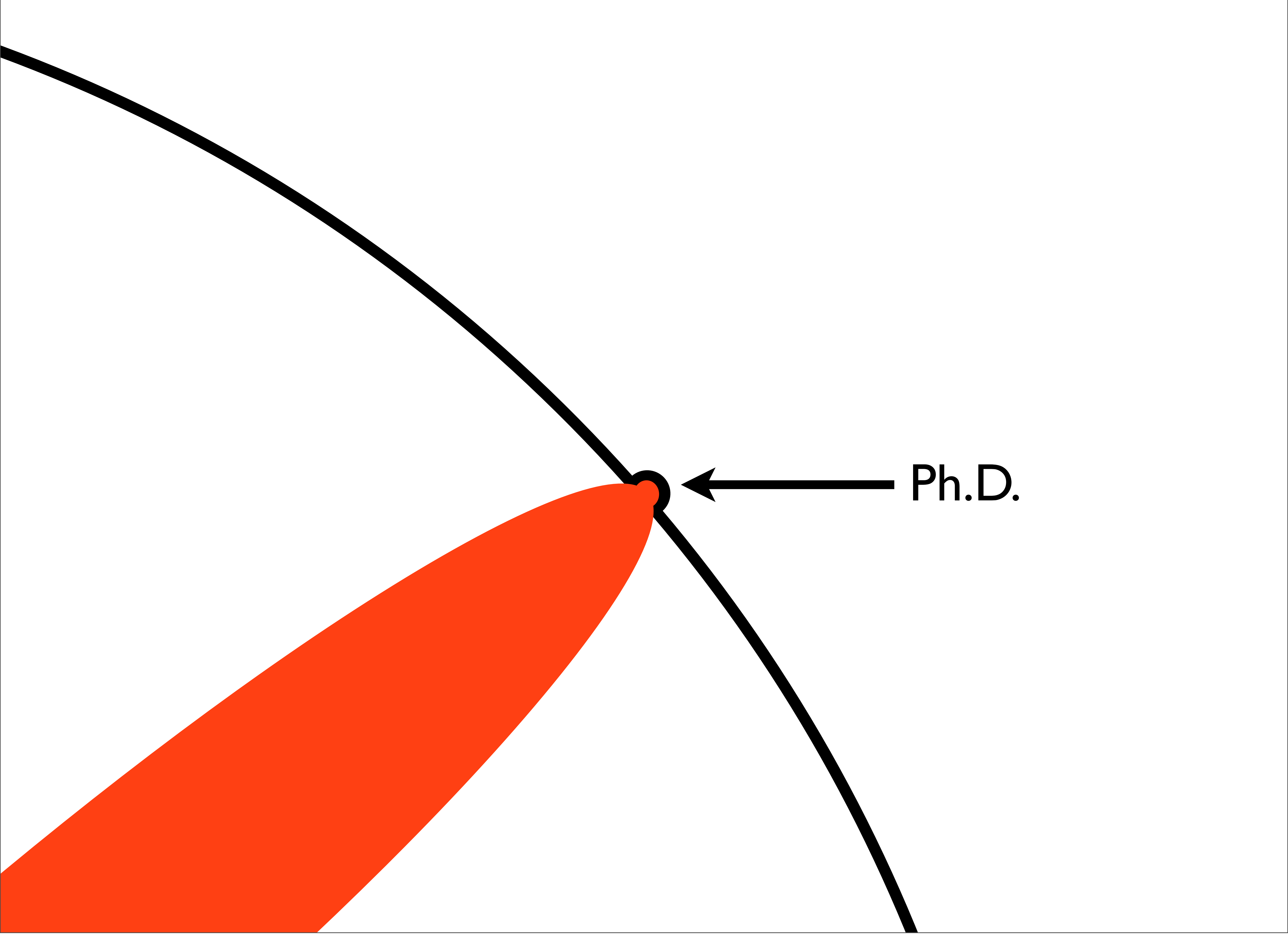


Matt Might, The Illustrated Guide to a Ph.D.: http://matt.might.net/articles/phd-school-in-pictures

ONE SIZE DOES NOT FIT ALL

Streaming

OLAP

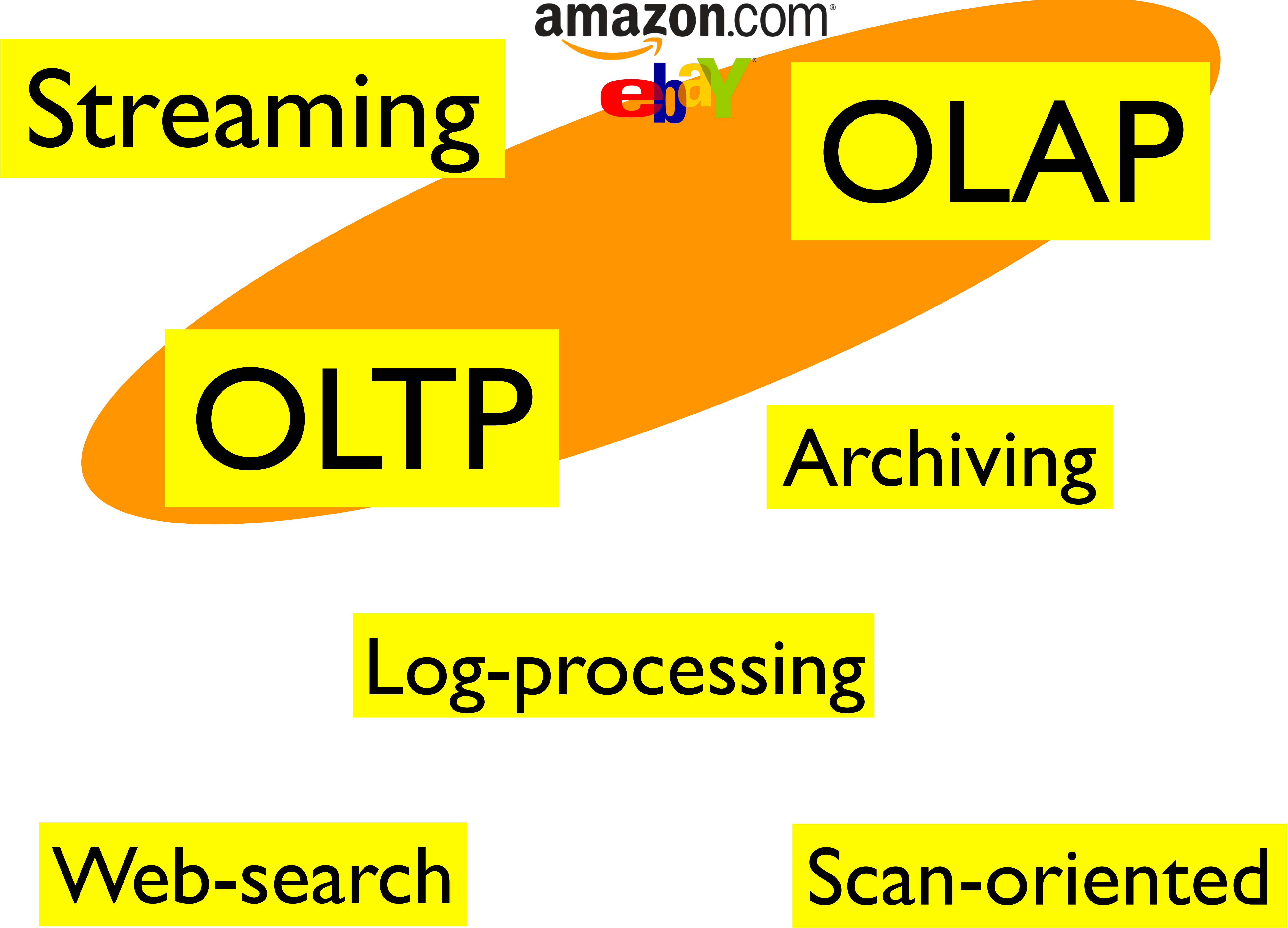
OLTP

Archiving

Log-processing

Web-search

Scan-oriented



amazon.com

ebay

Streaming

OLAP

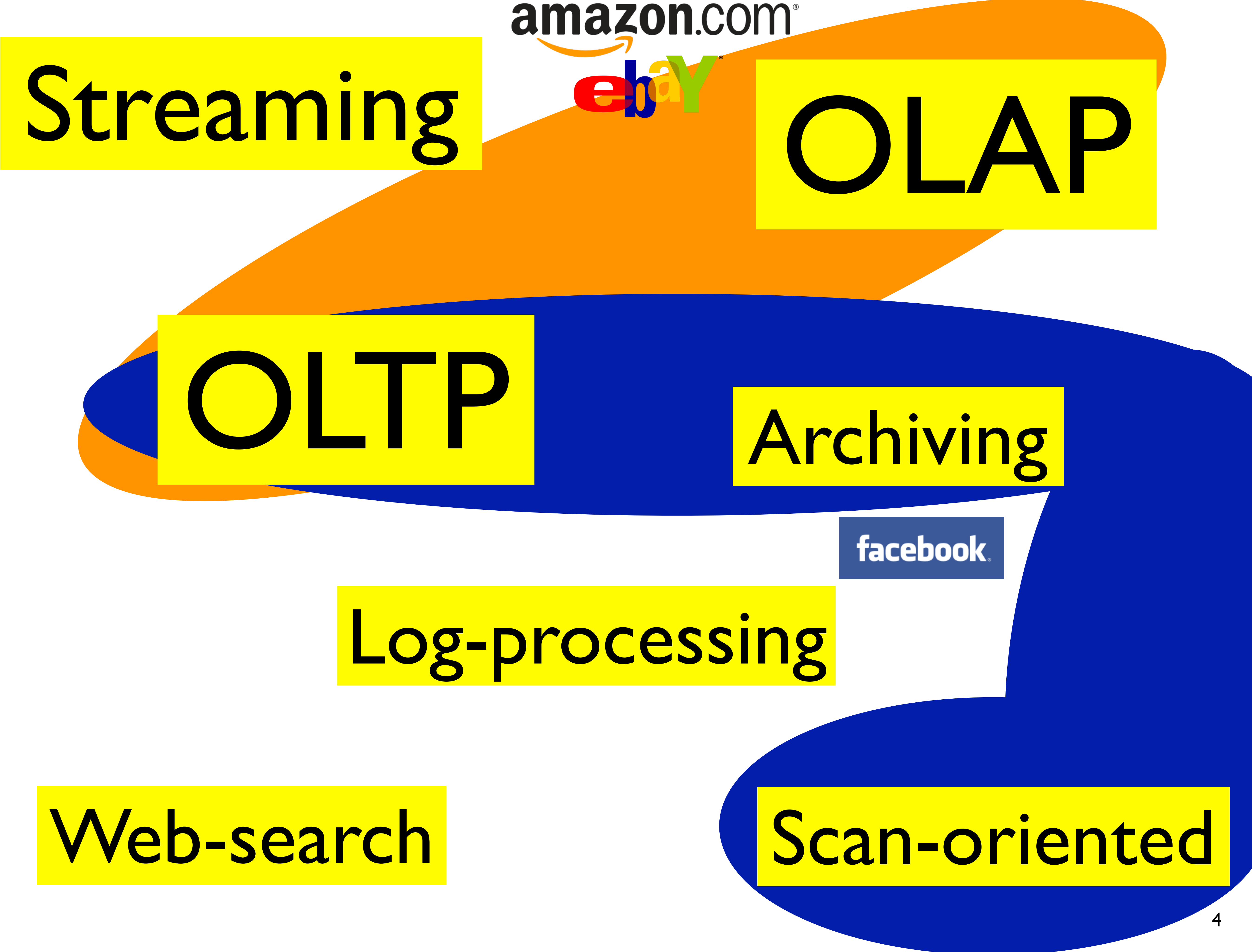
OLTP

Archiving

Log-processing

Web-search

Scan-oriented



Streaming

OLAP

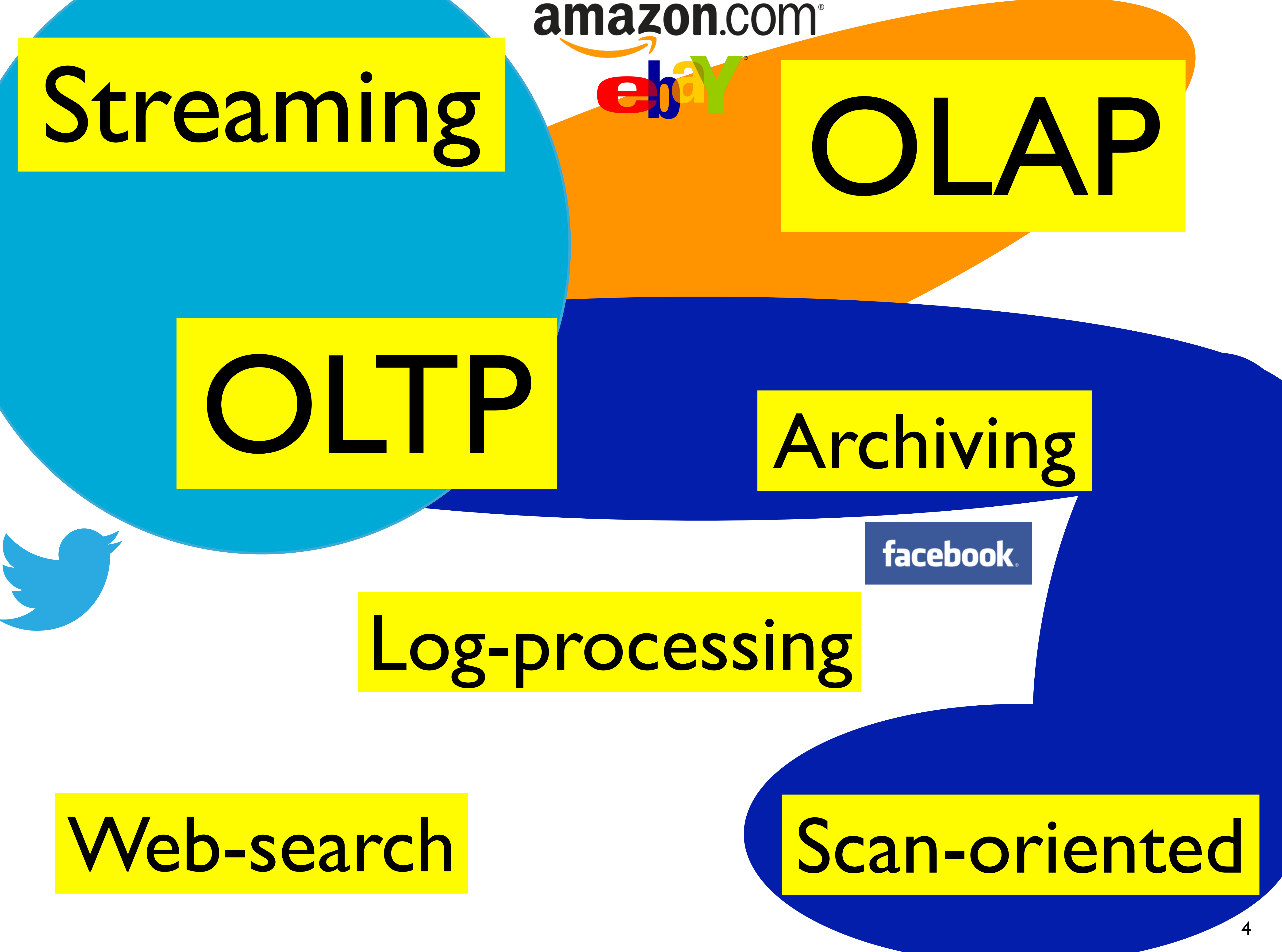
OLTP

Archiving

Log-processing

Web-search

Scan-oriented



Streaming

amazon.com

ebay

OLAP

OLTP

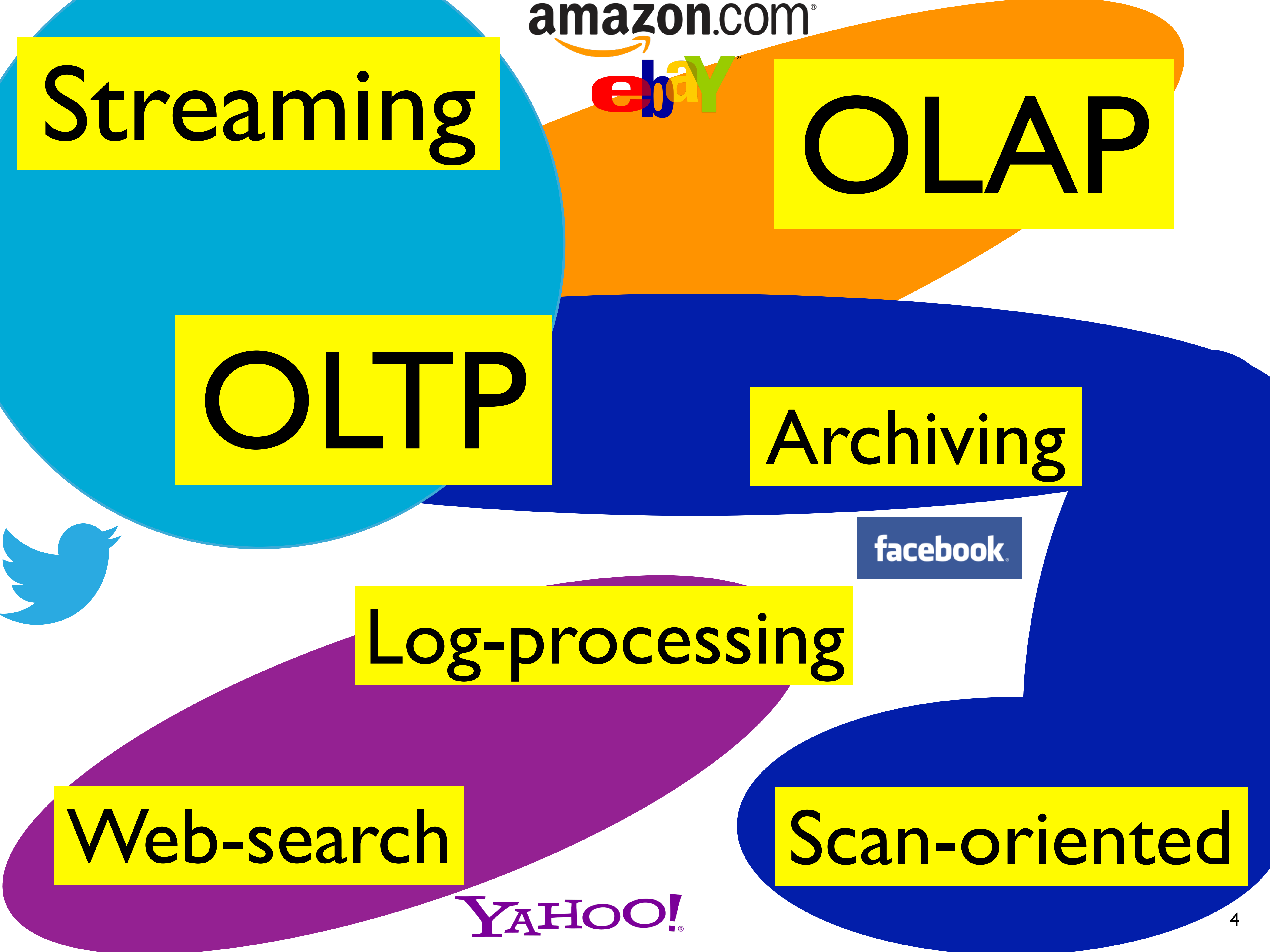
Archiving

facebook

Log-processing

Web-search

Scan-oriented



Streaming

OLAP

OLTP

Archiving

Log-processing

Web-search

Scan-oriented

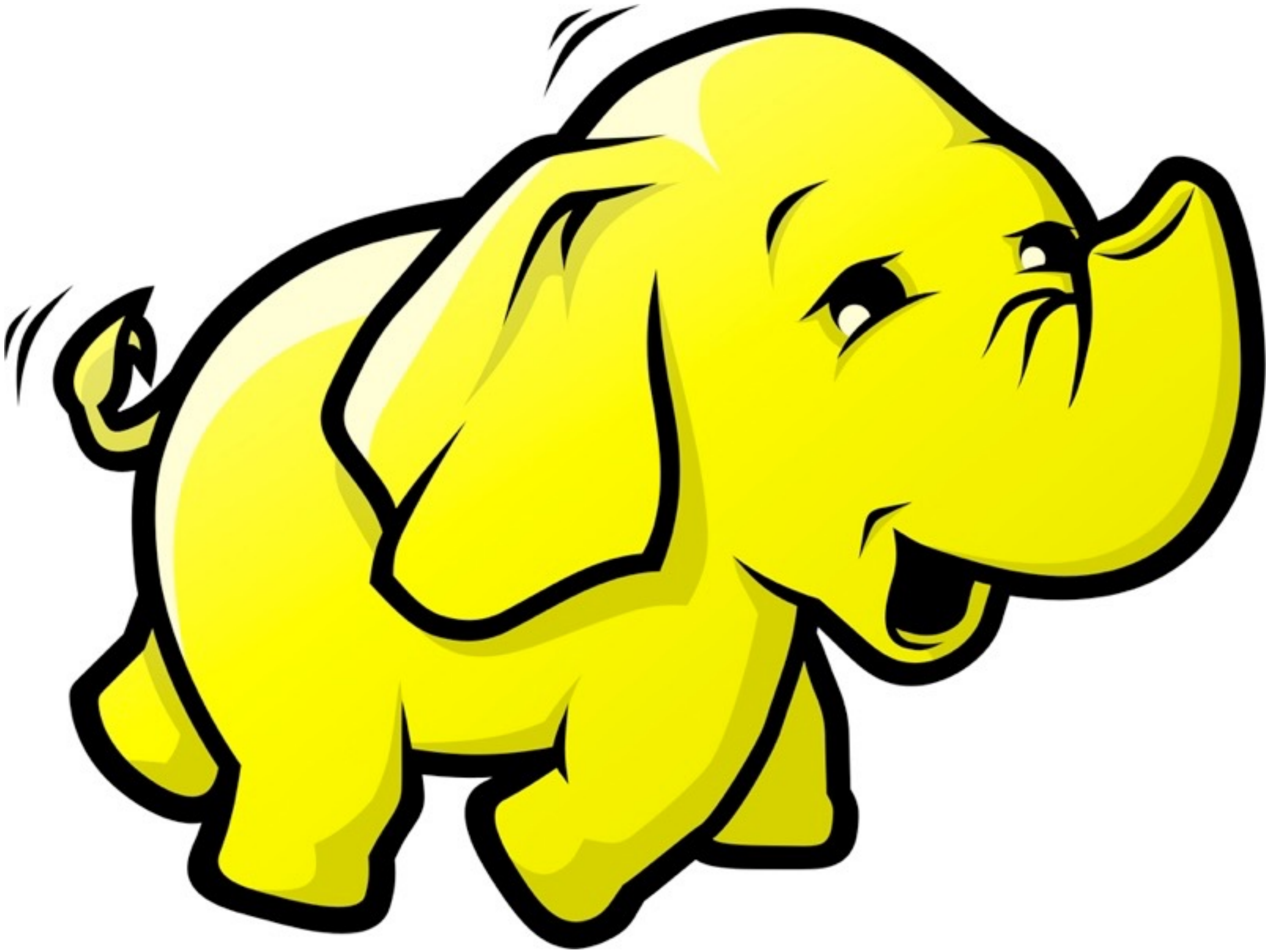
OLTP

OLAP

Archive

Streaming

Log-processing



Streaming

OLAP

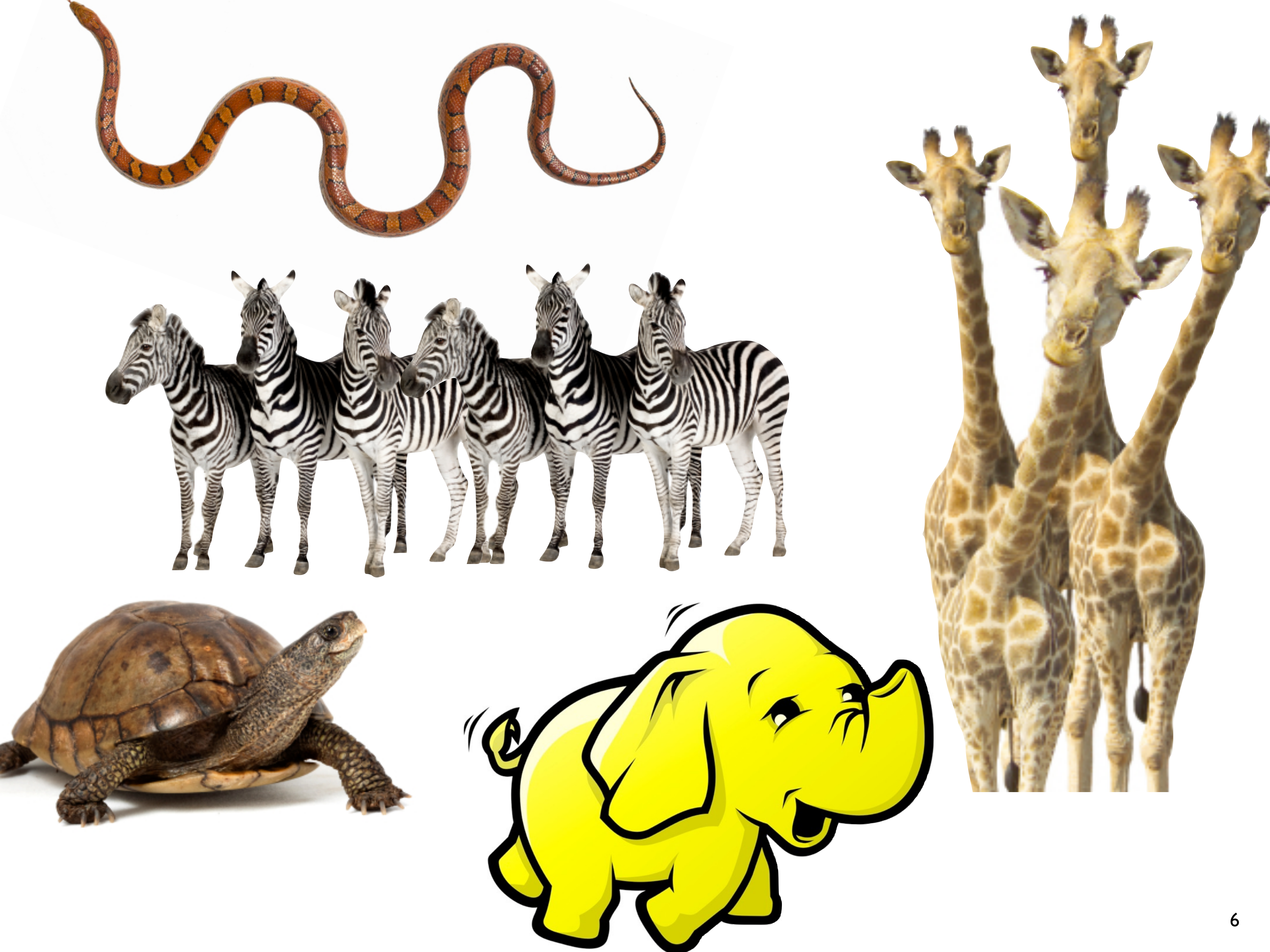
OLTP

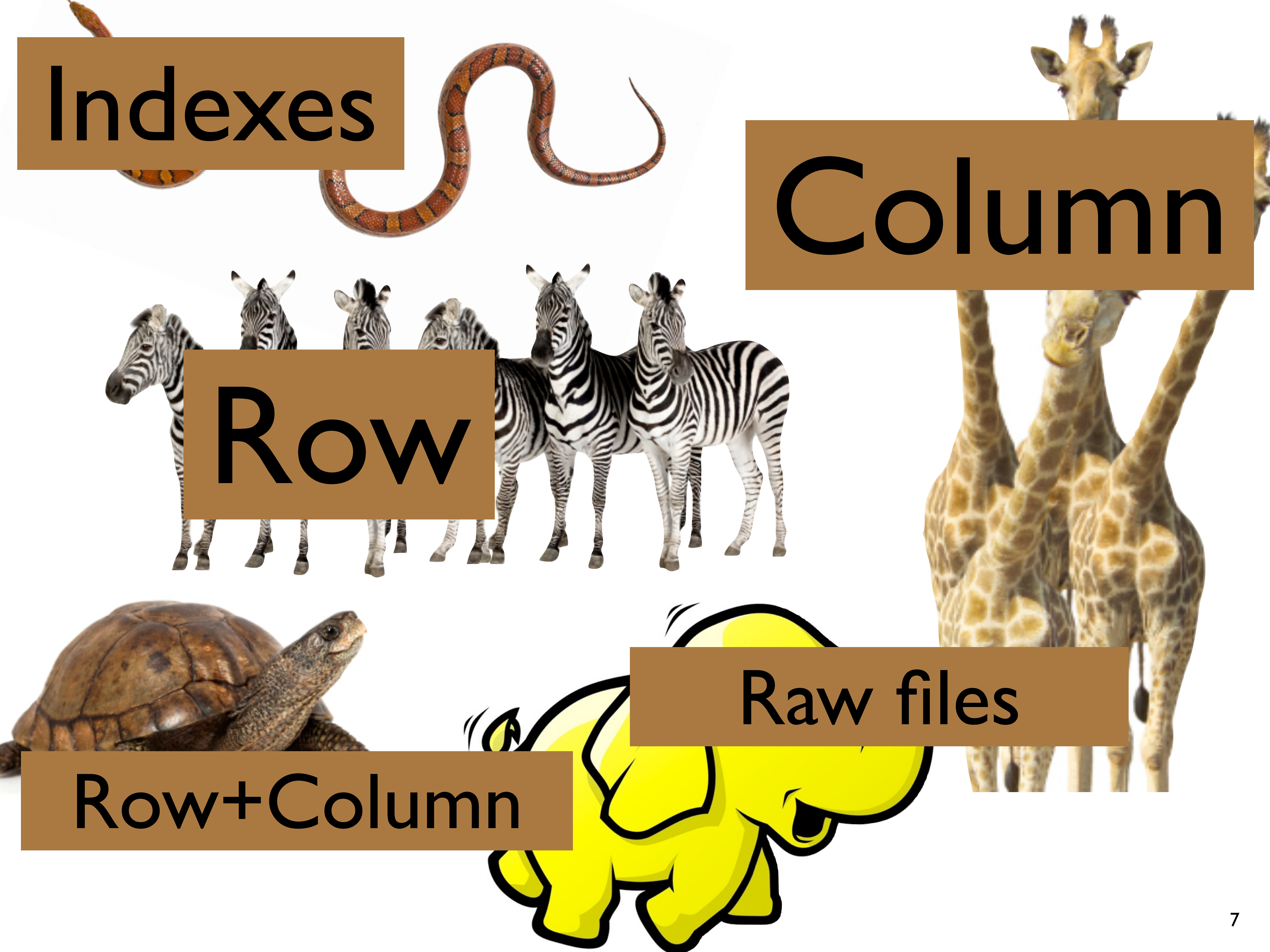
Archiving

Log-processing

Web-search

Scan-oriented





Indexes

Column

Row

Row+Column

Raw files

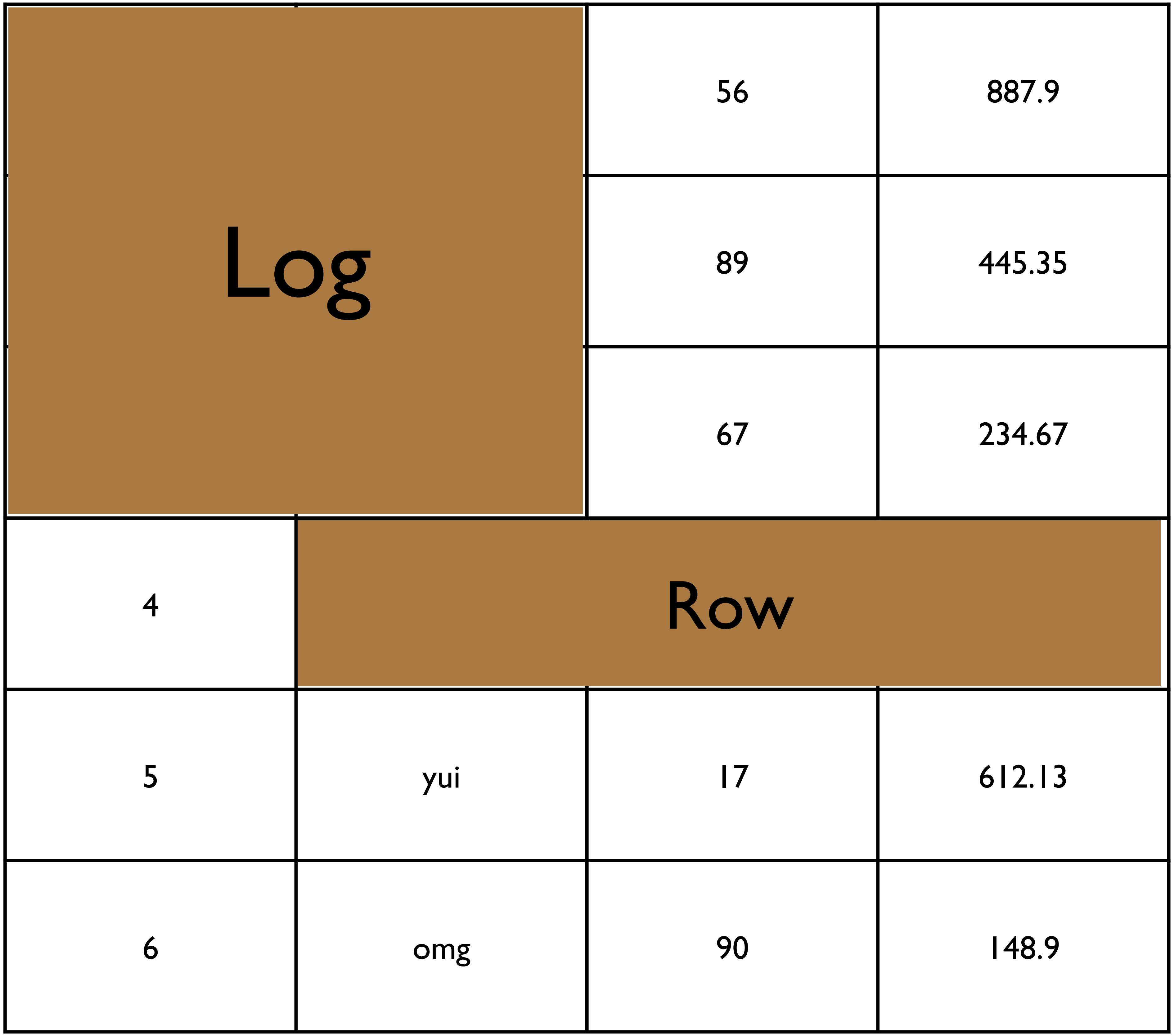
Storage Views

1	abc	56	887.9
2	fdg	89	445.35
3	poe	67	234.67
4	lkj	12	385.92
5	yui	17	612.13
6	omg	90	148.9

Storage Views

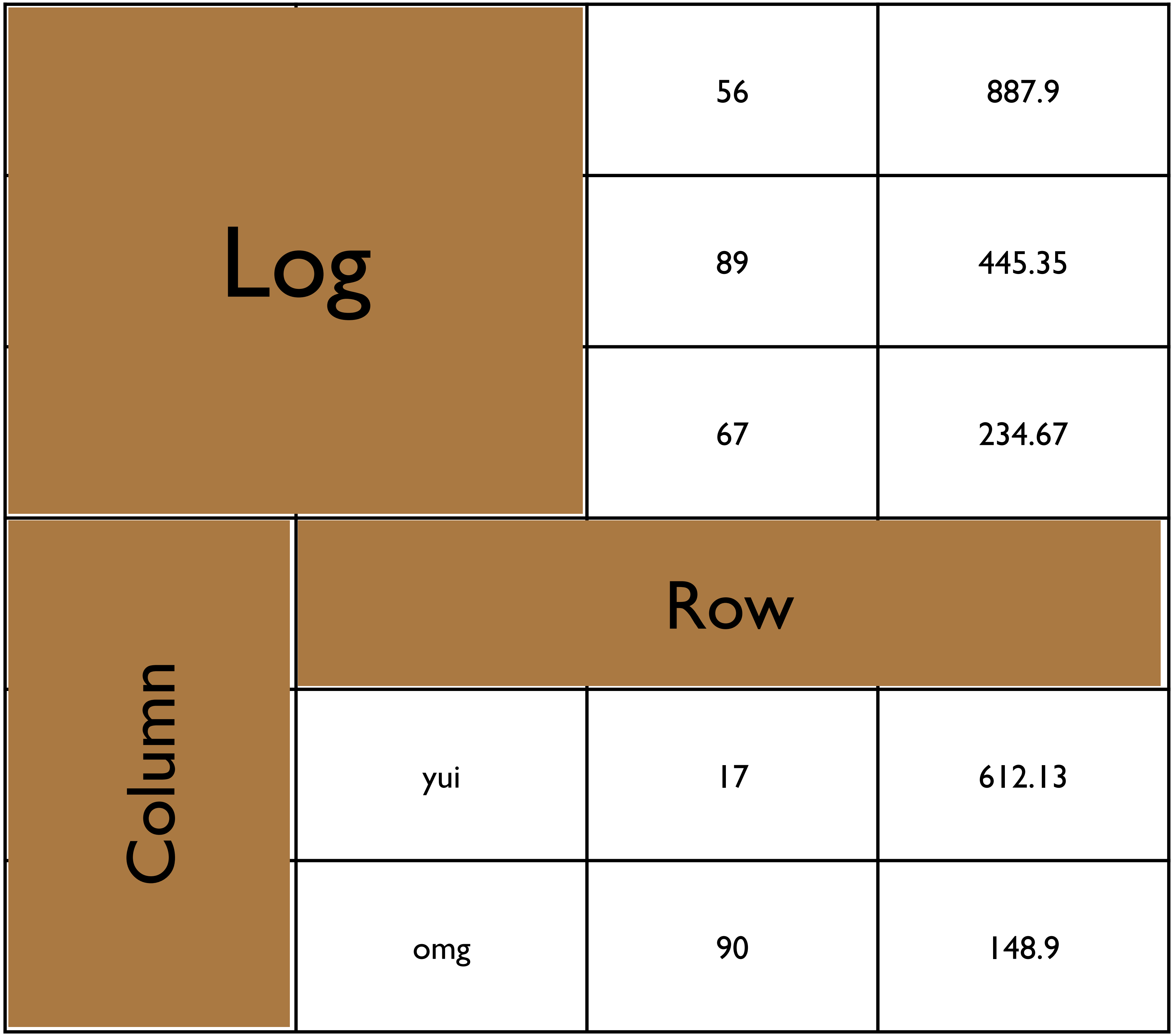
<table>
<tr><td rowspan="3" colspan="2">Log</td><td>56</td><td>887.9</td></tr>
<tr><td>89</td><td>445.35</td></tr>
<tr><td>67</td><td>234.67</td></tr>
<tr><td>4</td><td>lkj</td><td>12</td><td>385.92</td></tr>
<tr><td>5</td><td>yui</td><td>17</td><td>612.13</td></tr>
<tr><td>6</td><td>omg</td><td>90</td><td>148.9</td></tr>
</table>

Storage Views



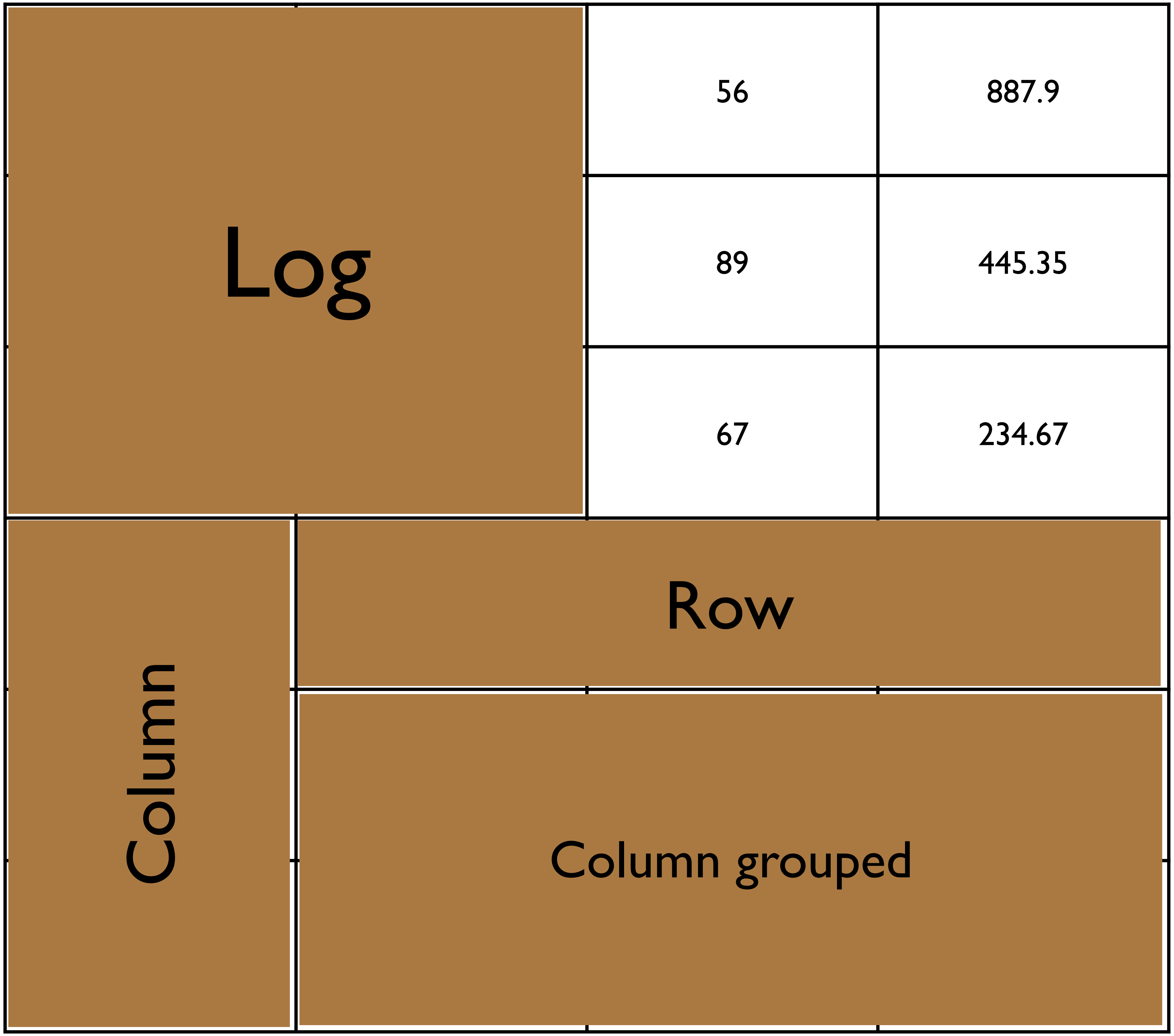
Log		56	887.9
		89	445.35
		67	234.67
4	Row		
5	yui	17	612.13
6	omg	90	148.9

Storage Views



Log		56	887.9
		89	445.35
		67	234.67
Column	Row		
	yui	17	612.13
	omg	90	148.9

Storage Views



Log

56

887.9

89

445.35

67

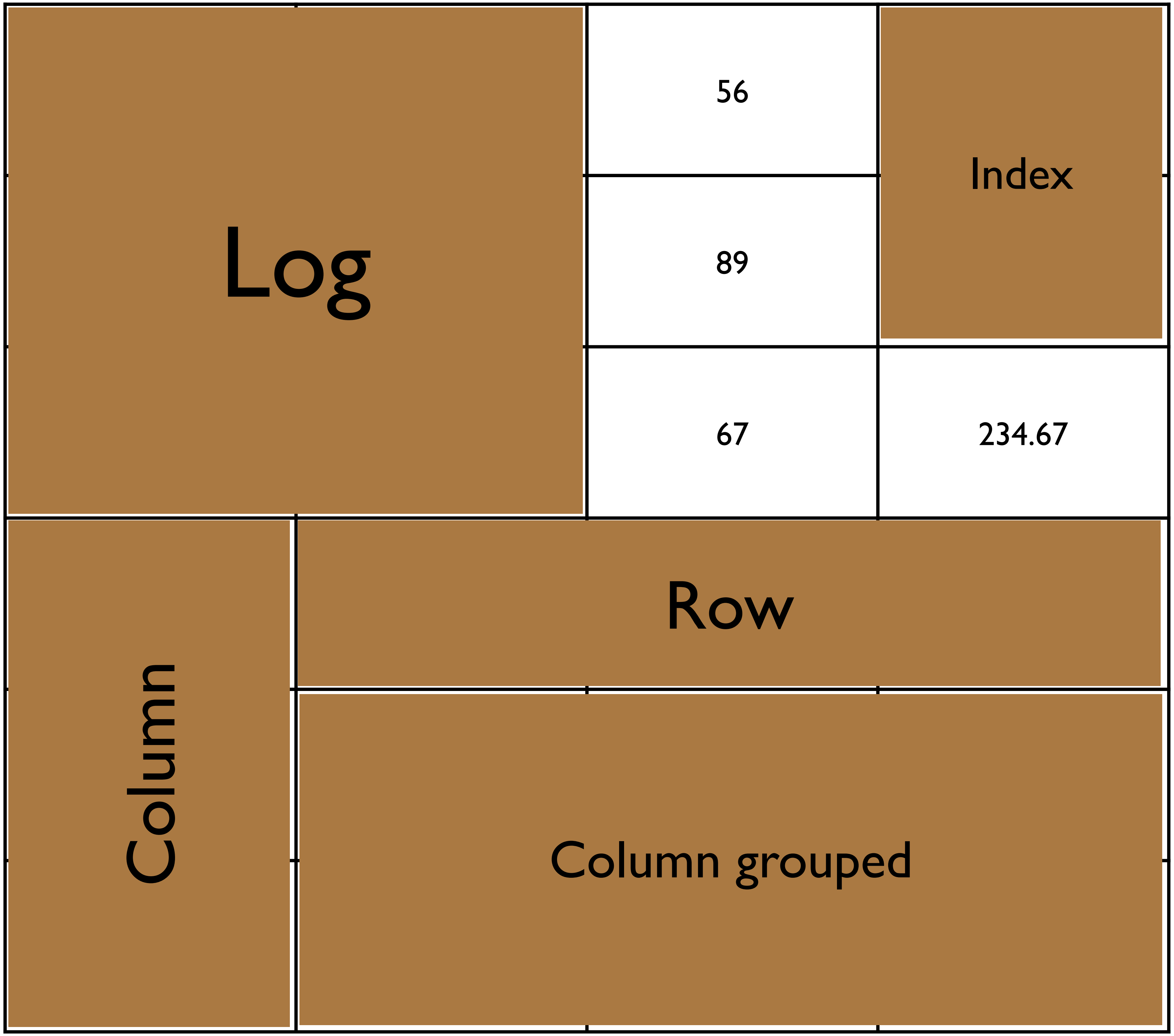
234.67

Column

Row

Column grouped

Storage Views



Log

56

89

67

Index

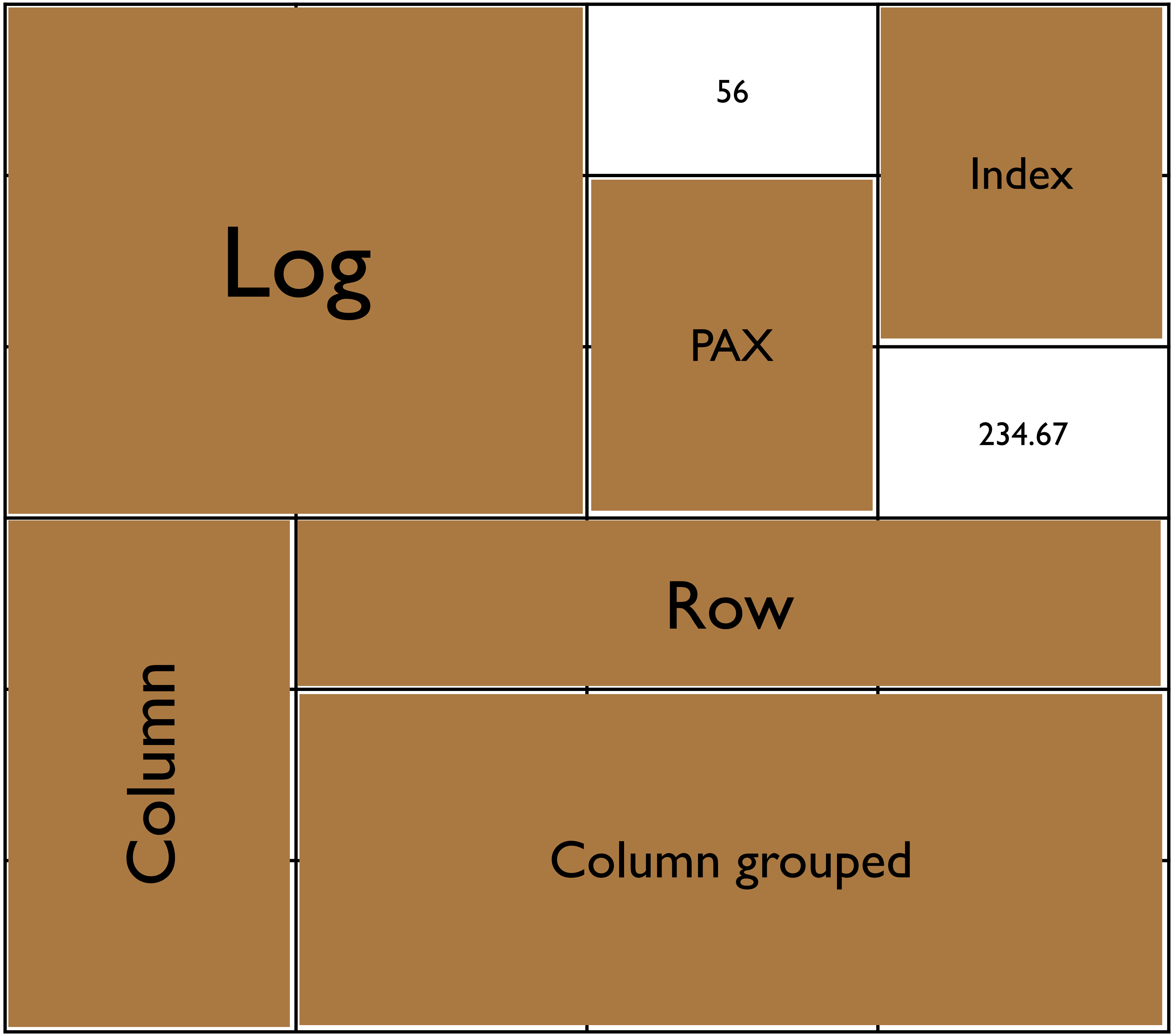
234.67

Column

Row

Column grouped

Storage Views



Log

56

Index

PAX

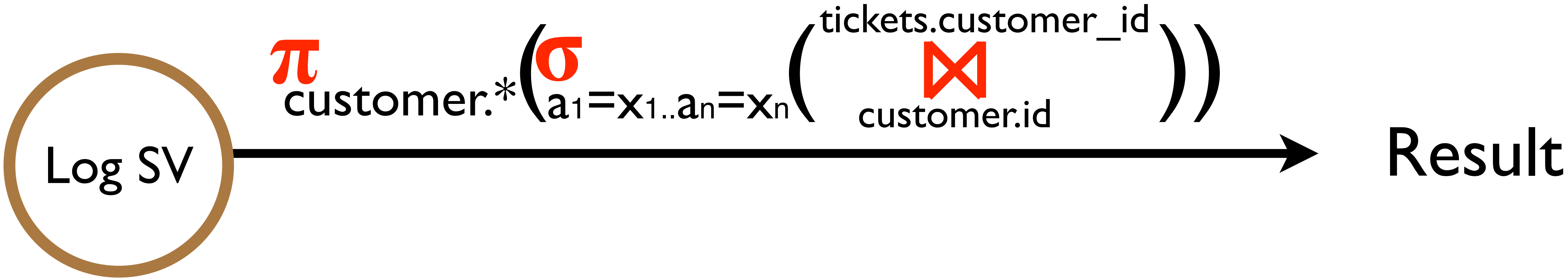
234.67

Column

Row

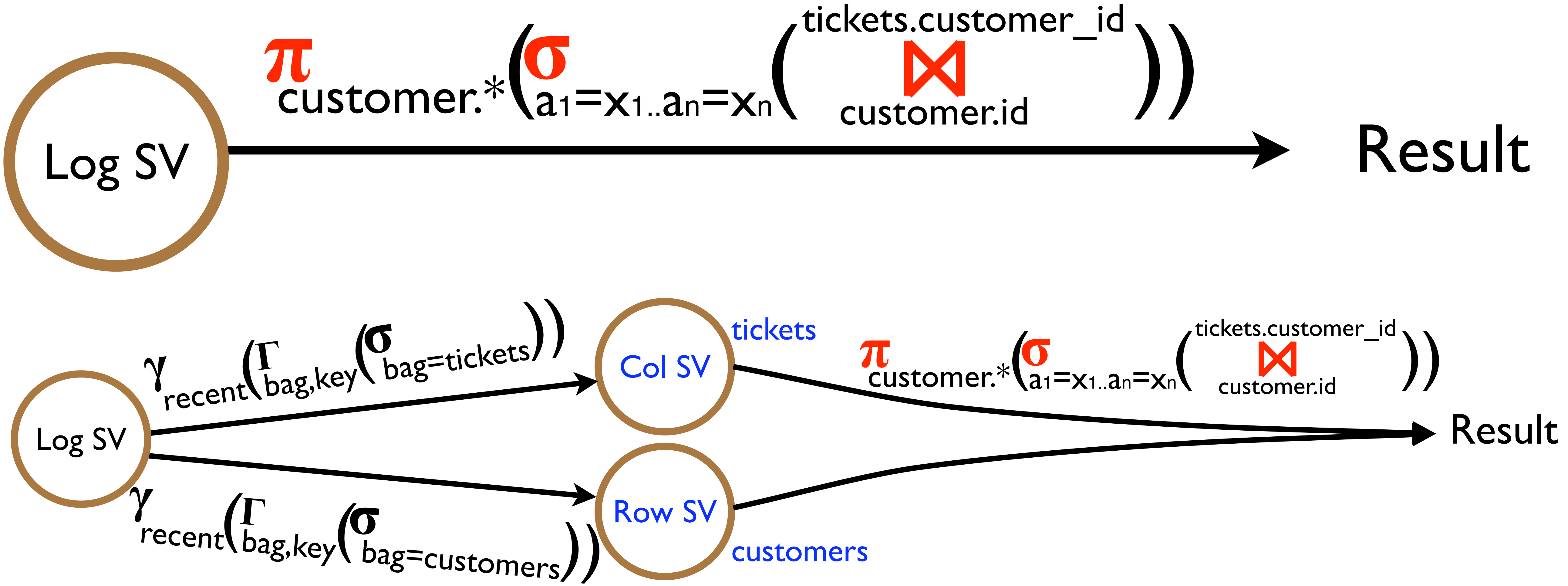
Column grouped

Example: Flight Tickets



$$\pi_{customer.*}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{tickets.customer\_id}{\underset{customer.id}{\bowtie}}\right)\right)$$

Example: Flight Tickets



Log SV

$$\pi_{\text{customer.*}}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}\right)\right)$$

Result

Log SV

$$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=tickets}}\right)\right)$$

$$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=customers}}\right)\right)$$

Col SV

tickets

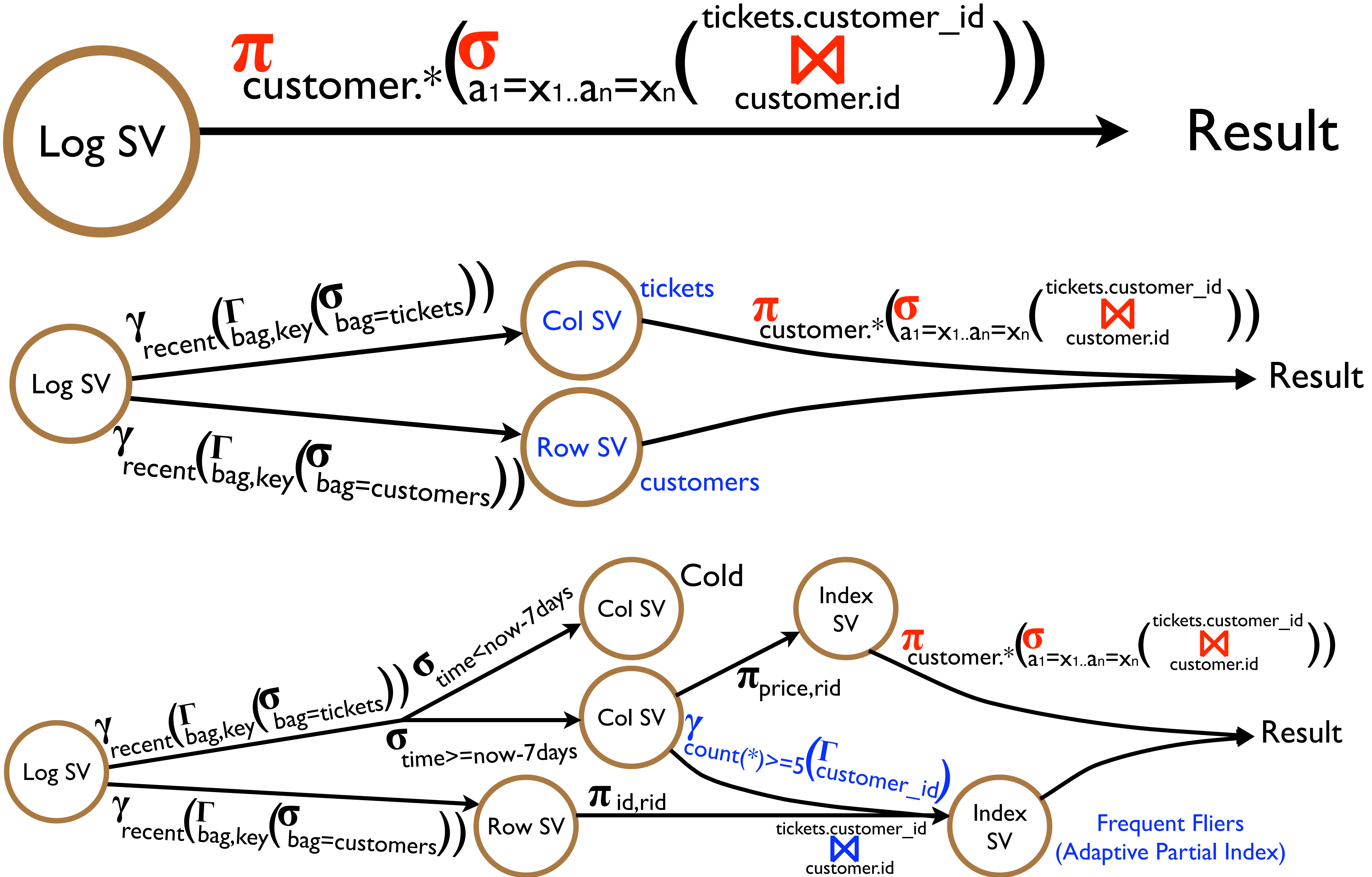
Row SV

customers

$$\pi_{\text{customer.*}}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}\right)\right)$$

Result

Example: Flight Tickets



Log SV

$\pi_{\text{customer.*}}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}\right)\right)$

Result

Log SV

$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=tickets}}\right)\right)$

Col SV

tickets

$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=customers}}\right)\right)$

Row SV

customers

$\pi_{\text{customer.*}}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}\right)\right)$

Result

Log SV

$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=tickets}}\right)\right)$

$\sigma_{\text{time<now-7days}}$

Col SV

Cold

$\sigma_{\text{time>=now-7days}}$

Col SV

$\pi_{\text{price,rid}}$

Index SV

$\pi_{\text{customer.*}}\left(\sigma_{a_1=x_1..a_n=x_n}\left(\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}\right)\right)$

Result

$\gamma_{\text{count(*)>=5}}\left(\Gamma_{\text{customer\_id}}\right)$

$\gamma_{\text{recent}}\left(\Gamma_{\text{bag,key}}\left(\sigma_{\text{bag=customers}}\right)\right)$

Row SV

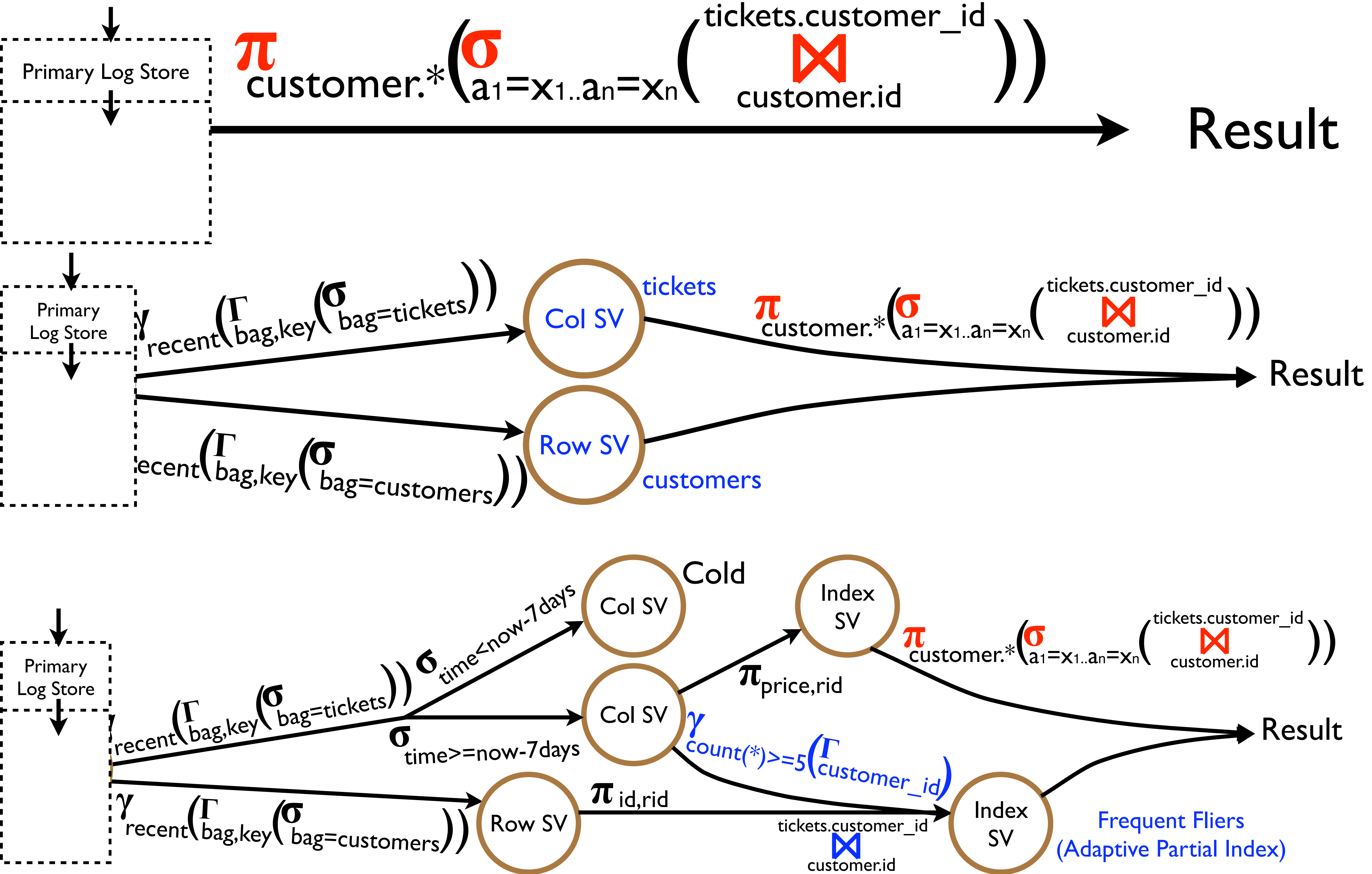
$\pi_{\text{id,rid}}$

$\overset{\text{tickets.customer\_id}}{\underset{\text{customer.id}}{\bowtie}}$

Index SV

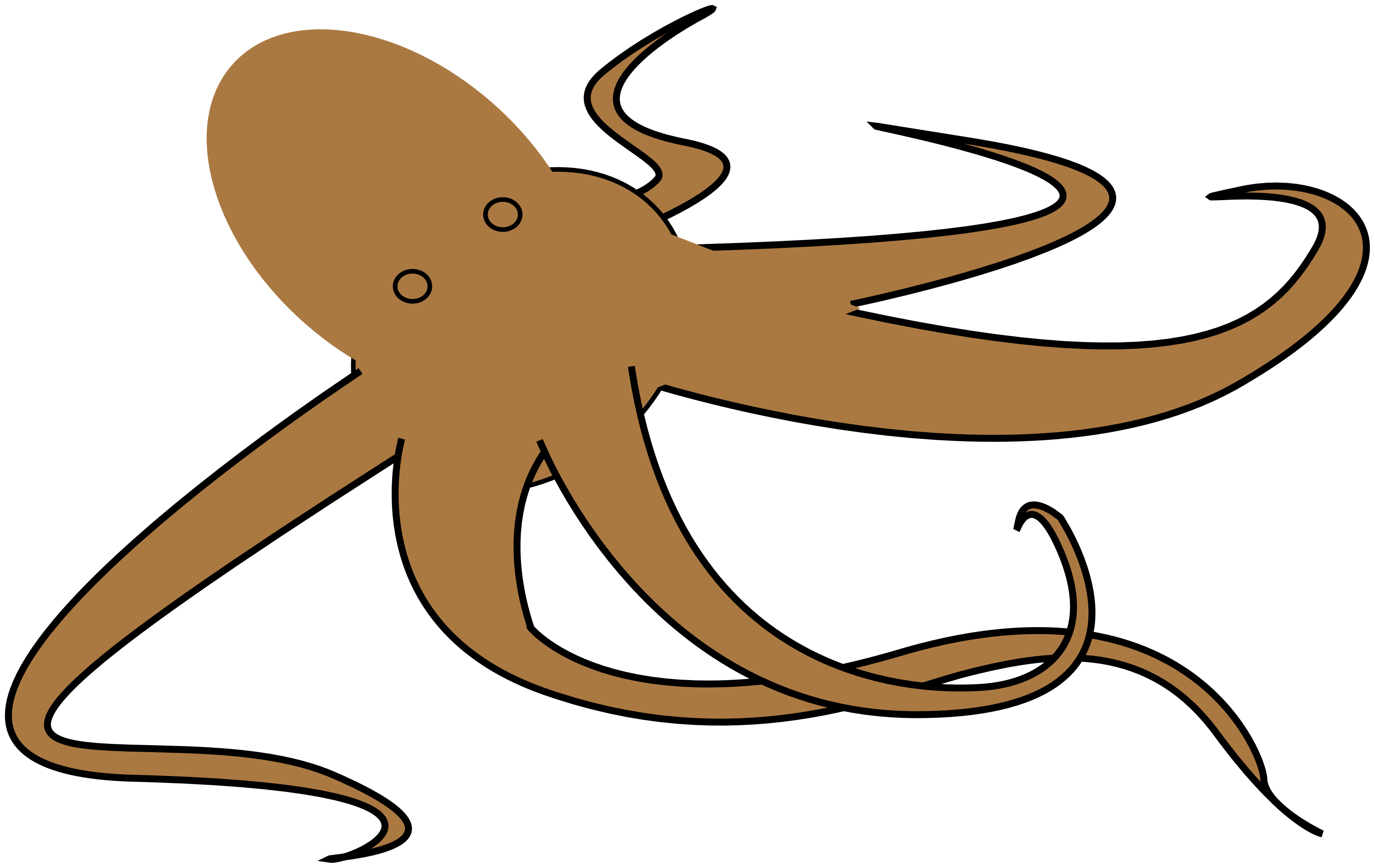
Frequent Fliers
(Adaptive Partial Index)

Example: Flight Tickets



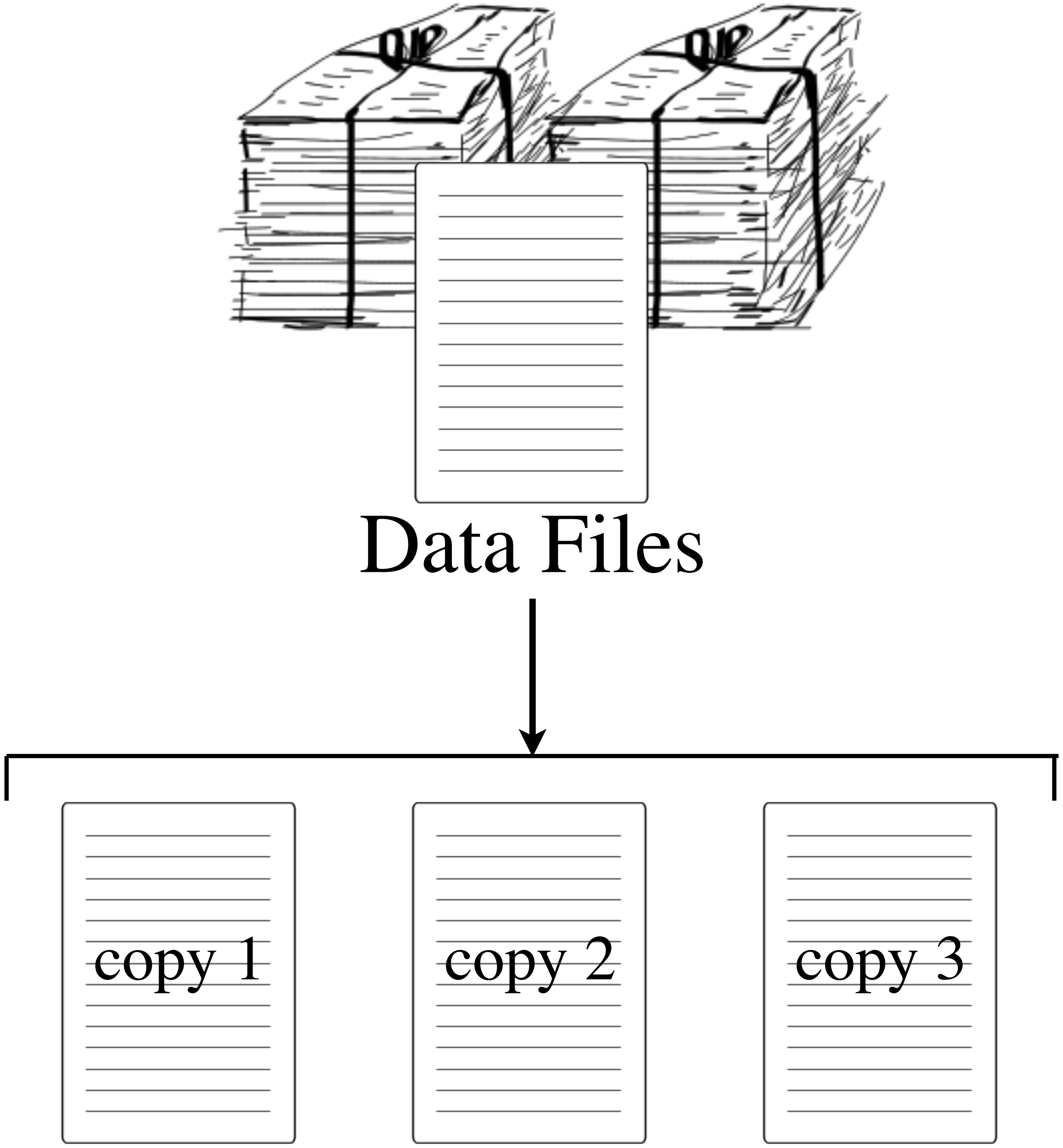
Primary Log Store
$\pi_{customer.*}(\sigma_{a_1=x_1..a_n=x_n}(\underset{customer.id}{\overset{tickets.customer\_id}{\bowtie}}))$
Result
Primary Log Store
$\gamma_{recent}(\Gamma_{bag,key}(\sigma_{bag=tickets}))$
Col SV
tickets
$\pi_{customer.*}(\sigma_{a_1=x_1..a_n=x_n}(\underset{customer.id}{\overset{tickets.customer\_id}{\bowtie}}))$
Result
$ecent(\Gamma_{bag,key}(\sigma_{bag=customers}))$
Row SV
customers
Primary Log Store
$recent(\Gamma_{bag,key}(\sigma_{bag=tickets}))$
$\sigma_{time<now-7days}$
Col SV
Cold
Index SV
$\pi_{customer.*}(\sigma_{a_1=x_1..a_n=x_n}(\underset{customer.id}{\overset{tickets.customer\_id}{\bowtie}}))$
$\sigma_{time>=now-7days}$
Col SV
$\pi_{price,rid}$
$\gamma_{count(*)>=5}(\Gamma_{customer\_id})$
Result
$\gamma_{recent}(\Gamma_{bag,key}(\sigma_{bag=customers}))$
Row SV
$\pi_{id,rid}$
$\underset{customer.id}{\overset{tickets.customer\_id}{\bowtie}}$
Index SV
Frequent Fliers
(Adaptive Partial Index)

WTF!
Where’s The Food!



Rodent Store

What to store?



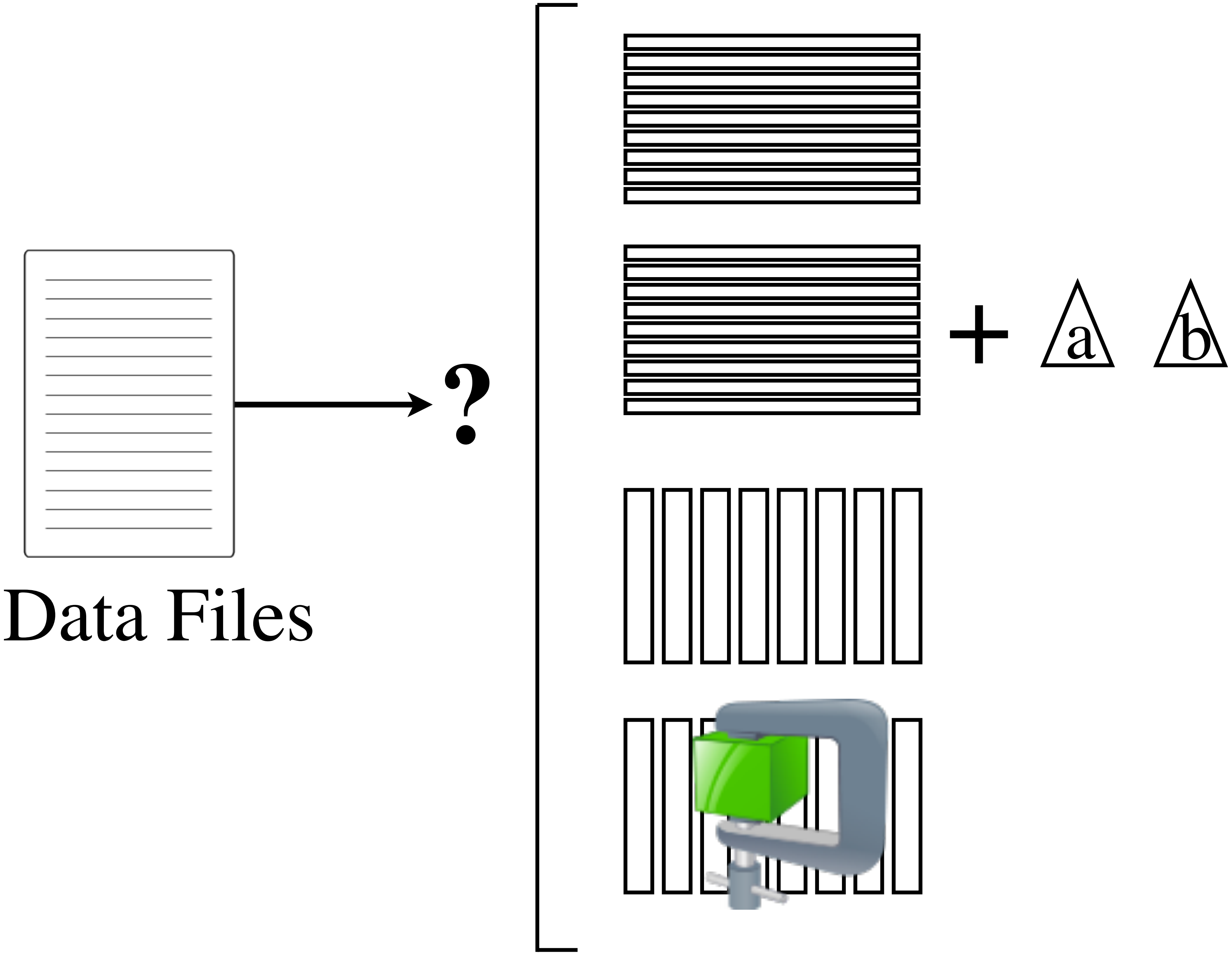
Data Files

copy 1

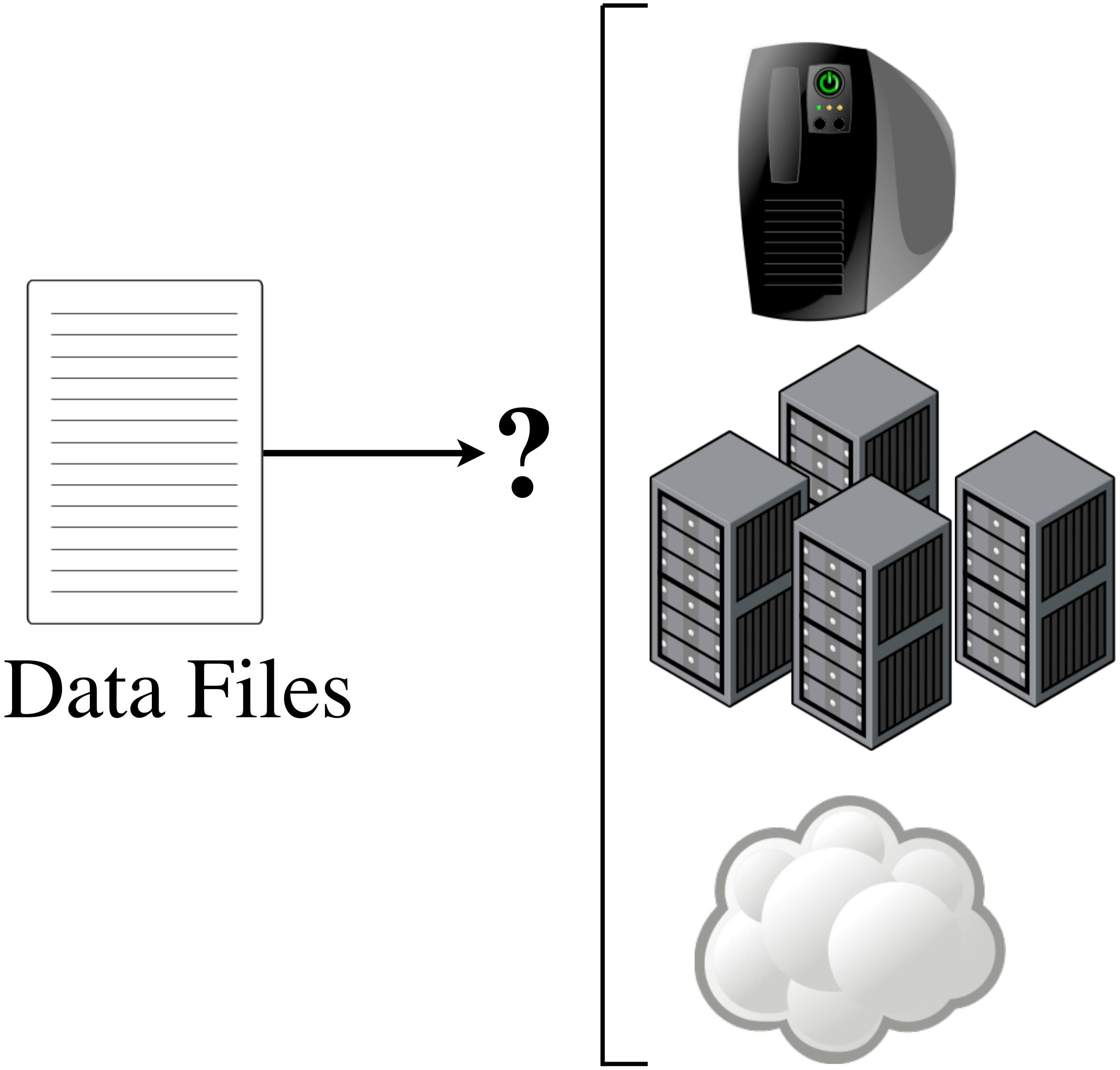
copy 2

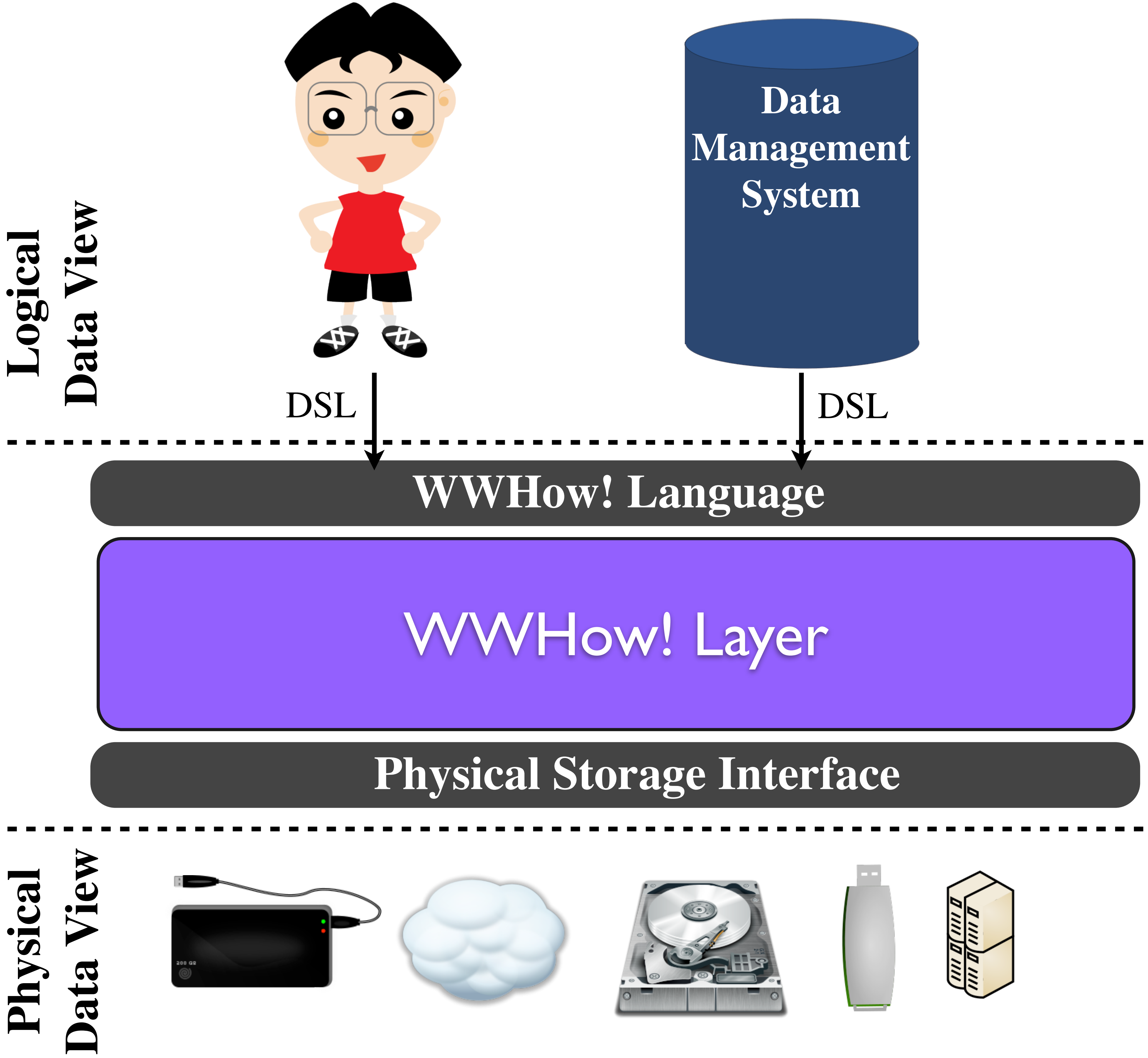
copy 3

How to store?



Where to store?





Logical Data View

Data Management System

DSL

DSL

WWHow! Language

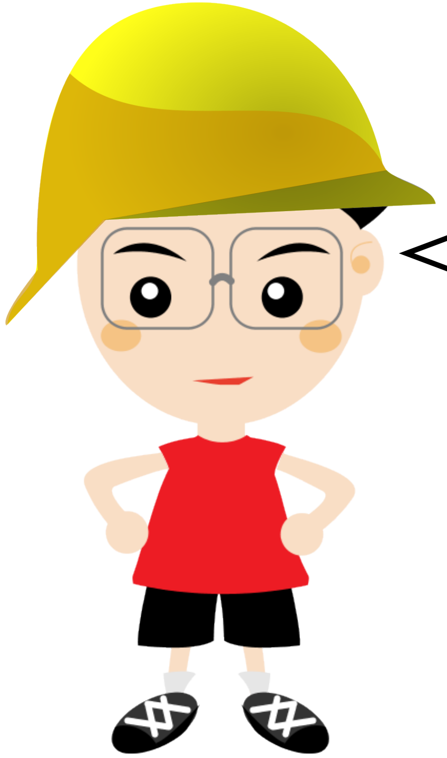
WWHow! Layer

Physical Storage Interface

Physical Data View

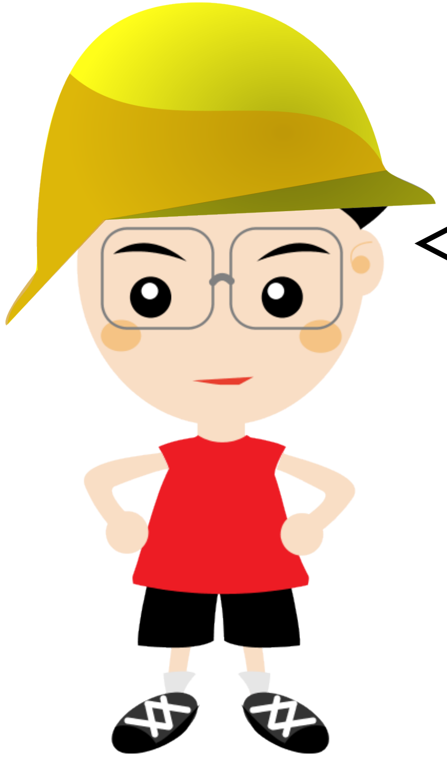
Example Use-cases

- WWHow! File System
- WWHow! RAID
- WWHow! Relational DBMS
- WWHow! Cloud



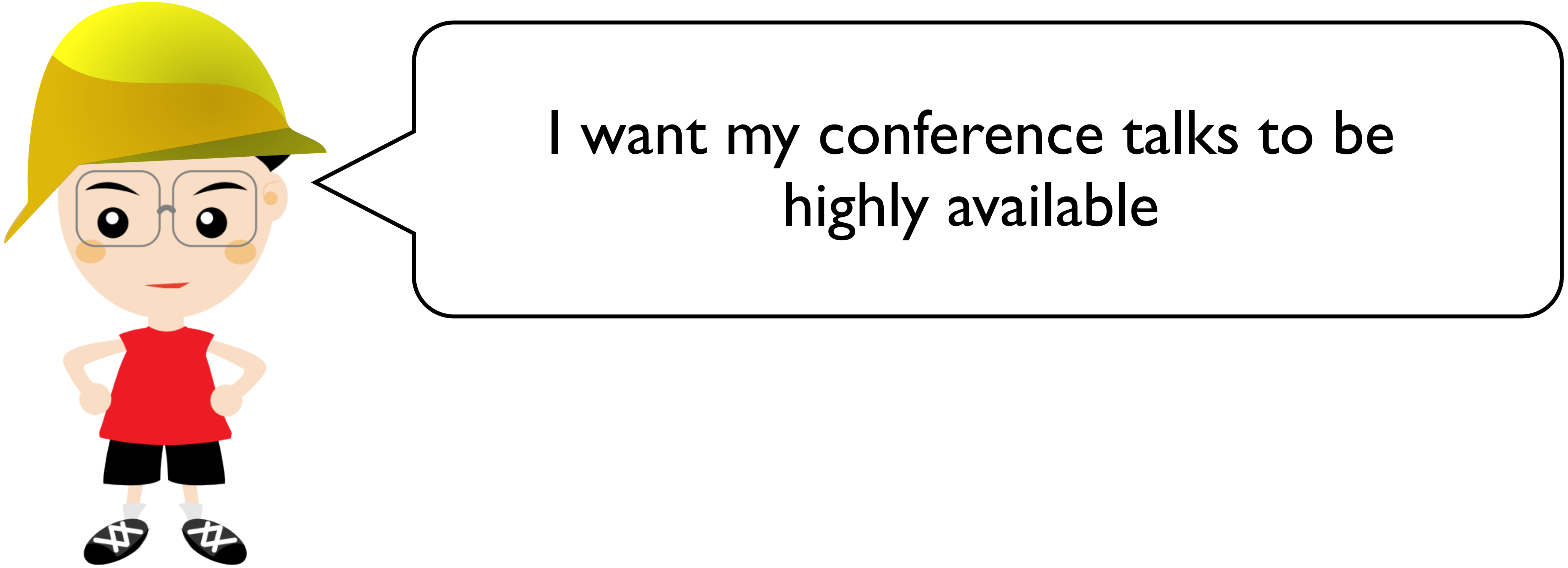
Store my conferences talks (PDFs 2x and PPTs 1x) using RSA compression on University server

```
STORE '/Users/Bob/Conferences/Talks/*.*'
WHAT *.(pdf | ppt), *.pdf
WHERE vise4
HOW encryption(rsa) FOR *;
```



I want my conference talks to be highly available

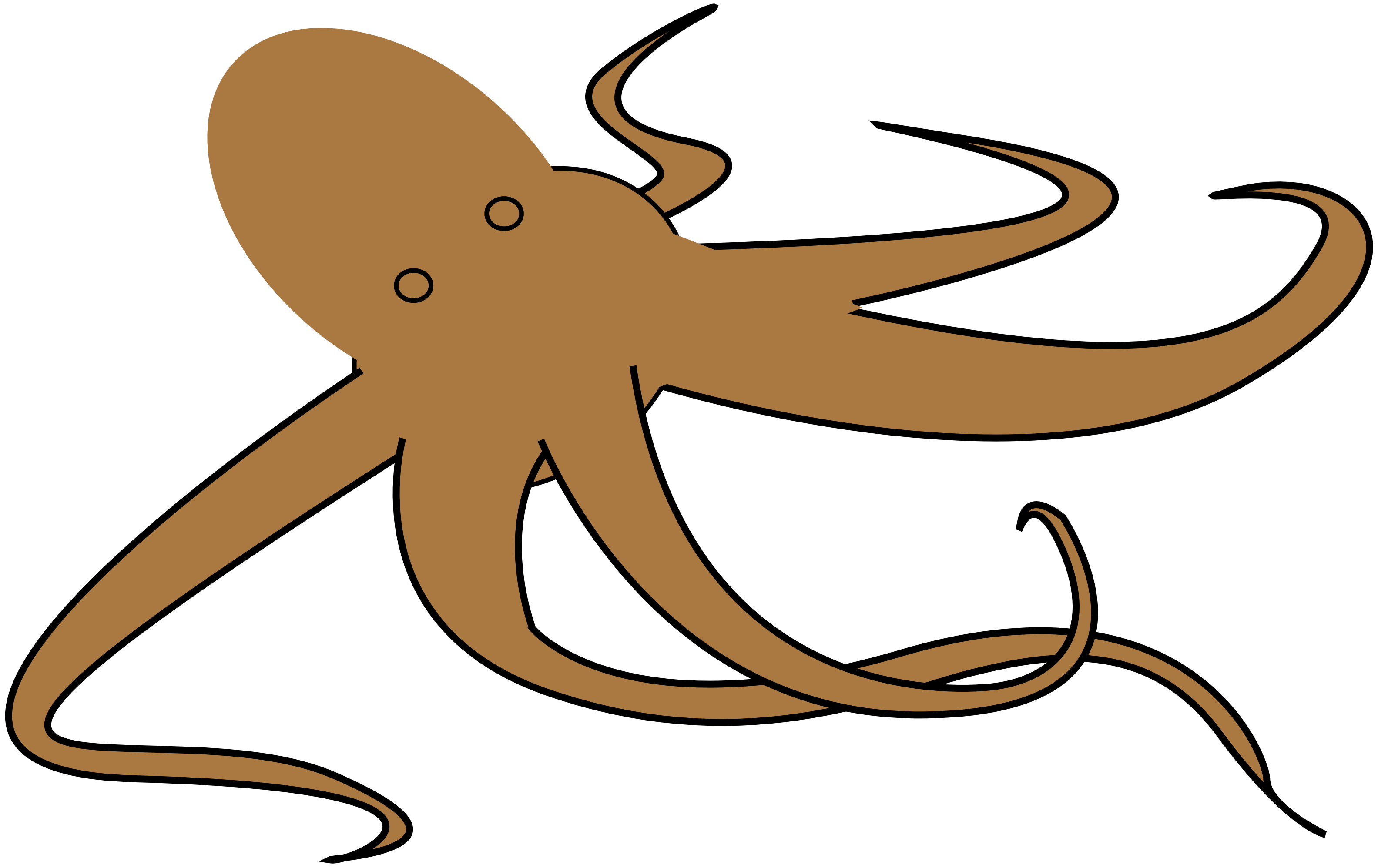
```
STORE ‘/Users/Bob/Conferences/Talks/*.*’
WHAT *.(pdf | ppt), *.pdf
HOW encryption(rsa) FOR *
PREFERENCE Availability=‘high’;
```



```
STORE '/Users/Bob/Conferences/Talks/*.*'
WHAT *.(pdf | ppt), *.pdf
HOW encryption(rsa) FOR *
PREFERENCE Availability='high';
```

job for the
WWhow! data storage optimizer

OctopusDB

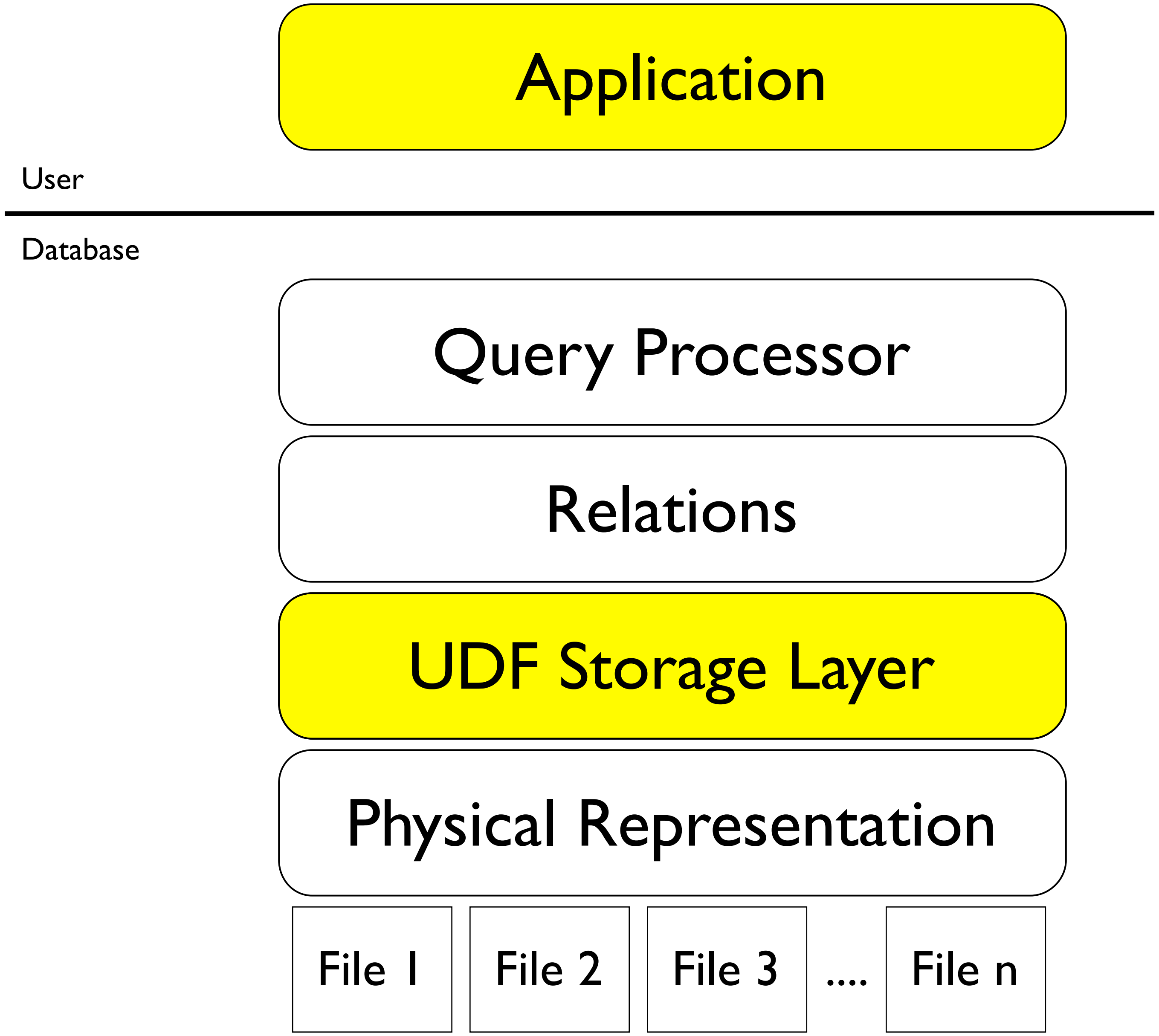


- Cool Vision
- Tough to realize

C-Store

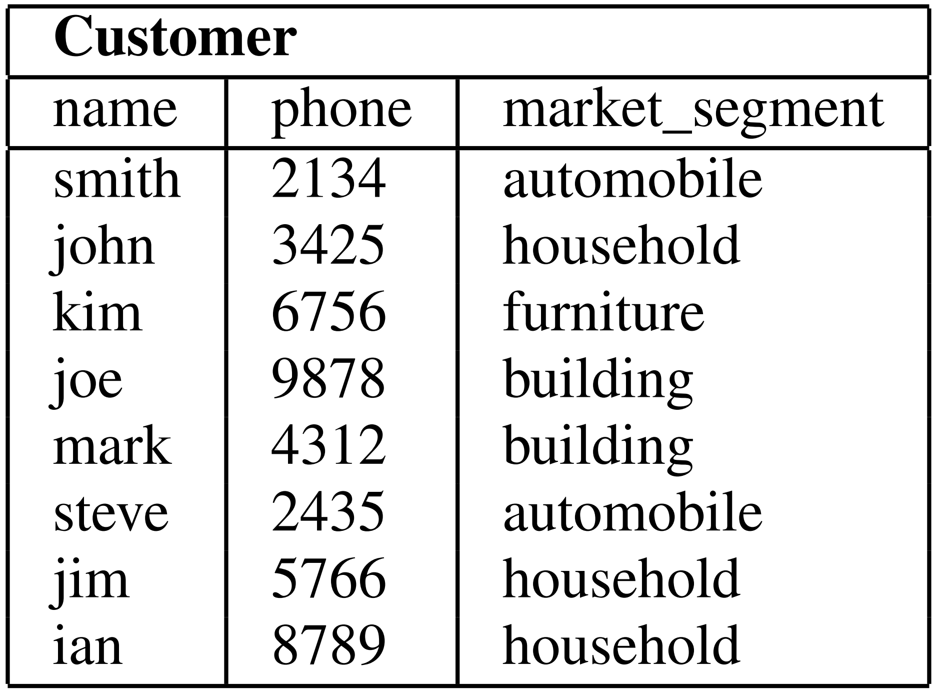
?

Trojan Columns



Trojan Columns

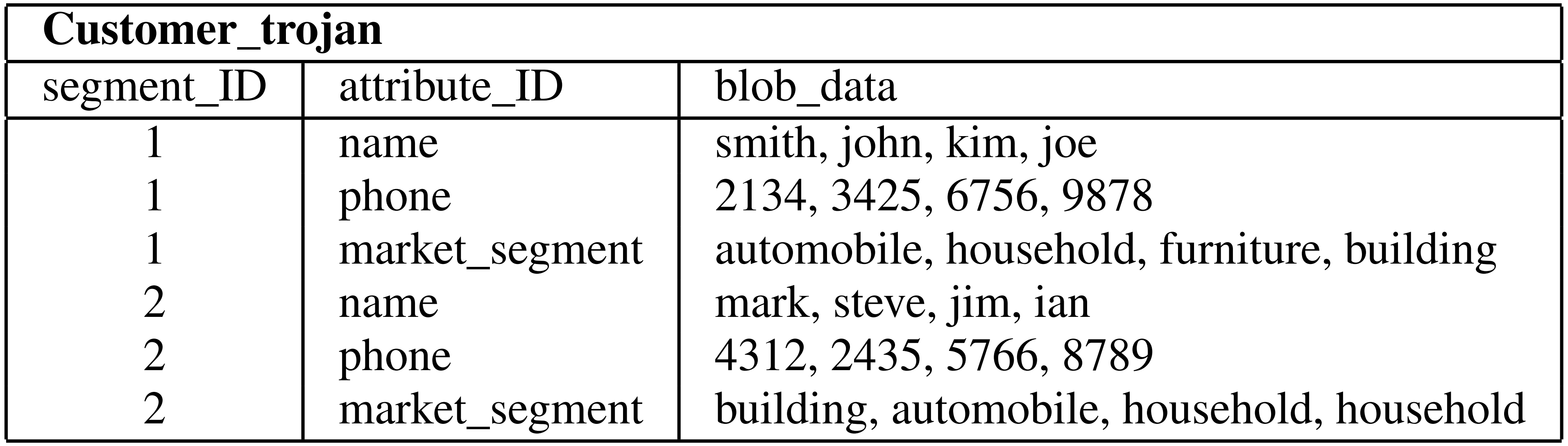
Relation



Customer

name	phone	market_segment
smith	2134	automobile
john	3425	household
kim	6756	furniture
joe	9878	building
mark	4312	building
steve	2435	automobile
jim	5766	household
ian	8789	household

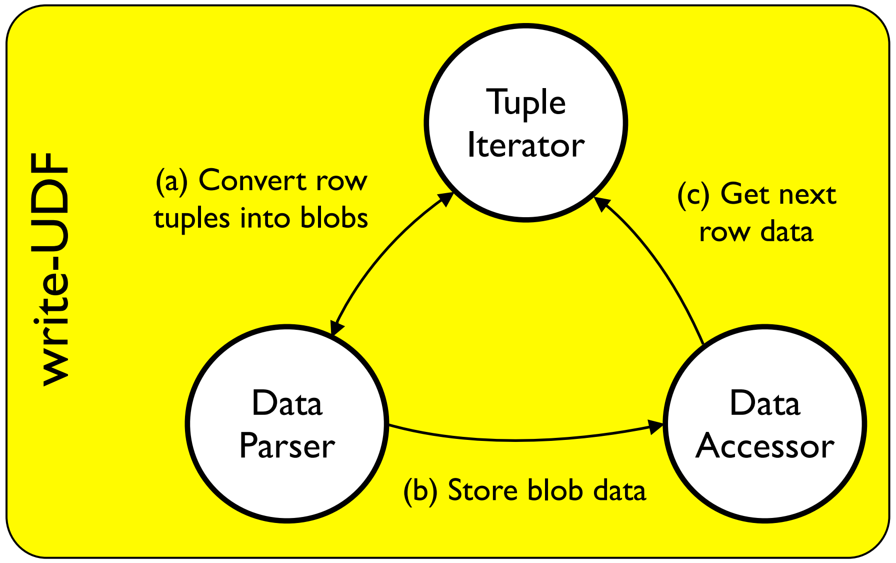
Physical Table



Customer_trojan

segment_ID	attribute_ID	blob_data
1	name	smith, john, kim, joe
1	phone	2134, 3425, 6756, 9878
1	market_segment	automobile, household, furniture, building
2	name	mark, steve, jim, ian
2	phone	4312, 2435, 5766, 8789
2	market_segment	building, automobile, household, household

Trojan Columns



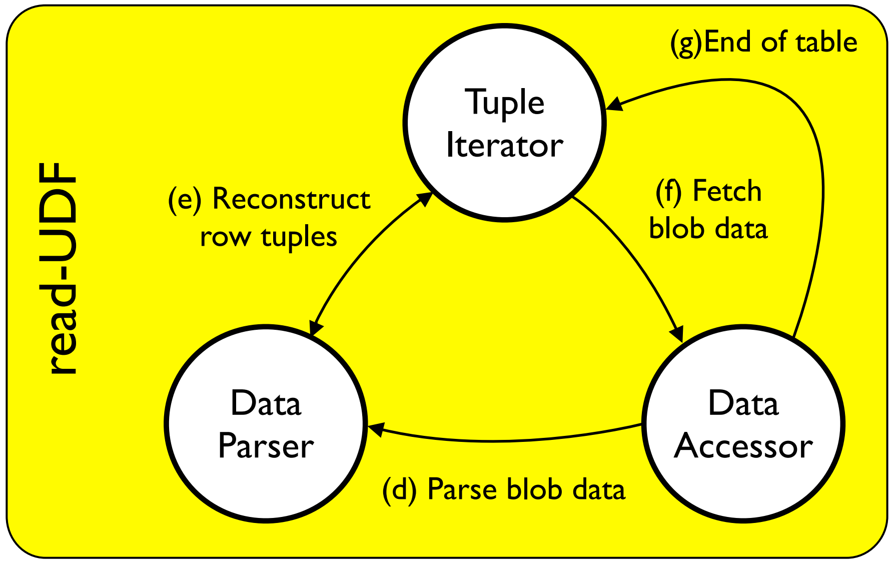
Relation

Customer		
name	phone	market_segment
smith	2134	automobile
john	3425	household
kim	6756	furniture
joe	9878	building
mark	4312	building
steve	2435	automobile
jim	5766	household
ian	8789	household

Physical Table

Customer_trojan		
segment_ID	attribute_ID	blob_data
1	name	smith, john, kim, joe
1	phone	2134, 3425, 6756, 9878
1	market_segment	automobile, household, furniture, building
2	name	mark, steve, jim, ian
2	phone	4312, 2435, 5766, 8789
2	market_segment	building, automobile, household, household

Trojan Columns



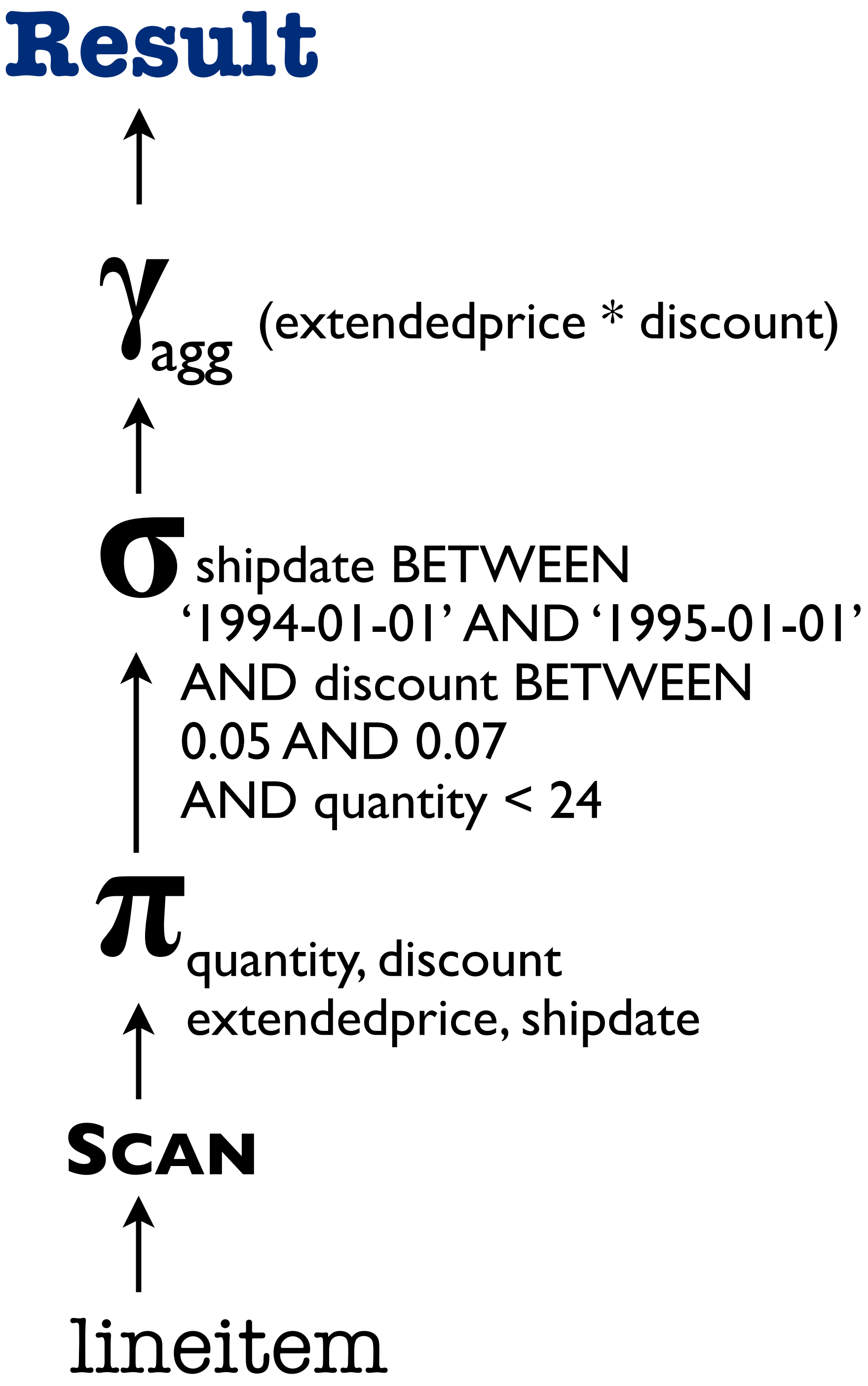
Relation

Customer		
name	phone	market_segment
smith	2134	automobile
john	3425	household
kim	6756	furniture
joe	9878	building
mark	4312	building
steve	2435	automobile
jim	5766	household
ian	8789	household

Physical Table

Customer_trojan		
segment_ID	attribute_ID	blob_data
1	name	smith, john, kim, joe
1	phone	2134, 3425, 6756, 9878
1	market_segment	automobile, household, furniture, building
2	name	mark, steve, jim, ian
2	phone	4312, 2435, 5766, 8789
2	market_segment	building, automobile, household, household

Example: TPC-H Query 6



Result

γ_{agg} (extendedprice * discount)

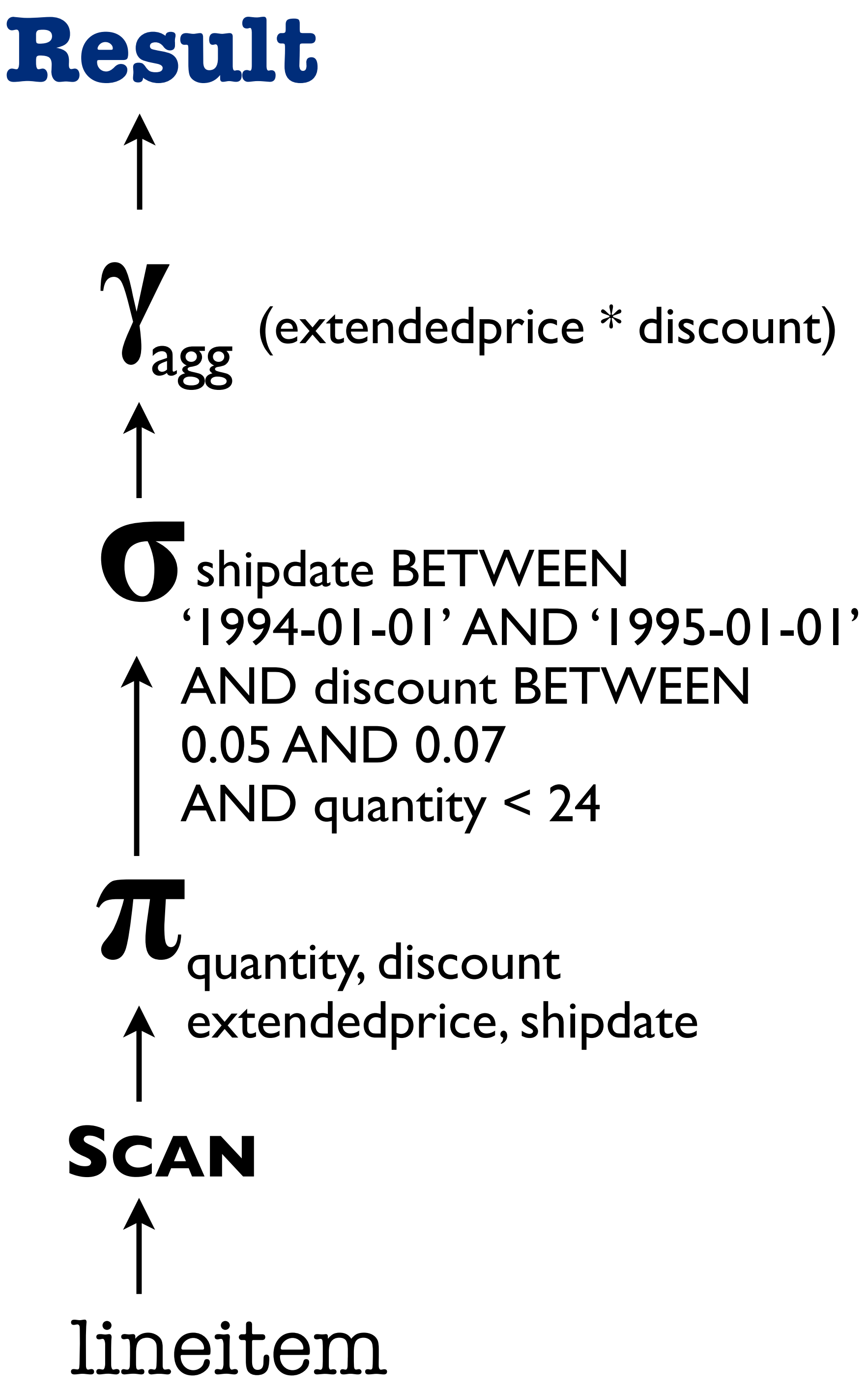
σ shipdate BETWEEN
'1994-01-01' AND '1995-01-01'
AND discount BETWEEN
0.05 AND 0.07
AND quantity < 24

$\pi_{\text{quantity, discount extendedprice, shipdate}}$

Scan

lineitem

Example: TPC-H Query 6



Result

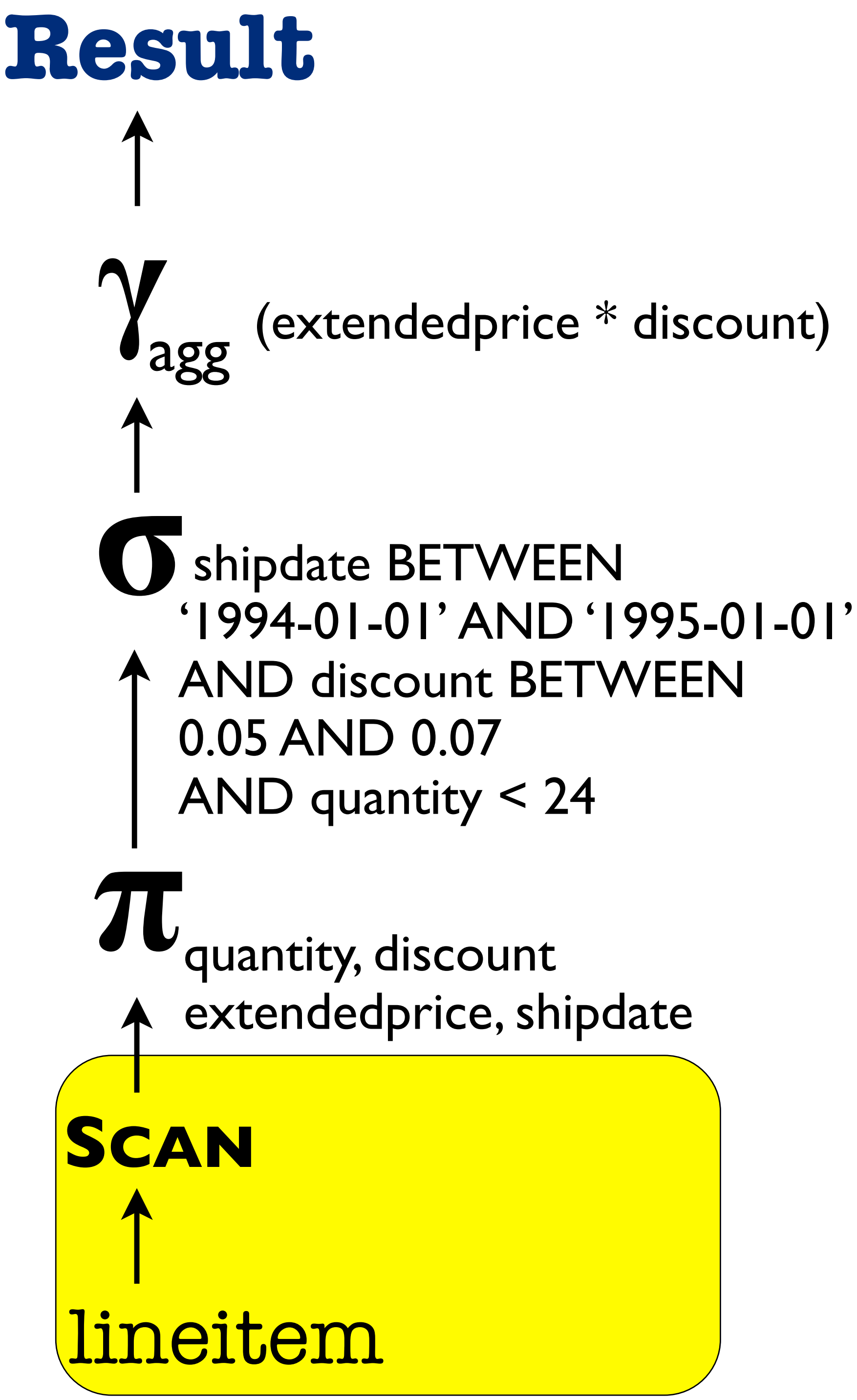
γ_{agg} (extendedprice * discount)

σ shipdate BETWEEN '1994-01-01' AND '1995-01-01' AND discount BETWEEN 0.05 AND 0.07 AND quantity < 24

$\pi_{\text{quantity, discount extendedprice, shipdate}}$

Scan

lineitem



Result

γ_{agg} (extendedprice * discount)

σ shipdate BETWEEN '1994-01-01' AND '1995-01-01' AND discount BETWEEN 0.05 AND 0.07 AND quantity < 24

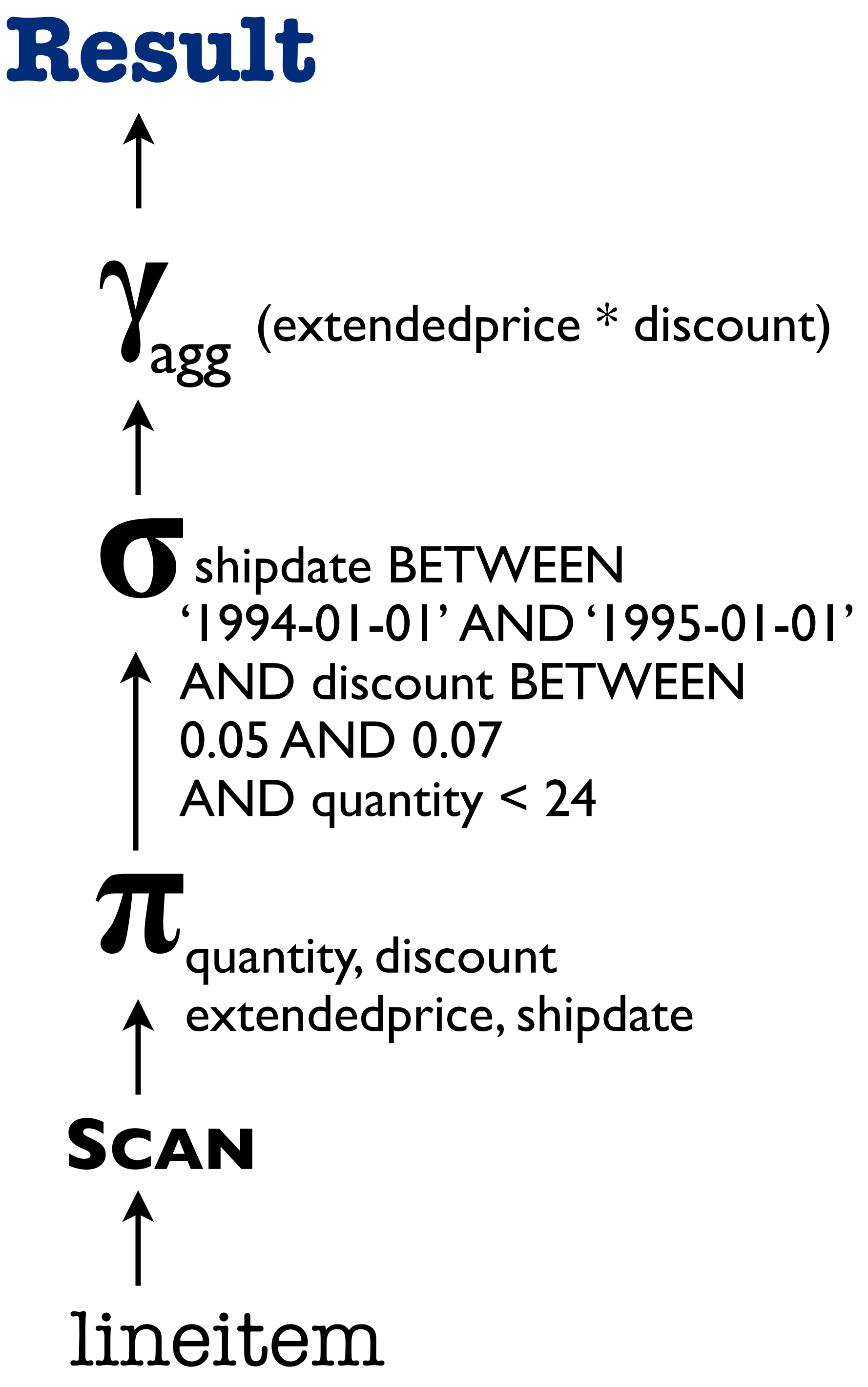
$\pi_{\text{quantity, discount extendedprice, shipdate}}$

Scan

lineitem

scanUDF

Example: TPC-H Query 6



Result

↑

γ_{agg} (extendedprice * discount)

↑

σ shipdate BETWEEN '1994-01-01' AND '1995-01-01' AND discount BETWEEN 0.05 AND 0.07 AND quantity < 24

↑

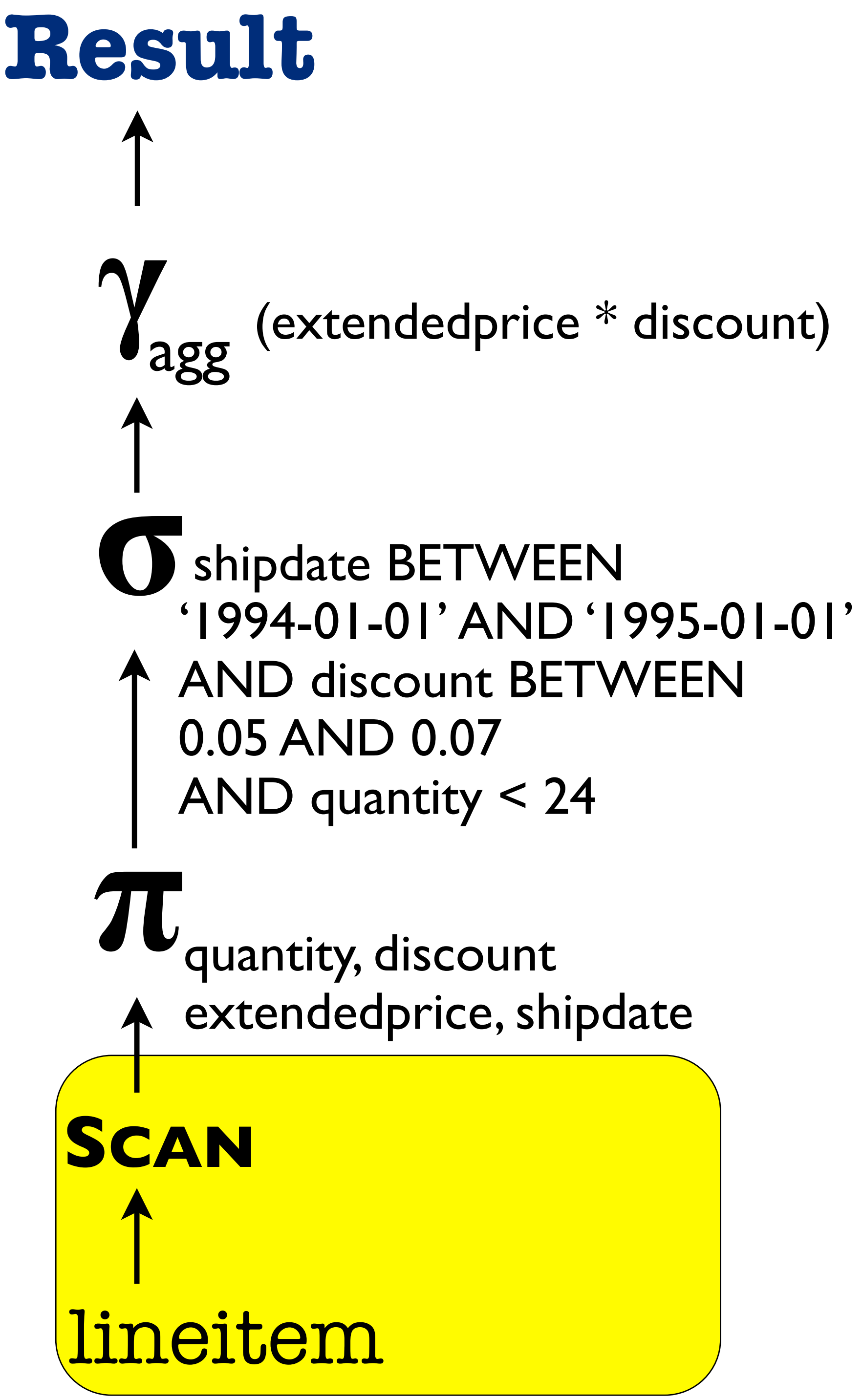
$\pi_{quantity, discount extendedprice, shipdate}$

↑

SCAN

↑

lineitem



Result

↑

γ_{agg} (extendedprice * discount)

↑

σ shipdate BETWEEN '1994-01-01' AND '1995-01-01' AND discount BETWEEN 0.05 AND 0.07 AND quantity < 24

↑

$\pi_{quantity, discount extendedprice, shipdate}$

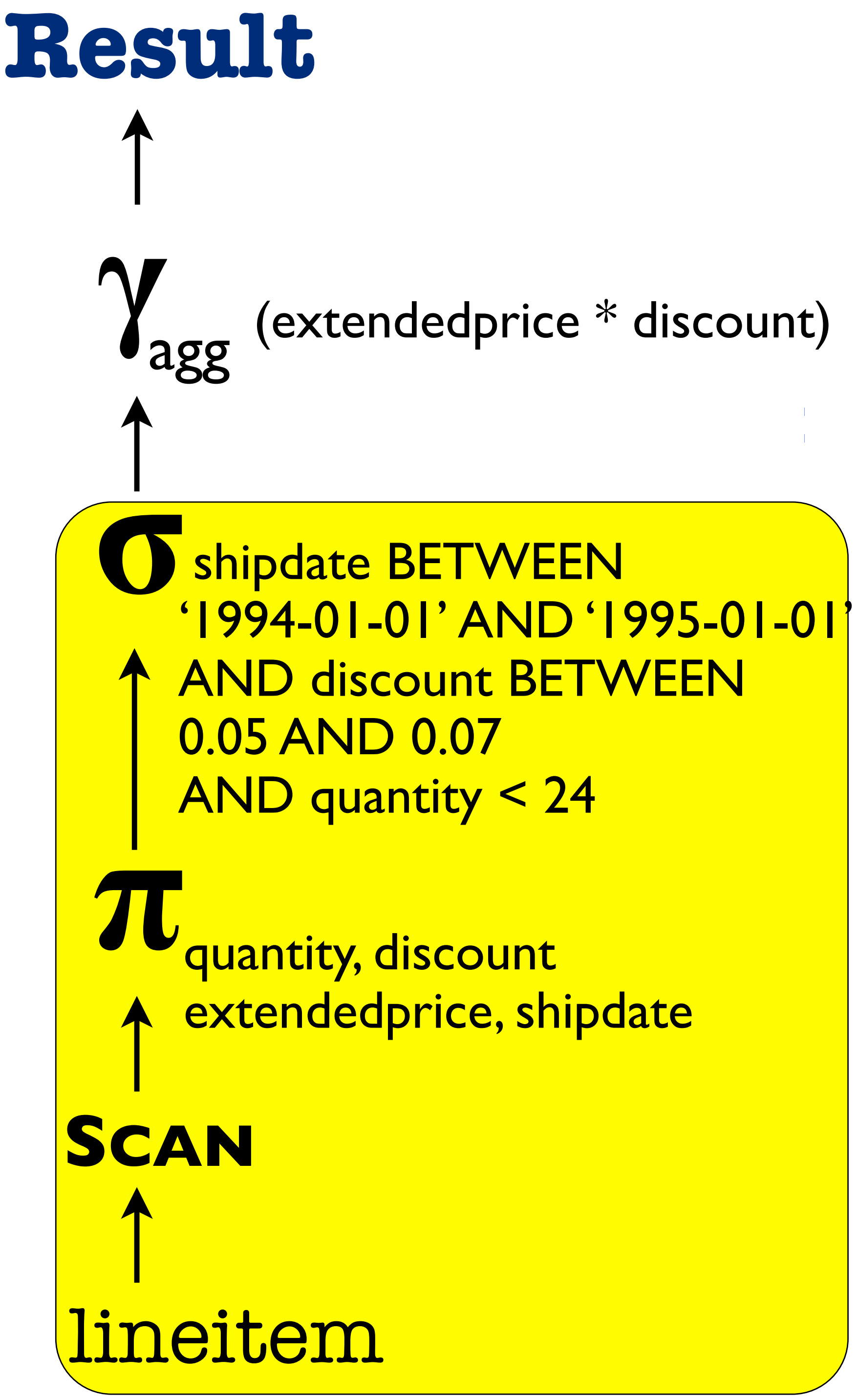
↑

SCAN

↑

lineitem

scanUDF



Result

↑

γ_{agg} (extendedprice * discount)

↑

σ shipdate BETWEEN '1994-01-01' AND '1995-01-01' AND discount BETWEEN 0.05 AND 0.07 AND quantity < 24

↑

$\pi_{quantity, discount extendedprice, shipdate}$

↑

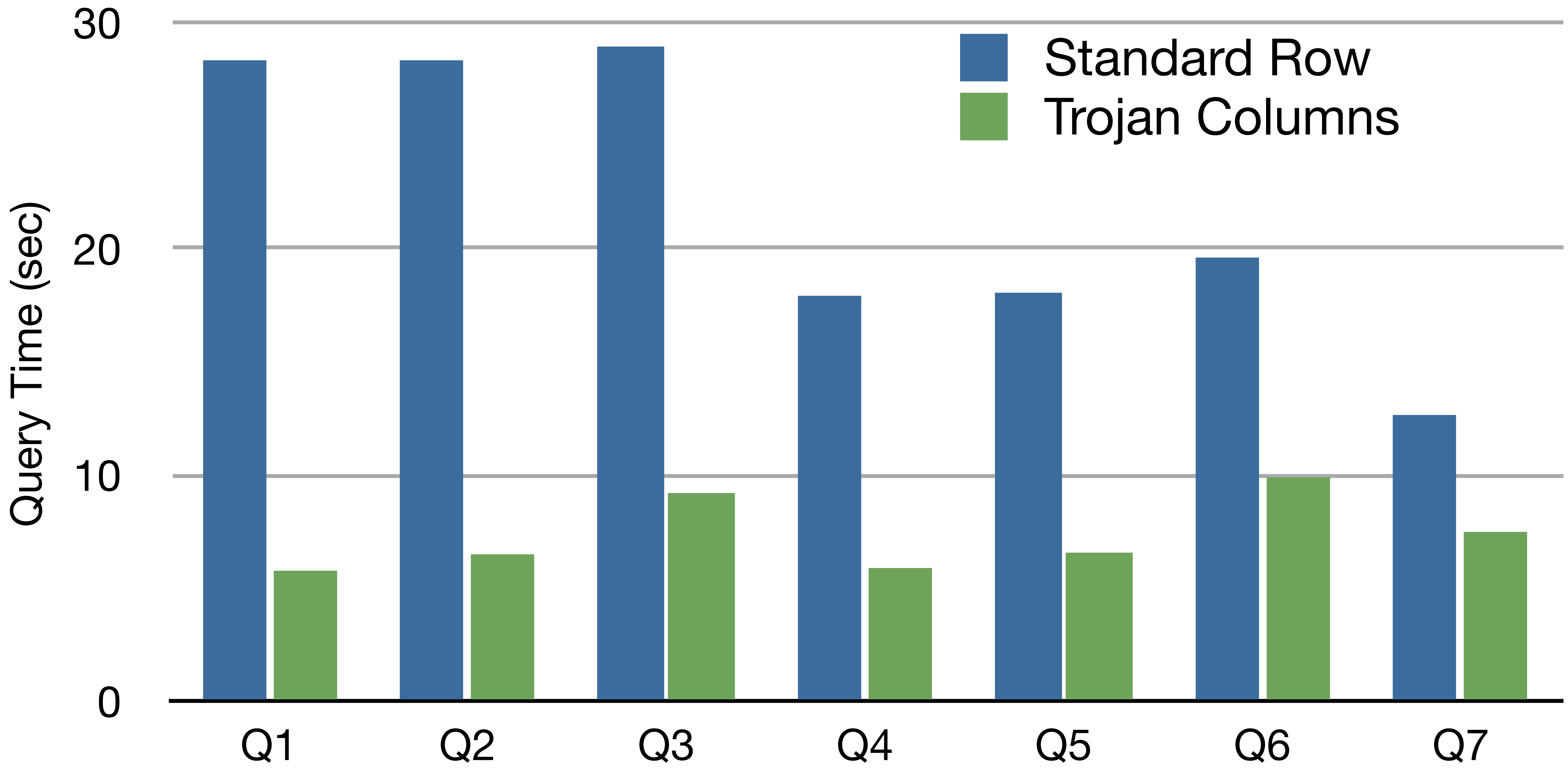
SCAN

↑

lineitem

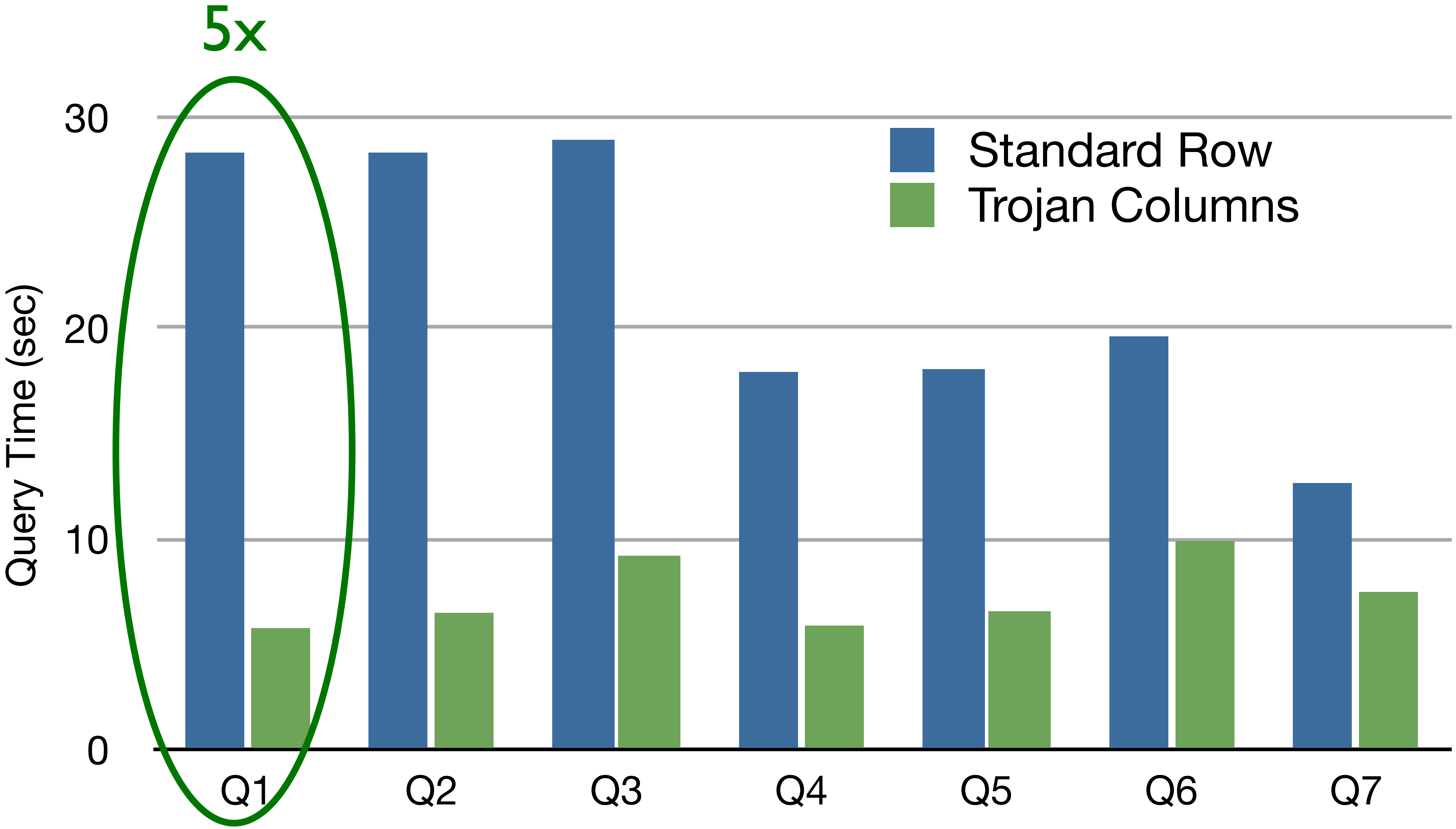
selectUDF

Benchmark Results *



* Mike Stonebraker et. al. C-Store: A Column Oriented DBMS. VLDB 2005

Benchmark Results *



* Mike Stonebraker et. al. C-Store: A Column Oriented DBMS. VLDB 2005

1980s

1990s

2000s

HYRISE

2010s

7 Vertical Partitioning Algorithms

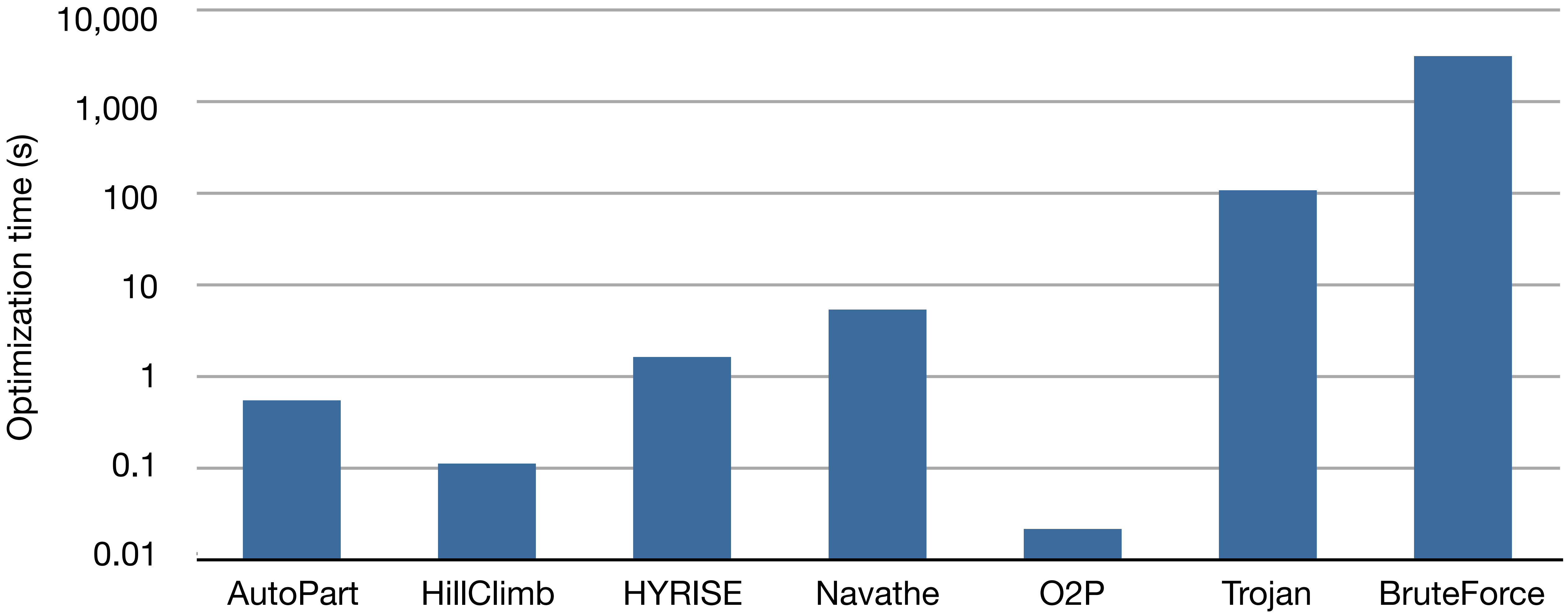
- Brute Force
- Navathe's Algorithm
- HillClimb
- AutoPart
- HYRISE
- O$_2$P
- Trojan

sv-timemachine.net

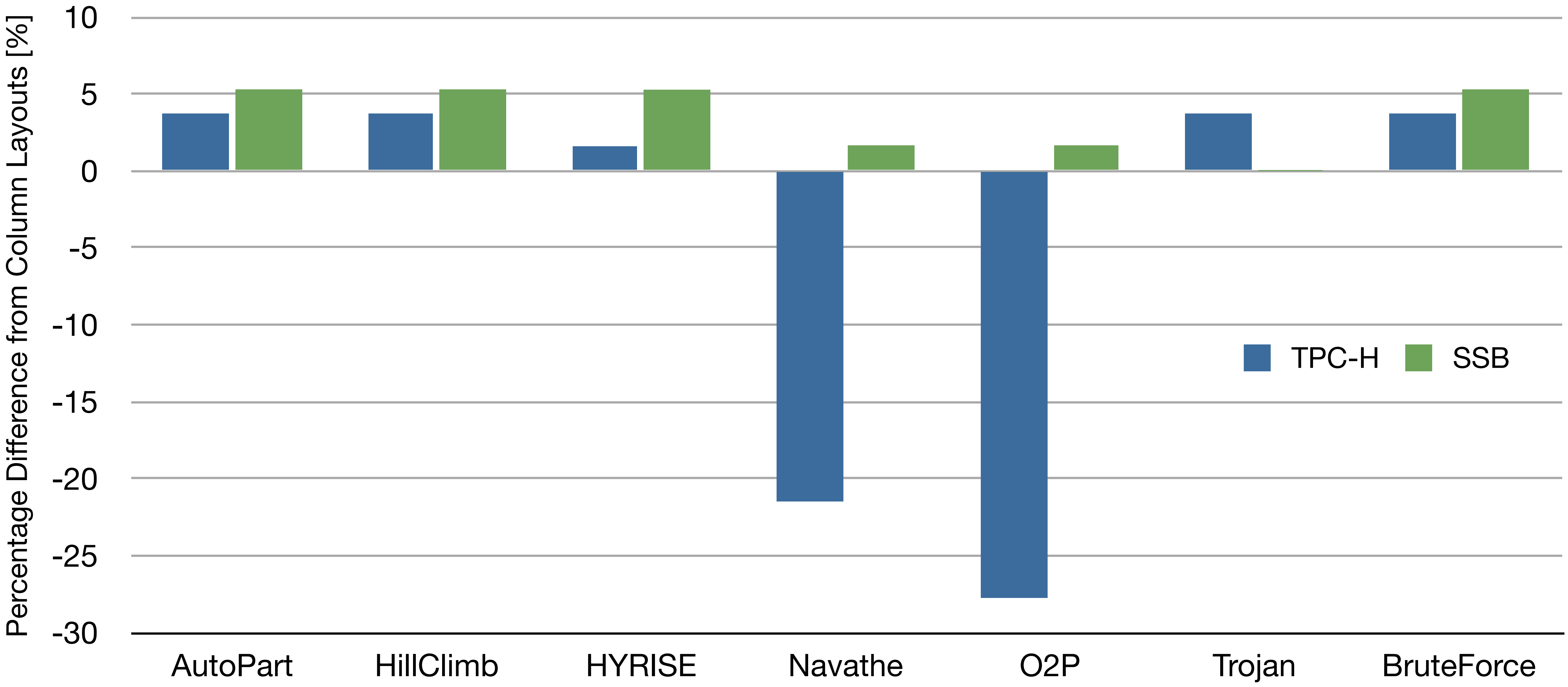
Four Comparison Metrics

- How Fast?
- How Good?
- How fragile?
- Where does it makes sense?

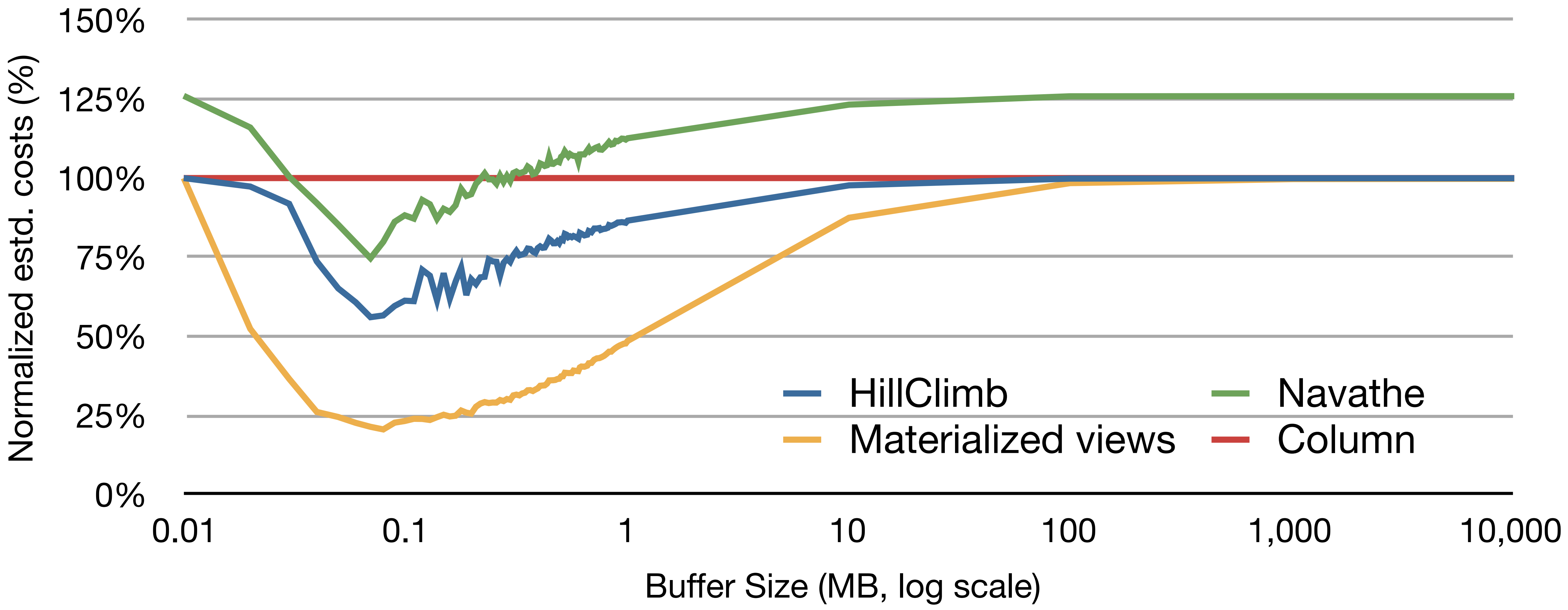
Optimization Runtime



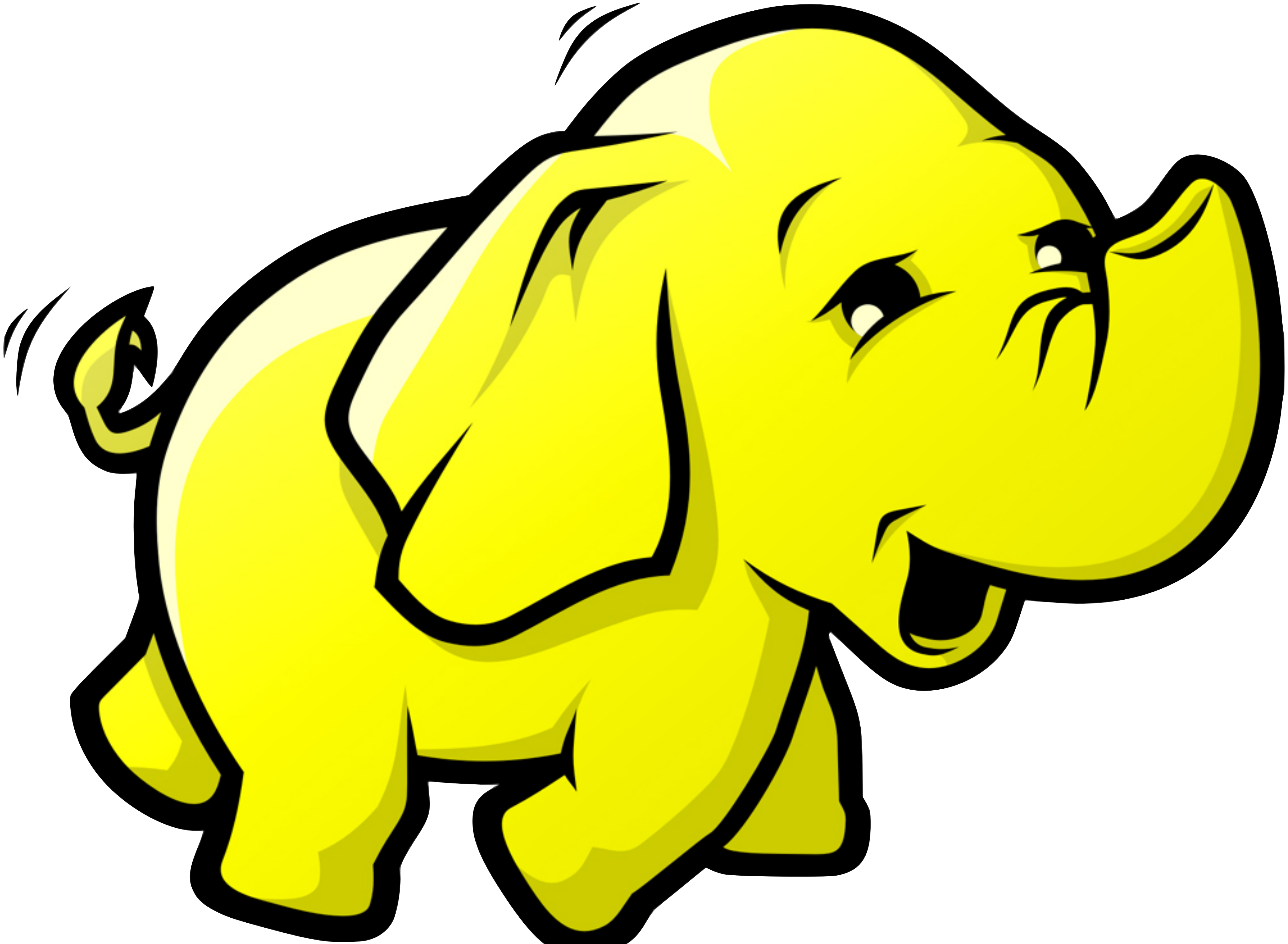
Distance from Column Layouts



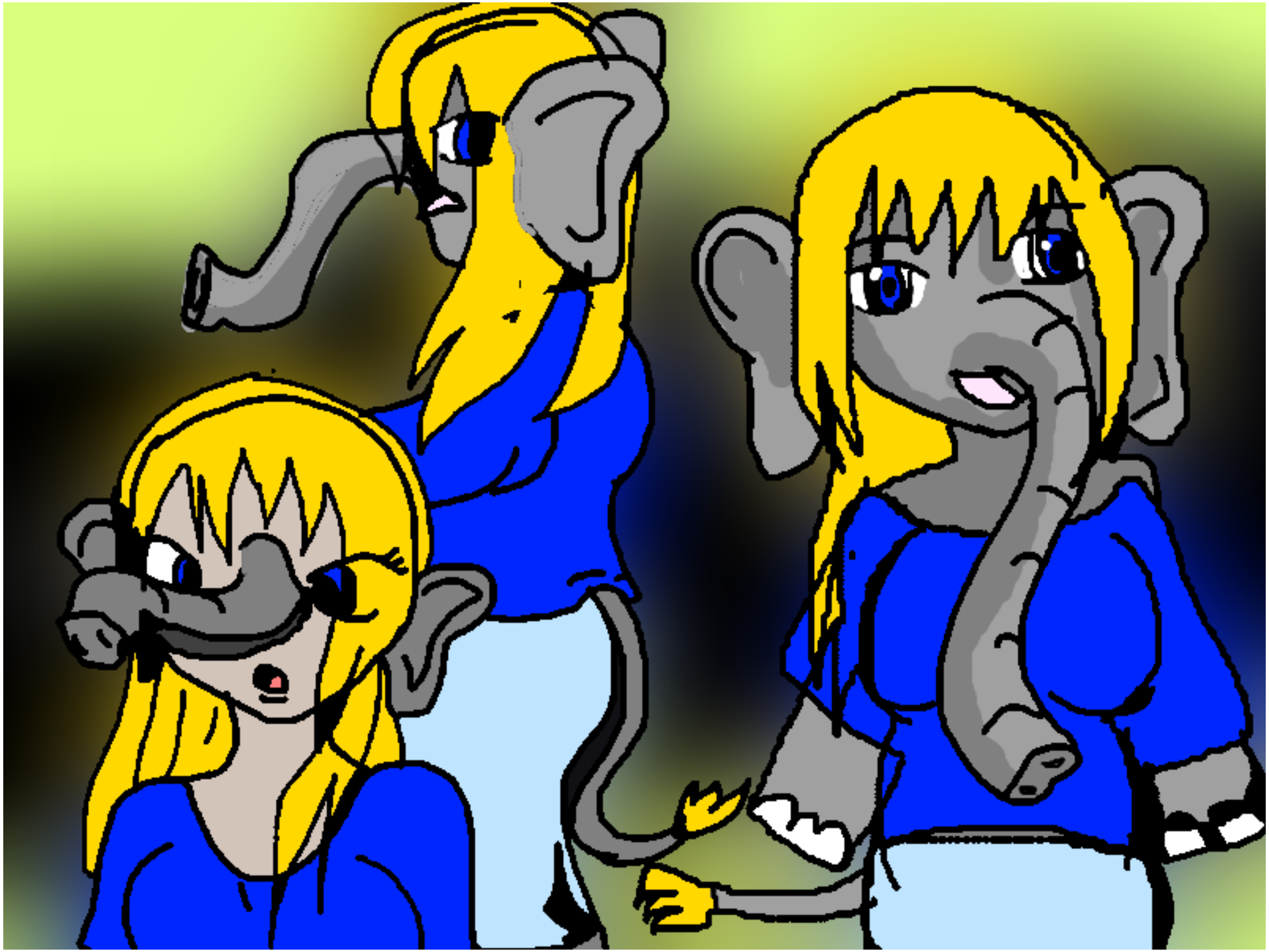
Effect of Buffer Size



Comparison’s Paper: Hadoop Vs PDBMS

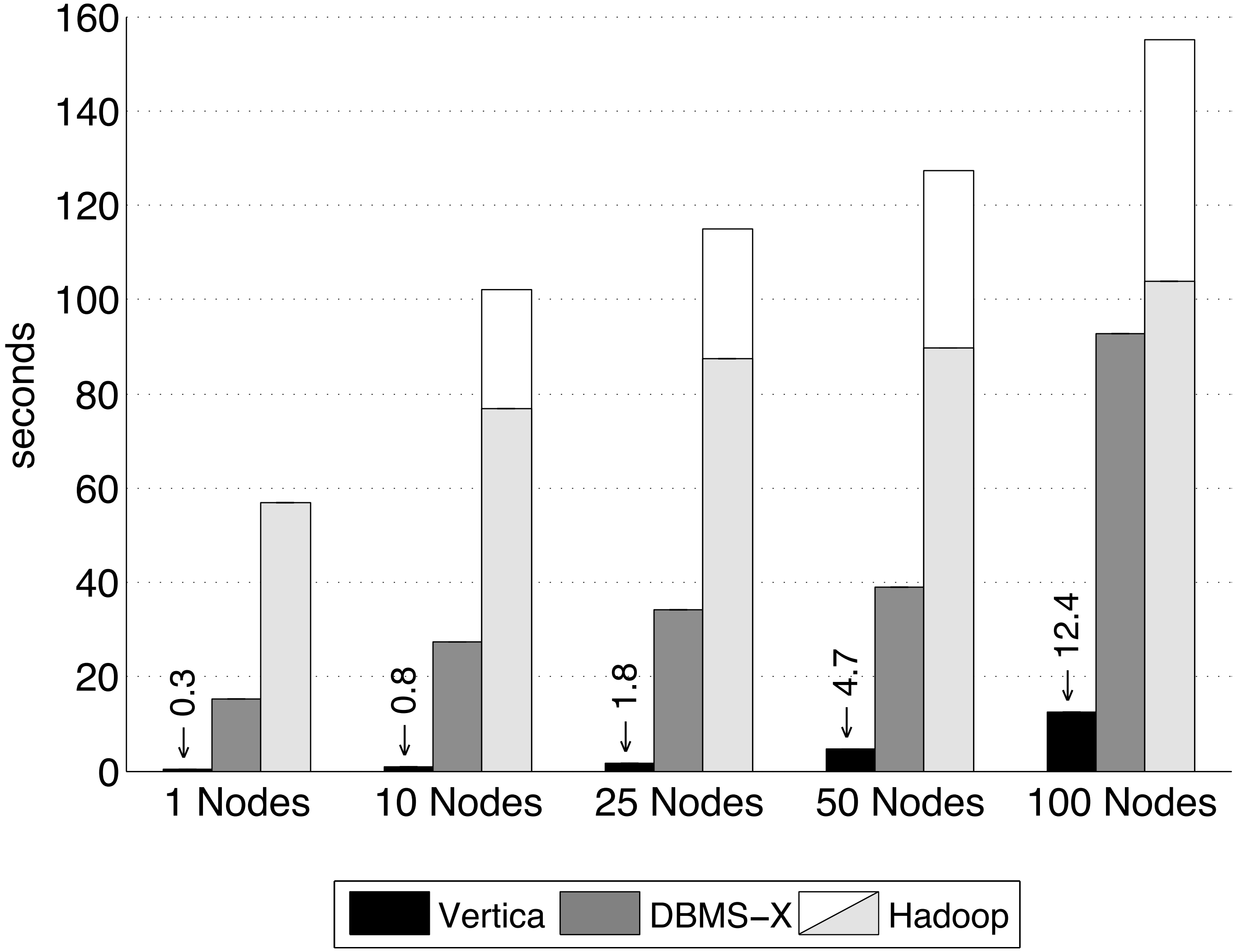


Comparison's Paper: Hadoop Vs PDBMS

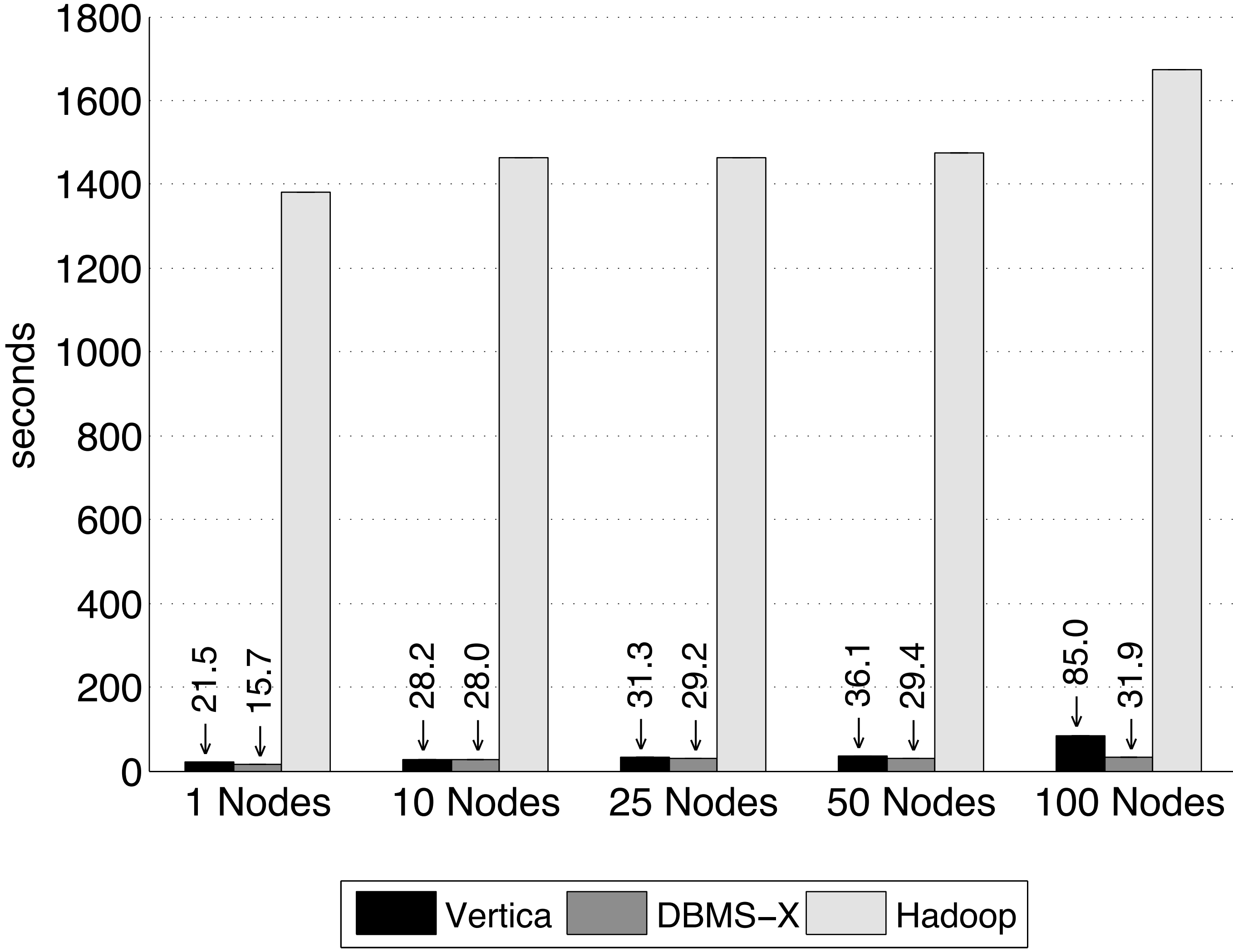


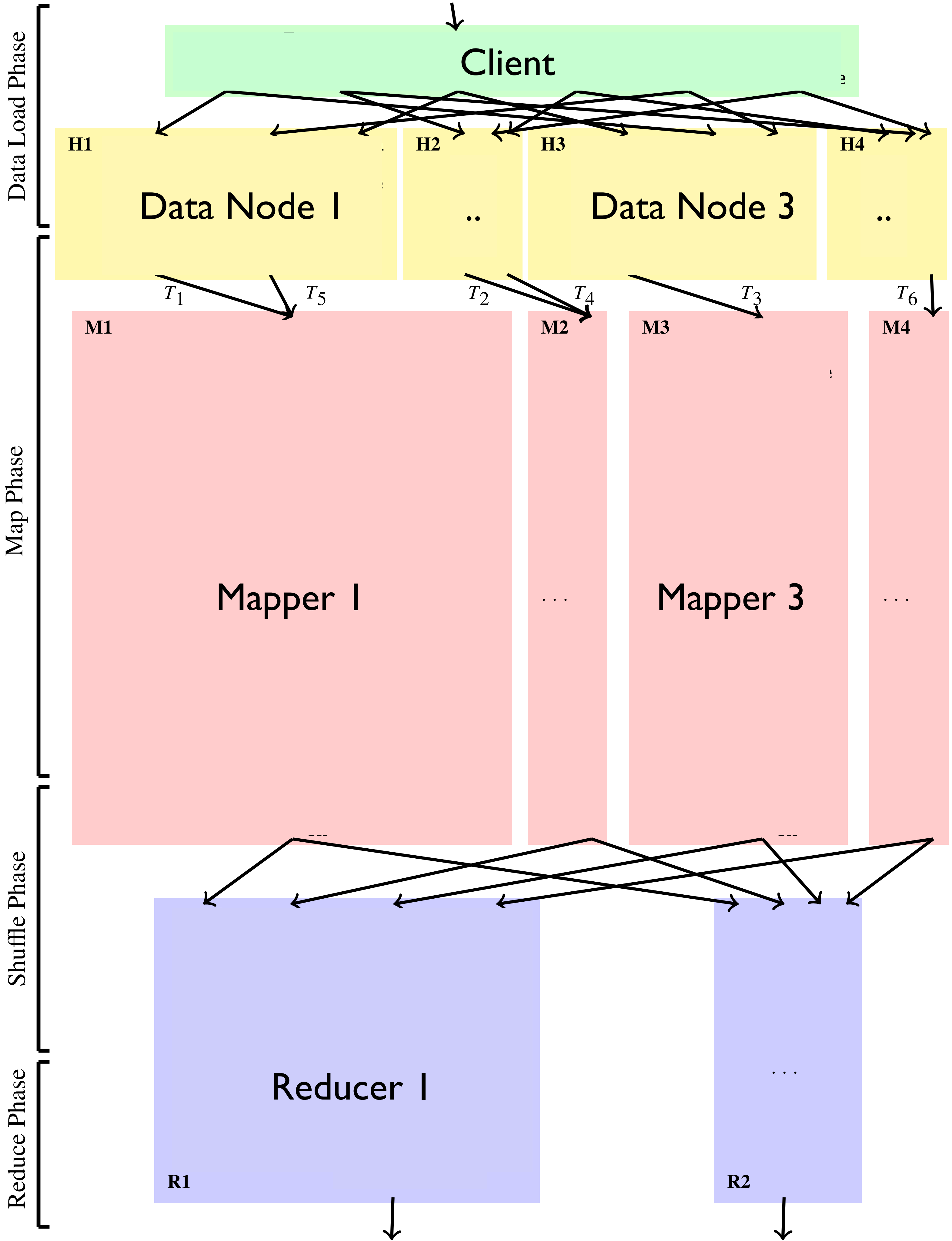
Analytical Query Performance

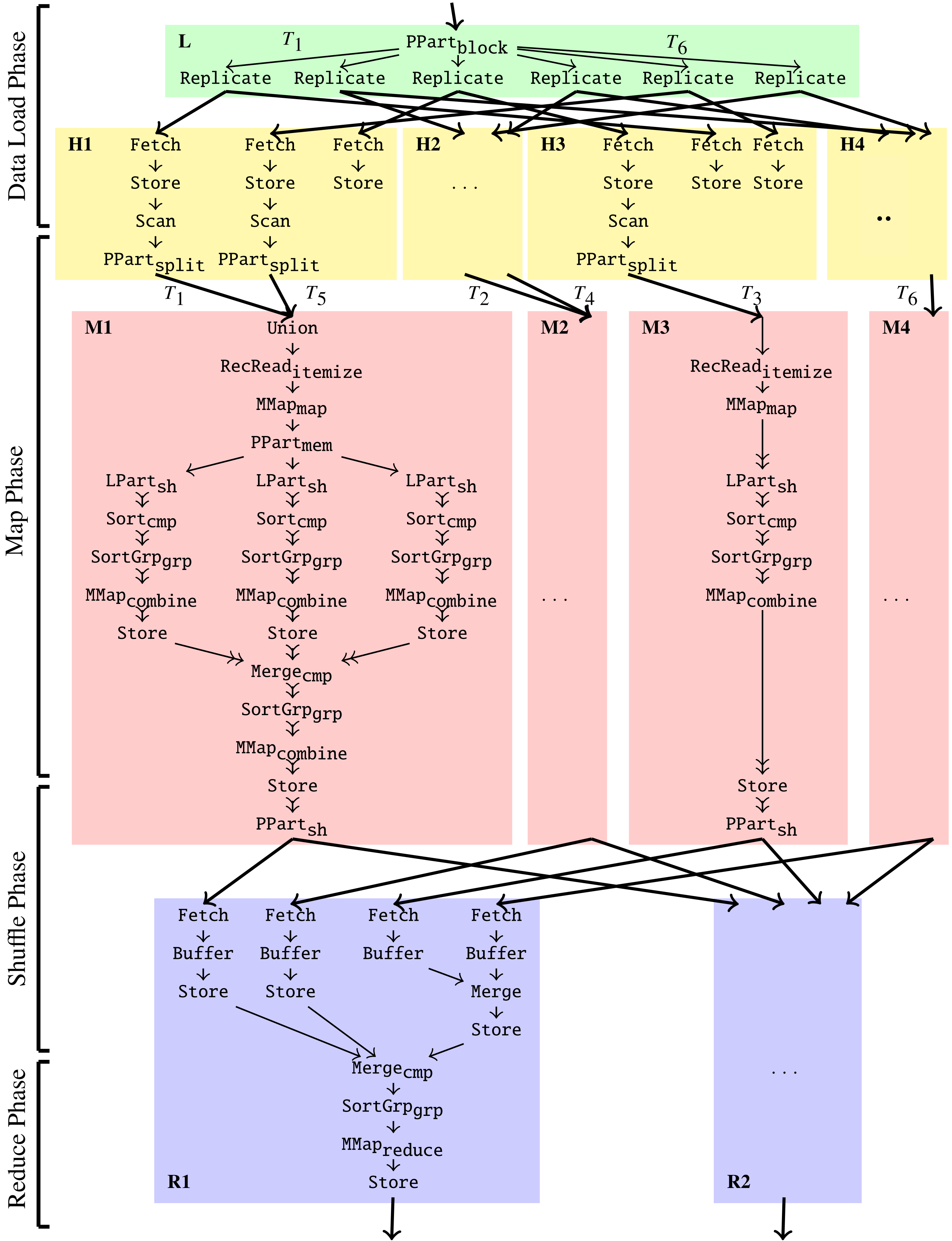
Selection Task

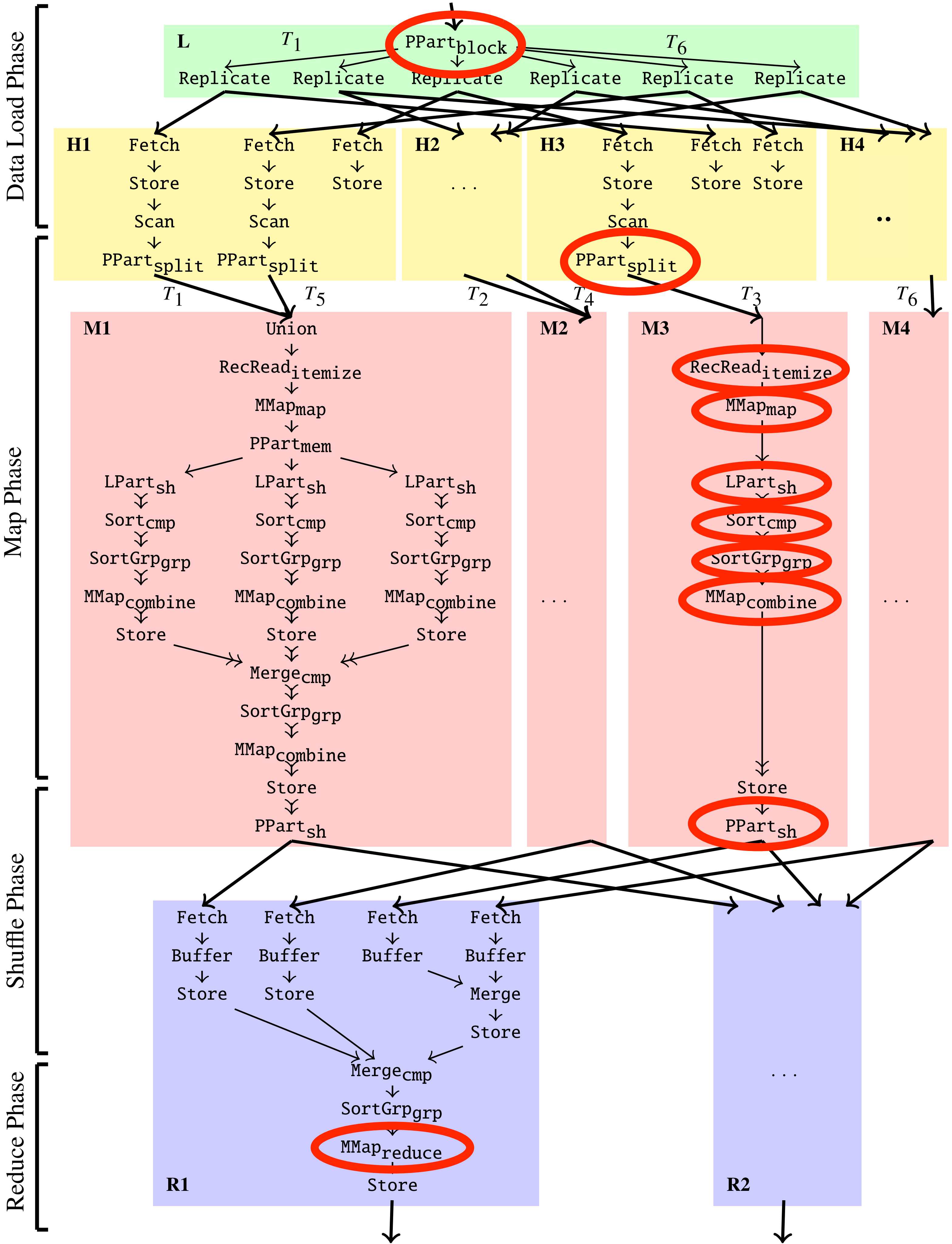


Join Task

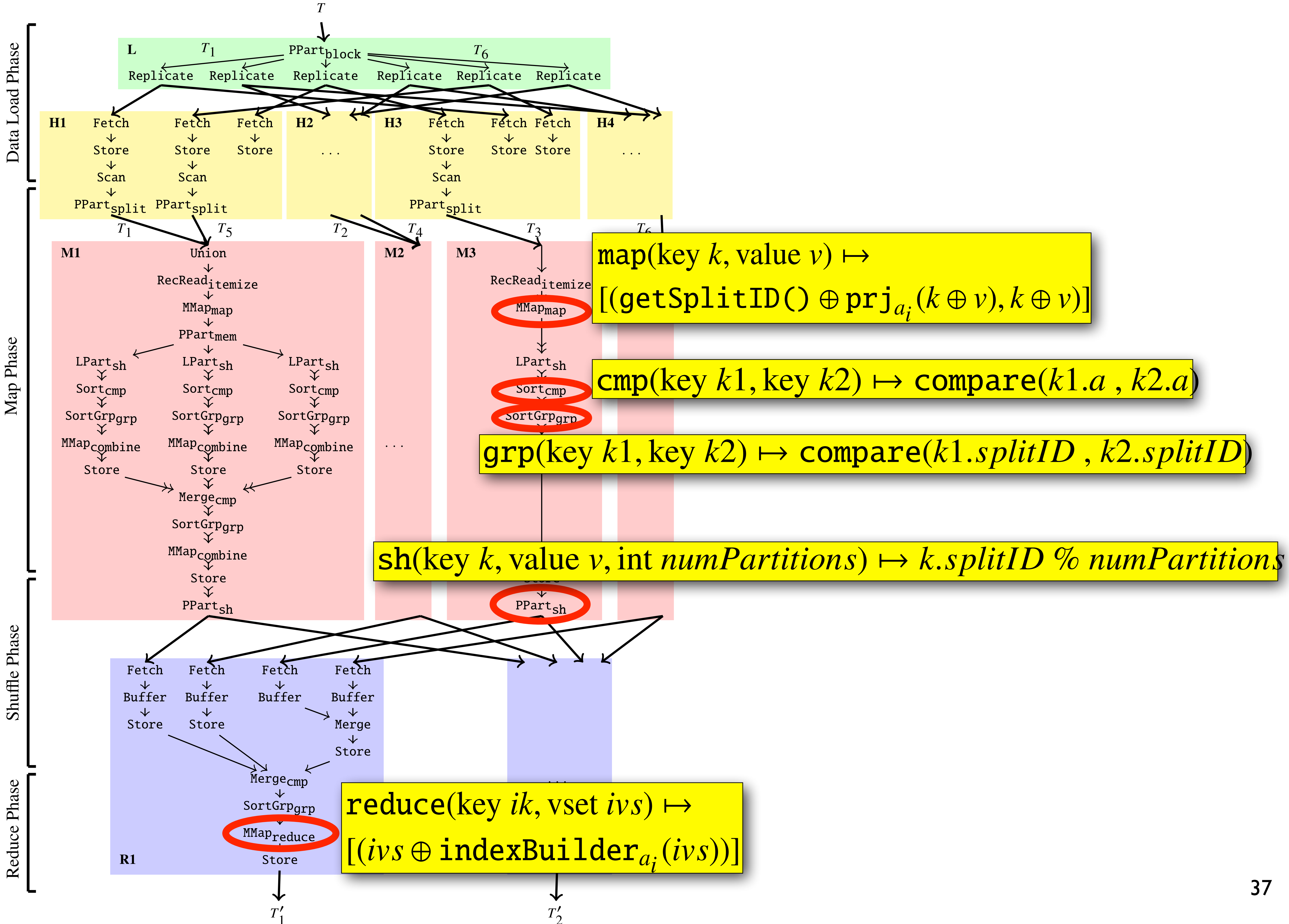



Data Load Phase
Map Phase
Shuffle Phase
Reduce Phase
Client
H1
Data Node 1
H2
..
H3
Data Node 3
H4
..
T_1
T_5
T_2
T_4
T_3
T_6
M1
Mapper 1
M2
. . .
M3
Mapper 3
M4
. . .
Reducer 1
R1
. . .
R2


Data Load Phase
Map Phase
Shuffle Phase
Reduce Phase
L
T1
PPartblock
T6
Replicate
Replicate
Replicate
Replicate
Replicate
Replicate
H1
Fetch
Store
Scan
PPartsplit
Fetch
Store
Scan
PPartsplit
Fetch
Store
H2
. . .
H3
Fetch
Store
Scan
PPartsplit
Fetch
Store
Fetch
Store
H4
..
T1
T5
T2
T4
T3
T6
M1
Union
RecReaditemize
MMapmap
PPartmem
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
Mergecmp
SortGrpgrp
MMapcombine
Store
PPartsh
M2
. . .
M3
RecReaditemize
MMapmap
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
PPartsh
M4
. . .
Fetch
Buffer
Store
Fetch
Buffer
Store
Fetch
Buffer
Fetch
Buffer
Merge
Store
Mergecmp
SortGrpgrp
MMapreduce
Store
R1
. . .
R2


Data Load Phase
Map Phase
Shuffle Phase
Reduce Phase
L
T1
PPartblock
T6
Replicate
Replicate
Replicate
Replicate
Replicate
Replicate
H1
Fetch
Store
Scan
PPartsplit
Fetch
Store
Scan
PPartsplit
Fetch
Store
H2
. . .
H3
Fetch
Store
Scan
PPartsplit
Fetch
Store
Fetch
Store
H4
..
T1
T5
T2
T4
T3
T6
M1
Union
RecReaditemize
MMapmap
PPartmem
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
Mergecmp
SortGrpgrp
MMapcombine
Store
PPartsh
M2
. . .
M3
RecReaditemize
MMapmap
LPartsh
Sortcmp
SortGrpgrp
MMapcombine
Store
PPartsh
M4
. . .
Fetch
Buffer
Store
Fetch
Buffer
Store
Fetch
Buffer
Fetch
Buffer
Merge
Store
Mergecmp
SortGrpgrp
MMapreduce
Store
R1
. . .
R2

Trojan Index Creation



$\mathrm{map}(\text{key } k, \text{value } v) \mapsto$
$[(\texttt{getSplitID()} \oplus \texttt{prj}_{a_i}(k \oplus v), k \oplus v)]$

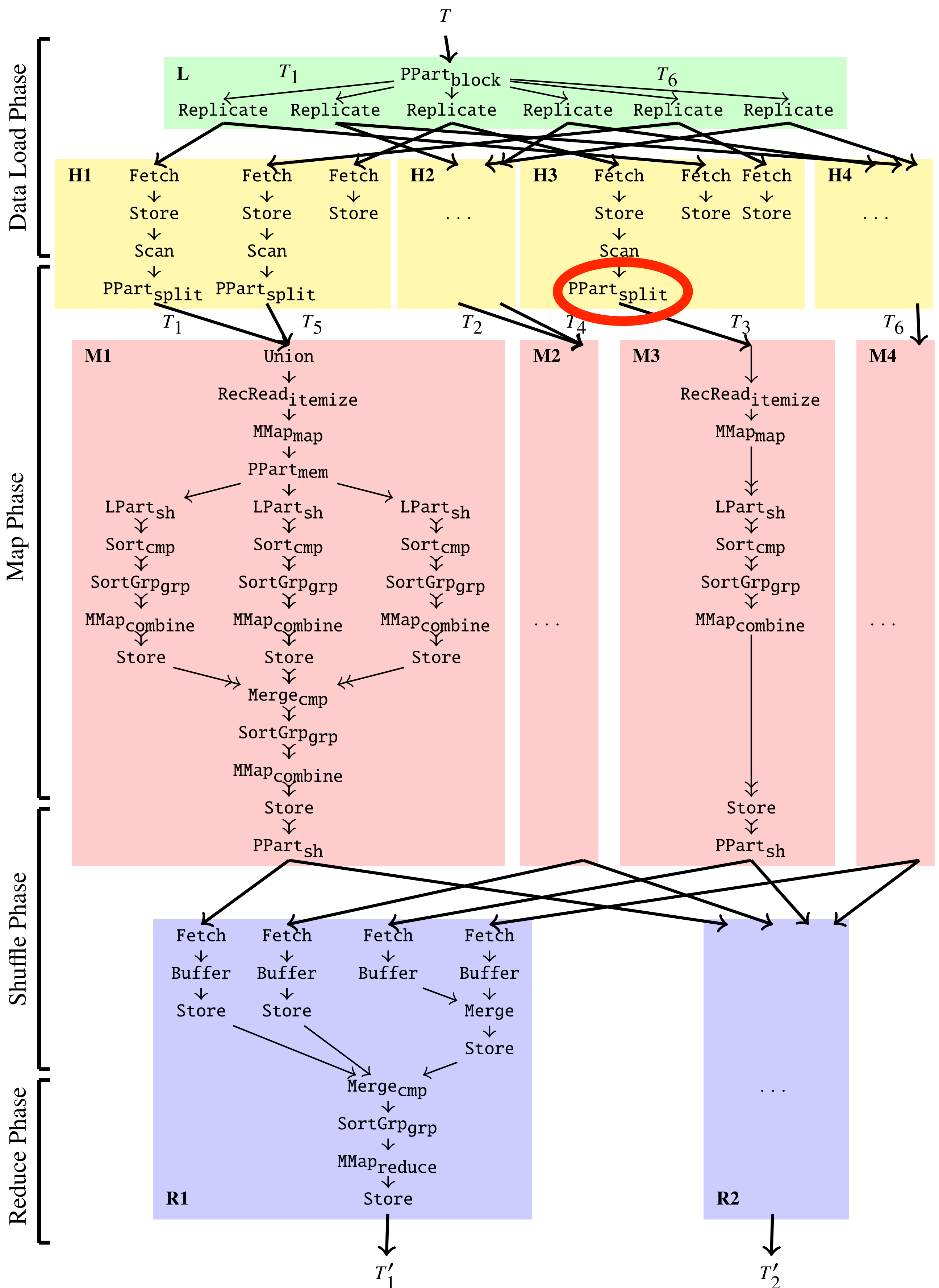
$\mathrm{cmp}(\text{key } k1, \text{key } k2) \mapsto \texttt{compare}(k1.a\ , k2.a)$

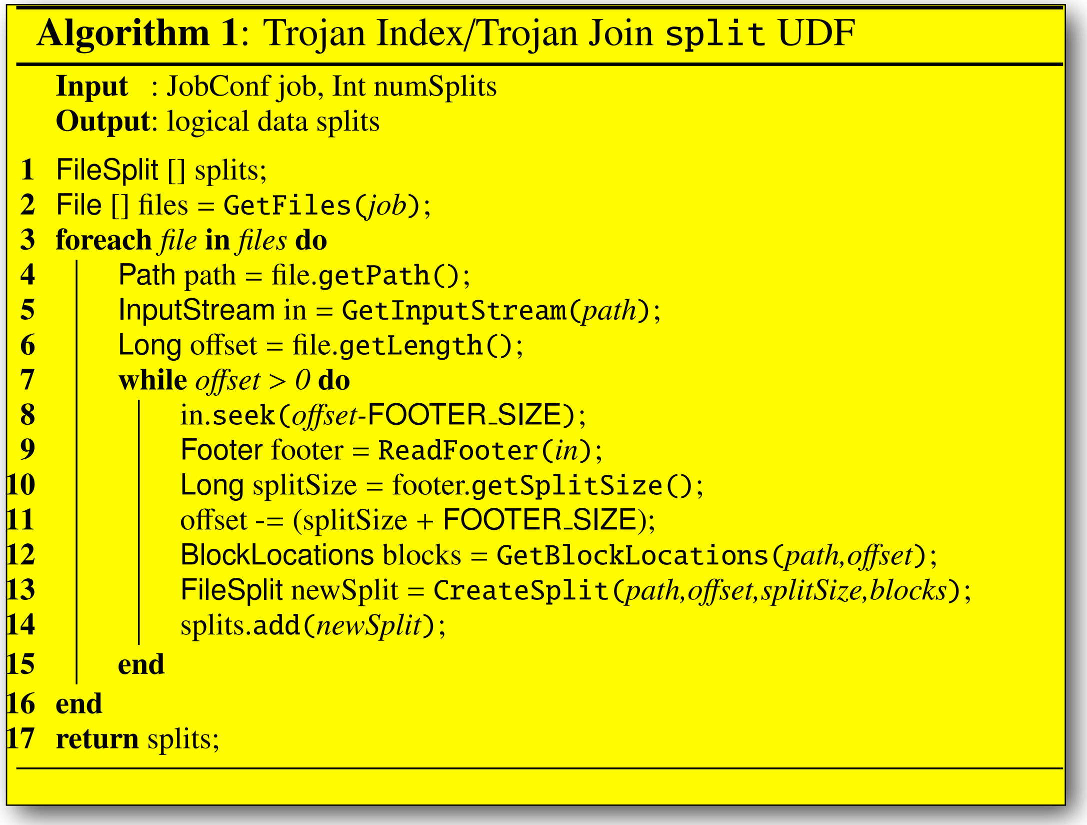
$\mathrm{grp}(\text{key } k1, \text{key } k2) \mapsto \texttt{compare}(k1.splitID\ , k2.splitID)$

$\mathrm{sh}(\text{key } k, \text{value } v, \text{int } numPartitions) \mapsto k.splitID \,\%\, numPartitions$

$\texttt{reduce}(\text{key } ik, \text{vset } ivs) \mapsto$
$[(ivs \oplus \texttt{indexBuilder}_{a_i}(ivs))]$

Trojan Index Access



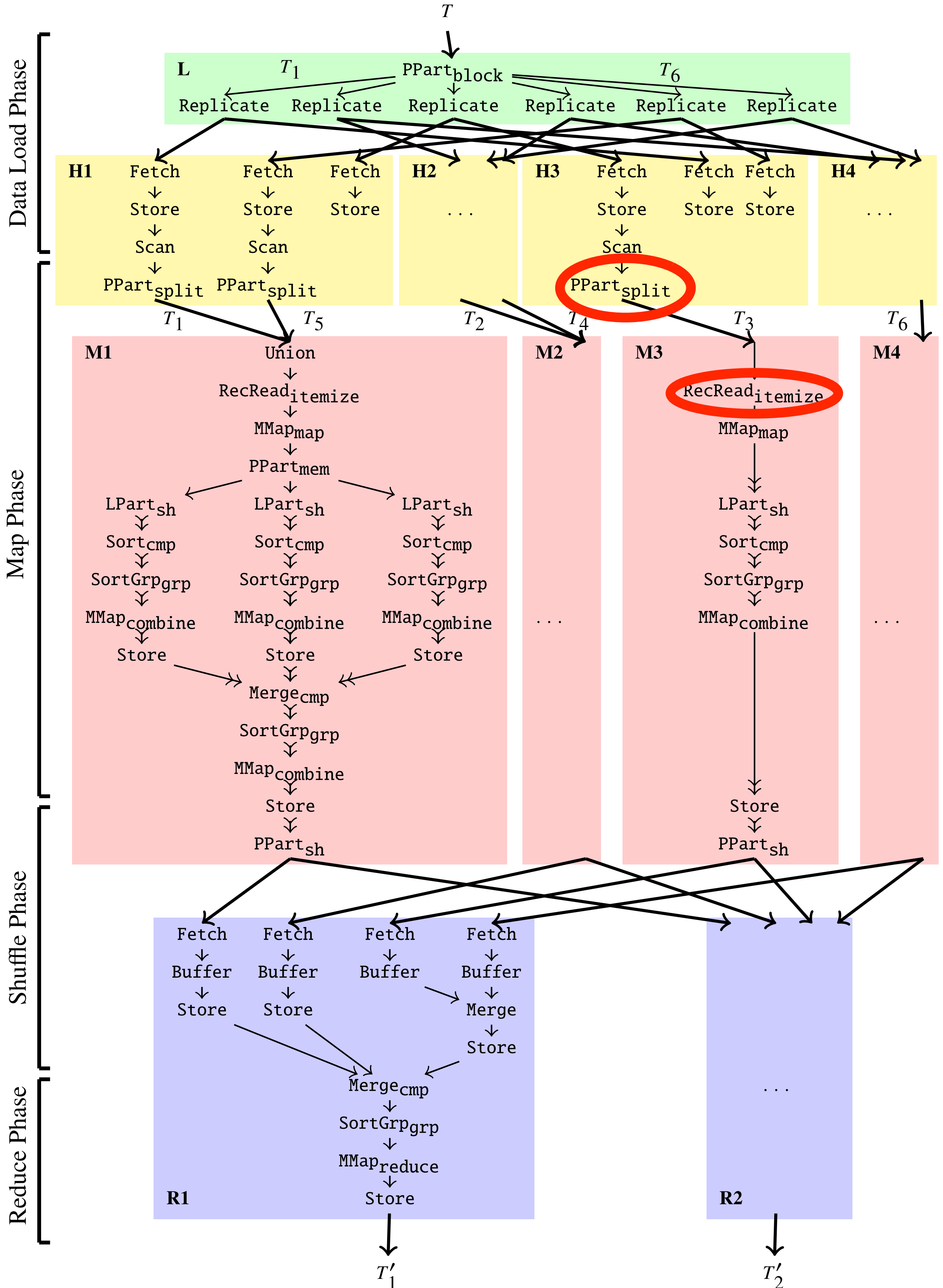


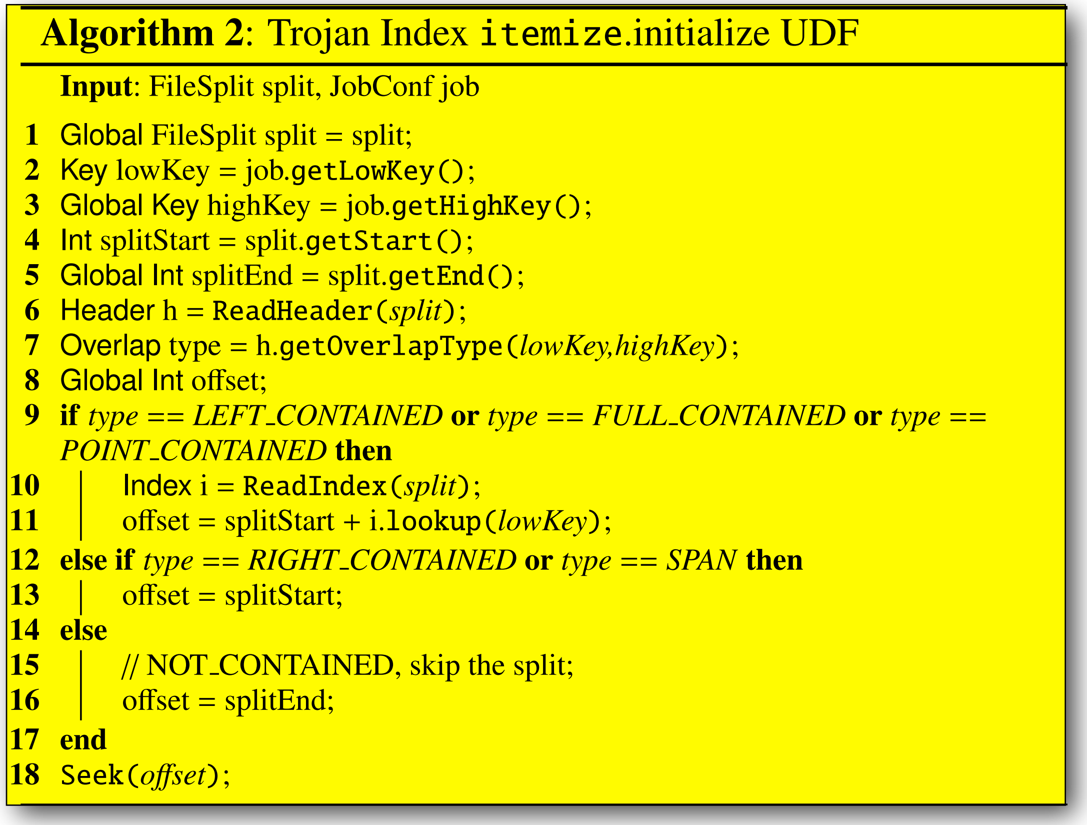
Algorithm 1: Trojan Index/Trojan Join split UDF

Input : JobConf job, Int numSplits
Output: logical data splits

```
FileSplit [] splits;
File [] files = GetFiles(job);
foreach file in files do
    Path path = file.getPath();
    InputStream in = GetInputStream(path);
    Long offset = file.getLength();
    while offset > 0 do
        in.seek(offset-FOOTER_SIZE);
        Footer footer = ReadFooter(in);
        Long splitSize = footer.getSplitSize();
        offset -= (splitSize + FOOTER_SIZE);
        BlockLocations blocks = GetBlockLocations(path,offset);
        FileSplit newSplit = CreateSplit(path,offset,splitSize,blocks);
        splits.add(newSplit);
    end
end
return splits;
```

Trojan Index Access

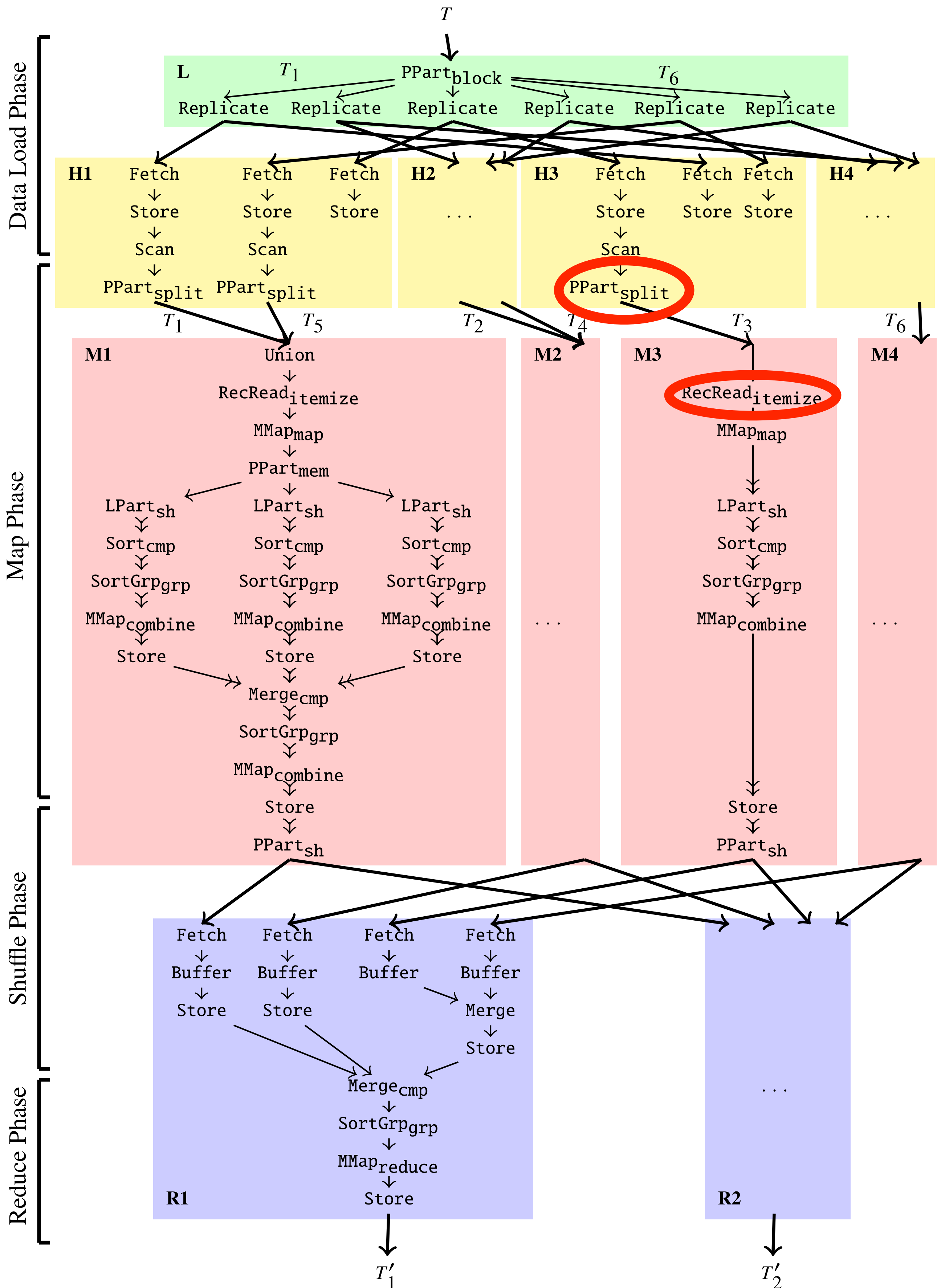




Algorithm 2: Trojan Index `itemize`.initialize UDF

```
Input: FileSplit split, JobConf job
Global FileSplit split = split;
Key lowKey = job.getLowKey();
Global Key highKey = job.getHighKey();
Int splitStart = split.getStart();
Global Int splitEnd = split.getEnd();
Header h = ReadHeader(split);
Overlap type = h.getOverlapType(lowKey,highKey);
Global Int offset;
if type == LEFT_CONTAINED or type == FULL_CONTAINED or type ==
POINT_CONTAINED then
    Index i = ReadIndex(split);
    offset = splitStart + i.lookup(lowKey);
else if type == RIGHT_CONTAINED or type == SPAN then
    offset = splitStart;
else
    // NOT_CONTAINED, skip the split;
    offset = splitEnd;
end
Seek(offset);
```

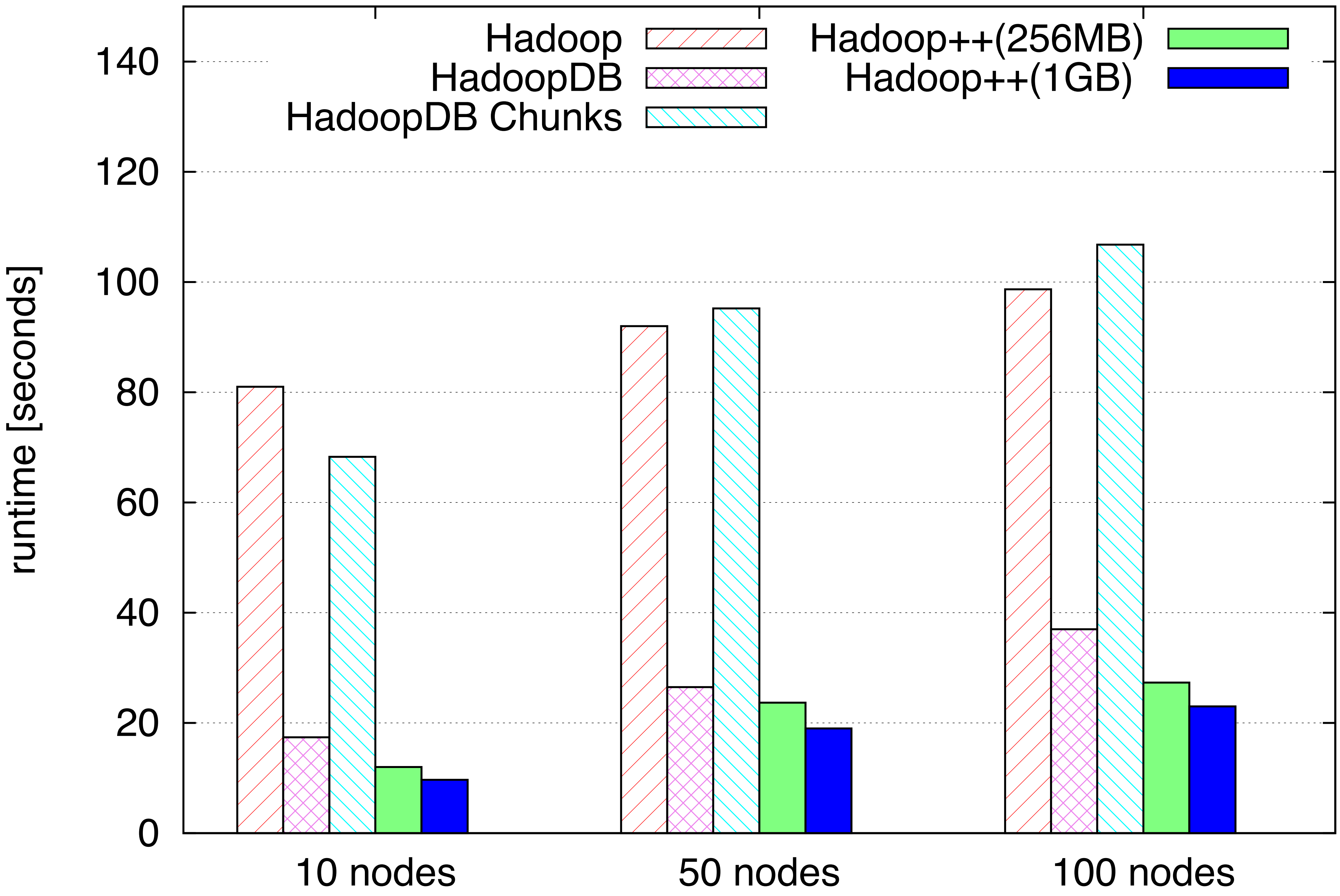
Trojan Index Access



Algorithm 3: Trojan Index `itemize`.next UDF

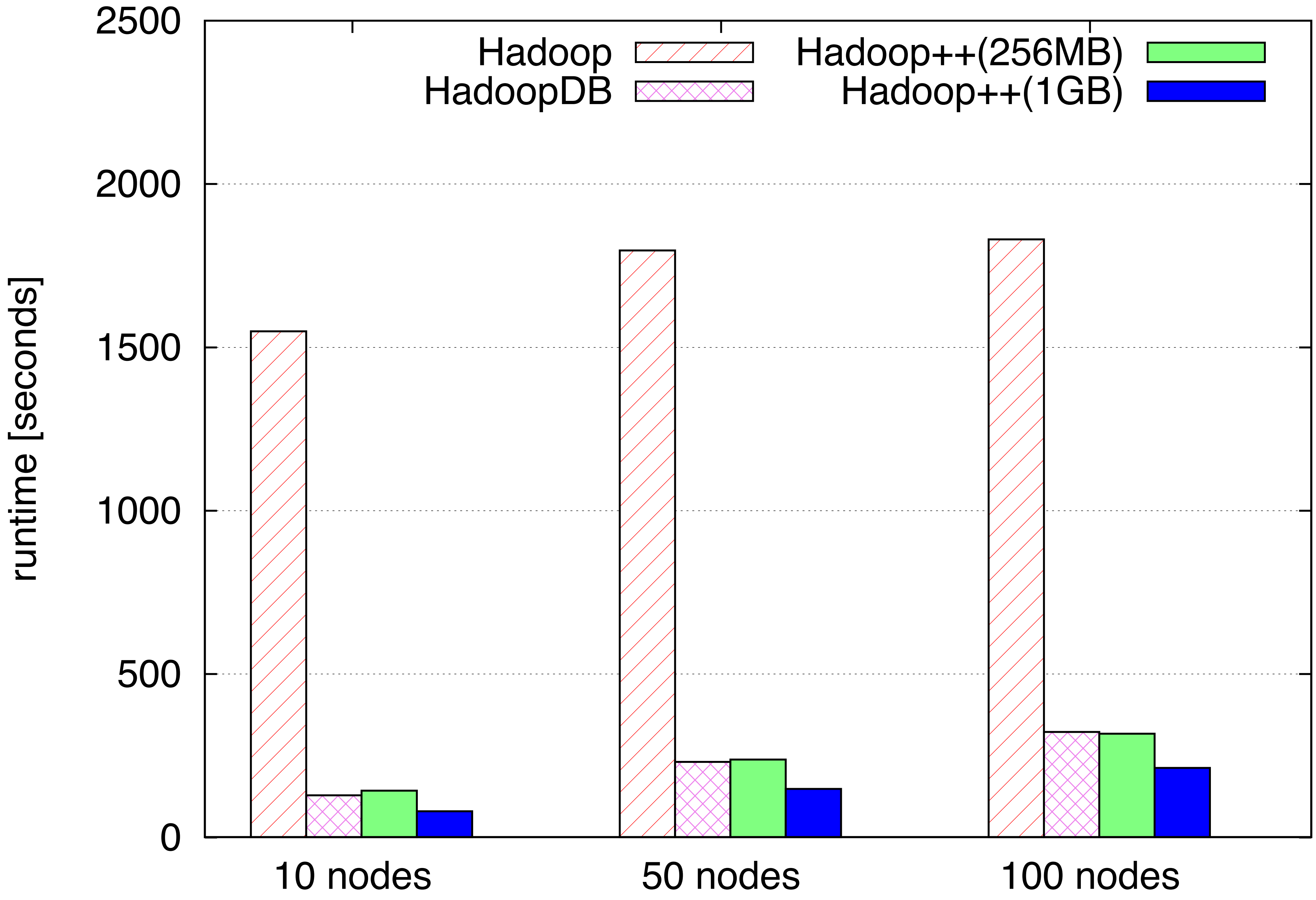
```
Input  : KeyType key, ValueType value
Output: has more records
if $offset < splitEnd$ then
    Record nextRecord = ReadNextRecord(split);
    offset += nextRecord.size();
    if $nextRecord.key < highKey$ then
        SetKeyValue(key, value, nextRecord);
        return true;
    end
end
return false;
```

Selection Analytical Task *



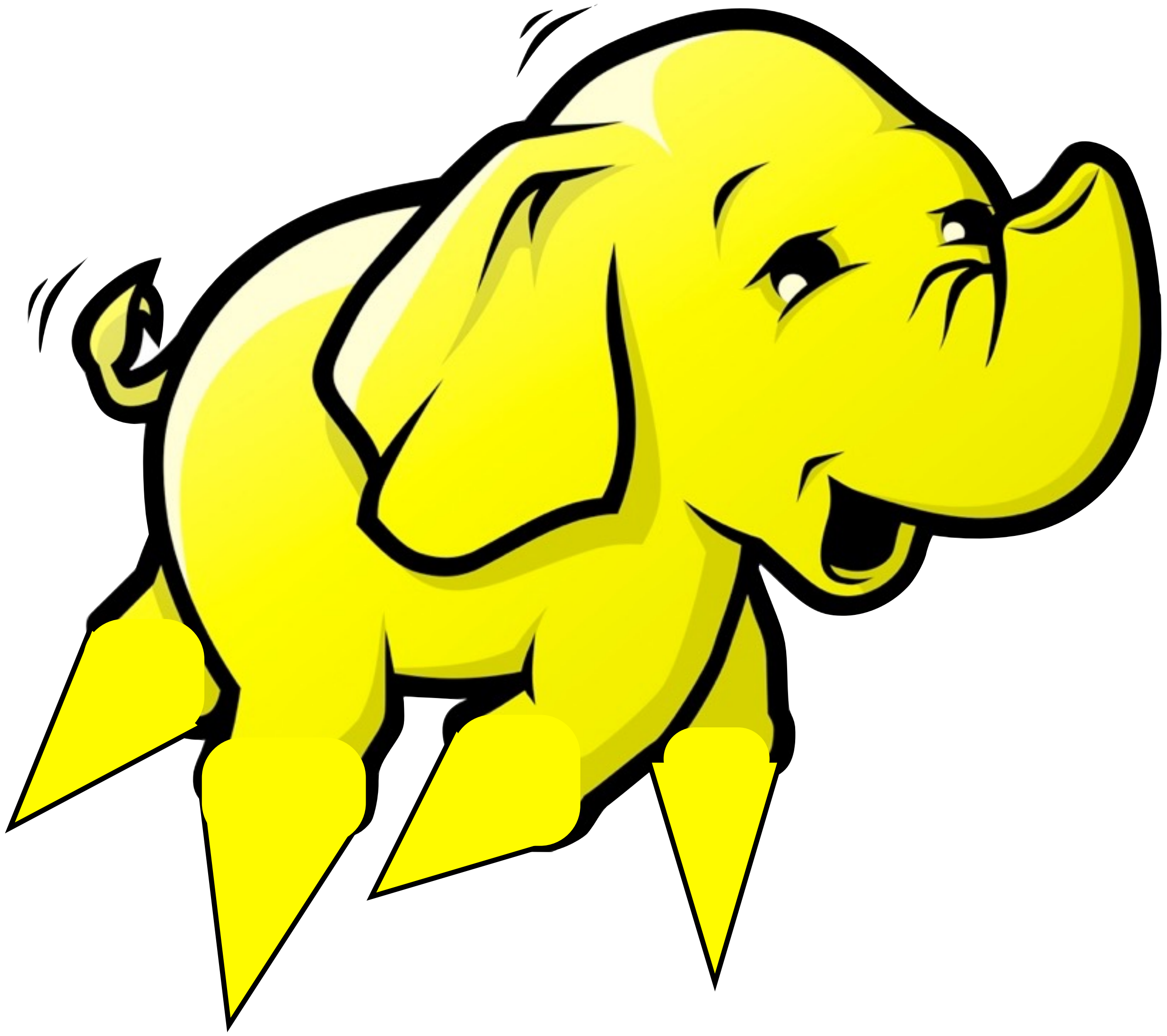
* Pavlo et. al. A Comparison of Approaches to large-Scale Data Analysis. SIGMOD 2009

Join Analytical Task *

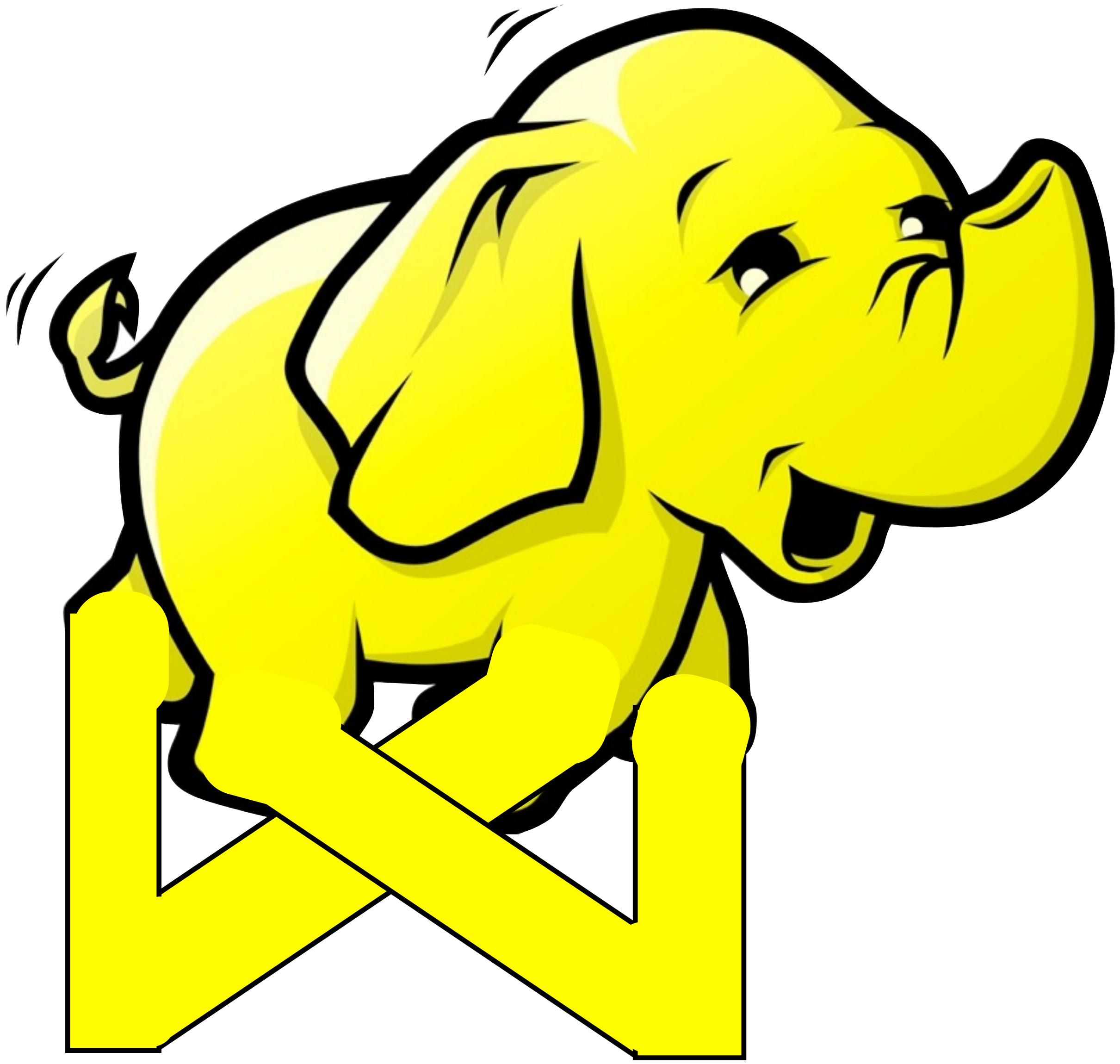


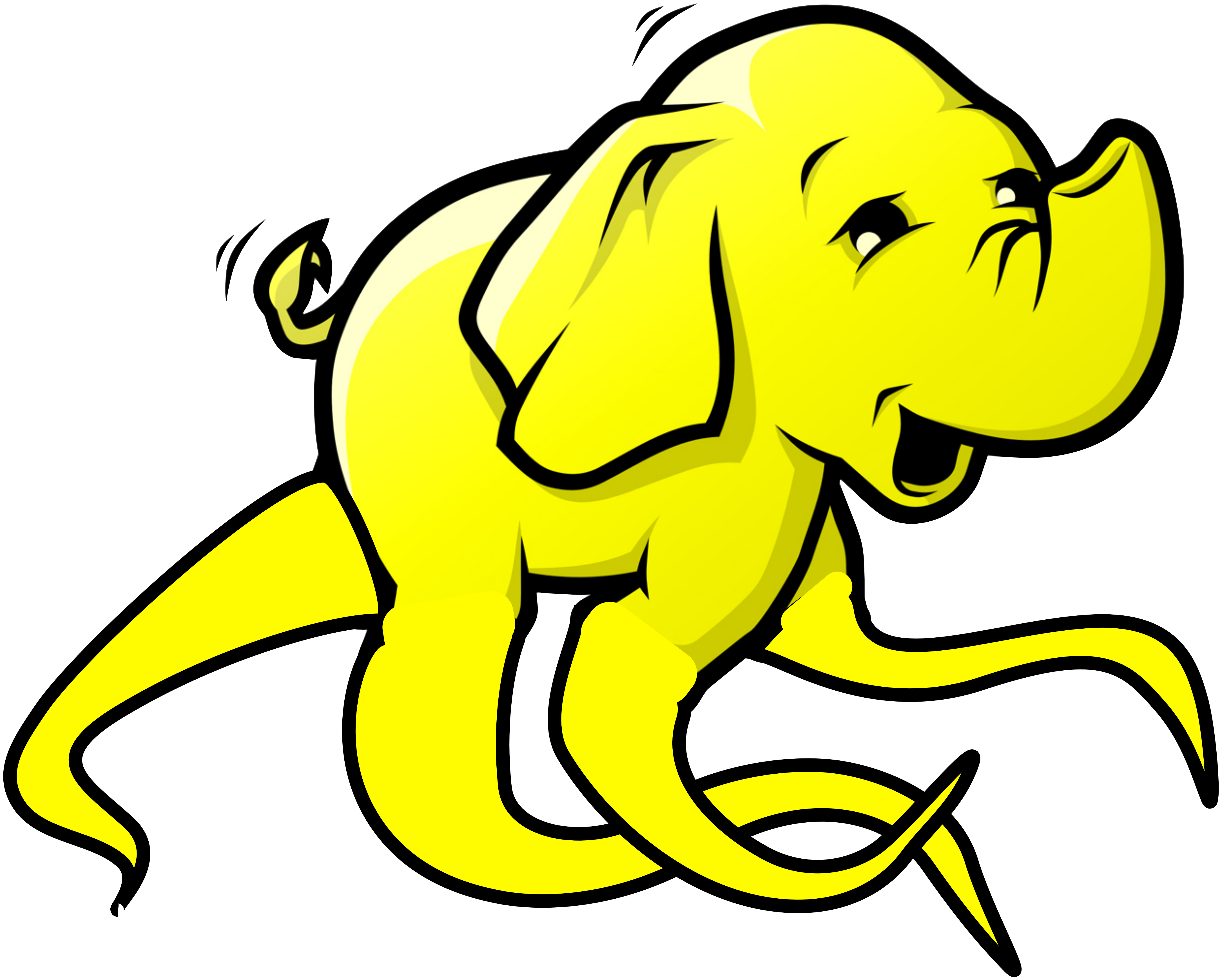
* Pavlo et. al. A Comparison of Approaches to large-Scale Data Analysis. SIGMOD 2009

Trojan Index



Trojan Join

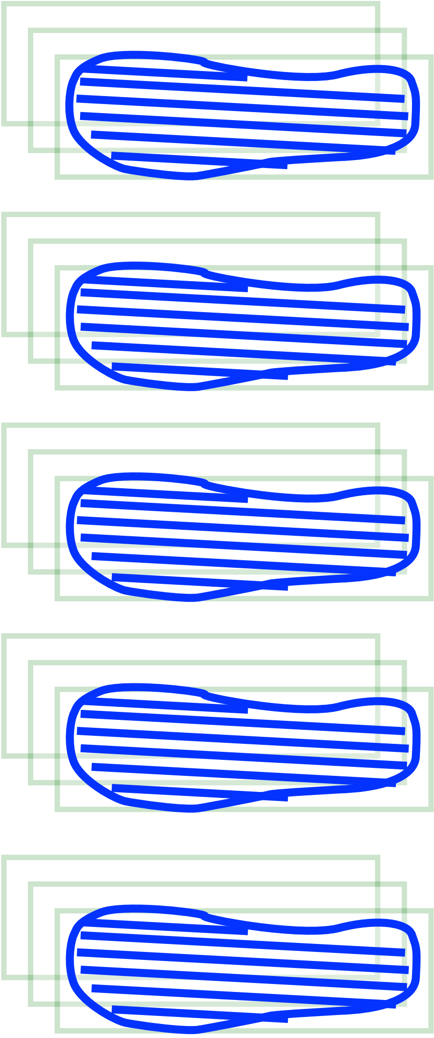




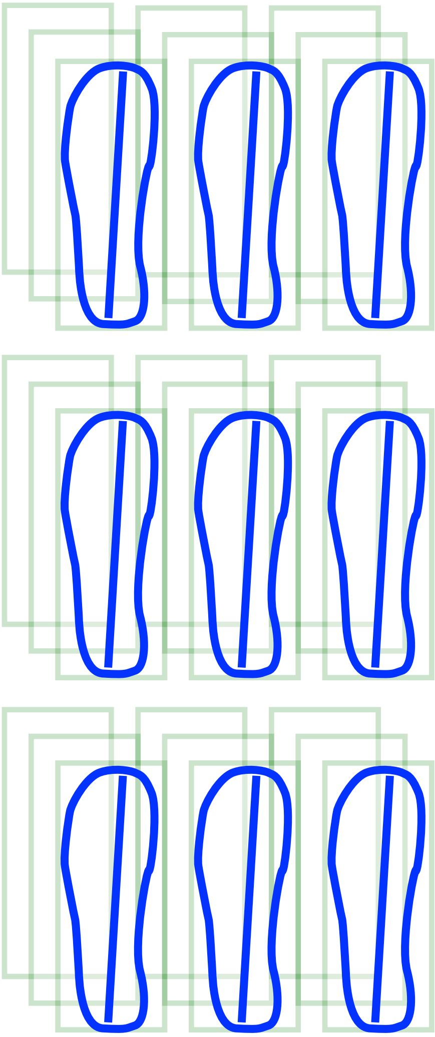
Traditional Layouts

Row

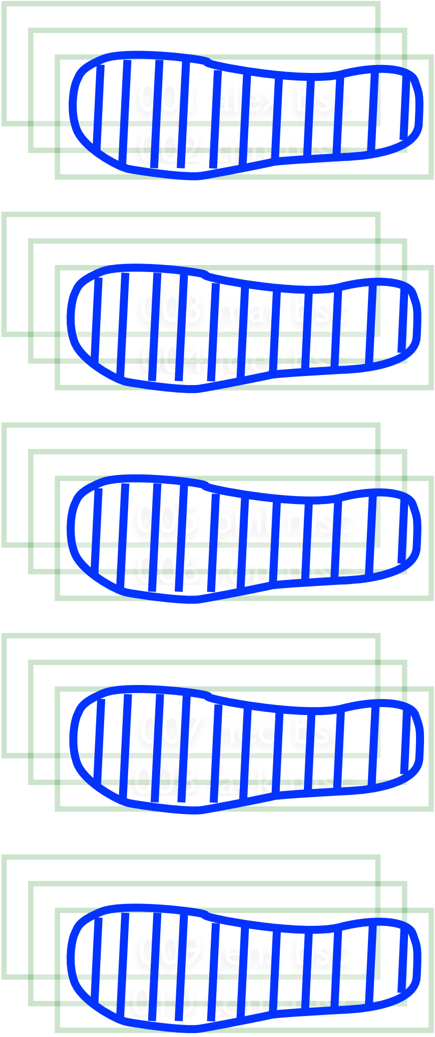
(default)



Column*



PAX**



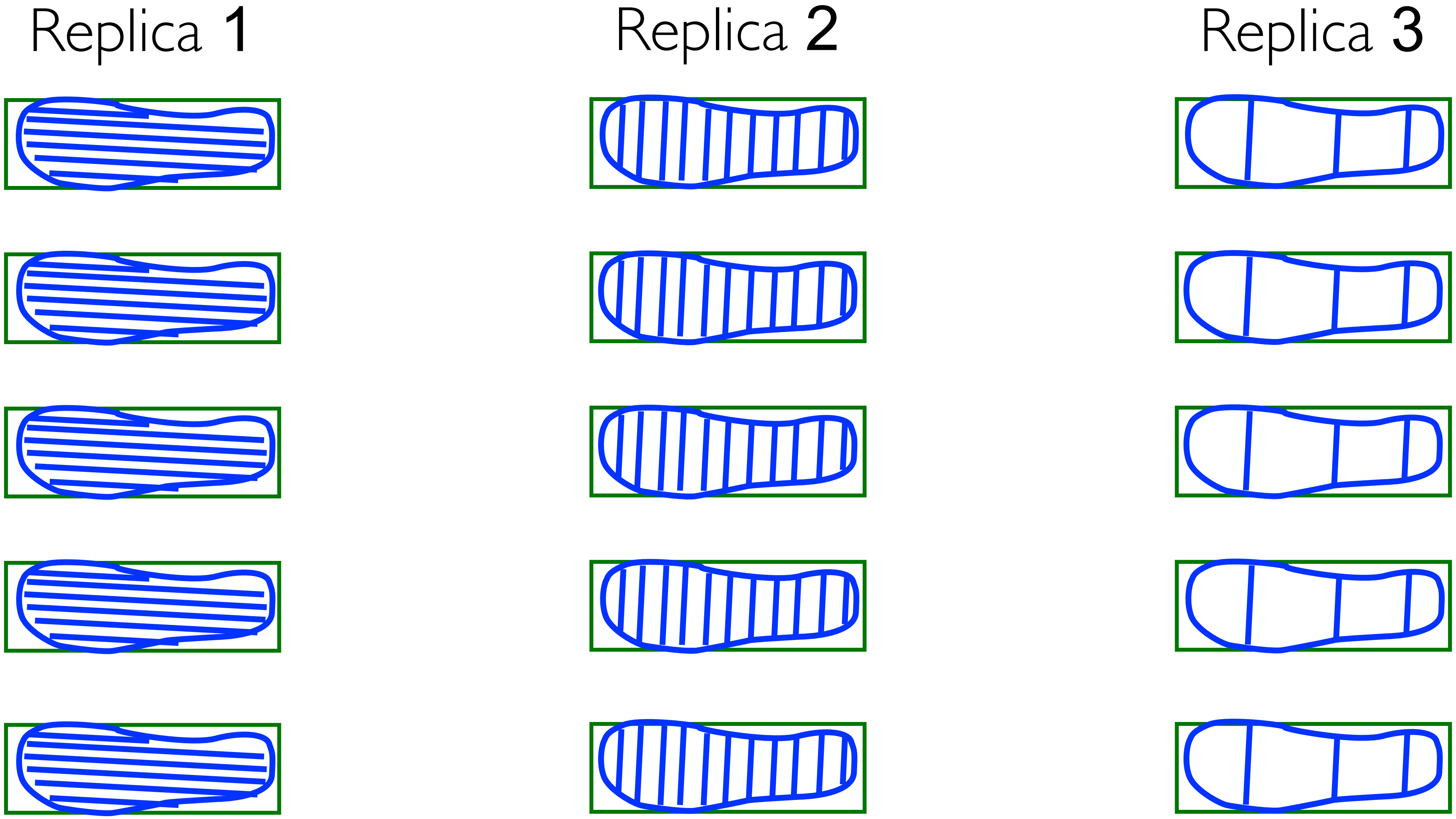
* A. Floratou et al. Column-Oriented Storage Techniques for MapReduce. PVLDB, April, 2011

** Y. He et al. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. ICDE, 2011

Traditional Layouts

	Row	Column	PAX
Non-required Reads	🟥	🟩	🟩
Network Costs	🟩	🟥	🟩
Data Block Placement	🟩	🟥	🟩
Tuple Reconstruction	🟩	🟥	🟥

Trojan Data Layouts



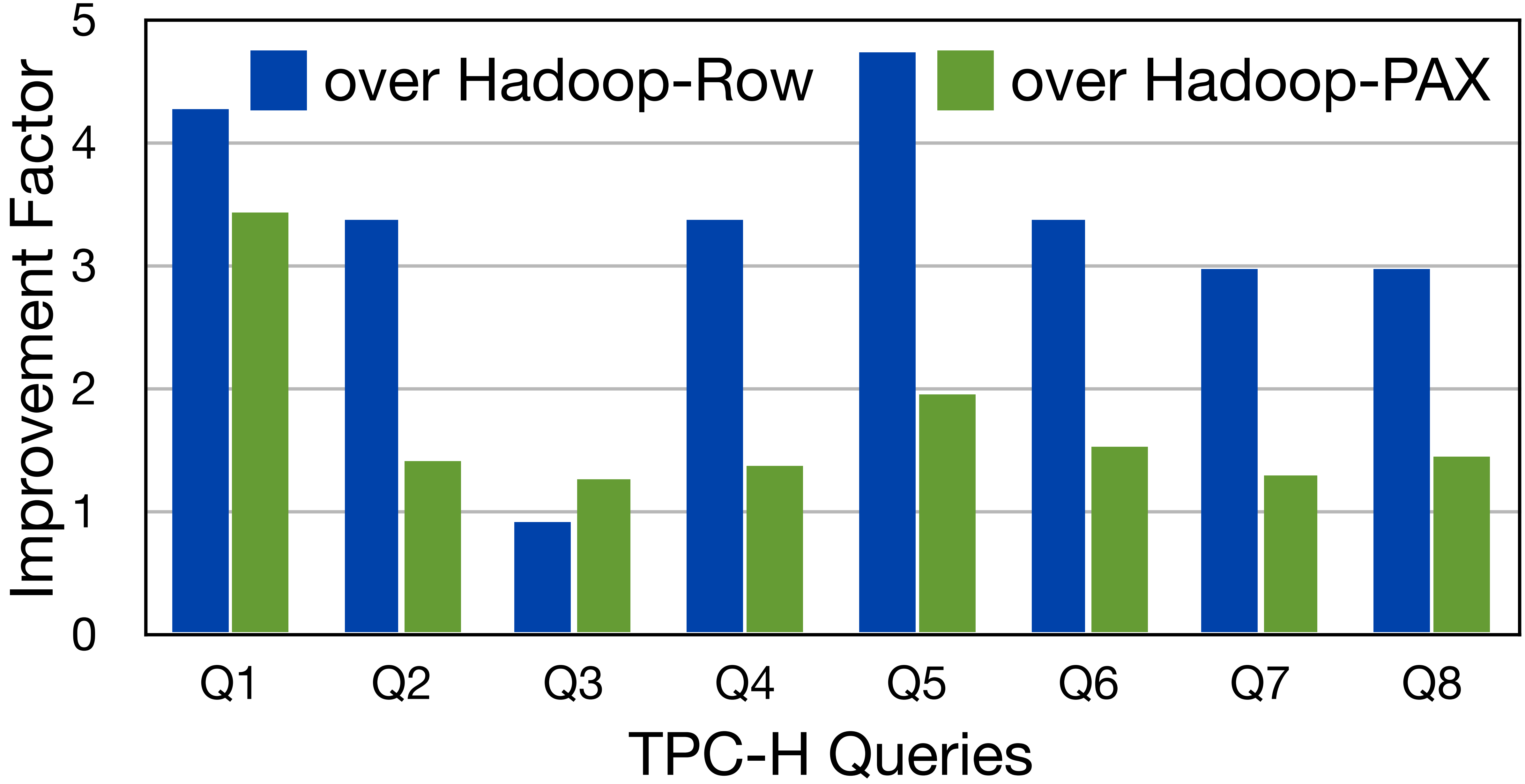
Trojan Data Layouts

	Row	Column	PAX	Trojan
Non-required Reads	🟥	🟩	🟩	🟩
Network Costs	🟩	🟥	🟩	🟩
Data Block Placement	🟩	🟥	🟩	🟩
Tuple Reconstruction	🟩	🟥	🟥	🟩

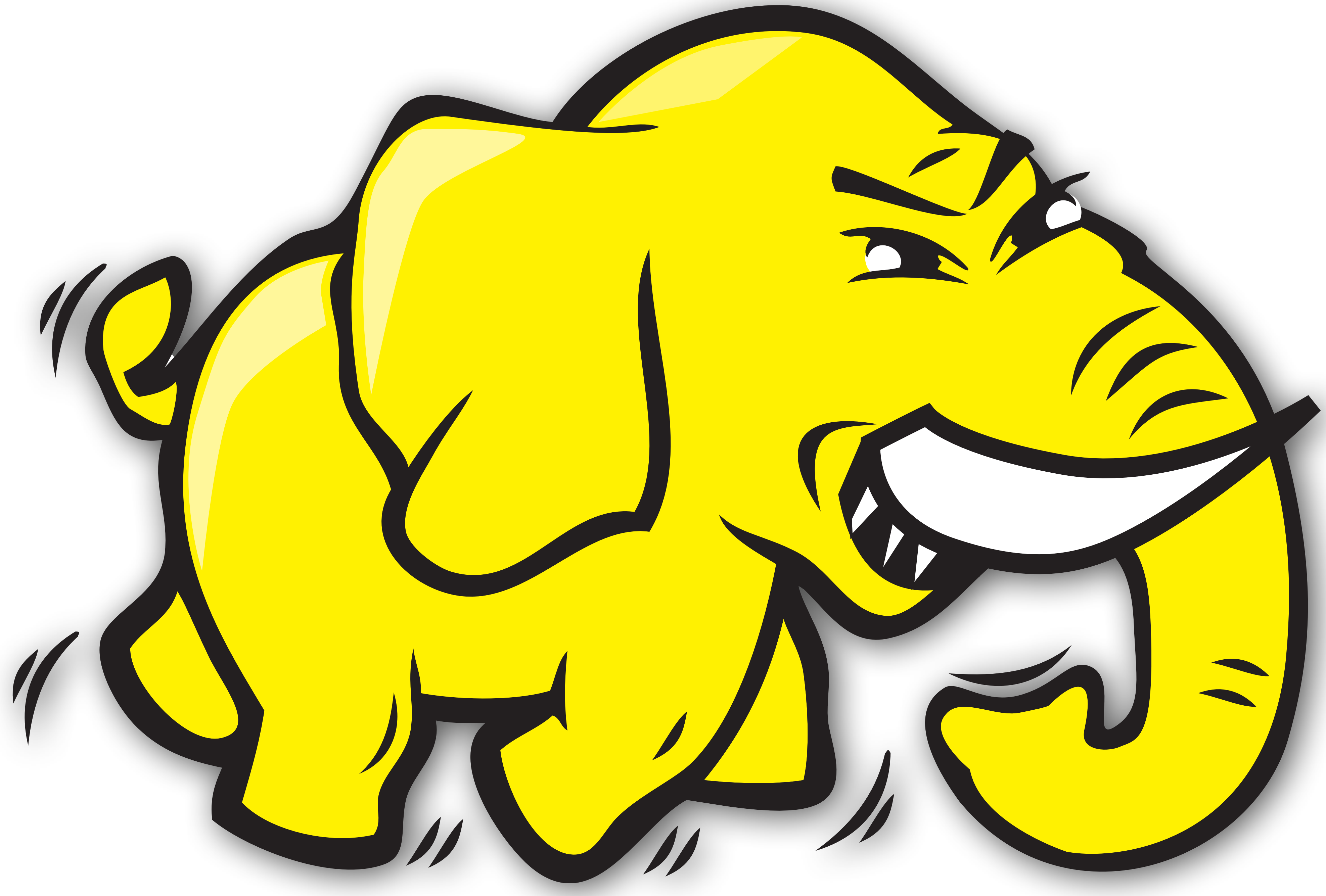
Layout Quality

	#Non-required Attributes Read	#Joins in Tuple Reconstruction
HADOOP-ROW	525	0
HADOOP-PAX	0	139
Trojan Layout	14	20

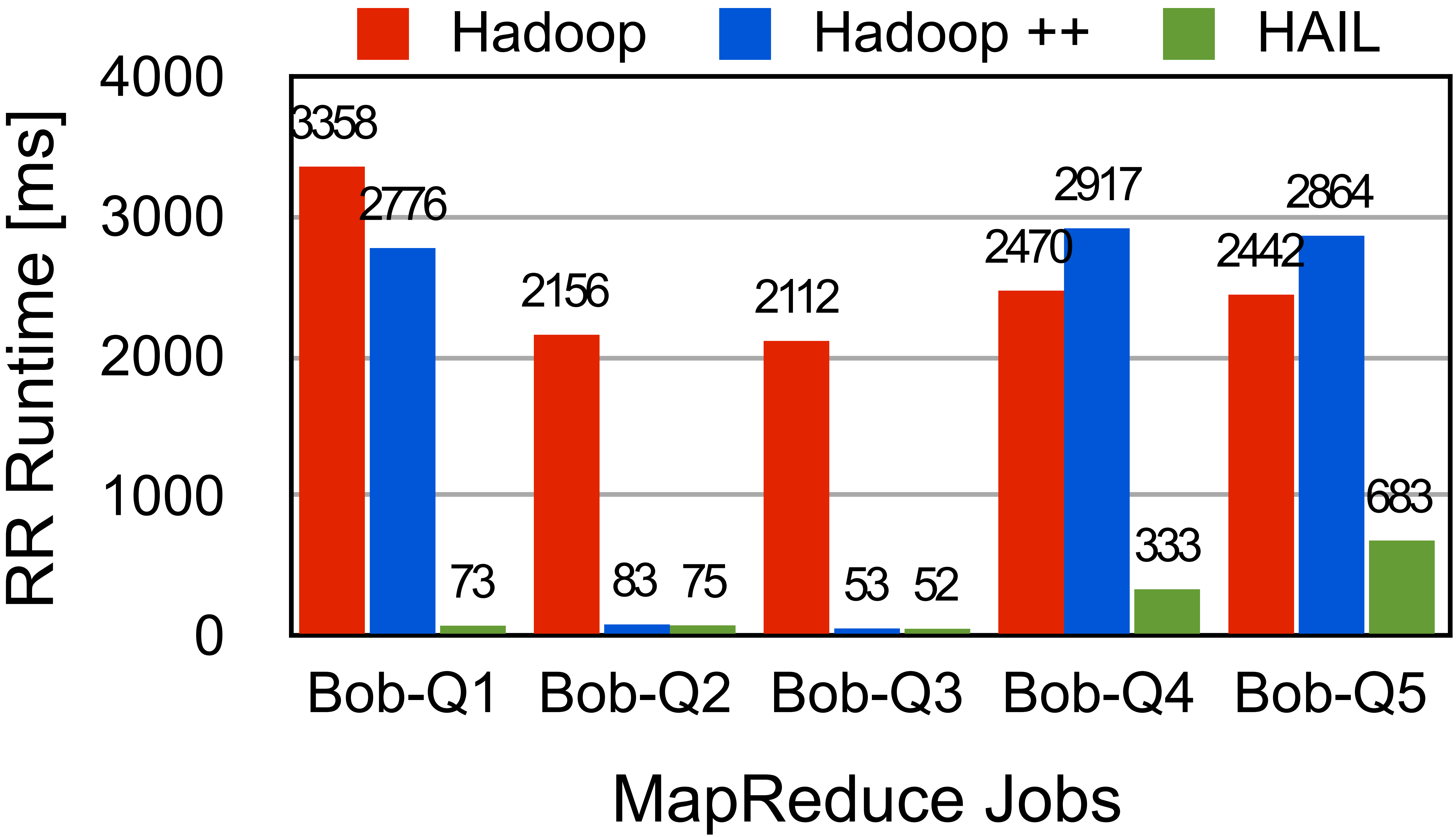
Projection Analytical Task



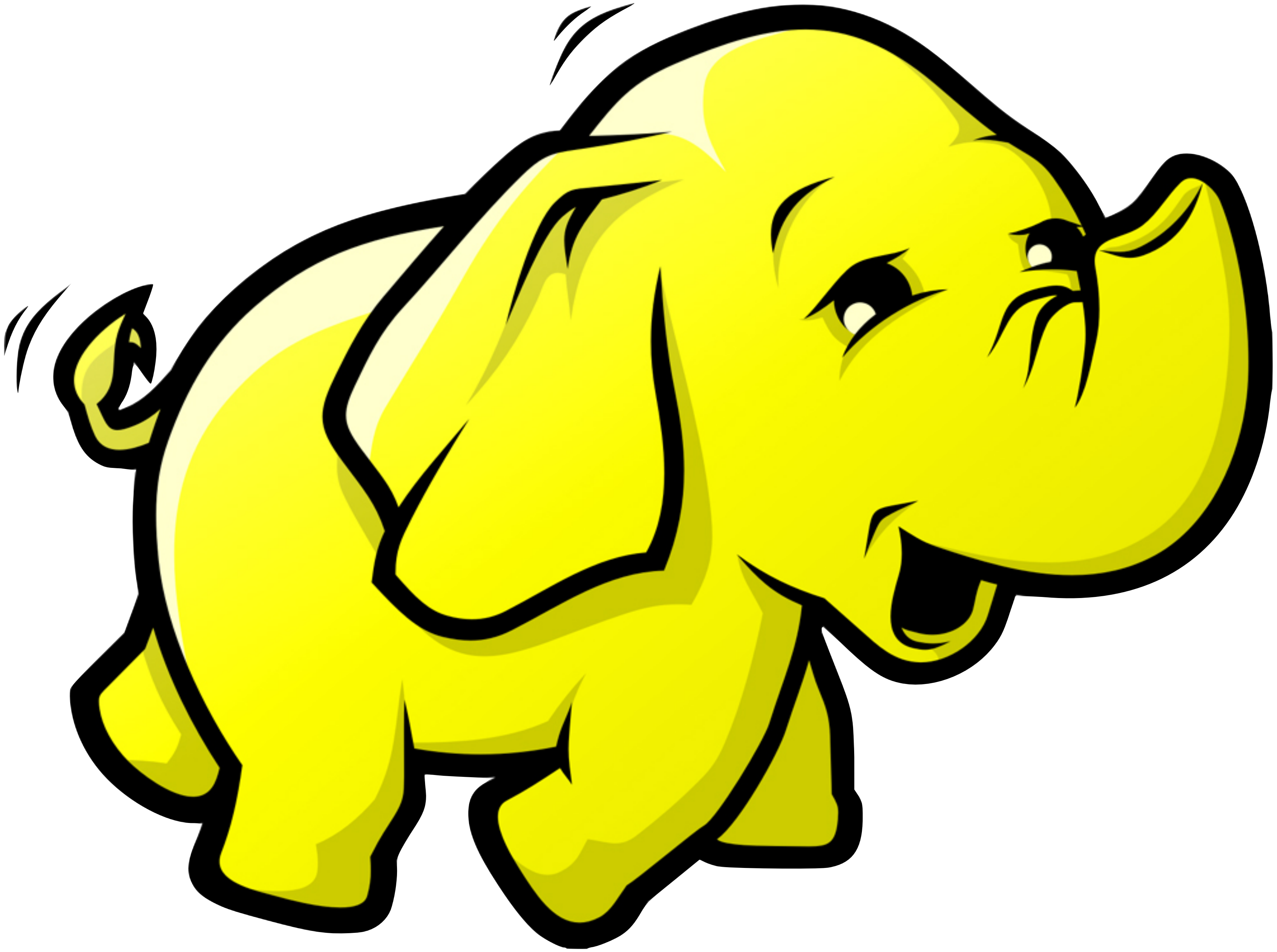
Hadoop Aggressive Indexing Library



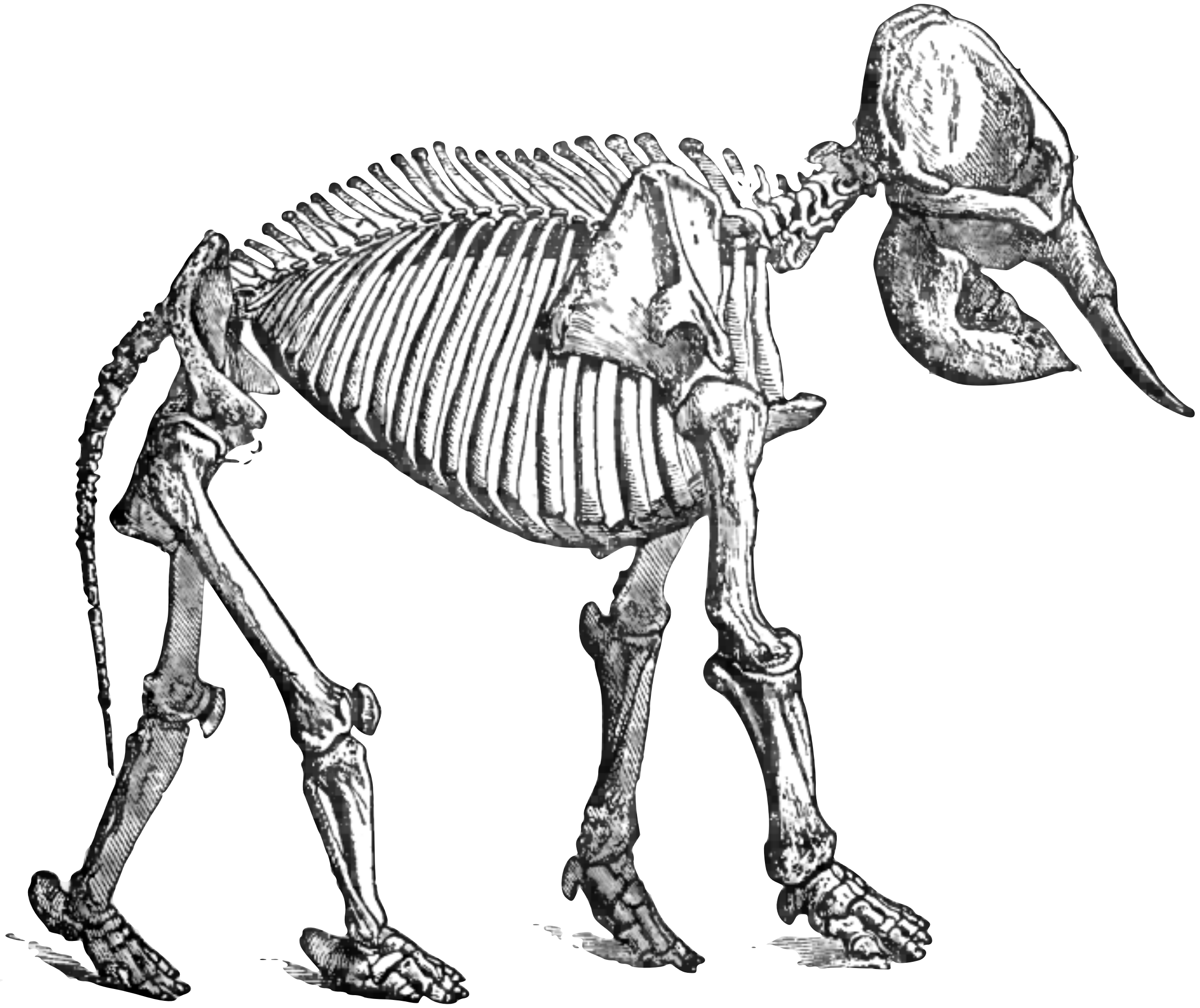
Individual Jobs: Weblog, RecordReader



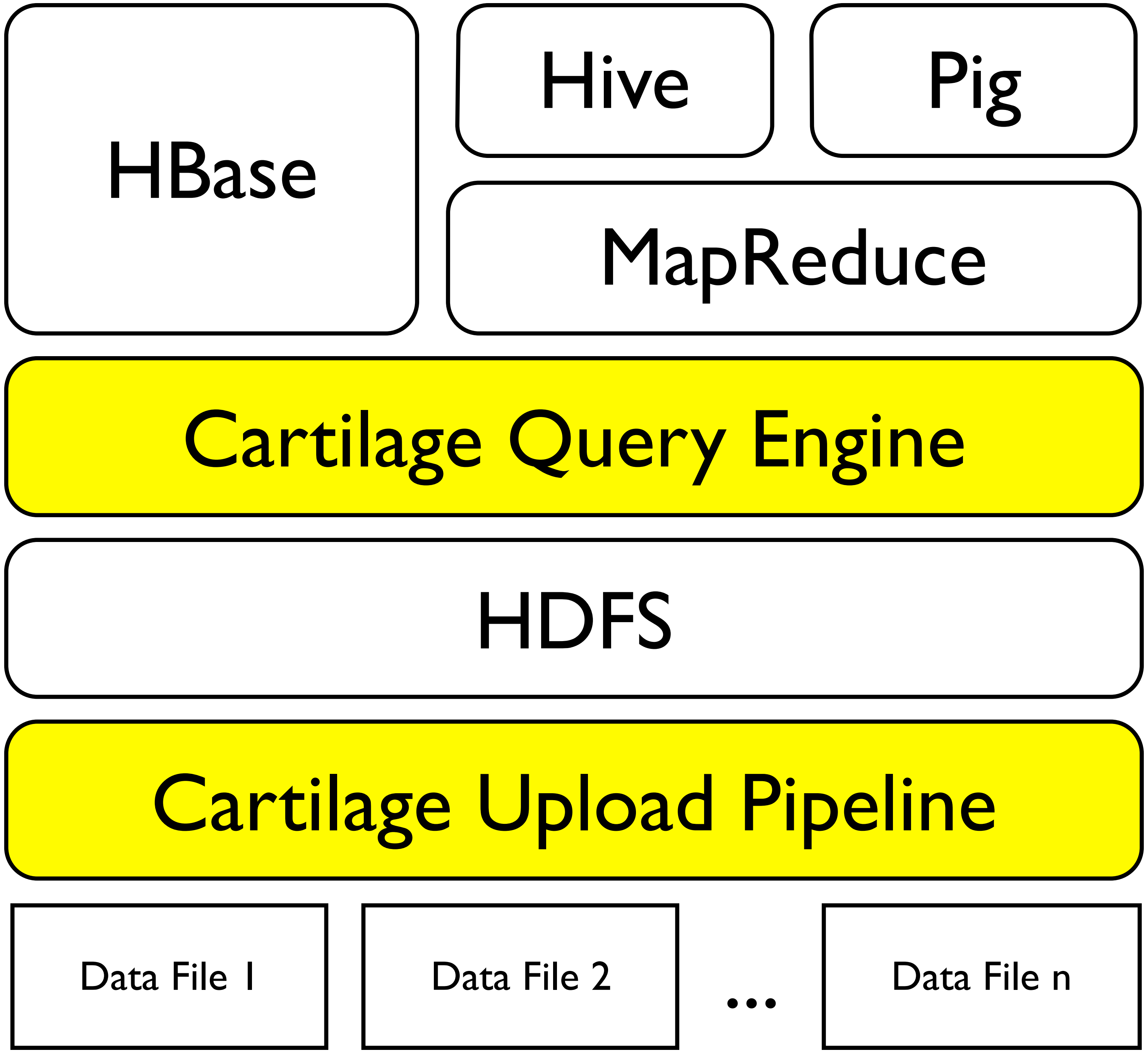
Cartilage



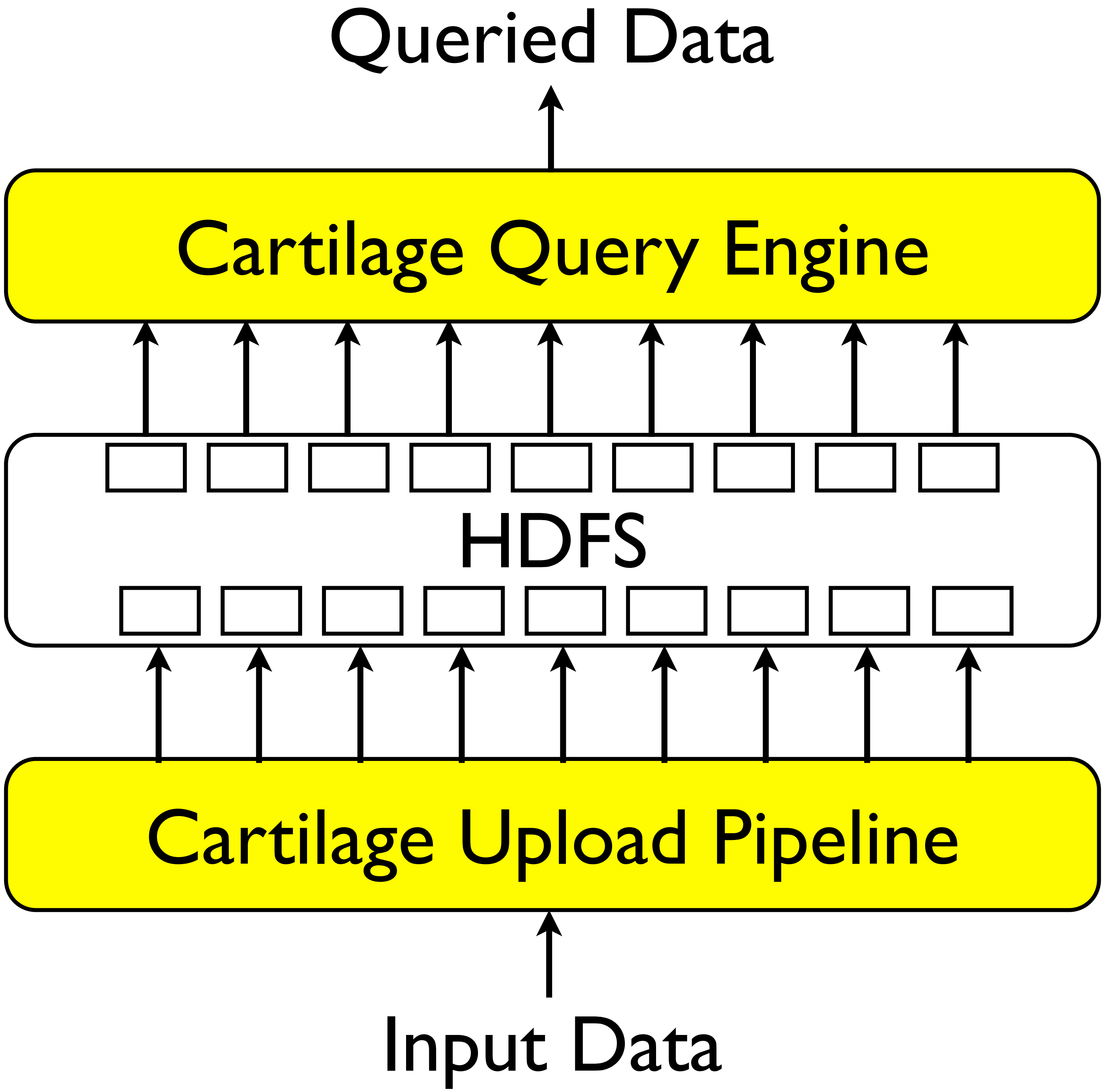
Cartilage



Hadoop Stack



Hadoop Stack



Queried Data

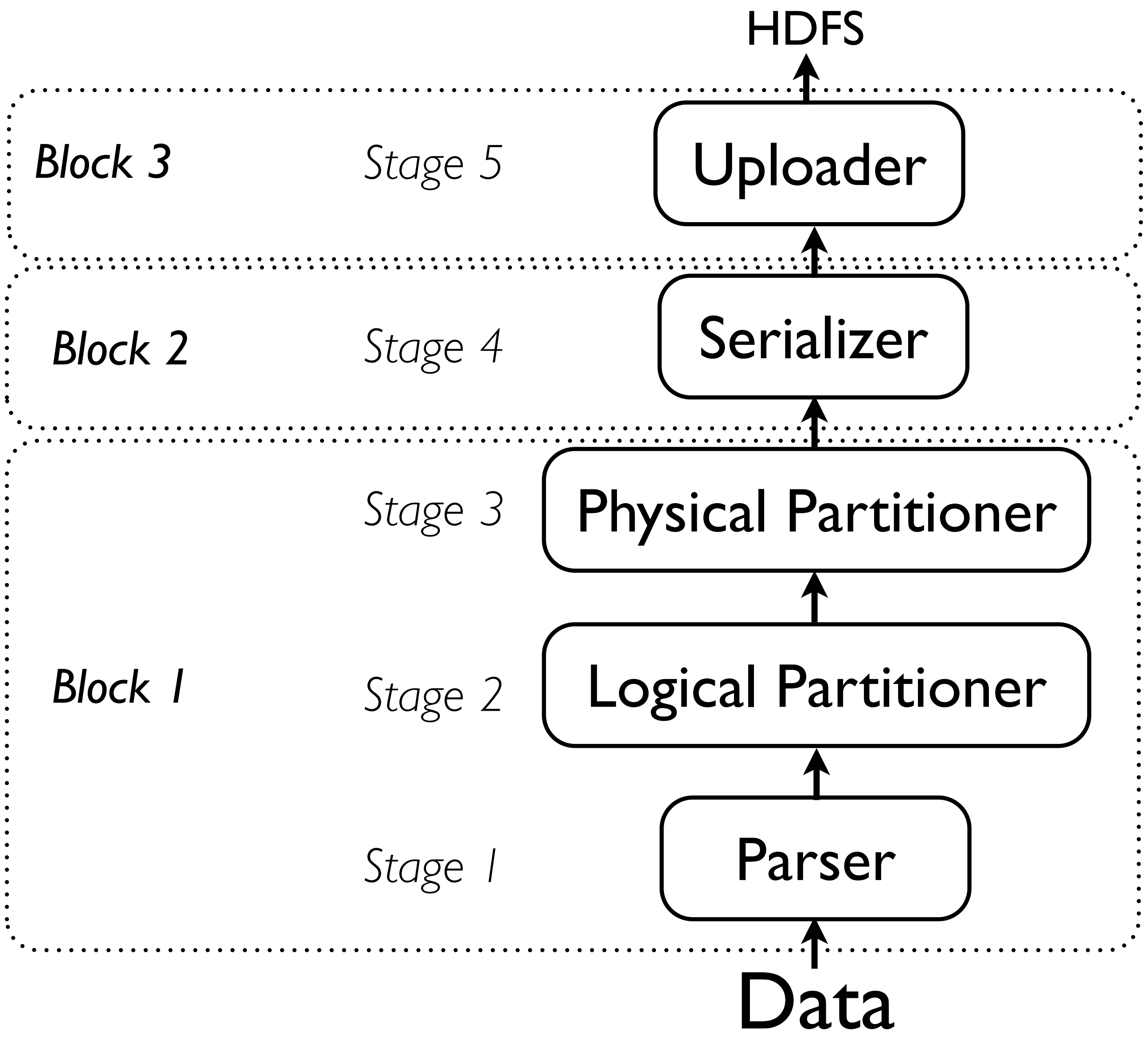
Cartilage Query Engine

HDFS

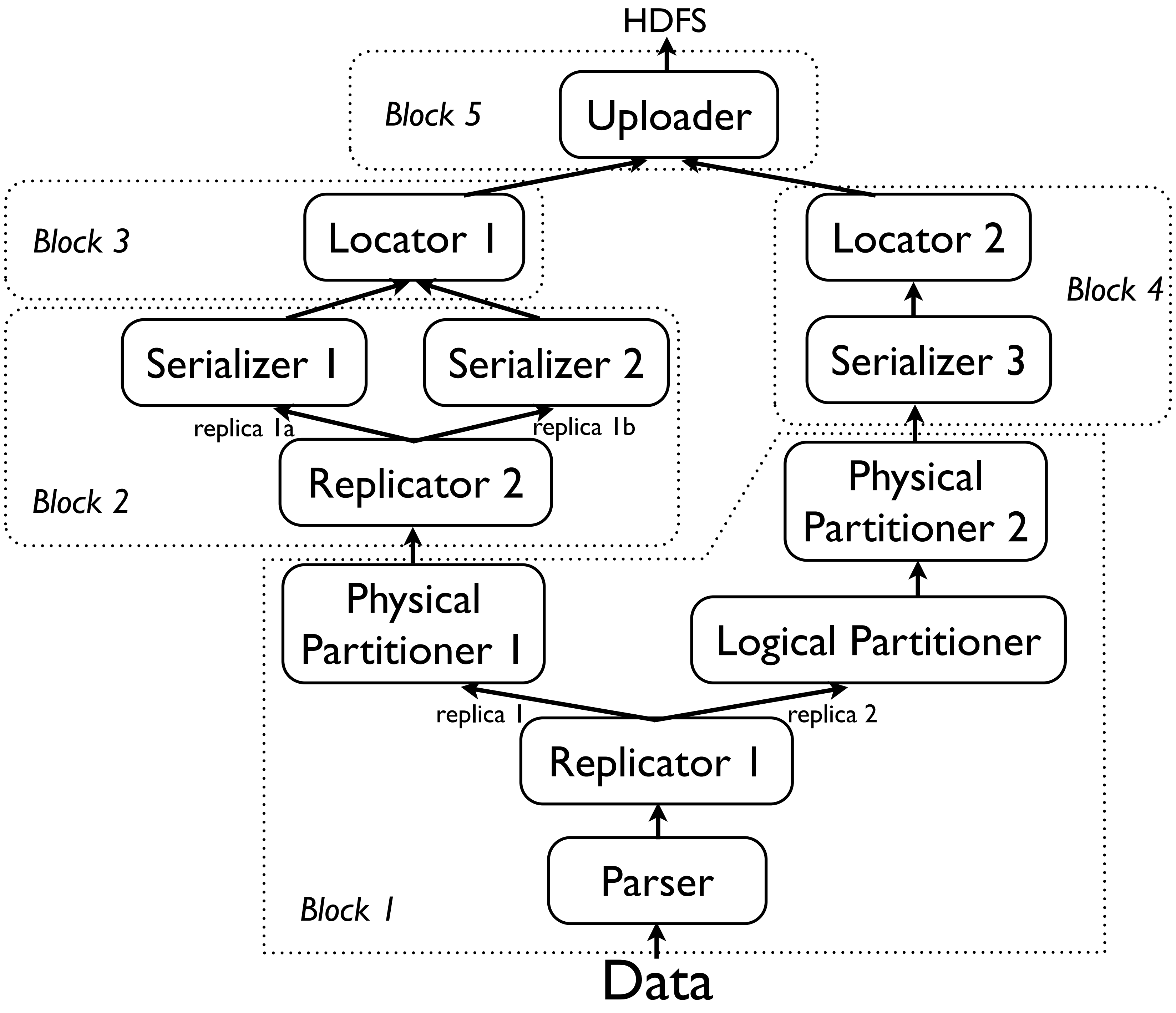
Cartilage Upload Pipeline

Input Data

Upload Plans

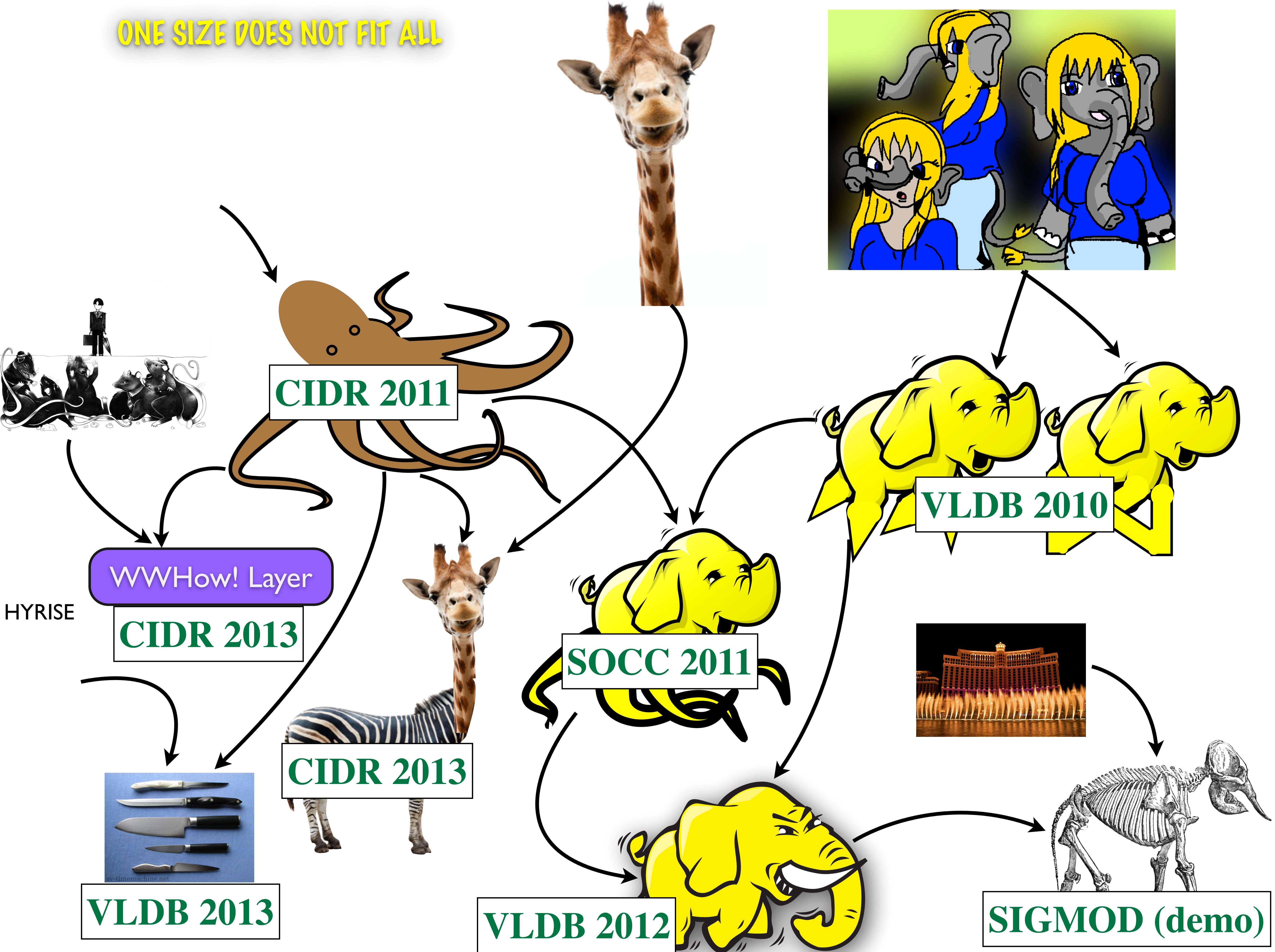


Upload Plans



Summary

ONE SIZE DOES NOT FIT ALL



CIDR 2011

VLDB 2010

WWHow! Layer

HYRISE

CIDR 2013

SOCC 2011

CIDR 2013

VLDB 2013

VLDB 2012

SIGMOD (demo)

Acknowledgements

- Jens Dittrich
- Jorge Quiane
- Felix Martin Schuhknecht
- Endre Palatinus
- Karen Khachatryan
- Stefan Richter
- Alexander Bunte
- Sam Madden
- Stefan Richter
- Stefan Schuh
- Joerg Schad
- Yagiz Kargin
- Vinay Setty
- Vladimir Pavlov