Lightning Fast & Space Efficient Inequality Joins

Data Analytics Team @QCRI

معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute

KAUST

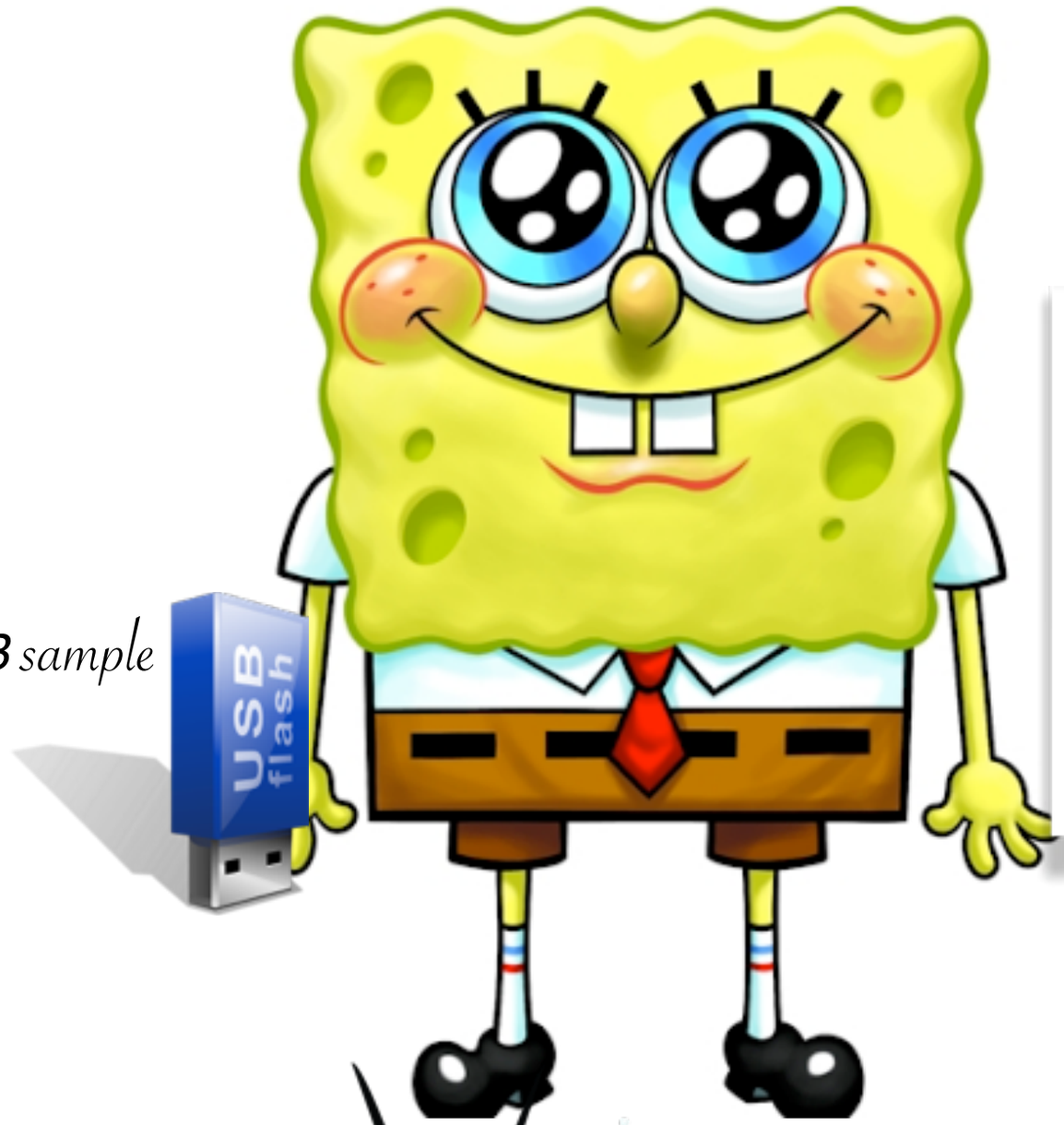
ASU Arizona State University

Once upon a time

at QCRI

Bob@QCRI

Bob@QCRI



1MB sample

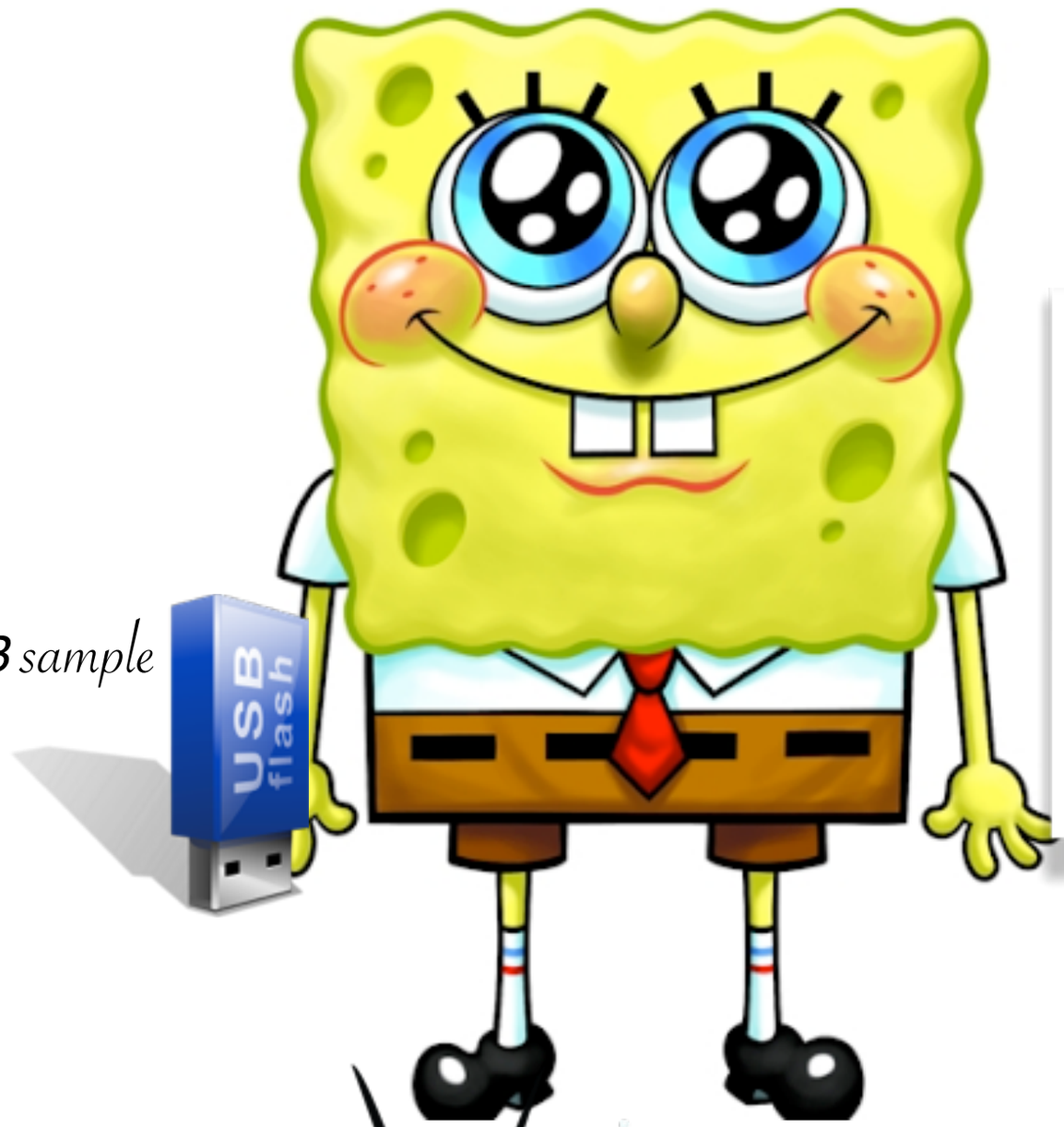
Data quality rule

Q1:

SELECT e.*id*
FROM *Employees e, Managers m*
WHERE e.*ssn = m.ssn*

NADEEF
RHEEM

Bob@QCRI



1MB sample

Data quality rule

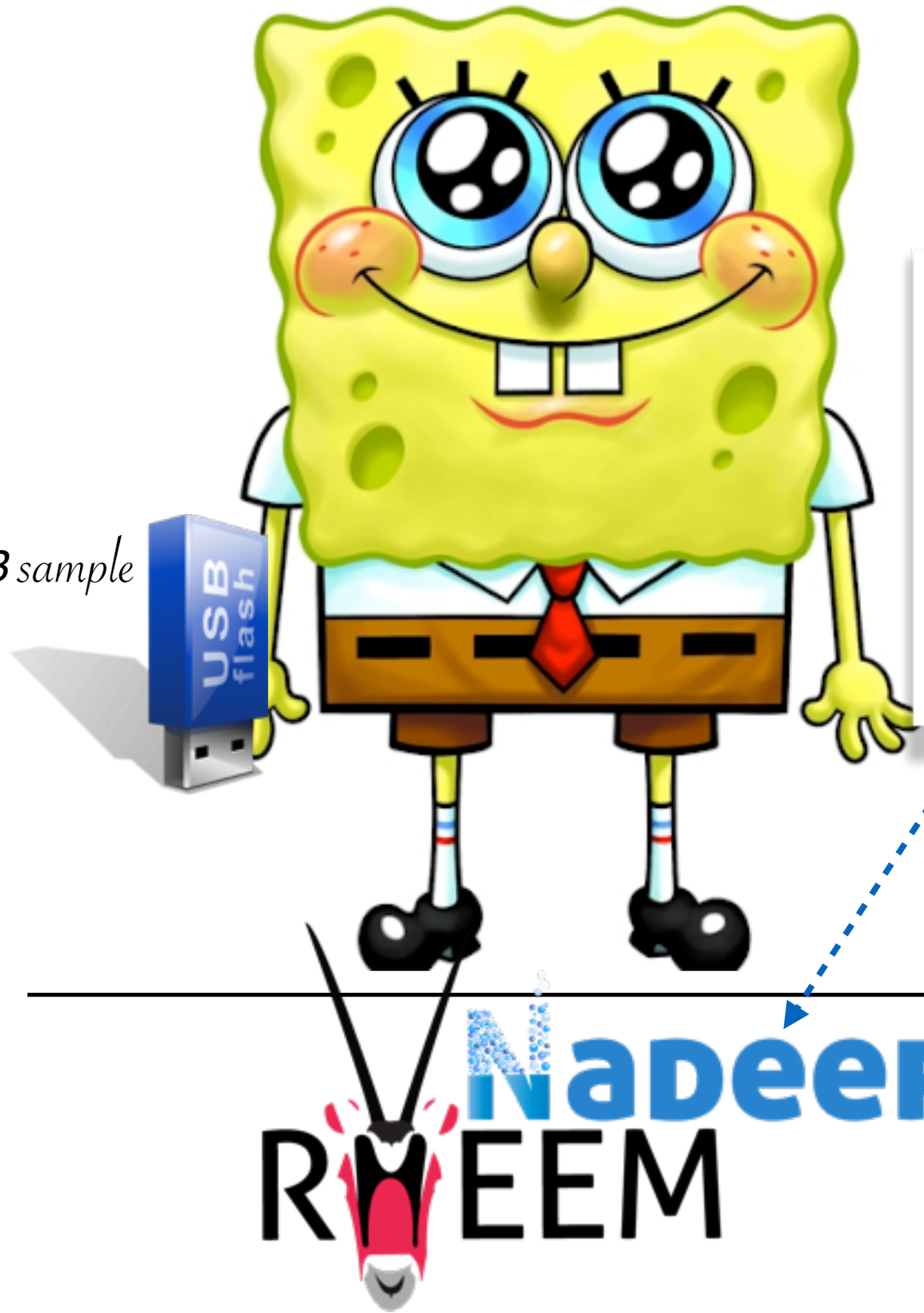
Q1:

SELECT e.*id*
FROM *Employees e, Managers m*
WHERE *e.ssn = m.ssn*

equi-join

NADEEF
RHEEM

Bob@QCRI



1MB sample

Data quality rule

Q1:

SELECT e.*id*
FROM *Employees e, Managers m*
WHERE *e.ssn = m.ssn*

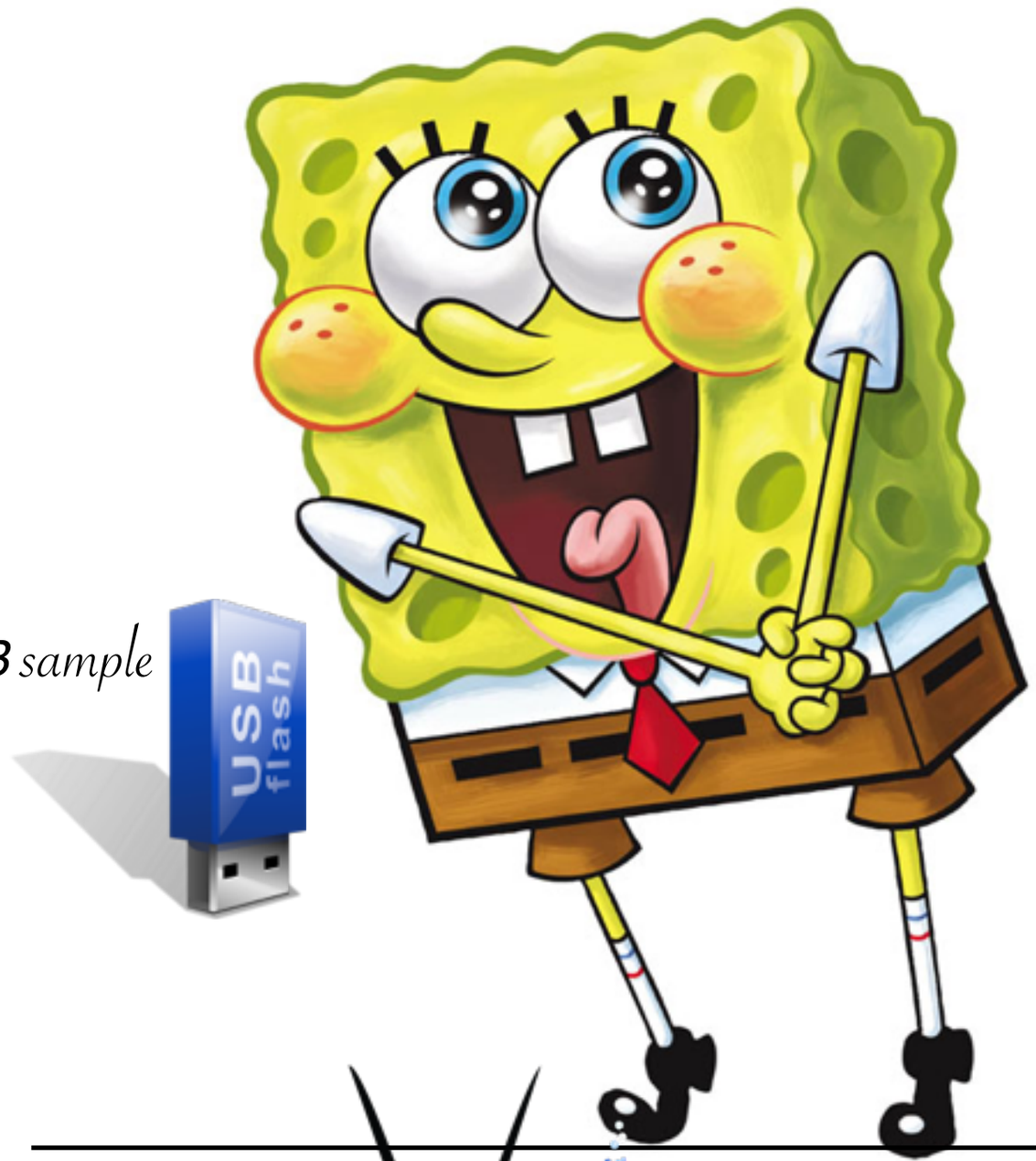
equi-join

results in few seconds!

NADEEF

RHEEM

Bob@QCRI



NADEEF

RHEEM

Bob@QCRI

1MB sample

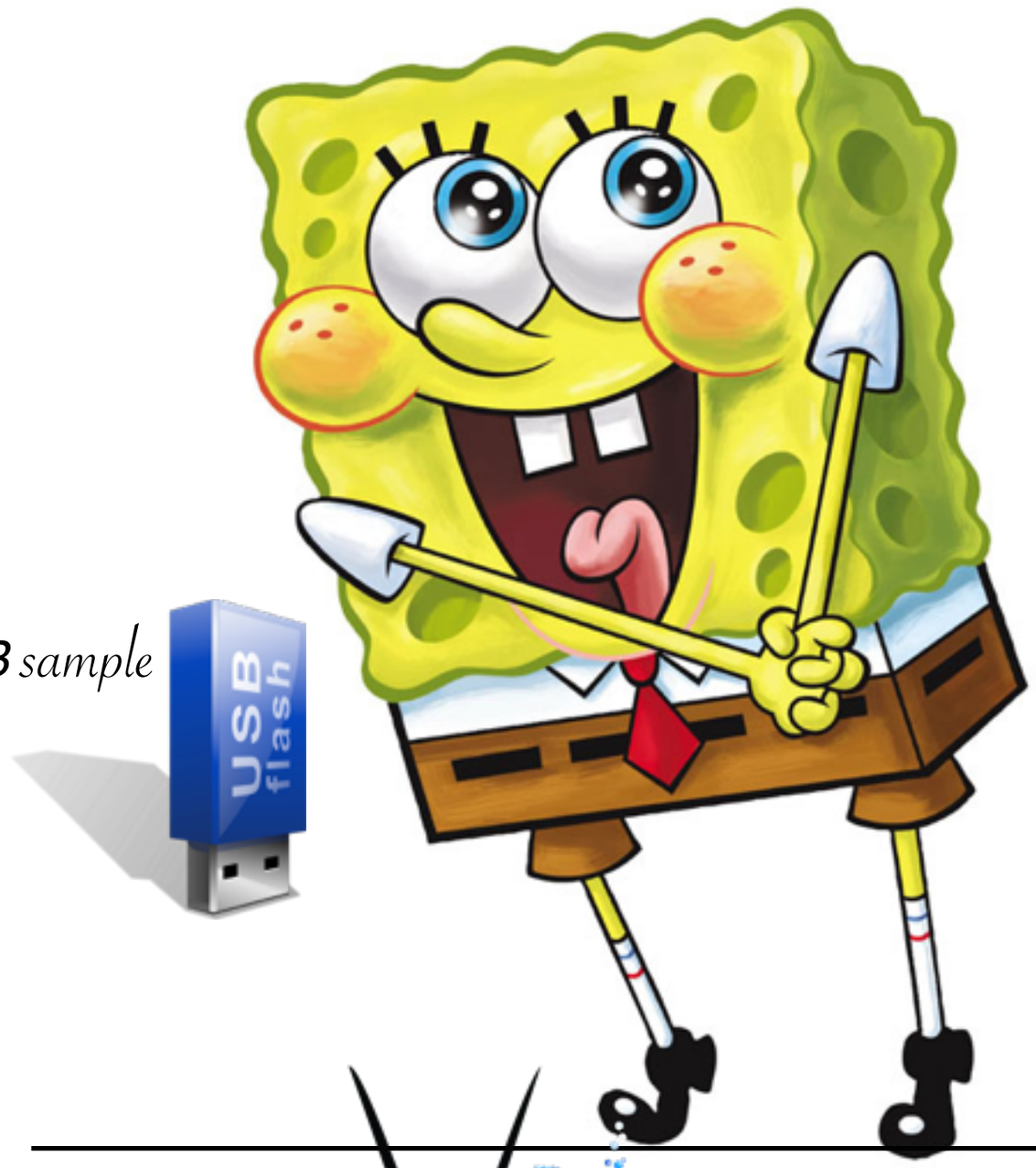
Data quality rule

Q2:

SELECT *e1.id, e2.id*
FROM *Employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

NADEEF
RHEEM

Bob@QCRI



Data quality rule

Q2:

SELECT *e1.id, e2.id*
FROM *Employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

nonequi-join

NADEEF

RHEEM

Bob@QCRI

1MB sample

Data quality rule

Q2:

```
SELECT e1.id, e2.id
FROM Employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

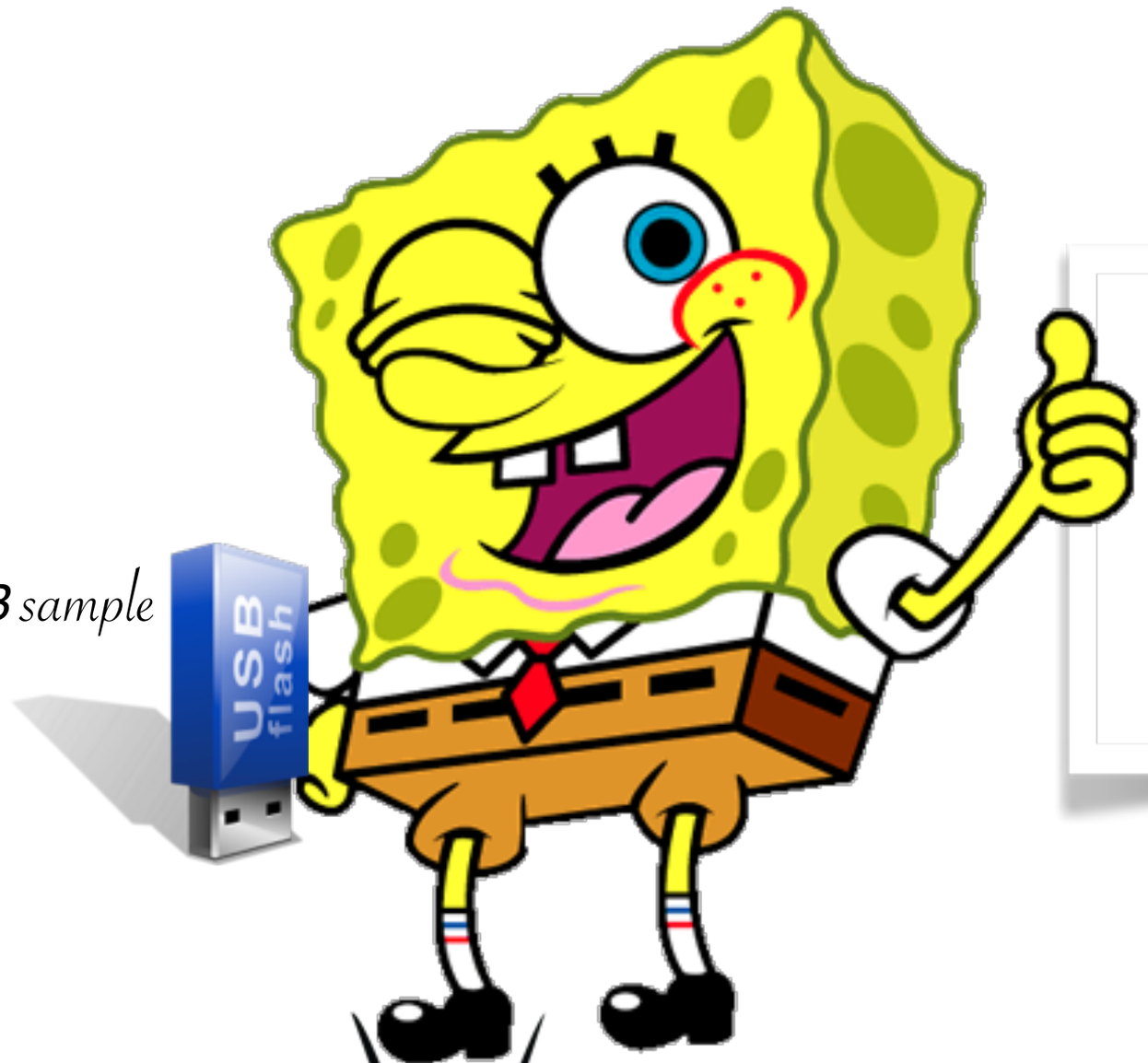
nonequi-join

no results after 1hr!

NADEEF

RHEEM

Bob@QCRI



Data quality rule

Q2:

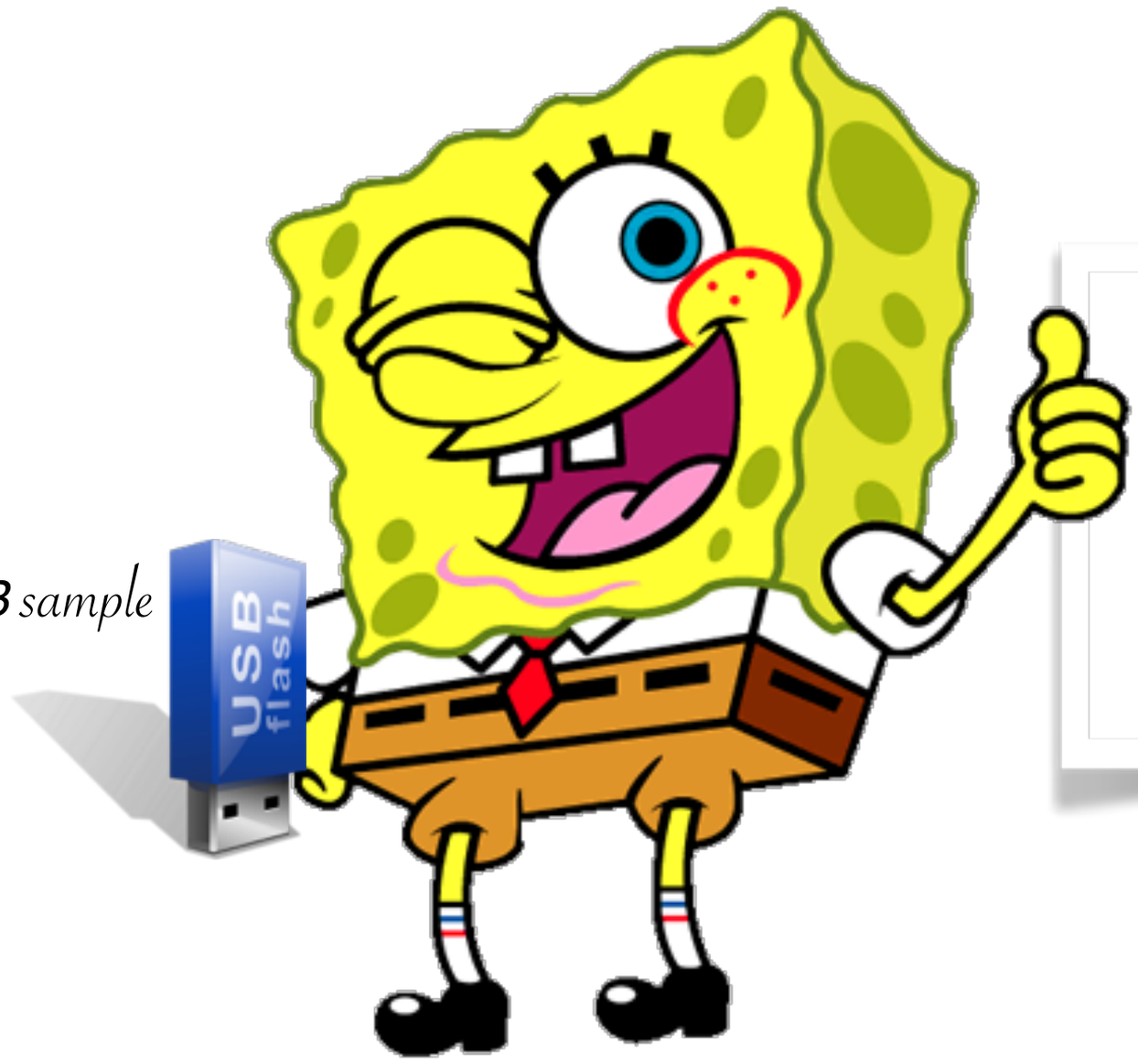
SELECT *e1.id, e2.id*
FROM *Employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

nonequi-join

NADEEF

RHEEM

Bob@QCRI



Data quality rule

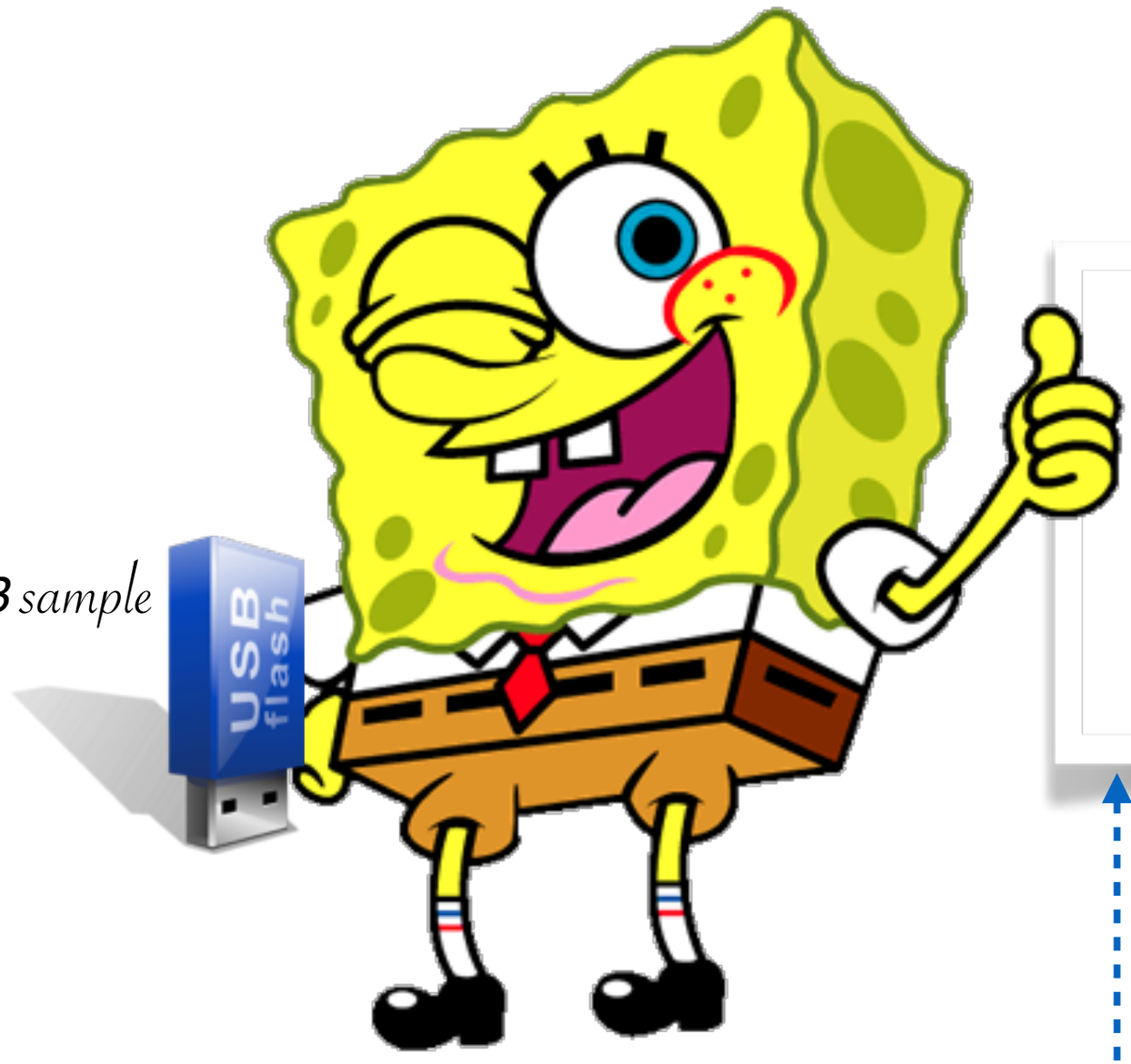
Q2:

SELECT *e1.id, e2.id*
FROM *Employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

nonequi-join

System-X

Bob@QCRI



Data quality rule

Q2:

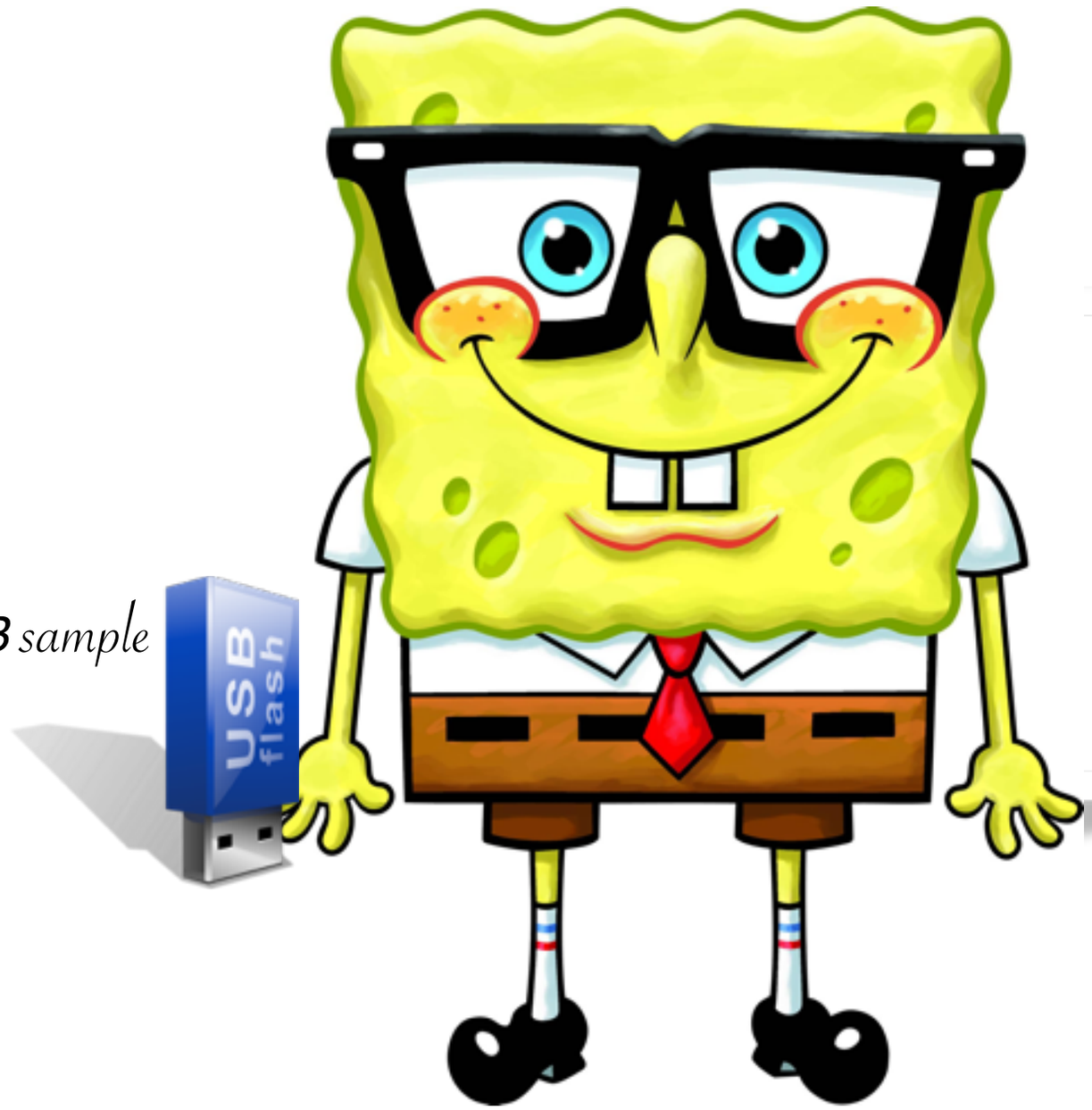
```
SELECT e1.id, e2.id
FROM Employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

nonequi-join

no results after 3hr!

System-X

Bob@QCRI



1MB sample

Data quality rule

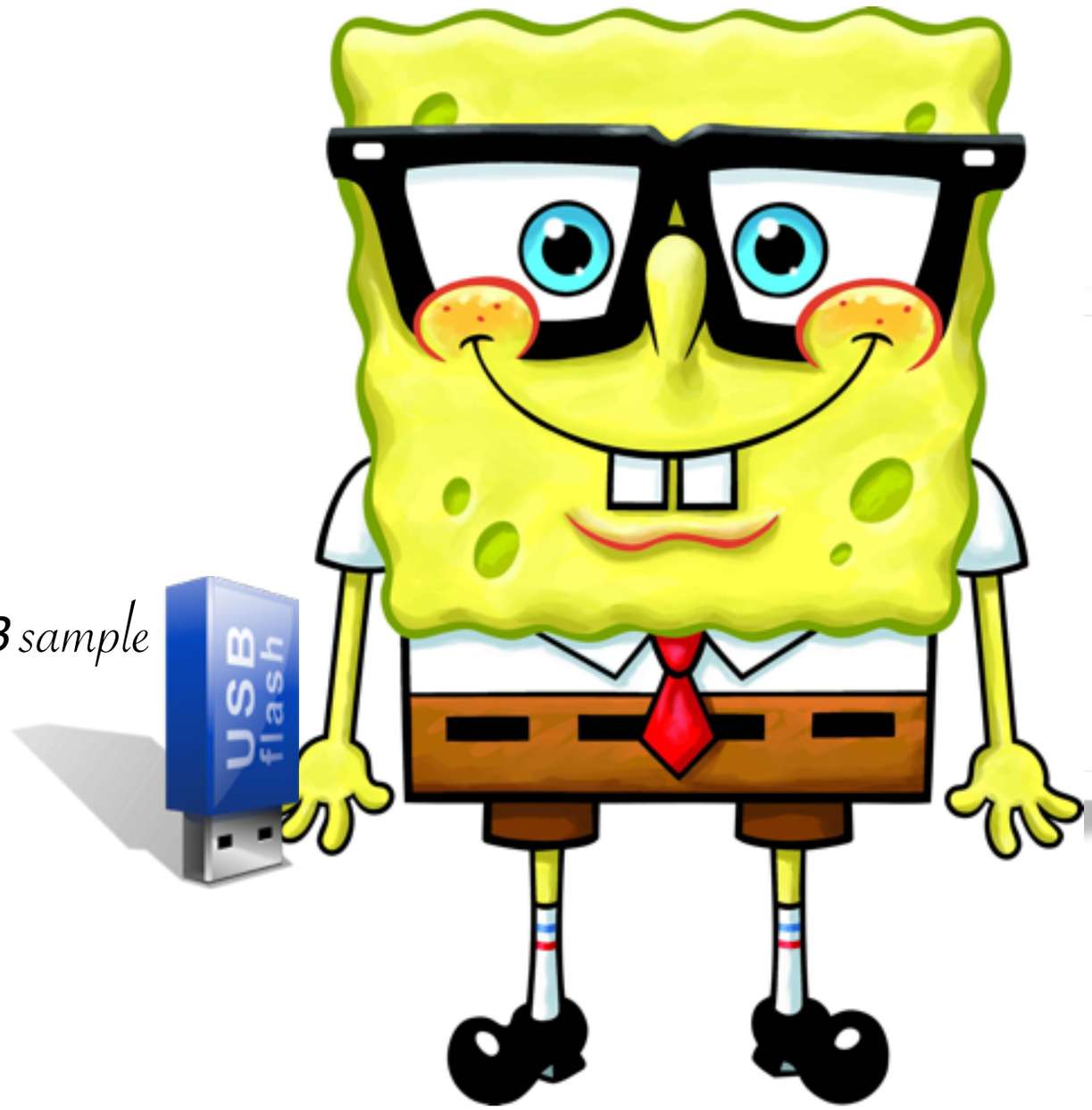
Q2:

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

nonequi-join

System-X

Bob@QCRI



1MB sample

Data quality rule

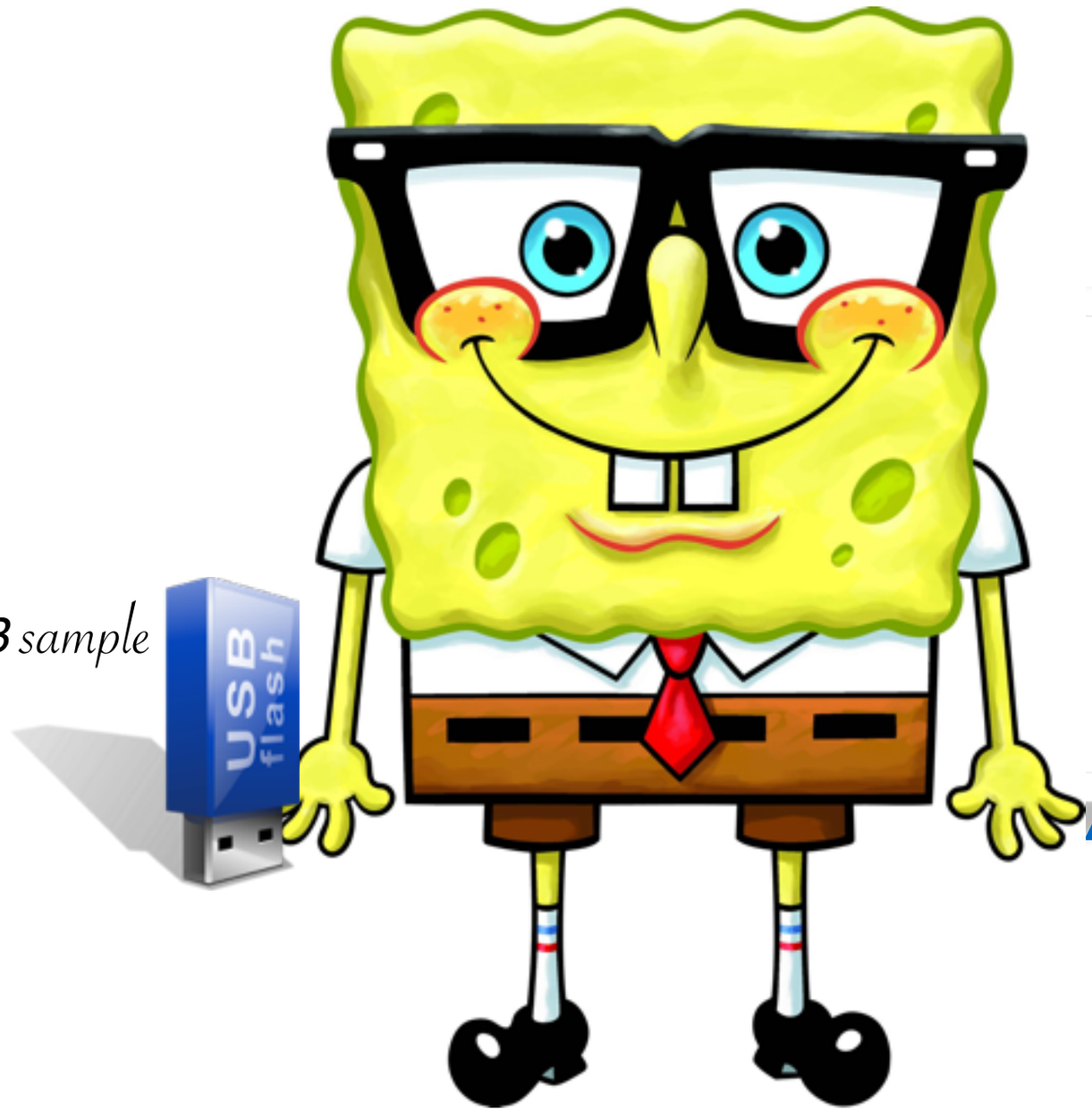
Q2:

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

nonequi-join

DBMS-X

Bob@QCRI



Data quality rule

Q2:

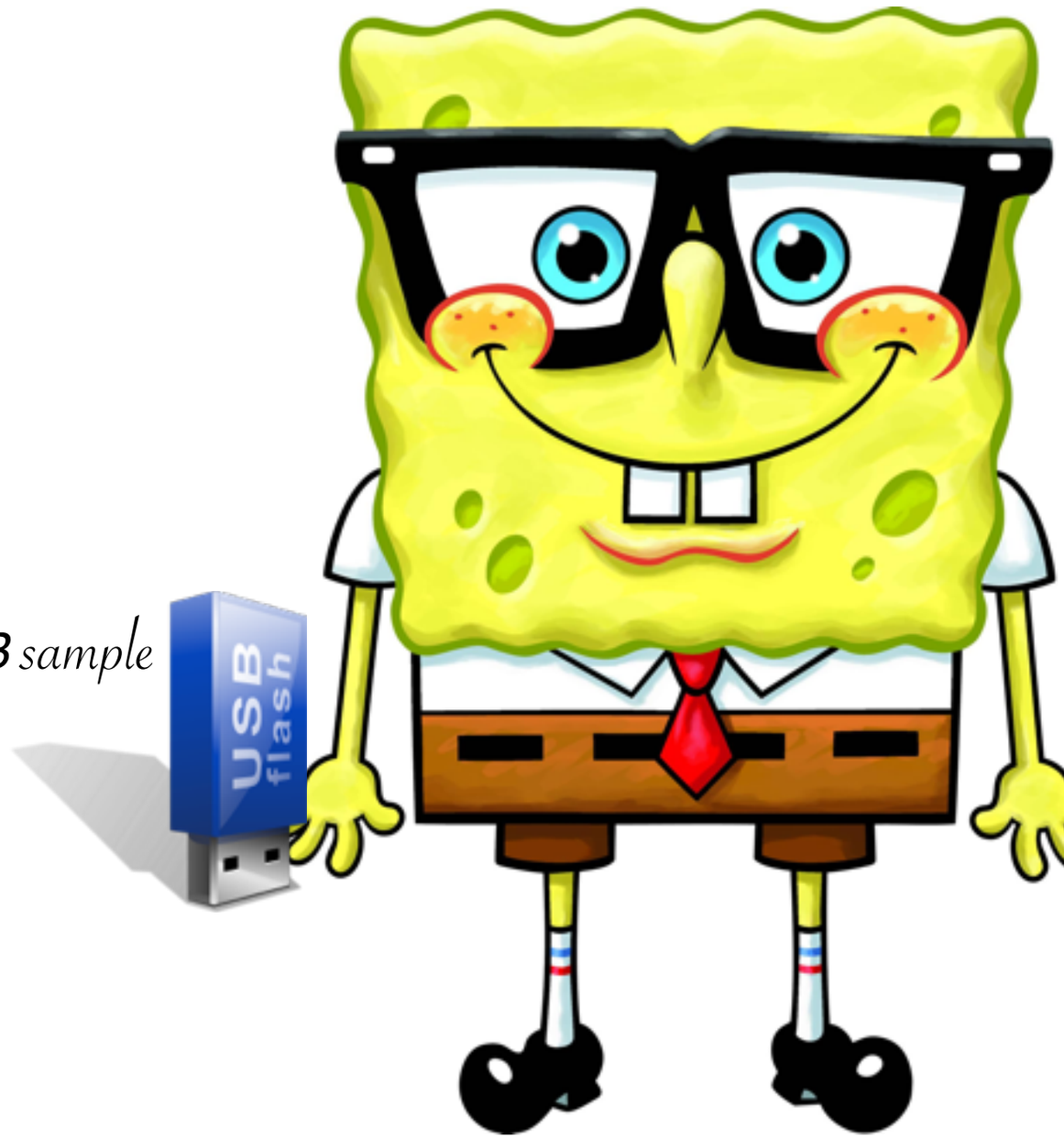
```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

nonequi-join

results after 2hr!

DBMS-X

Bob@QCRI



1MB sample

Data quality rule

Q2:

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
AND e1.tax < e2.tax;
```

nonequi-join

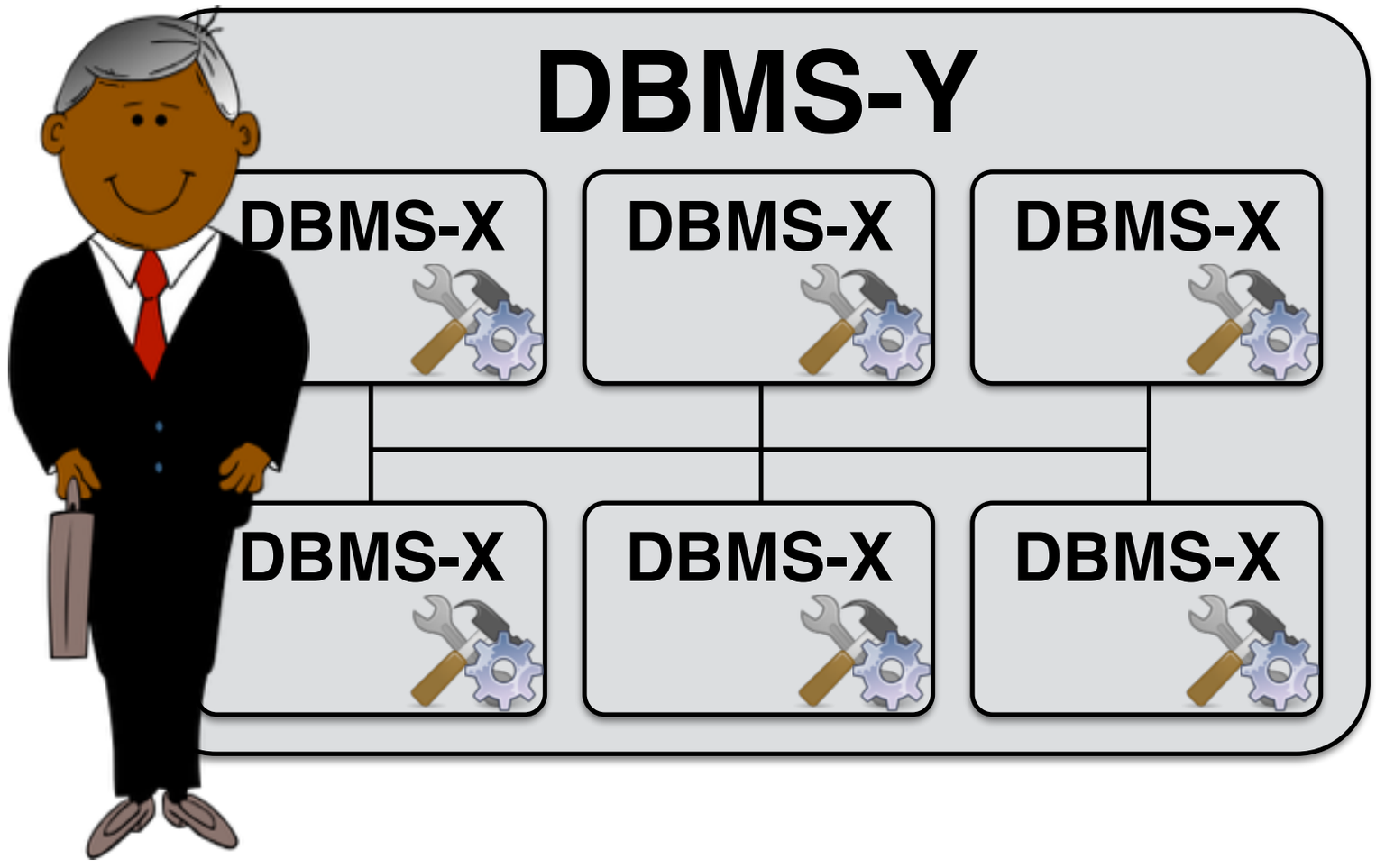
results after 2hr!

DBMS-X

$\times + \sigma$

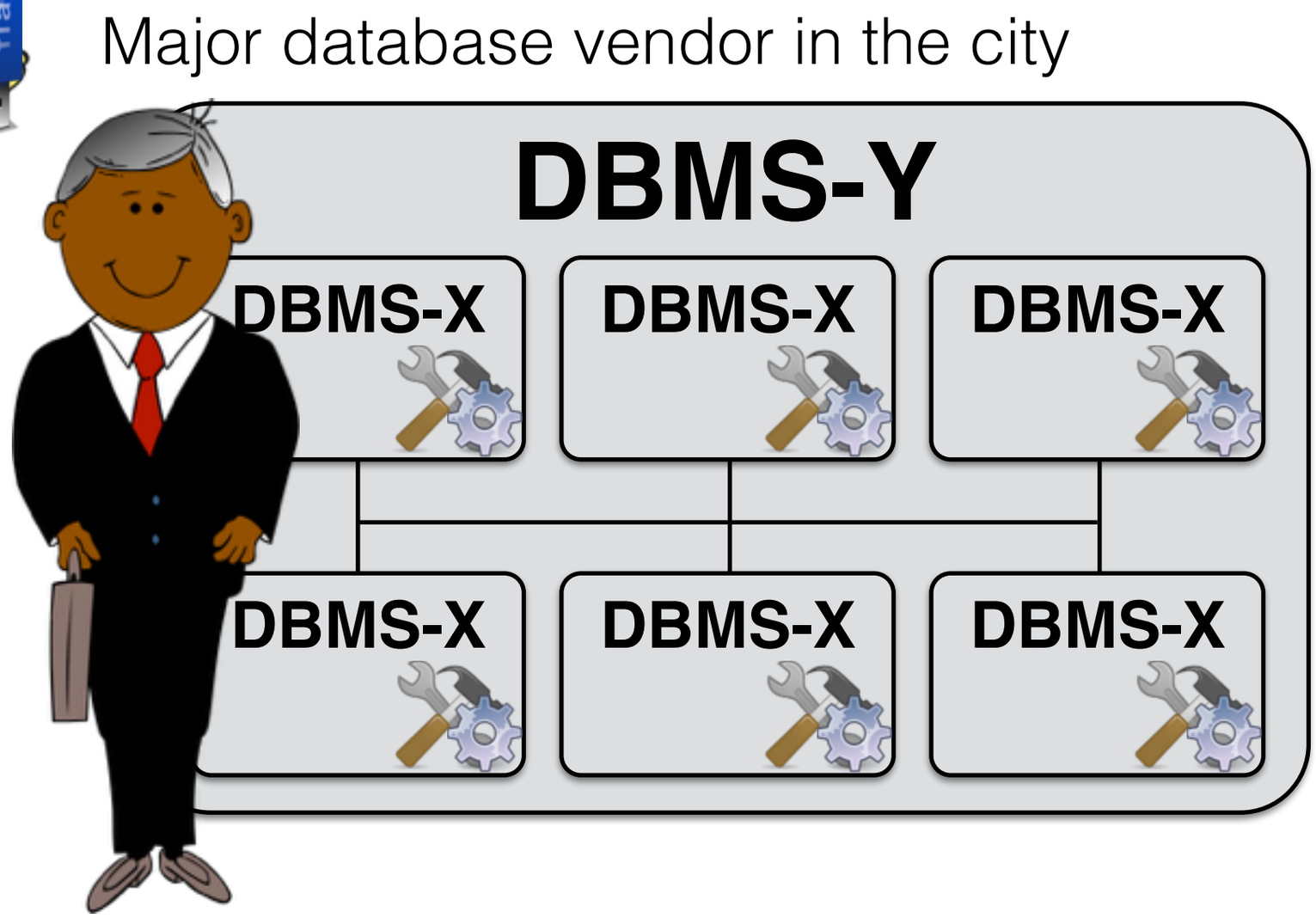
Bob@QCRI

Major database vendor in the city



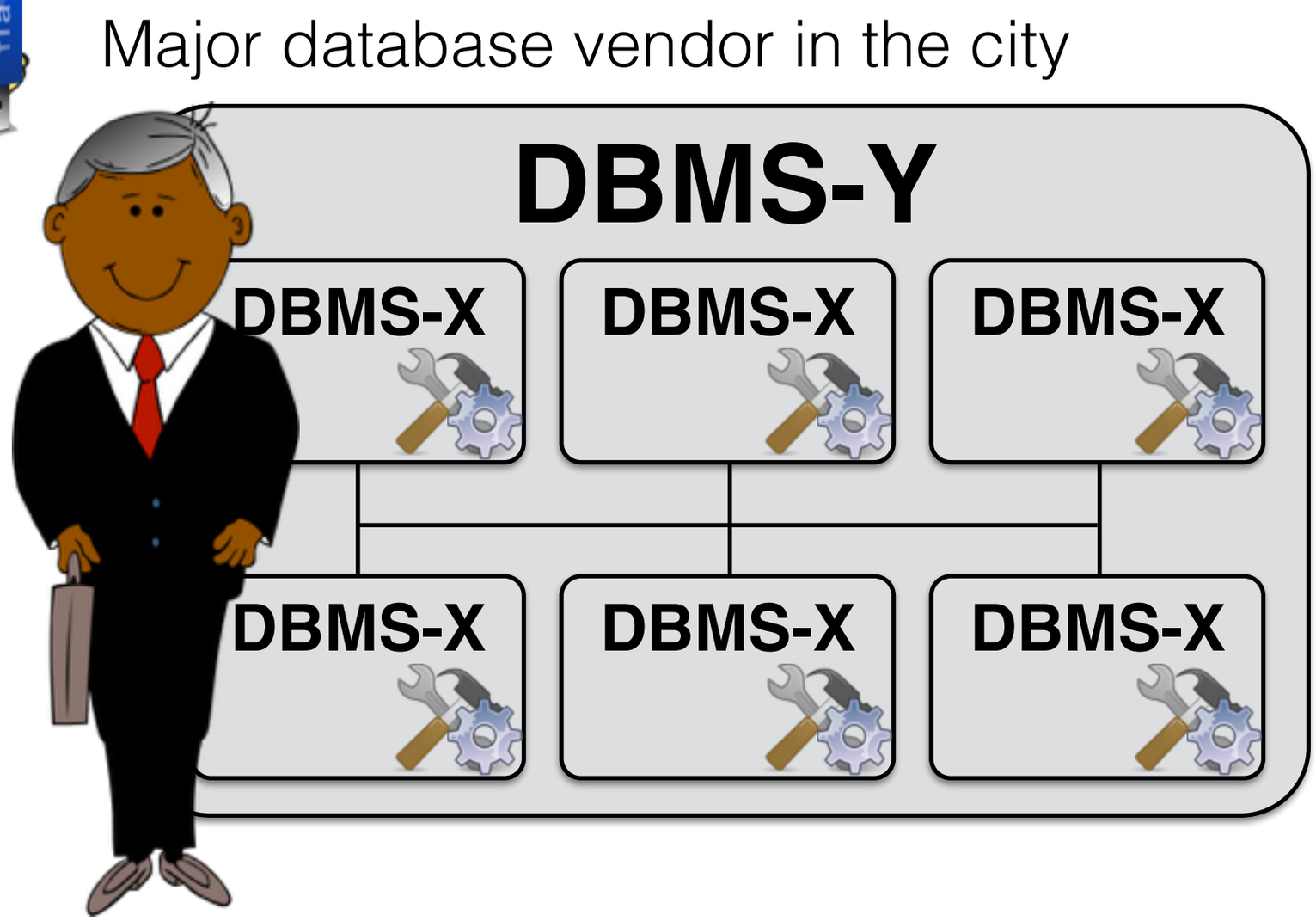
Bob@QCRI


1MB sample
1GB sample
USB flash
Query 2


Major database vendor in the city
DBMS-Y
DBMS-X
DBMS-X
DBMS-X
DBMS-X
DBMS-X
DBMS-X

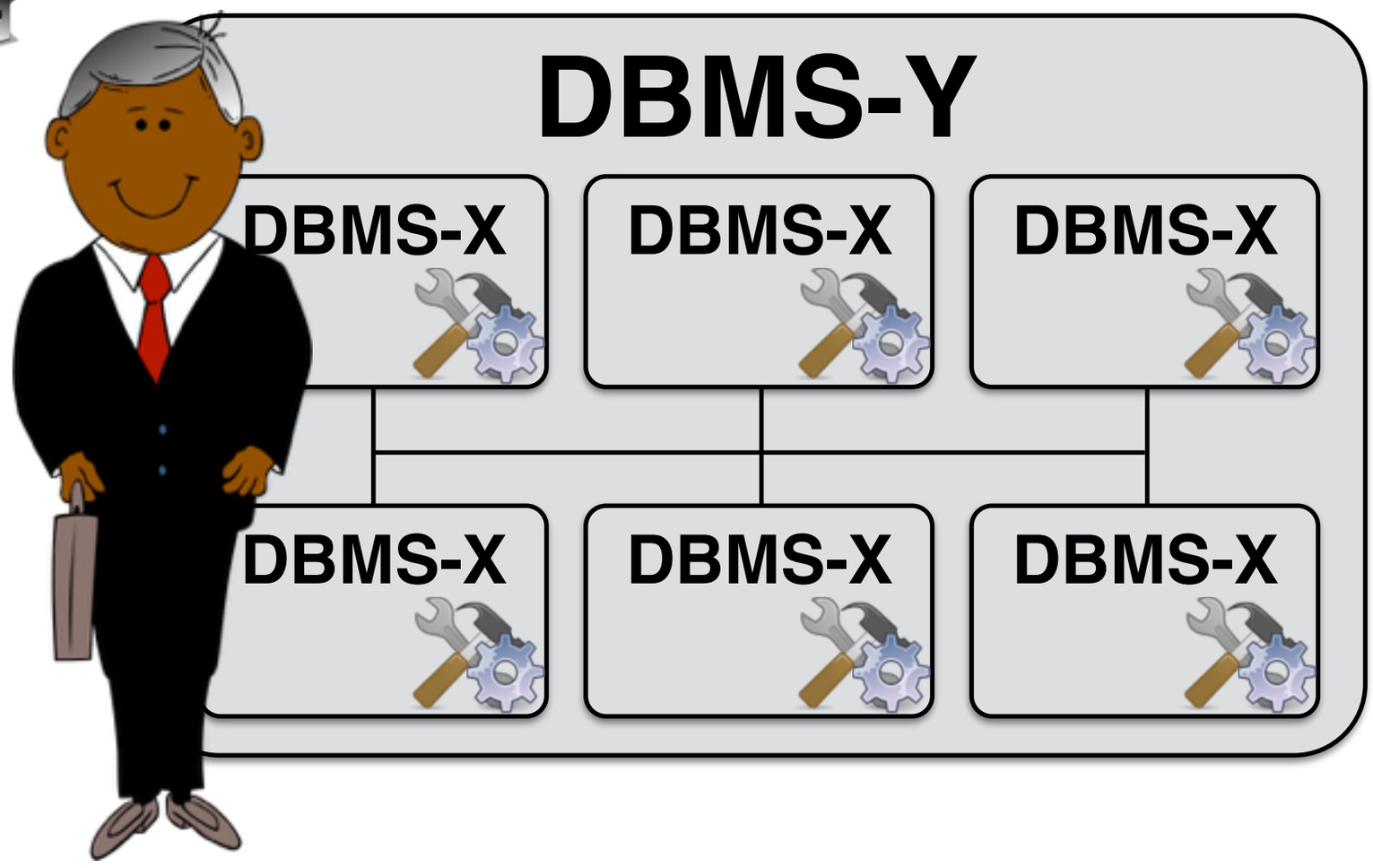
Bob@QCRI


1MB sample
1GB sample
USB flash
Query 2


Major database vendor in the city
DBMS-Y
DBMS-X
DBMS-X
DBMS-X
DBMS-X
DBMS-X
DBMS-X

Bob@QCRI

Major database vendor in the city



no results after 45mins!

Bob@QCRI

— cross product doesn't scale!—

Bob@QCRI

— cross product doesn't scale! —

1GB dataset x 1GB dataset

Bob@QCRI
— cross product doesn't scale! —

1GB dataset x 1GB dataset

↓

100 trillion comparisons!

(0.000*1ms per comparison*)

Bob@QCRI
— cross product doesn't scale! —

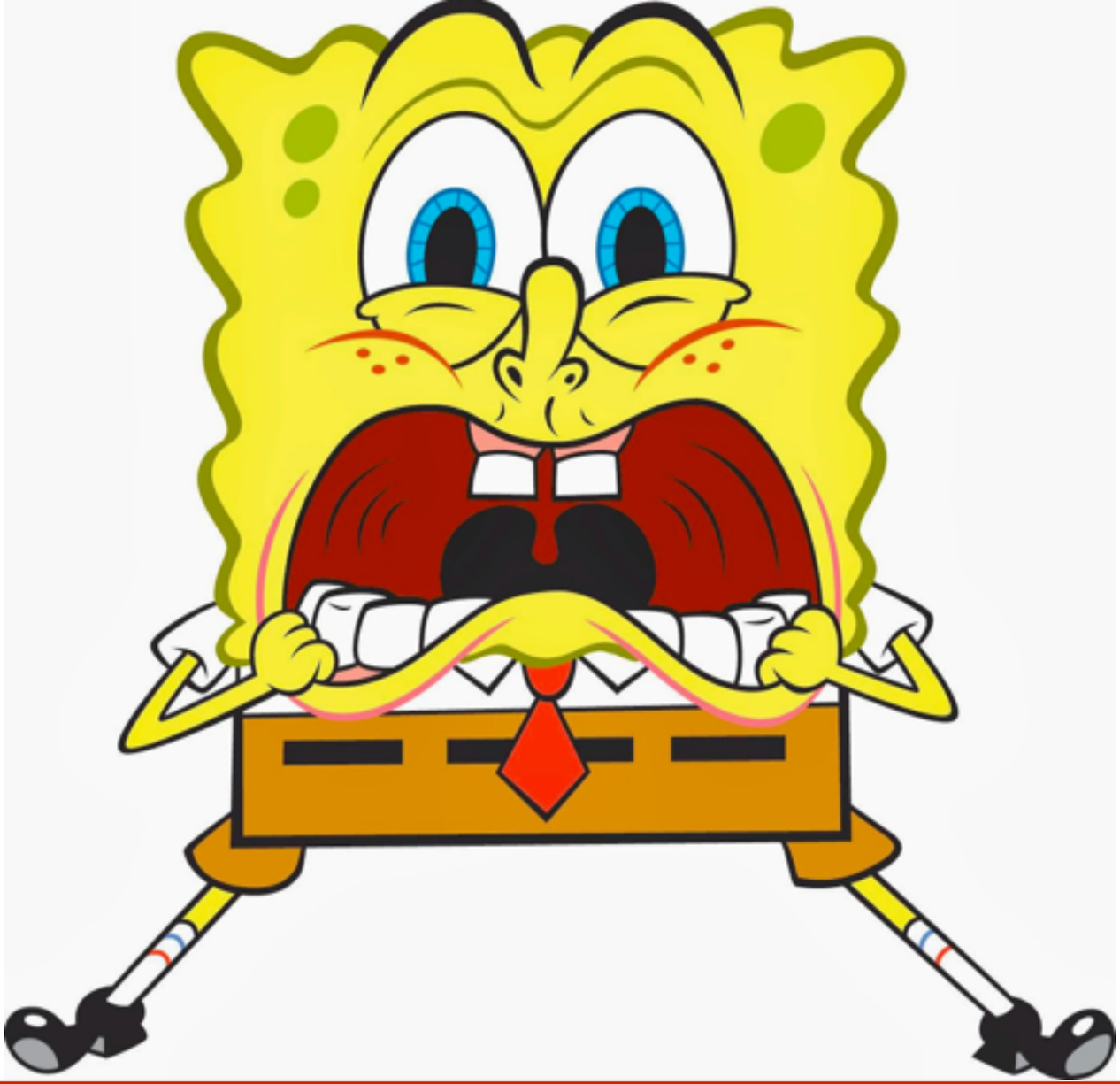
1GB dataset x 1GB dataset

↓

100 trillion comparisons!
(0.000*1ms per comparison*)

↓

> 115 days!



> 300 thousands years for 1TB!!!

Bob@QCRI

Bob@

Can you handle nonequi-joins?

Cross product
+
selection

DBMS-Z

Bob@QCRI

Distributed frameworks:

- MapReduce
- Spark

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Joins:

- Sort-merge
- Band
- Interval

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Joins:

- ~~Sort-merge~~
- ~~Band~~
- ~~Interval~~

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Joins:

- ~~Sort-merge~~
- ~~Band~~
- ~~Interval~~

But, can we avoid a cross product?

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Joins:

- ~~Sort-merge~~
- ~~Band~~
- ~~Interval~~

But, can we avoid a cross product?

Indexes:

- R-trees
- Space filling curves
- Bitmap
- Gist

Bob@QCRI

Distributed frameworks:

- ~~MapReduce~~
- ~~Spark~~

Joins:

- ~~Sort-merge~~
- ~~Band~~
- ~~Interval~~

Indexes:

- R-trees
- ~~Space filling curves~~
- ~~Bitmap~~
- Gist

But, can we avoid a cross product?

No existing technique solves Bob's problem!

DataAnalyticsTeam@QCRI

DataAnalyticsTeam@QCRI

More applications:

- Temporal DBs,
- Spatial DBs,
- Sensor networks,
- Social networks, …

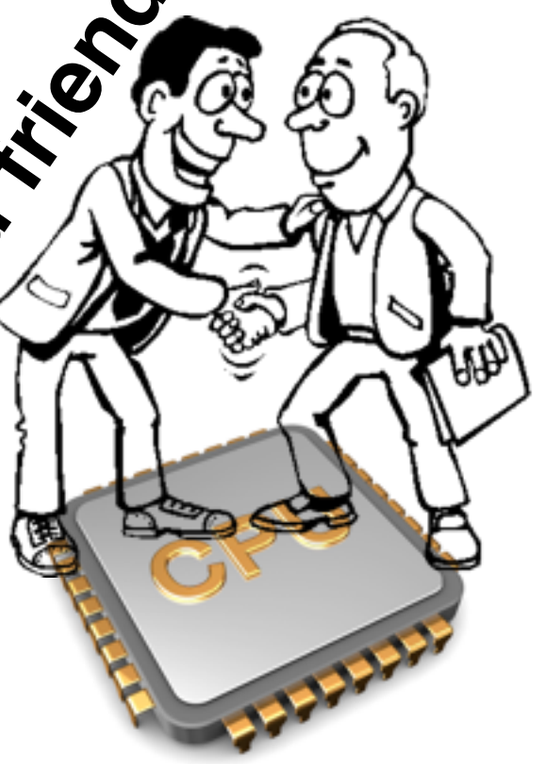
RAM locality is King!

— Jim Gray —

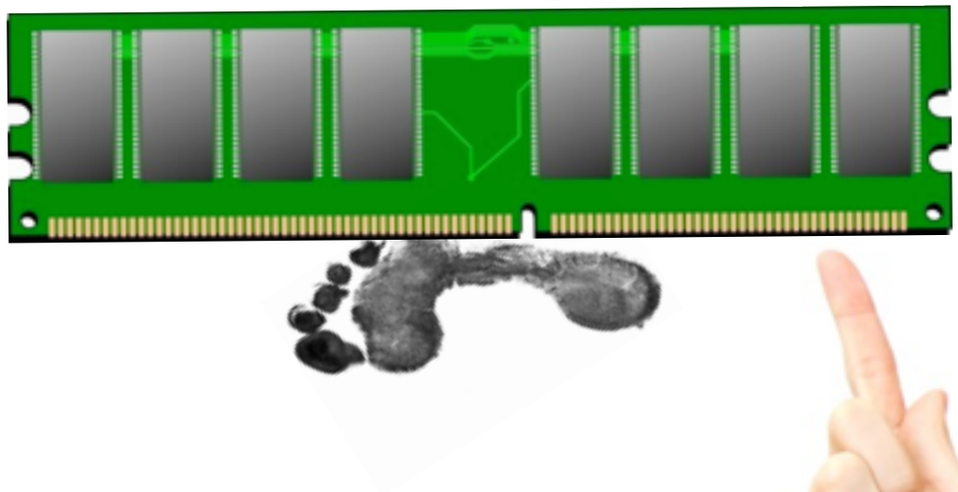
RAM locality is King!

— Jim Gray —

cpu friendly



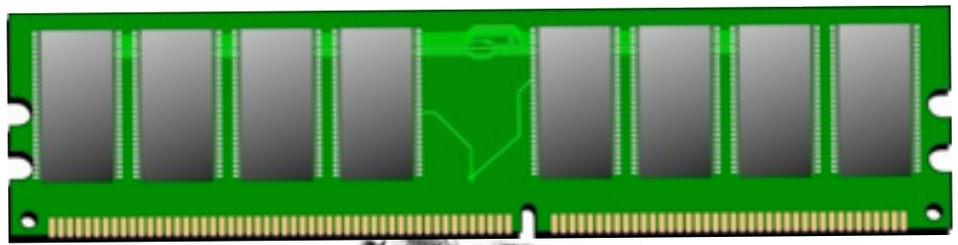
small memory footprint



RAM locality is King!

— Jim Gray —

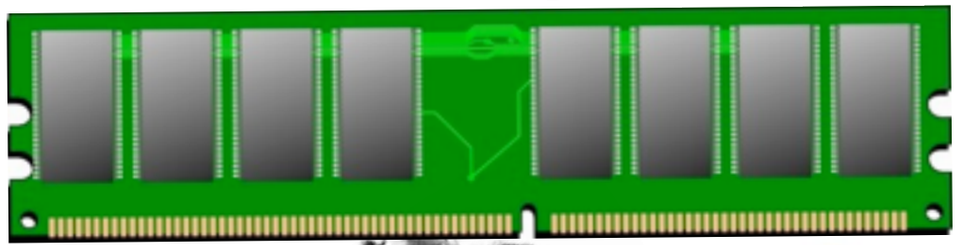
small memory footprint



RAM locality is King!

— Jim Gray —

small memory footprint



Fast!

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Employees

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*

Employees

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*

Employees

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

Sort on Salary

salary	tax	id
160	14	102
150	15	104
140	40	103
100	10	100
90	5	101

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
  AND e1.tax < e2.tax;
```

Employees

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

Sort on Salary

salary	tax	id
160	14	102
150	15	104
140	40	103
100	10	100
90	5	101

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
  AND e1.tax < e2.tax;
```

Employees

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

Sort on Salary

salary	tax	id
160	14	102
150	15	104
140	40	103
100	10	100
90	5	101

Sort on Tax

salary	tax	id
140	40	103
150	15	104
160	14	102
100	10	100
90	5	101

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id	salary	tax
102	160	14
104	150	15
103	140	40
100	100	10
101	90	5

Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id	salary	tax
102	160	14
104	150	15
100	100	10
103	140	40
101	90	5

Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

PA

order_sal
2
1
0
3
4

IEJoin Algorithm

Data quality rule

Q2:

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id	salary	tax
102	160	14
104	150	15
103	140	40
100	100	10
101	90	5

Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

PA

order_sal
2
1
0
3
4

Bit Array

0	0	0	0	0

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id	salary	tax
102	160	14
104	150	15
103	140	40
100	100	10
101	90	5

Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

PA

order_sal
2
1
0
3
4

Bit Array

0	0	0	0	0

IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id	salary	tax
102	160	14
104	150	15
103	140	40
100	100	10
101	90	5

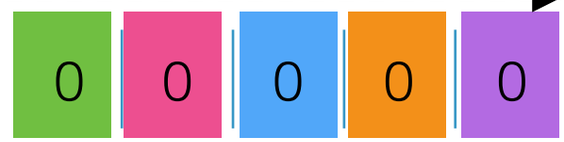
Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

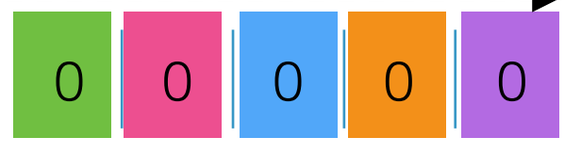
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

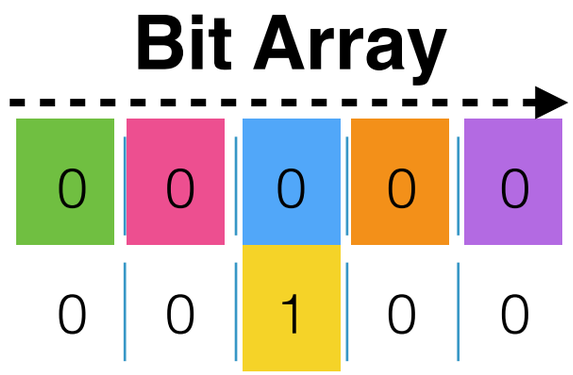
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

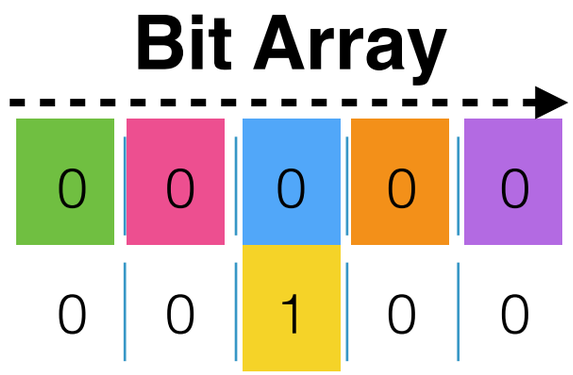
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

SELECT *e1.id, e2.id*
FROM *employees e1, e2*
WHERE *e1.salary > e2.salary*
AND *e1.tax < e2.tax;*

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

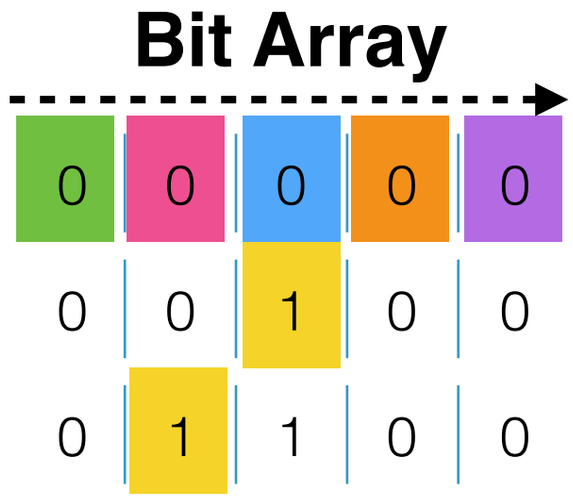
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

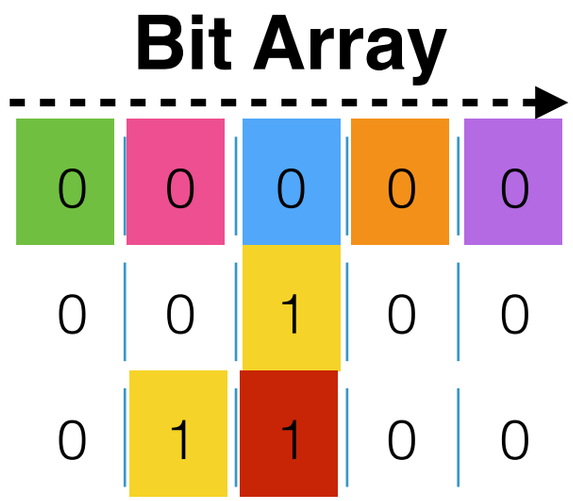
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

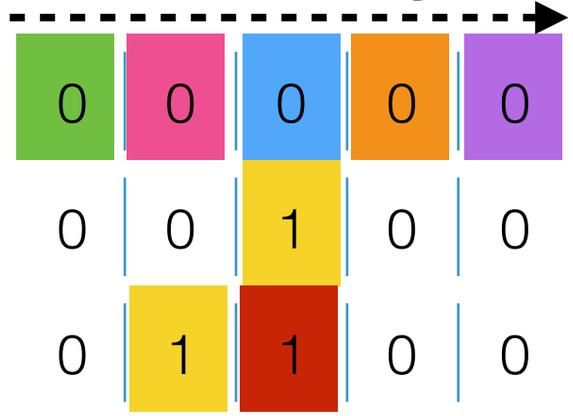
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	**103**

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

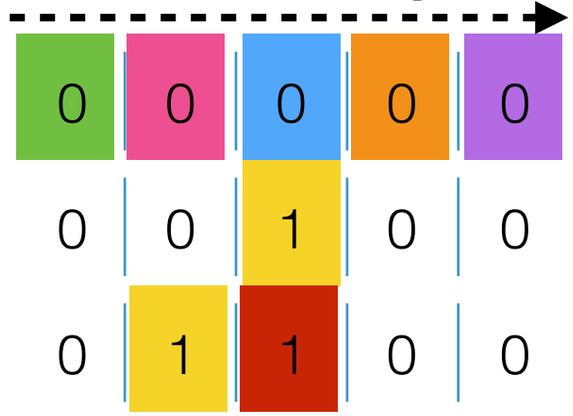
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	**103**

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

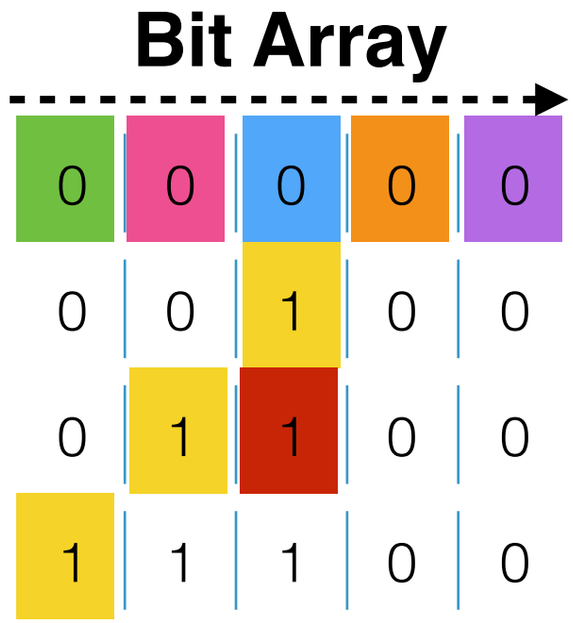
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	**103**

IEJoin Algorithm

Data quality rule

Q2:

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

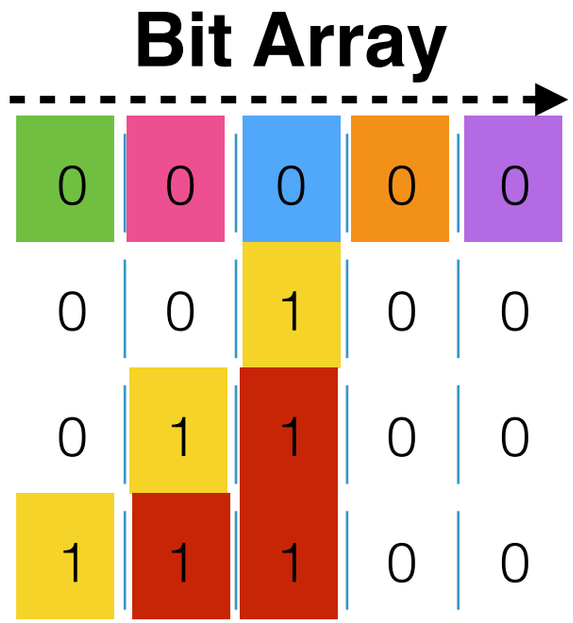
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	103
102	**104**
102	**103**

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

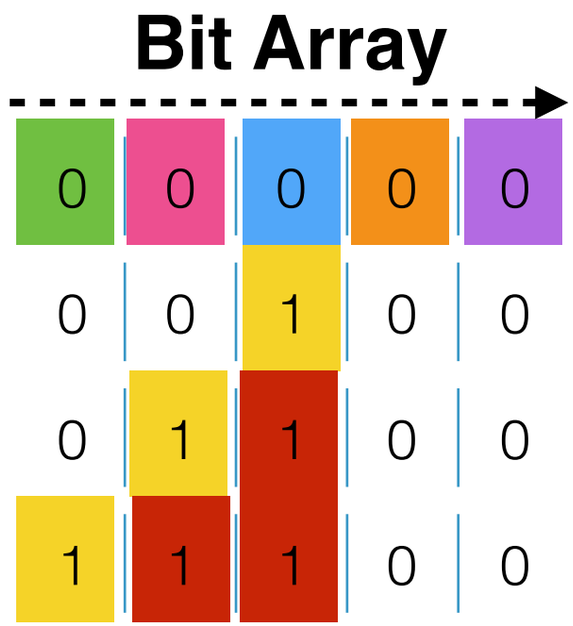
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	103
102	104
102	103

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

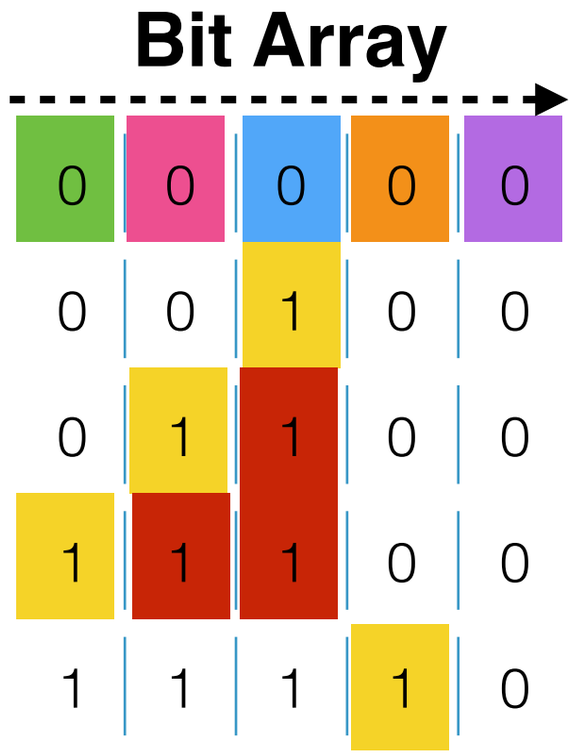
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	103
102	104
102	103

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

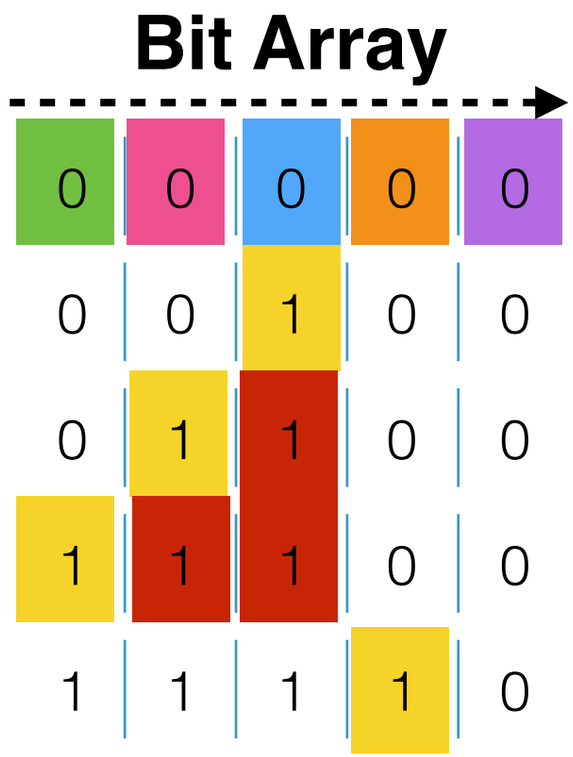
id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Bit Array



Results

e1.id	e2.id
104	103
102	104
102	103

IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

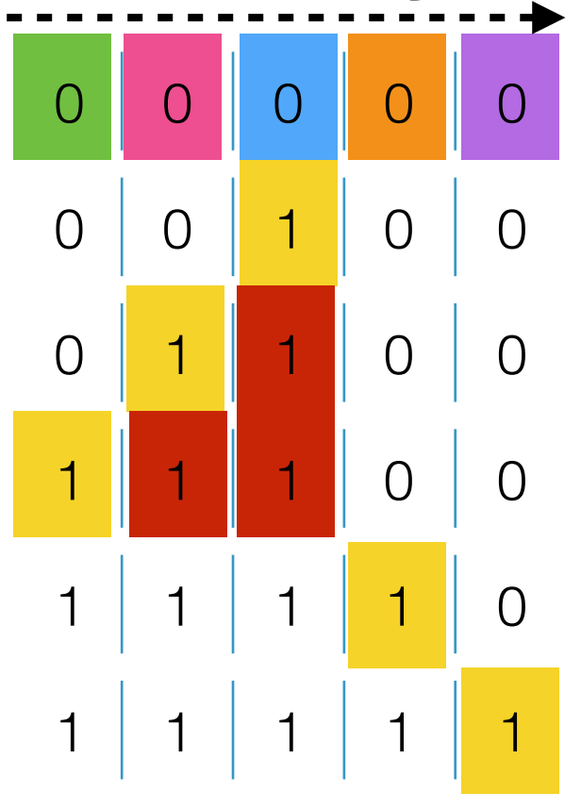
PA

order_sal
2
1
0
3
4

Results

e1.id	e2.id
104	103
102	104
102	103

Bit Array



IEJoin Algorithm

Q2: *Data quality rule*

```
SELECT e1.id, e2.id
FROM employees e1, e2
WHERE e1.salary > e2.salary
    AND e1.tax < e2.tax;
```

Sort on Salary

id
102
104
103
100
101

salary	tax
160	14
150	15
140	40
100	10
90	5

Sort on Tax

id
103
104
102
100
101

salary	tax
140	40
150	15
160	14
100	10
90	5

PA

order_sal
2
1
0
3
4

Results

e1.id	e2.id
104	103
102	104
102	103

Bit Array

0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1

Voila!

Enhancing IEJoin Algorithm

Emp1

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp2

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Enhancing IEJoin Algorithm

Emp1

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp1: Sort on Salary

salary	tax
160	14
⋮	

Emp1: Sort on Tax

salary	tax
140	40
⋮	

Emp1: PA

order_sal
2
⋮

Emp2

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp2: Sort on Salary

salary	tax
210	42
⋮	

Emp2: Sort on Tax

salary	tax
99	23
⋮	

Emp2: PA

order_sal
2
⋮

Enhancing IEJoin Algorithm

Emp1

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp1: Sort on Salary

salary	tax
160	14
⋮	

Emp1: Sort on Tax

salary	tax
140	40
⋮	

Emp1: PA

order_sal
2
⋮

Emp1: O1

order_sal
3
⋮

Emp1: O2

order_sal
5
⋮

Emp2

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp2: Sort on Salary

salary	tax
210	42
⋮	

Emp2: Sort on Tax

salary	tax
99	23
⋮	

Emp2: PA

order_sal
2
⋮

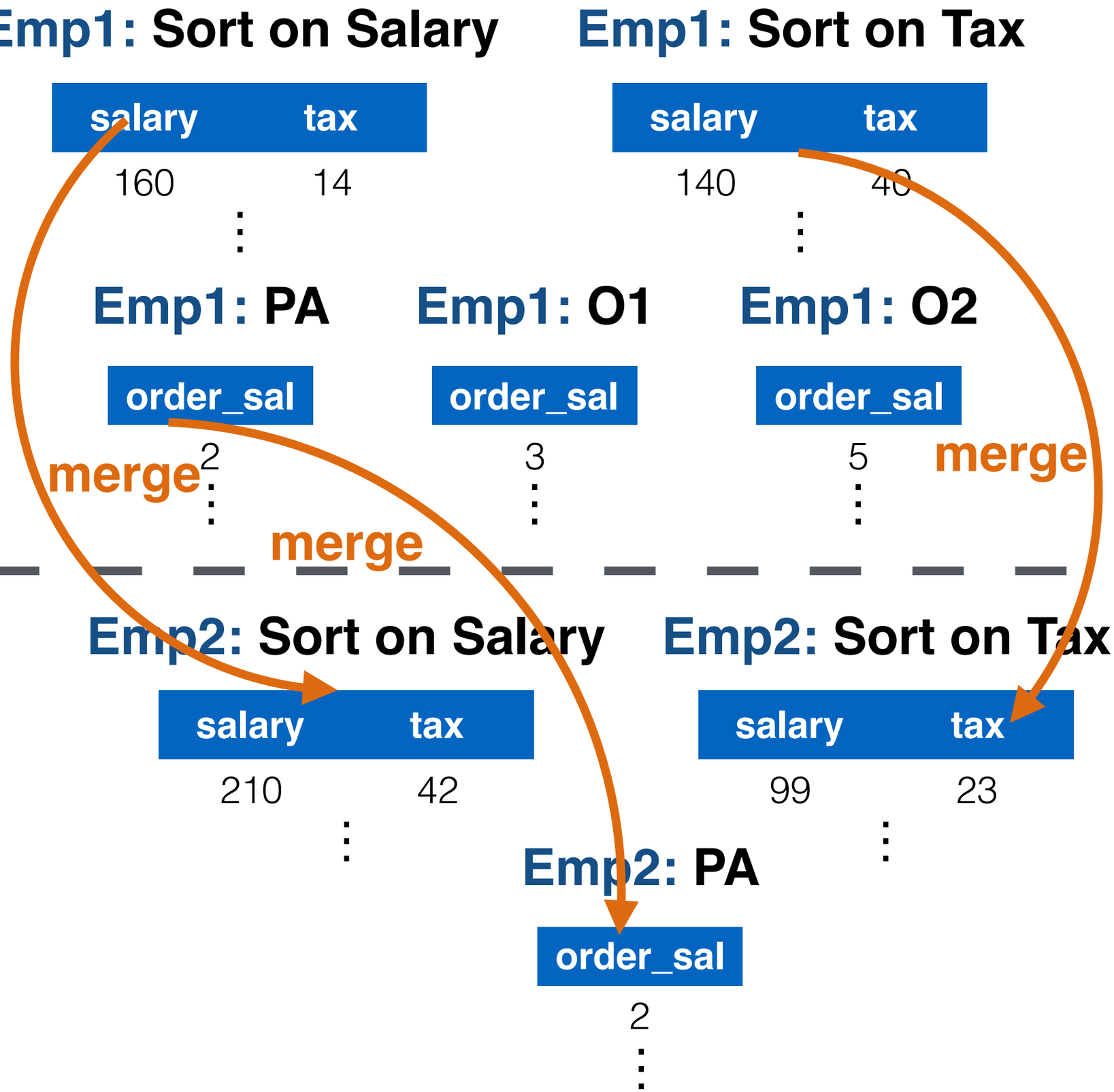
Enhancing IEJoin Algorithm

Emp1

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp2

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15



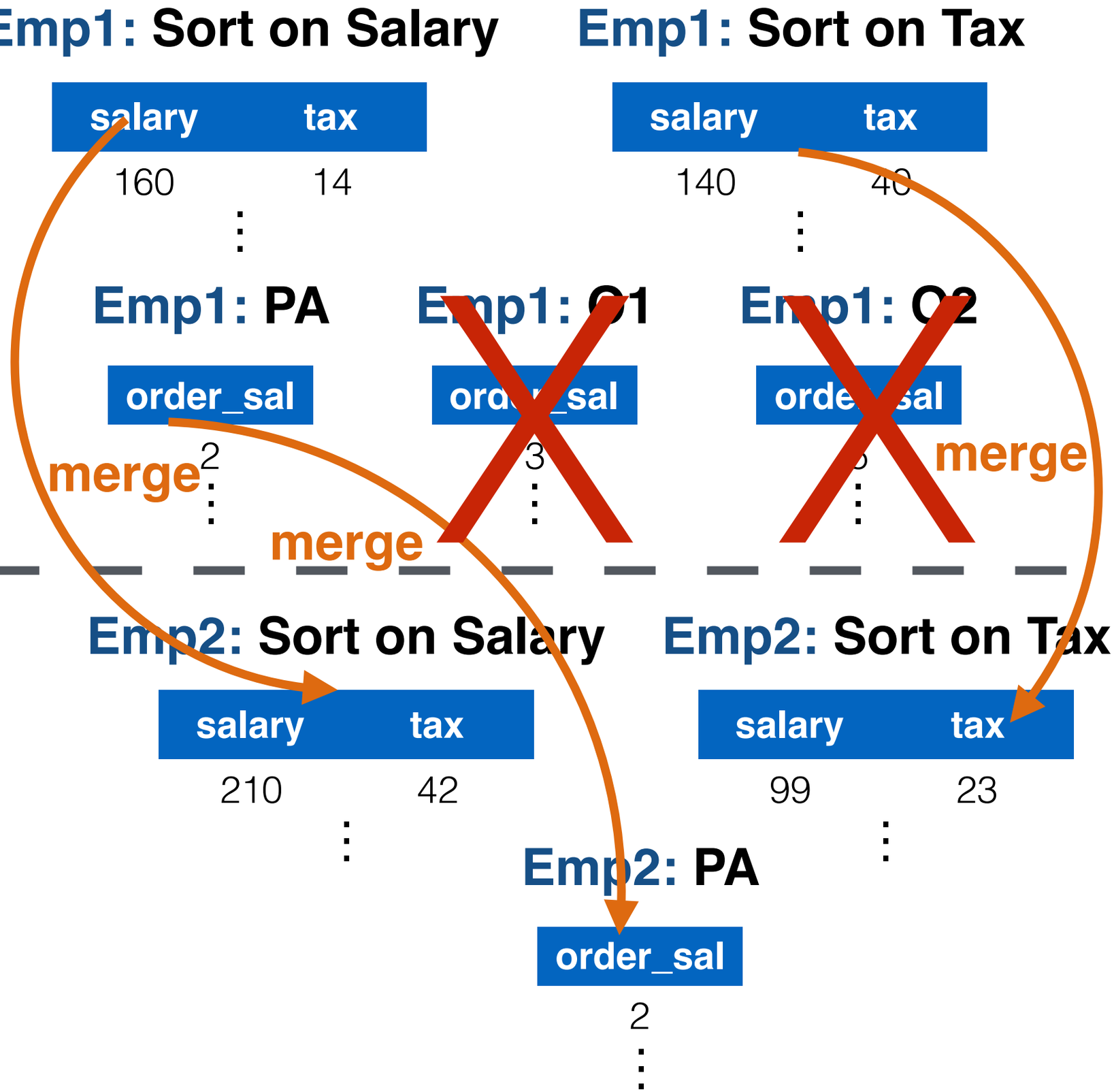
Enhancing IEJoin Algorithm

Emp1

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15

Emp2

ssn	salary	tax
3456	100	10
9876	90	5
6790	160	14
4567	140	40
2095	150	15



Enhancing IEJoin Algorithm

Sort on Salary

salary	tax
210	42
⋮	

Sort on Tax

salary	tax
99	23
⋮	

PA

order_sal
2
⋮

Bit Array

0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Enhancing IEJoin Algorithm

Sort on Salary

salary	tax
210	42
⋮	

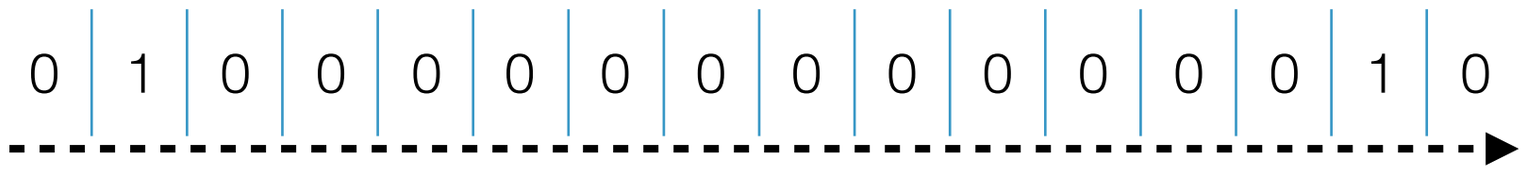
Sort on Tax

salary	tax
99	23
⋮	

PA

order_sal
2
⋮

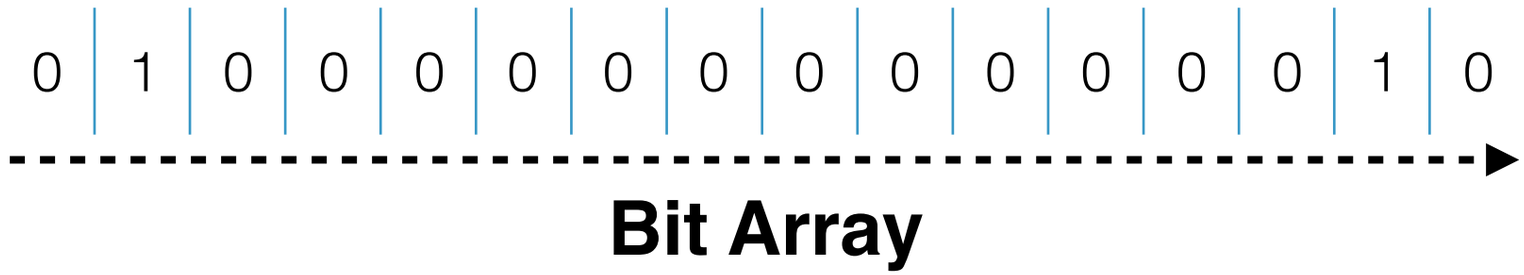
Bit Array



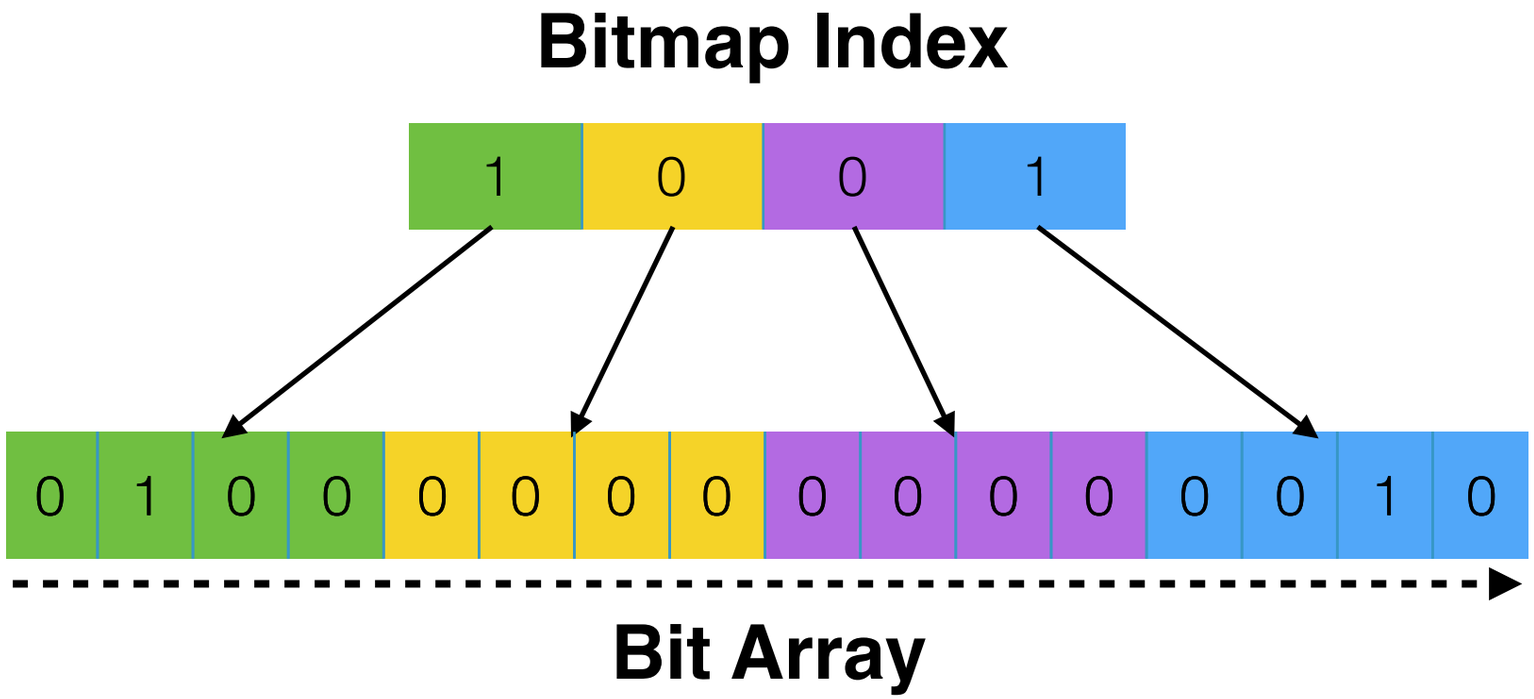
+

Postfiltering

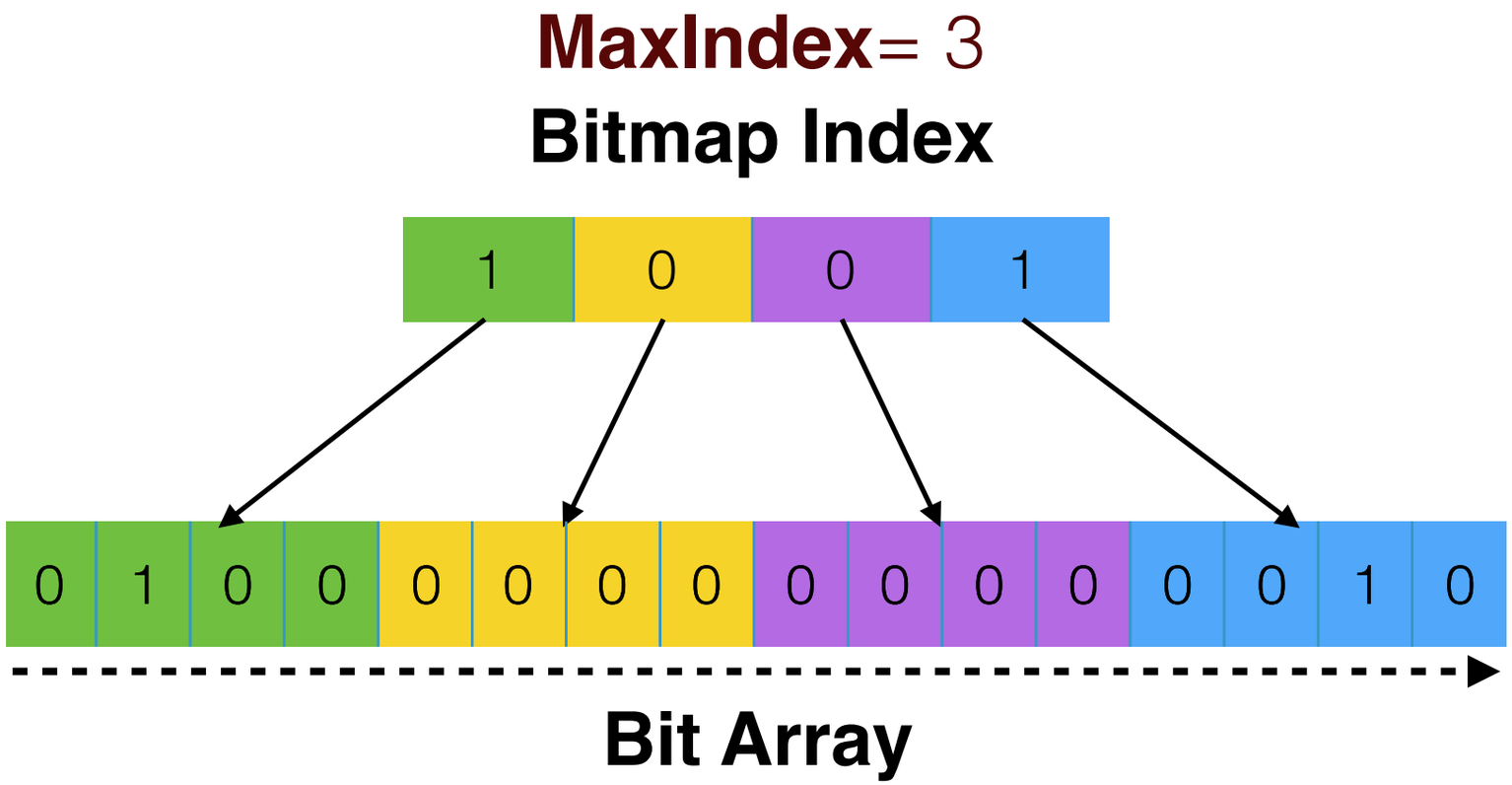
Enhancing IEJoin Algorithm



Enhancing IEJoin Algorithm



Enhancing IEJoin Algorithm



Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

Scaling IEJoin Up

R

S

Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

Scaling IEJoin Up

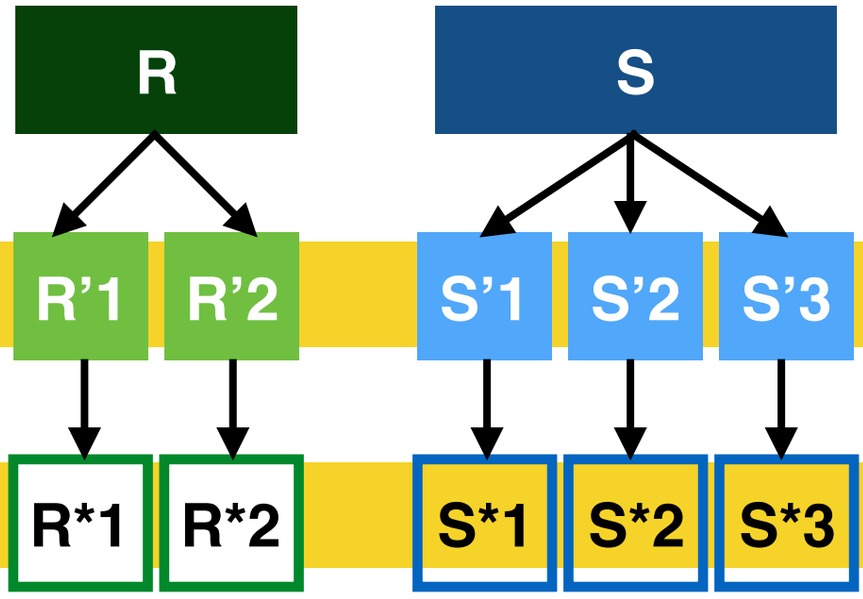
Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

sort on 1 att. + projection + partition

metadata (min-max values)

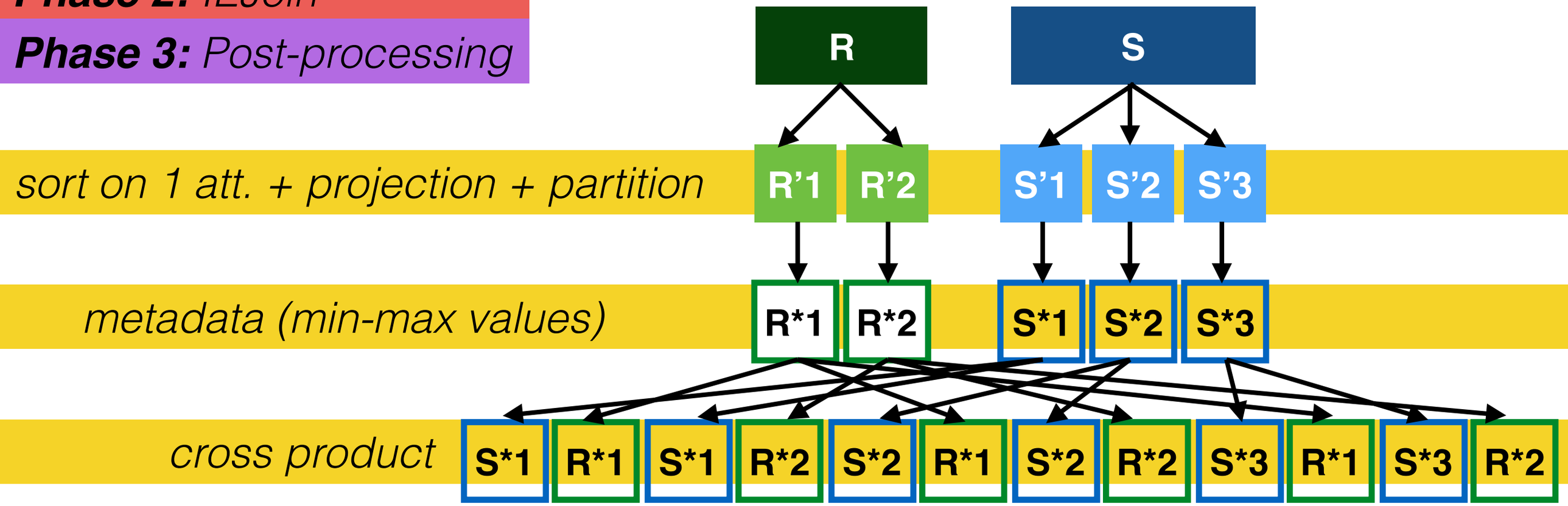


Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

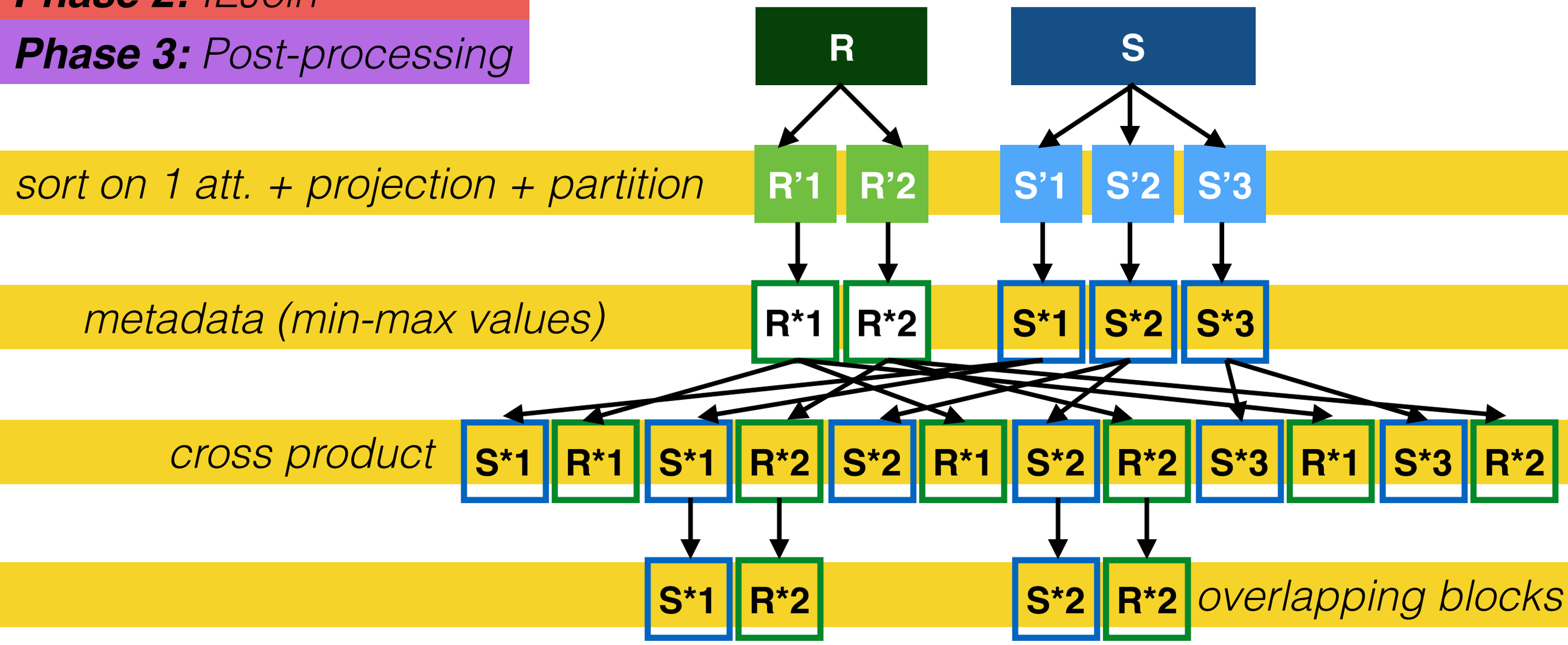


Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

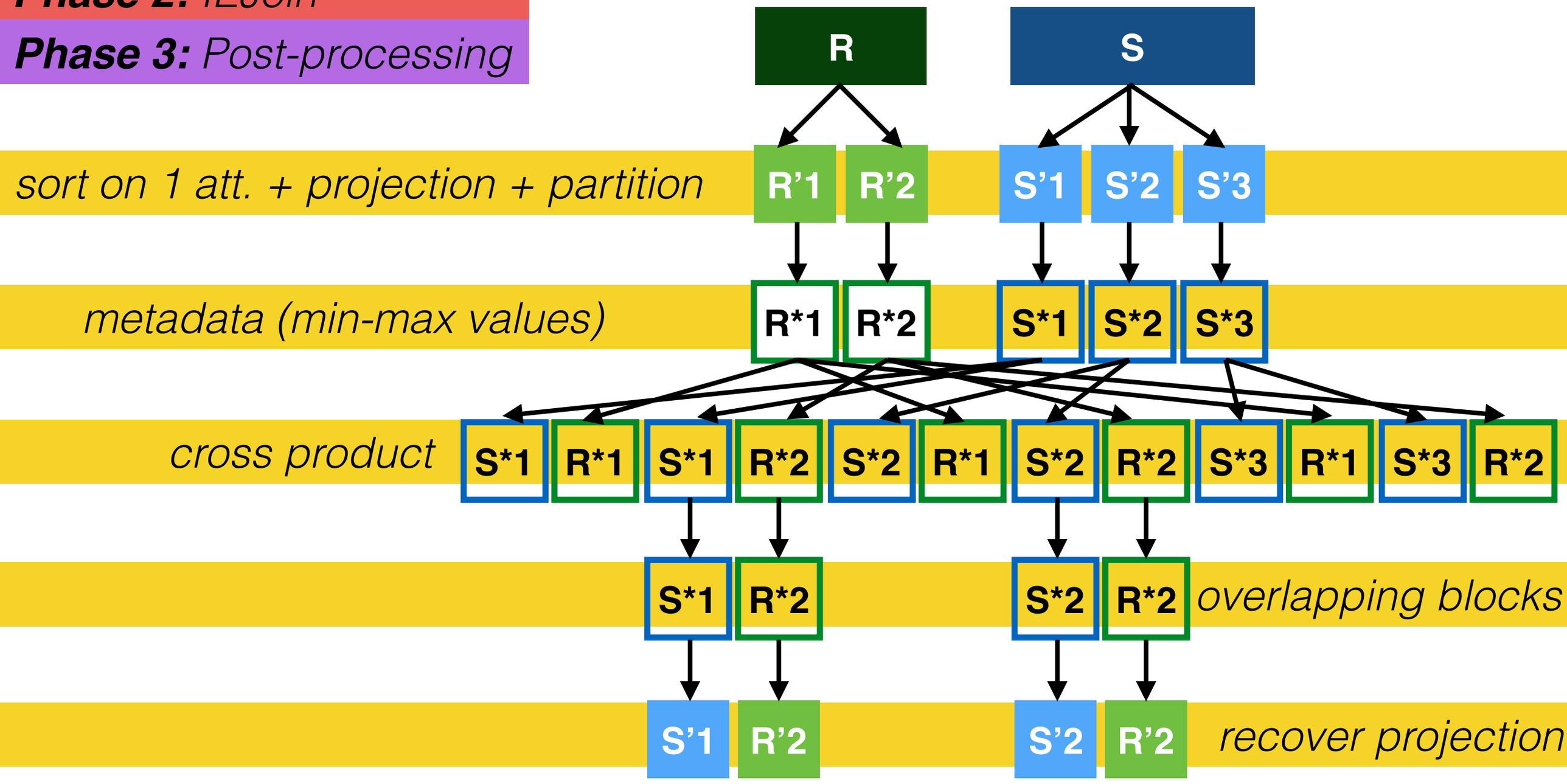


Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

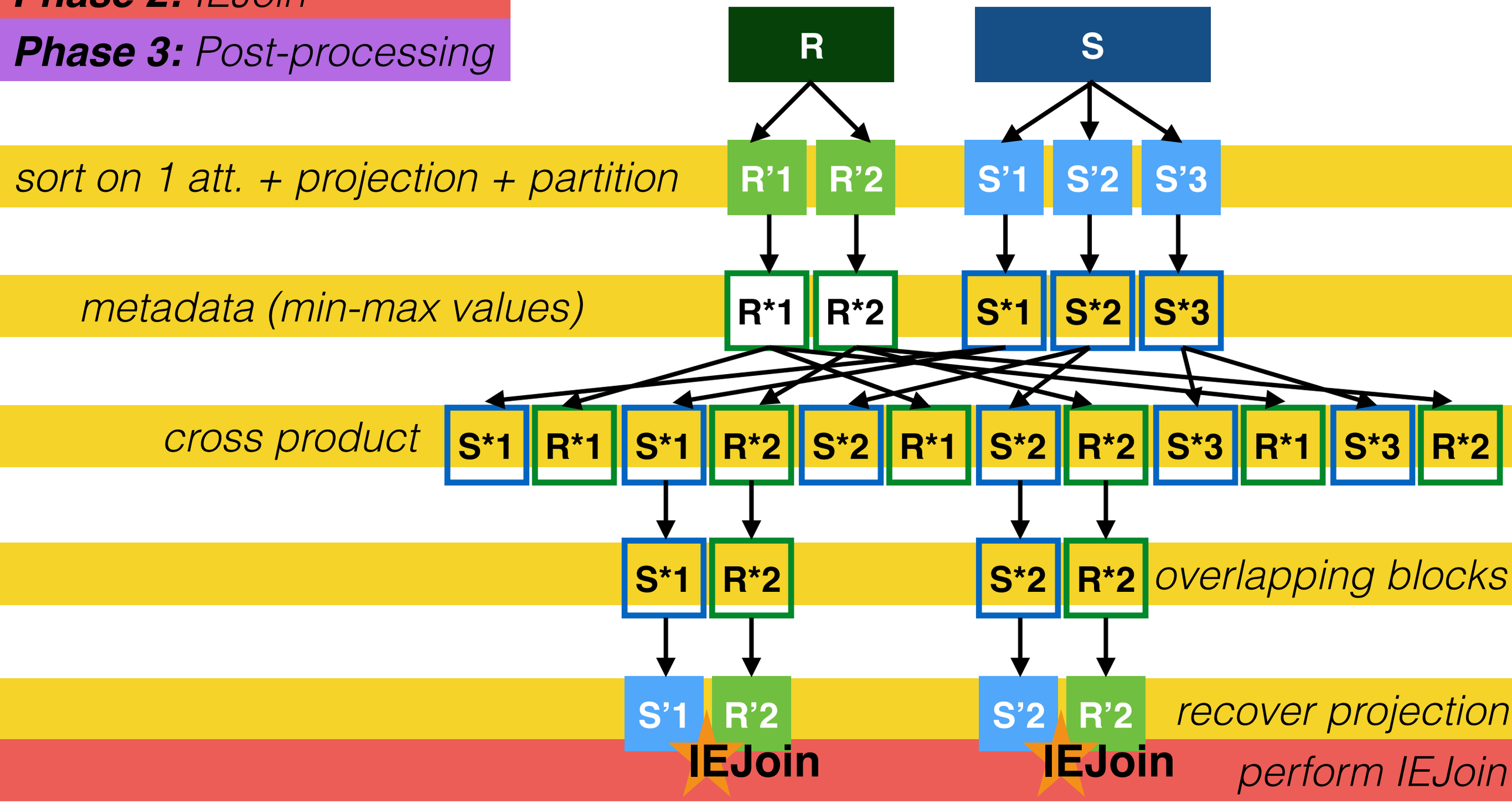


Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*

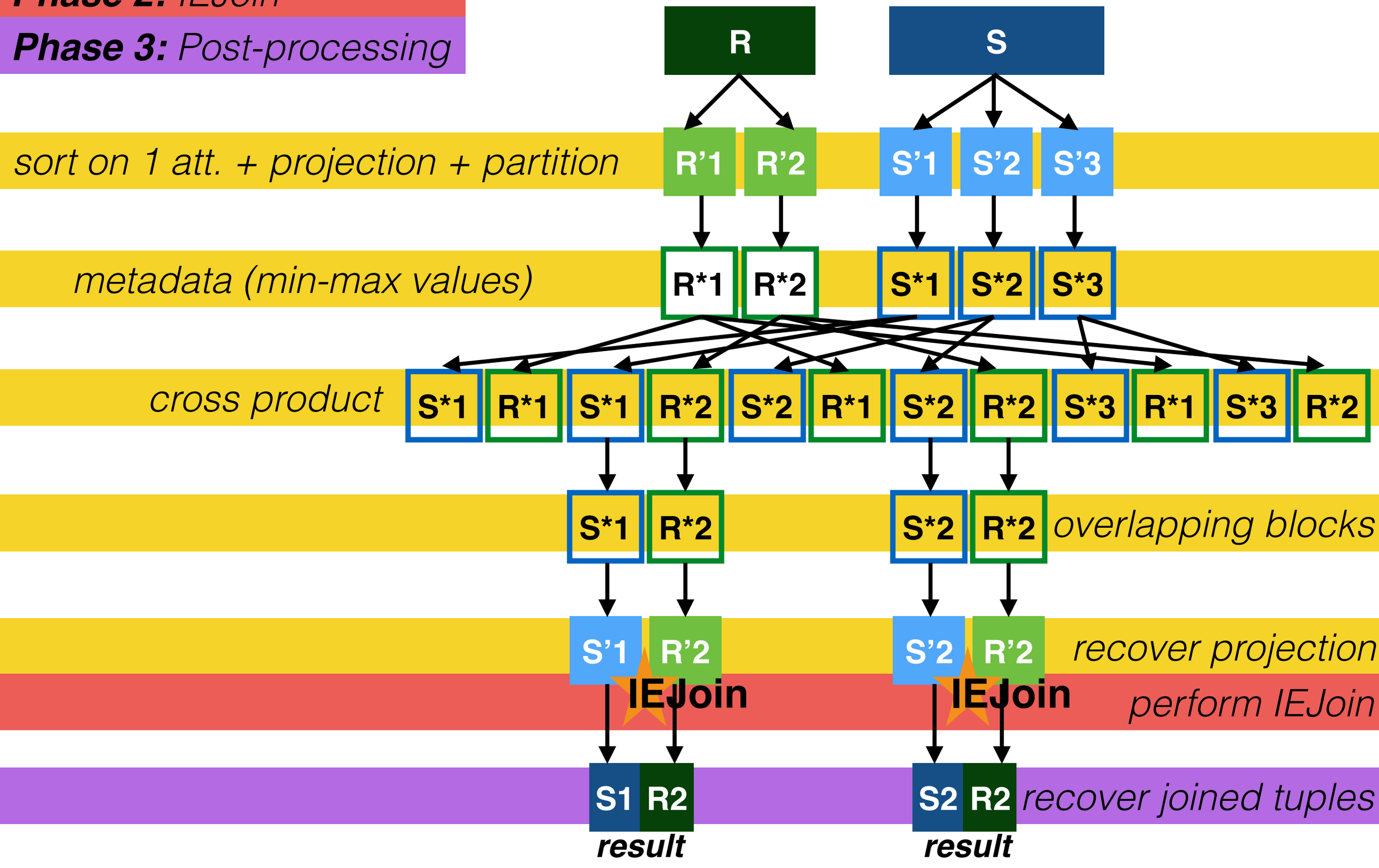


Scaling IEJoin Up

Phase 1: *Pre-processing*

Phase 2: *IEJoin*

Phase 3: *Post-processing*



More Optimizations…

Query Optimization &
Incremental IEJoin

— Fast and Scalable Inequality Joins —
VLDBJ, special issue on best VLDB'16 papers

Implemented on…

IEJoin

Compared with…

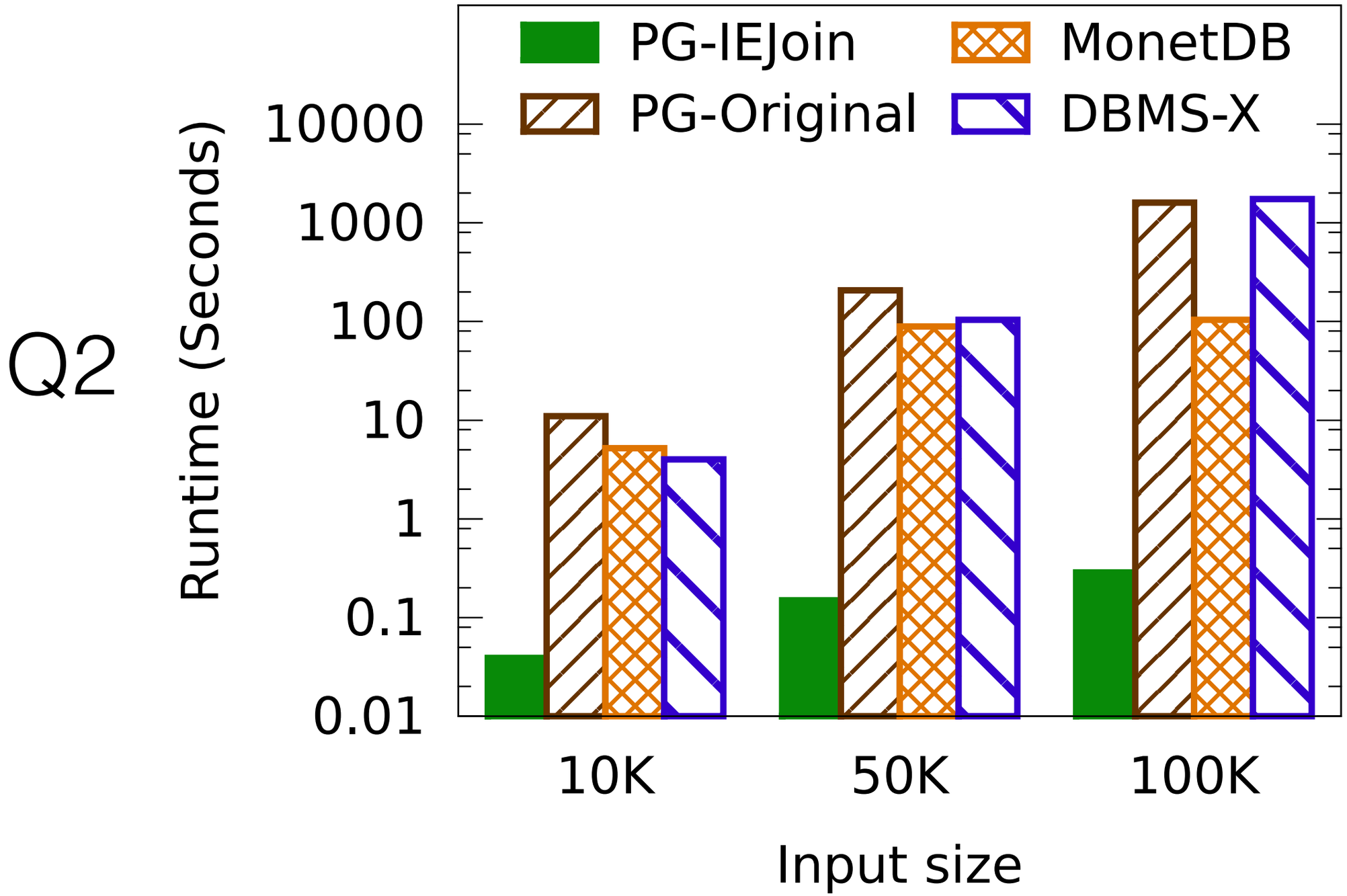
BTree

GiST

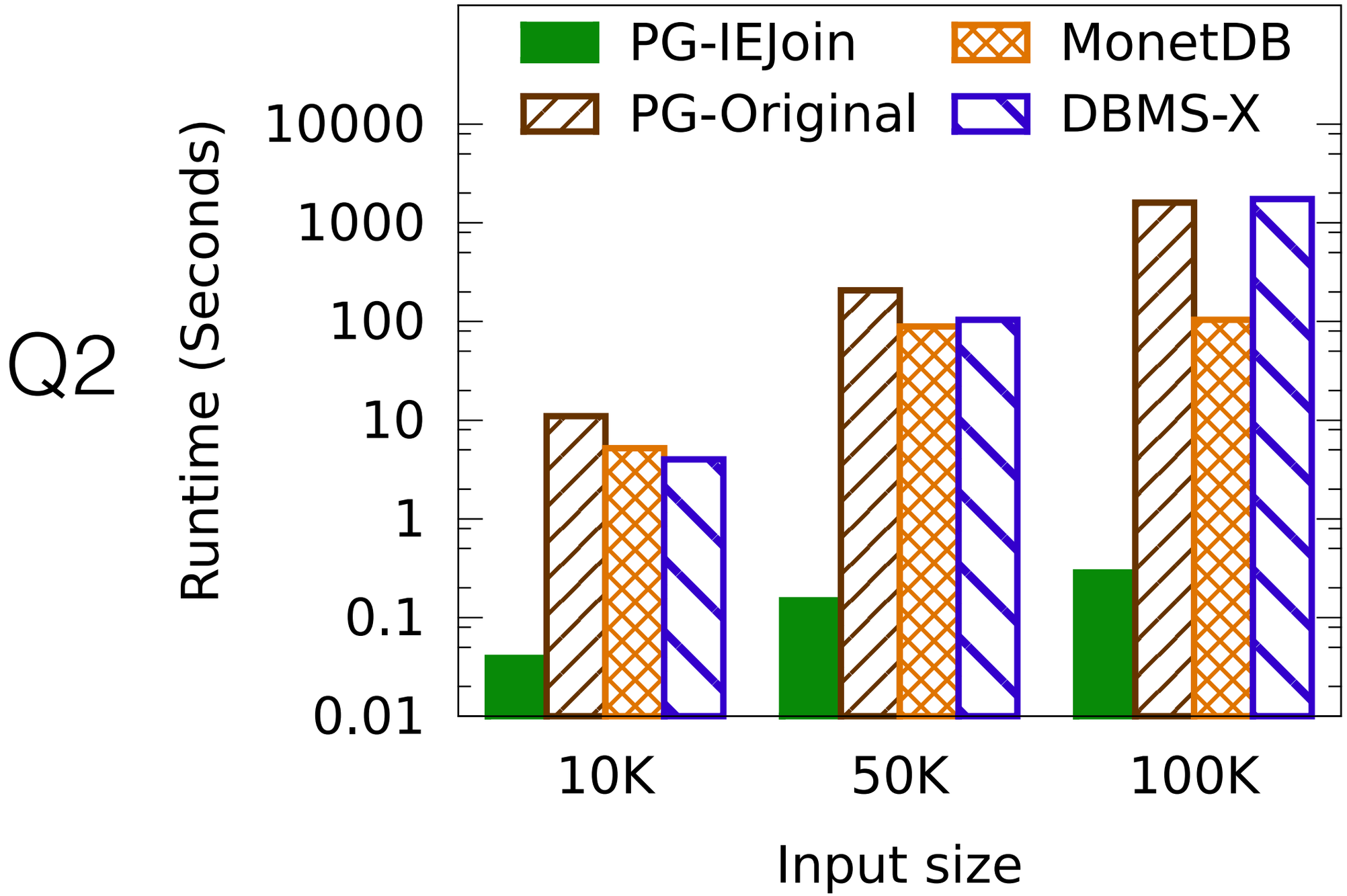
PostgreSQL

DBMS-X

Single-Node Performance

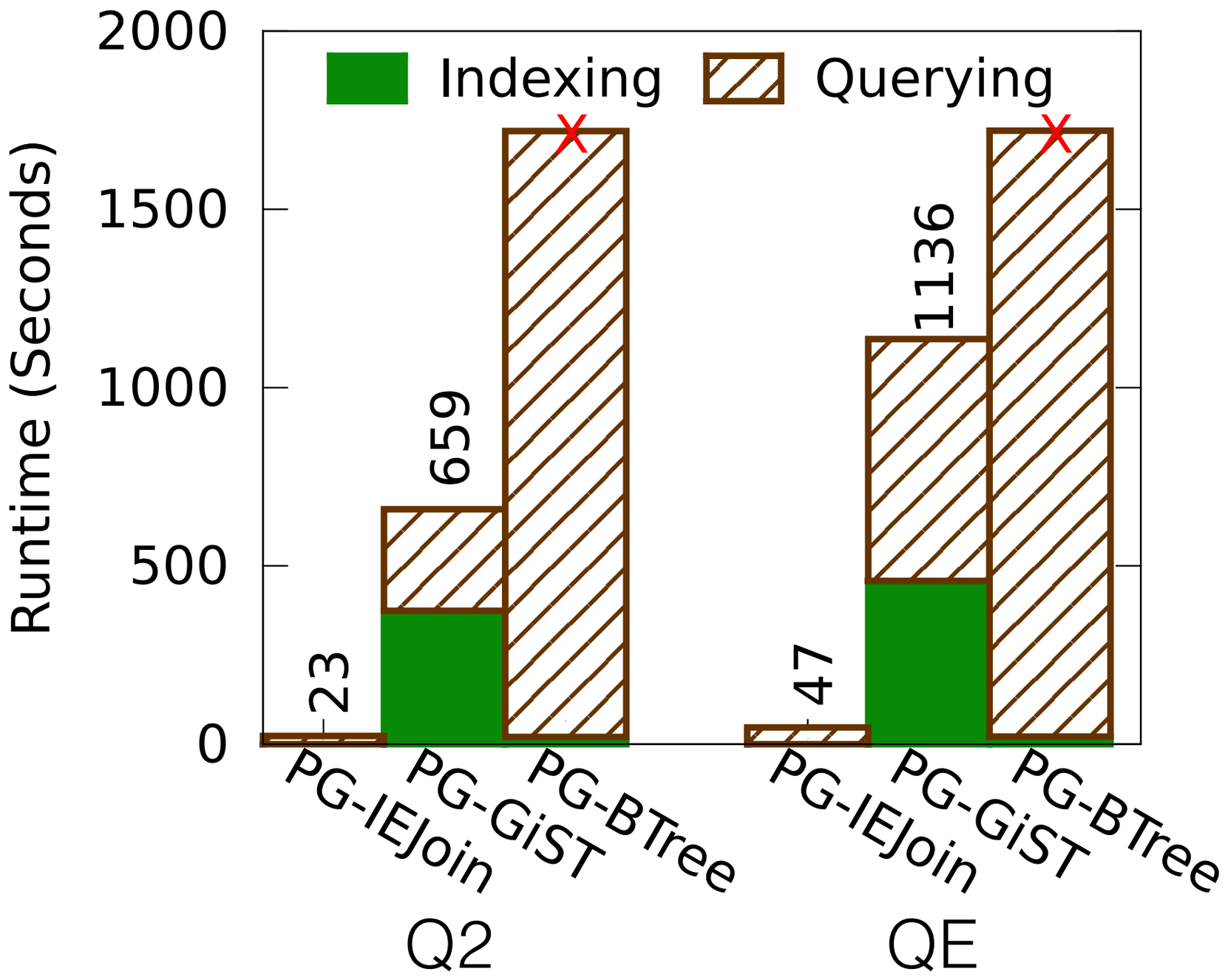


Single-Node Performance

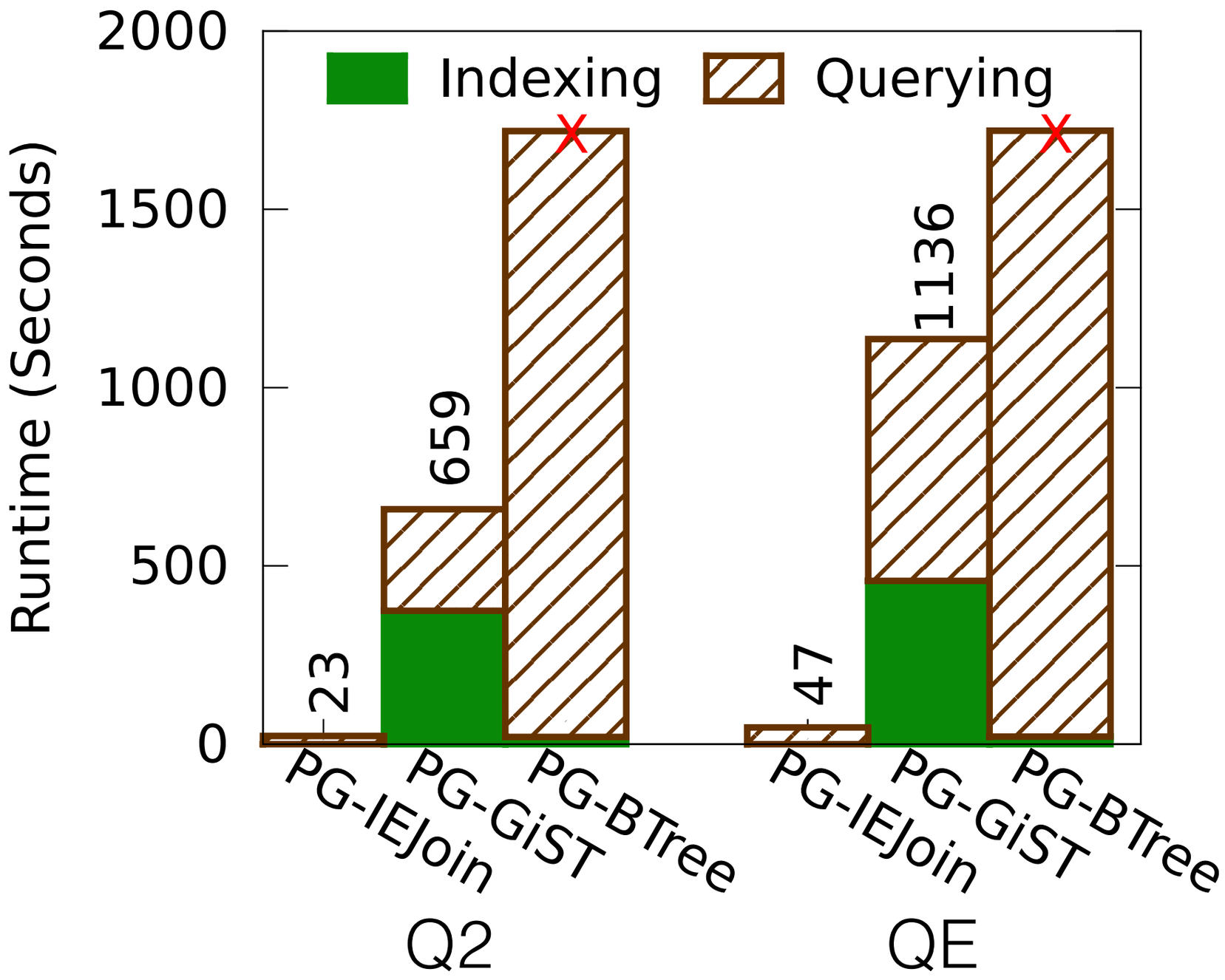


up to more than **3 orders of magnitude**!

IEJoin vs Indexing

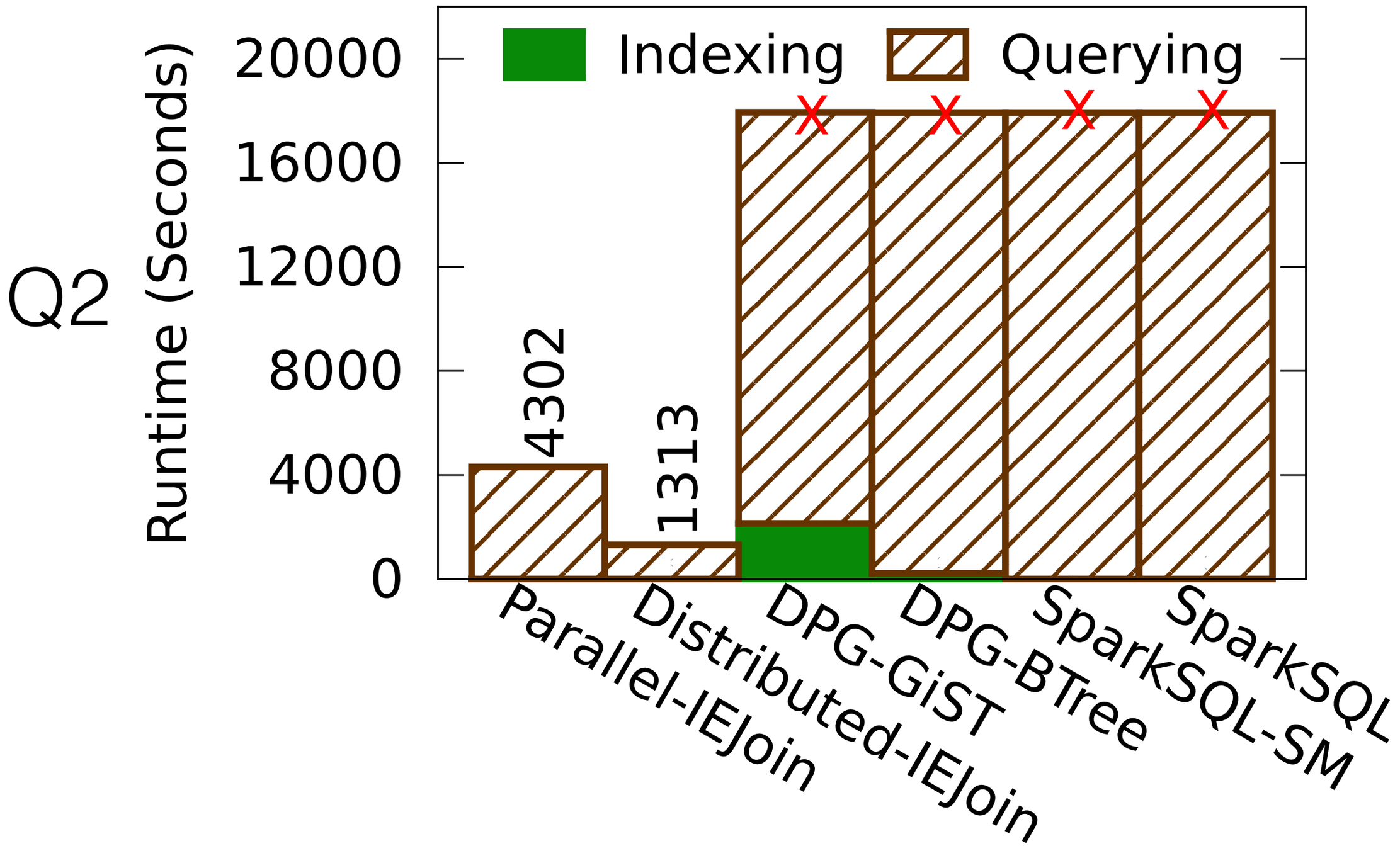


IEJoin vs Indexing

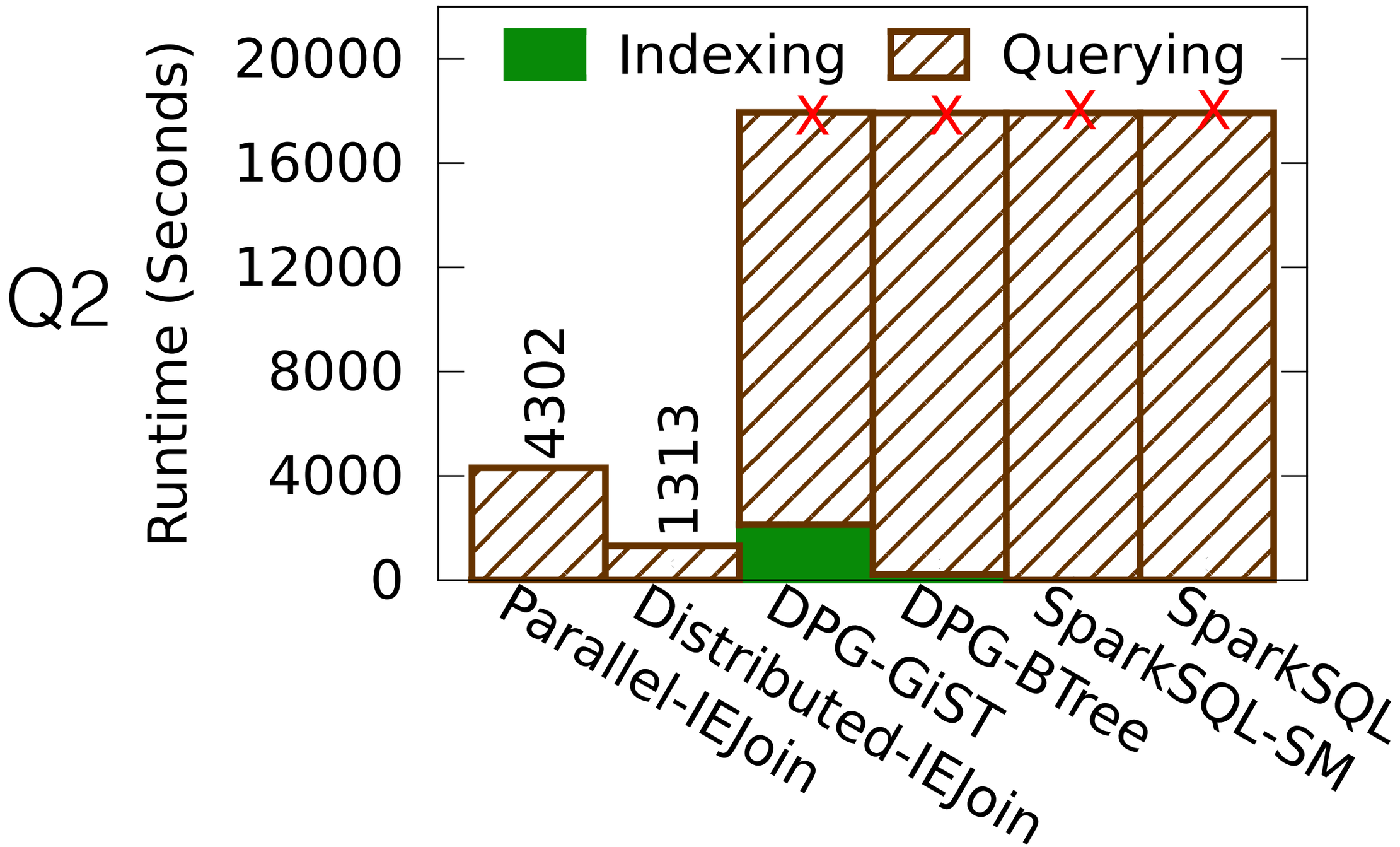


more than 1 **order of magnitude**!

Multi-Node Performance

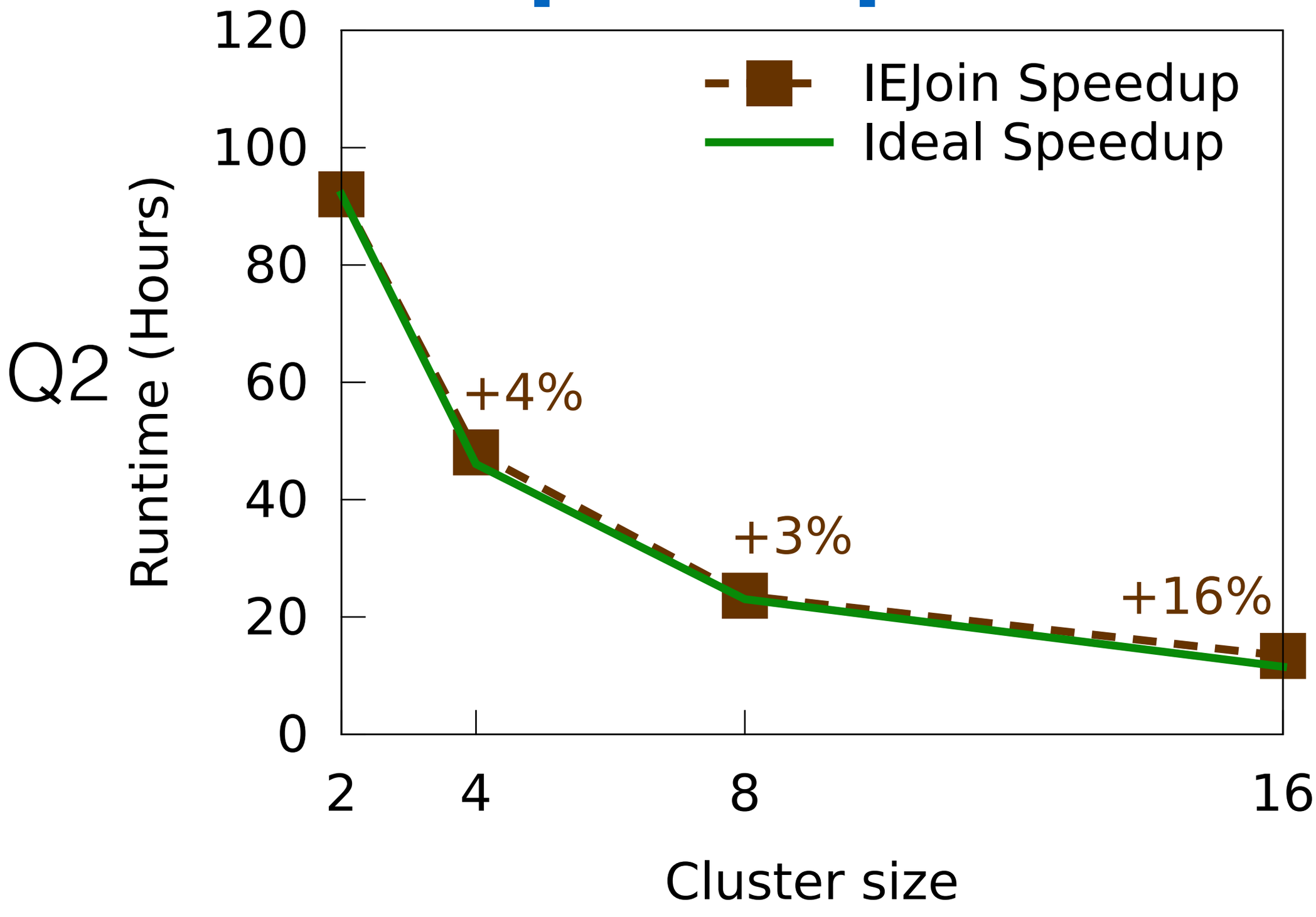


Multi-Node Performance

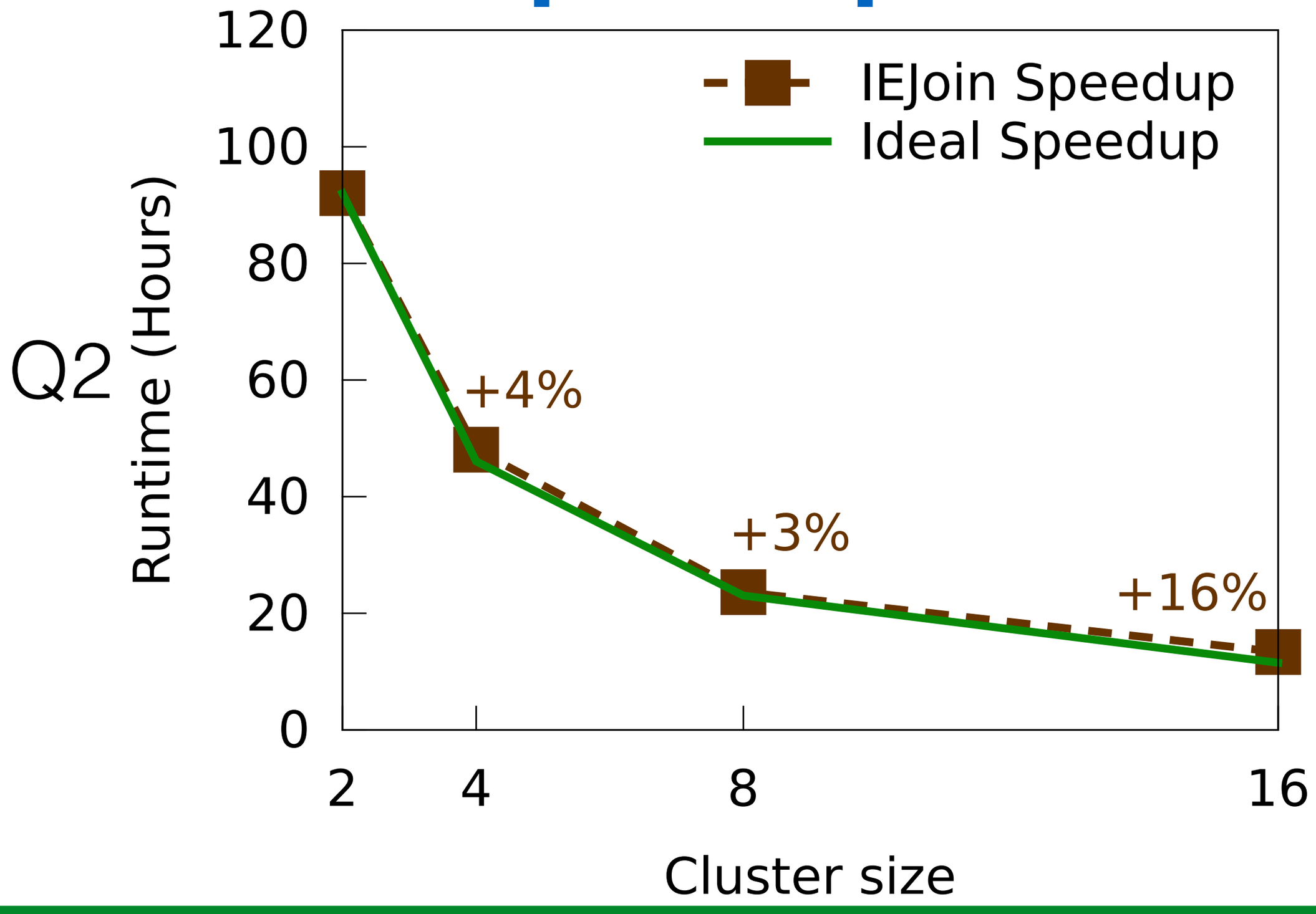


more than **2 orders of magnitude**!

Speedup


IEJoin Speedup
Ideal Speedup
Q2
Runtime (Hours)
120
100
80
60
40
20
0
+4%
+3%
+16%
2
4
8
16
Cluster size

Speedup



close to **ideal** speedup!

Summary

Simple

Employees

id
102
104
103
100
101

id	ssn	salary	tax
100	3456	100	10
101	9876	90	5
102	6790	160	14
103	4567	140	40
104	2095	150	15

Sort on Salary

id	salary	tax
102	160	14
104	150	15
103	140	40
100	100	10
101	90	5

Sort on Tax

id	salary	tax
103	140	40
104	150	15
102	160	14
100	100	10
101	90	5

PA

order_sal
2
1
0
3
4

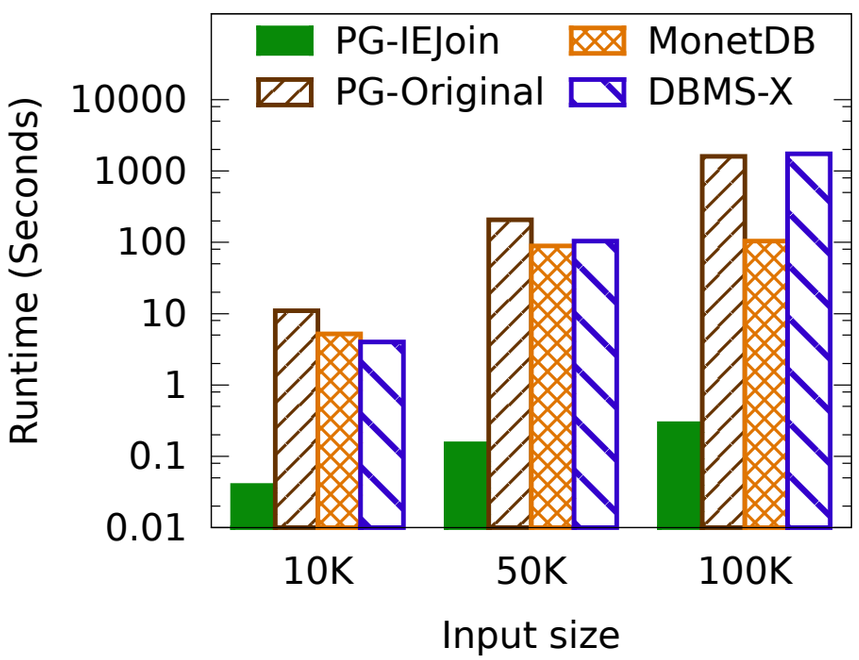
Results

e1.id	e2.id
104	103
102	104
102	103

Bit Array

0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	1	1	0	0
1	1	1	1	0
1	1	1	1	1

Fast



Q2