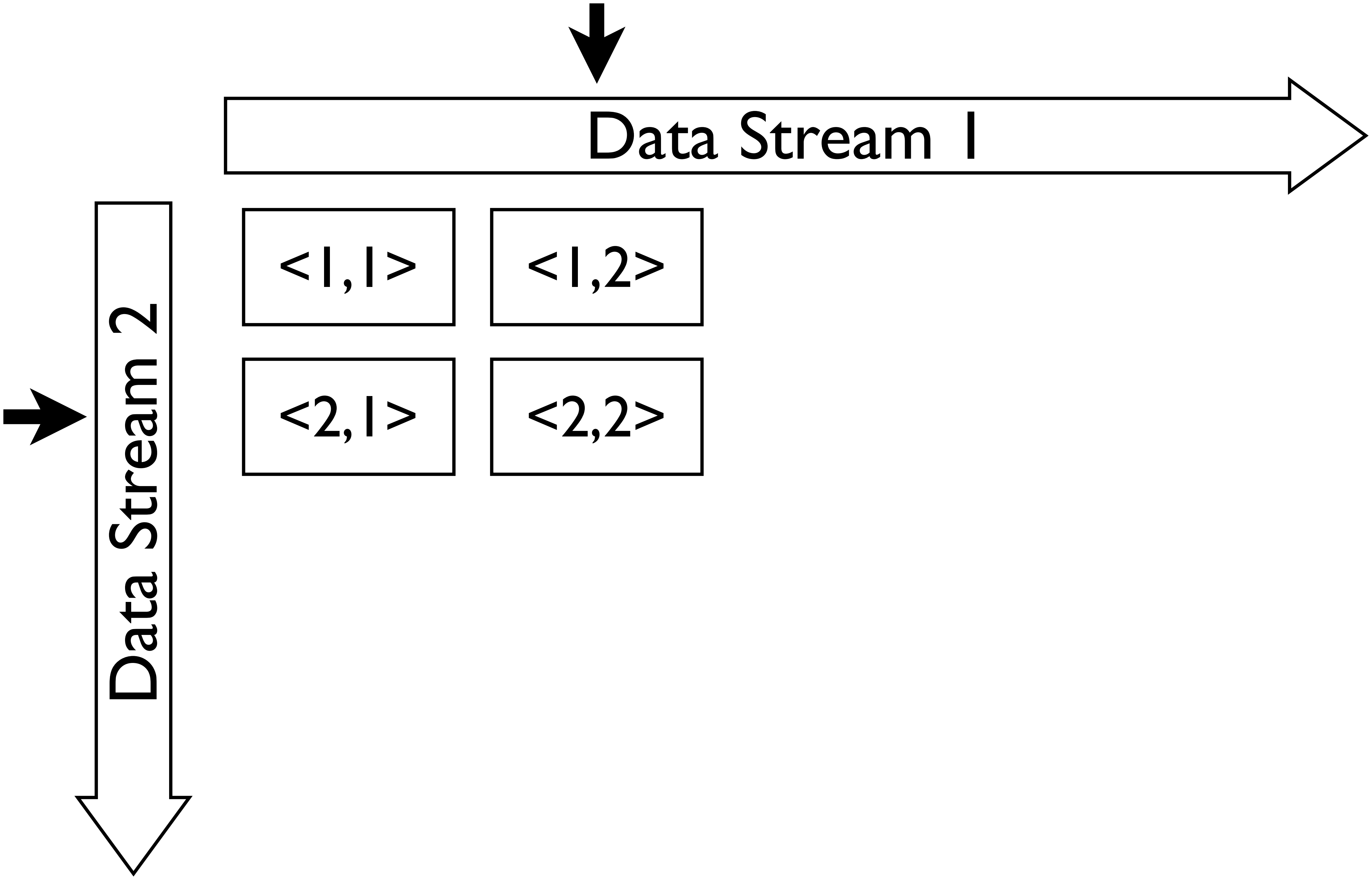
Ripple Join



Ripple Join



Ripple Join



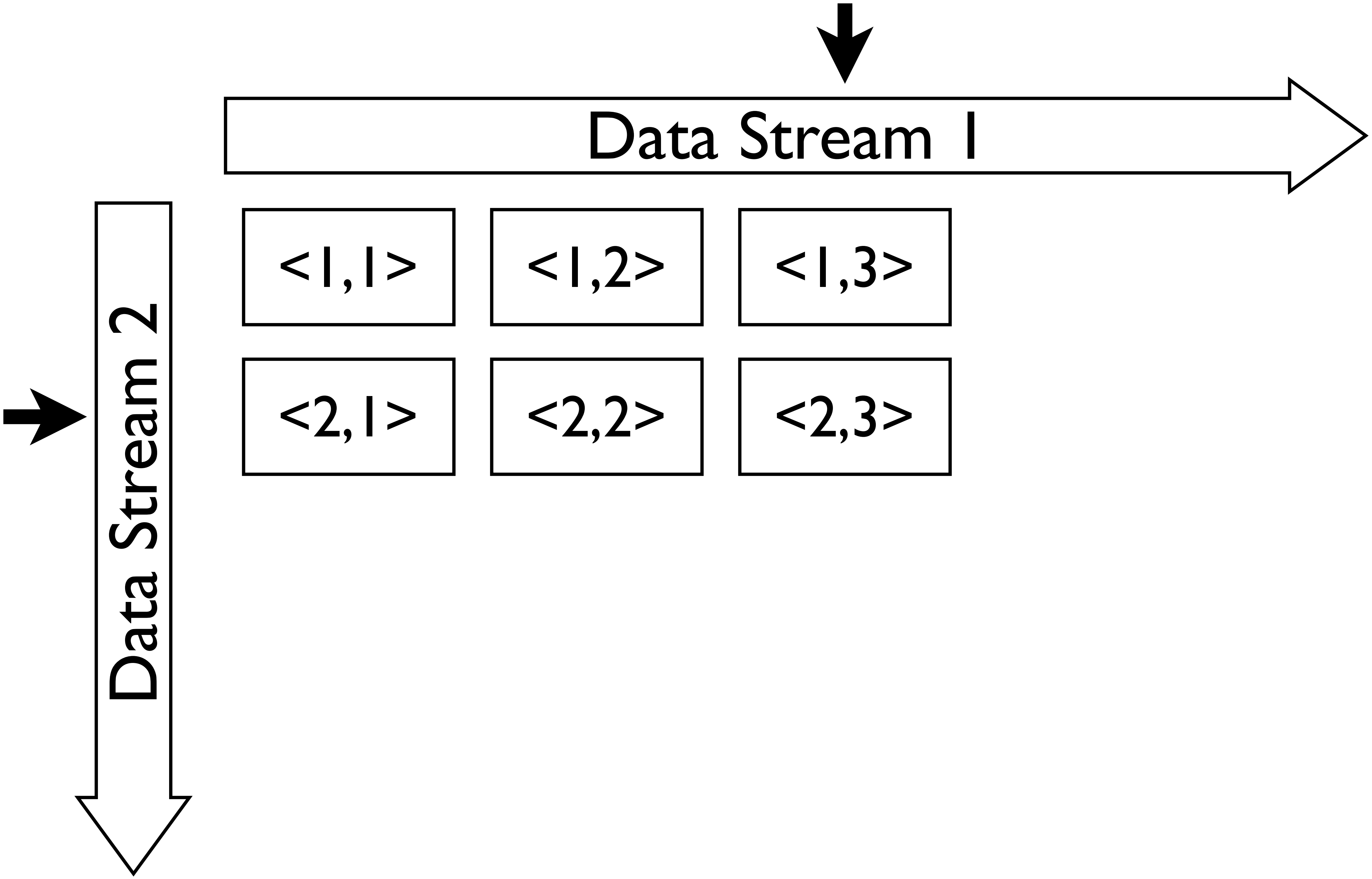
Data Stream 1

Data Stream 2

<1,1> <1,2>

<2,1> <2,2>

Ripple Join



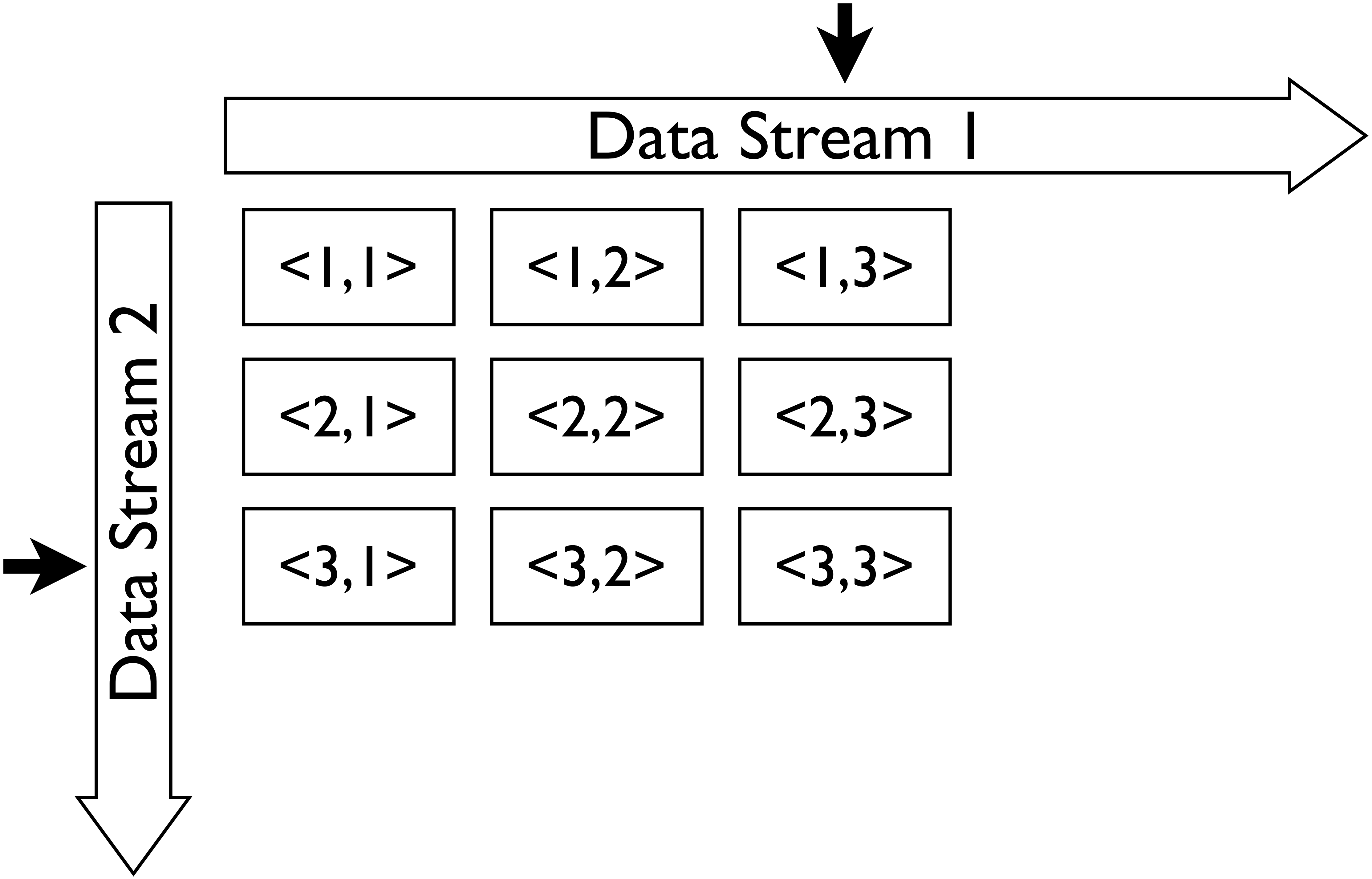
Data Stream 1

Data Stream 2

<1,1> <1,2> <1,3>

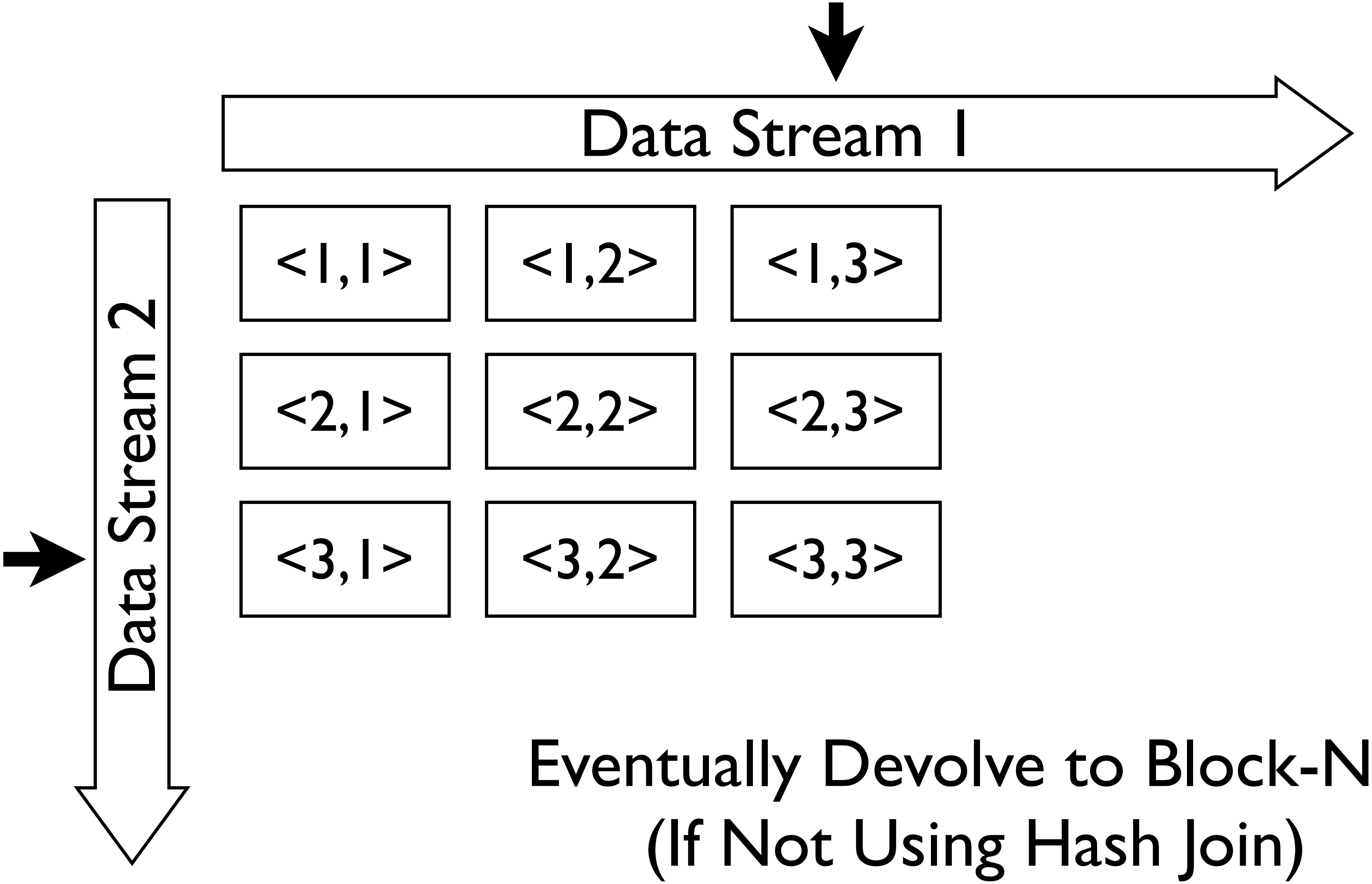
<2,1> <2,2> <2,3>

Ripple Join



<1,1>	<1,2>	<1,3>
<2,1>	<2,2>	<2,3>
<3,1>	<3,2>	<3,3>

Ripple Join



<1,1>	<1,2>	<1,3>
<2,1>	<2,2>	<2,3>
<3,1>	<3,2>	<3,3>

Eventually Devolve to Block-NLJ
(If Not Using Hash Join)

Online Aggregation

- Estimate Aggregate Results by Sampling.
- Continuously Sample and Update Results.
    - Should we sample with replacement?
- Online Aggregation Still an Open Challenge:
    - Random Access Expensive, Even on Flash.
    - Data is Changing Rapidly