Updated 1/26/2021

Topic 1: Data models and query languages
Unit 1: SQL (continued)
Lecture 3

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp21)

https://northeastern-datalab.github.io/cs7240/sp21/

1/26/2021

Pre-class conversations

- Last class recapitulation
  - Additional examples: conceptual evaluation strategy, SJs, decorrelating queries (postponed for after relational algebra)
- Any questions on class procedures?
  - Contrasting 4 times "scribing" (class illustrations) against 4 more assignments: you can choose what to focus on ☺. Plus you choose the question that you answer. That's key in research.
  - Example "minimum examples" today in class

- today:
  - SQL continued (with connection to table integration)
  - perhaps start of calculus

Topic 1: Data models and query languages

PRELIMINARY

- **Lecture 1 (Tue 1/19):** Course introduction, SQL refresher, PostgreSQL setup, SQL files
- **Lecture 2 (Fri 1/22):** SQL continued, Logic & relational calculus
- **Lecture 3 (Tue 1/26):** Relational calculus & relational algebra
- **Lecture 4 (Fri 1/29):** Relational algebra & Codd's theorem, Datalog
- **Lecture 5 (Tue 2/2):** Datalog and more expressive variations
- **Lecture 6 (Fri 2/5): (A1 due)** Datalog vs. stable model semantics
- **Lecture 7 (Tue 2/9):** Datalog evaluation strategies, NoSQL

Pointers to relevant concepts & supplementary material:

- **1. SQL**: SQL refresher [SAMS'12], [Cow'03] Ch3 & Ch5, [Complete'08] Ch6
- **2. Logic**: First-order logic, relational calculus: [Barland+'08] 4.1.2 & 4.2.1 & 4.4, [Genesereth+] Ch6, [Halpern+'01], [Cow'03] Ch4.3 & 4.4, [Elmasri, Navathe'15] Ch8.6 & Ch8.7, [Silberschatz+'10] Ch6.2 & Ch6.3 [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4
- **3. Algebra**: Relational algebra, Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'10] Ch6.1, [Alice'95] Ch4.4 & Ch5.4
- **4. Datalog**: Datalog, stable model semantics: [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [Gatterbauer, Suciu'10]
- **5. Data models**: Alternative data models, NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Source: https://northeastern-datalab.github.io/cs7240/sp21/

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins (CONT.)
  - Top-k

Coalesce function

M

a
1
2

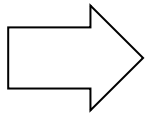
N

a
2
3

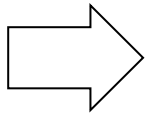
```
SELECT M.a, N.a, COALESCE(M.a, N.a) as b
FROM M
FULL JOIN N
ON M.a = N.a
```

COALESCE: takes first non-NULL value

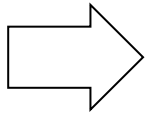
```
SELECT COALESCE(1, NULL)
```



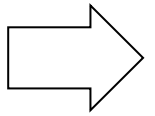
```
SELECT COALESCE(NULL, 3)
```



```
SELECT COALESCE(1, 2)
```

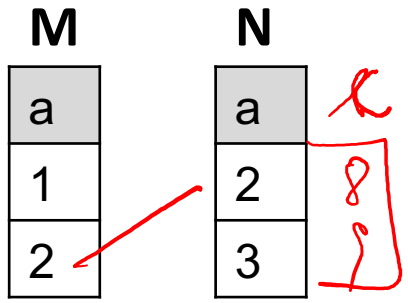


```
SELECT COALESCE(NULL, NULL)
```



?

Coalesce function




```
SELECT M.a, N.a, COALESCE(M.a, N.a) as b
FROM M
FULL JOIN N
ON M.a = N.a
```

Result

M.a	N.a	b

? ?

COALESCE: takes first non-NULL value

```
SELECT COALESCE(1, NULL)
```
⇨ 1

```
SELECT COALESCE(NULL, 3)
```
⇨ 3

```
SELECT COALESCE(1, 2)
```
⇨ 1

```
SELECT COALESCE(NULL, NULL)
```
⇨ NULL

Coalesce function

M

a
1
2

N

a
2
3

```
SELECT M.a, N.a, COALESCE(M.a, N.a) as b
FROM M
FULL JOIN N
ON M.a = N.a
```

Result

M.a	N.a	b
1	NULL	1
2	2	2
NULL	3	3

COALESCE: takes first non-NULL value

```
SELECT COALESCE(1, NULL)
```
→ 1

```
SELECT COALESCE(NULL, 3)
```
→ 3

```
SELECT COALESCE(1, 2)
```
→ 1

```
SELECT COALESCE(NULL, NULL)
```
→ NULL

Coalesce, Natural Outer Join, Union

M

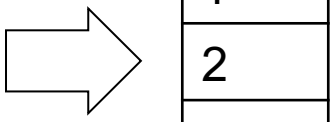
a
1
2

N

a
2
3

```
SELECT *
FROM M
NATURAL FULL JOIN N
```

Result



a
1
2
3

Natural full join models "coalesce"

Join vs. Union – it is actually the same:
Union is a special case of a join ☺
(under set semantics)

Commutativity & Associativity

Multiplication

$$(3 \cdot 2) \cdot 4 = 24$$

$$3 \cdot (2 \cdot 4) = 24$$

Multiplication is associative ☺

Order of operations can be exchanged

$$4 \cdot 2$$

and commutative ☺

Order of operands can be exchanged

Matrix multiplication

$$\begin{pmatrix} & \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 49 \end{pmatrix}$$

$$\begin{pmatrix} 11 & 16 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 3 \end{pmatrix} \cdot \left(\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 1 \end{pmatrix} \right) = \begin{pmatrix} 49 \end{pmatrix}$$

$$\begin{pmatrix} 5 \\ 13 \end{pmatrix}$$

Matrix multipl. is associative ☺

$$\begin{pmatrix} 3 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

#col ≠ #row

... but *not* commutative ☹

It turns out this is mainly a problem of syntax, not semantics. Einstein notation solves that. See e.g. Laue et al. *A Simple and Efficient Tensor Calculus*. AAAI 2020. https://arxiv.org/abs/2010.03313

Commutativity & Associativity

R

A	B
1	2

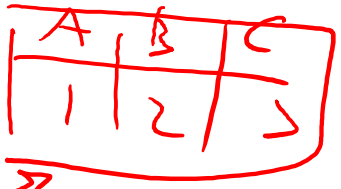
S

B	C
2	3

T

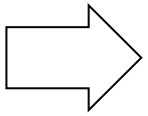
A	C
4	5

Outer joins



```
SELECT A, B, C
FROM (R
NATURAL FULL JOIN S)
NATURAL FULL JOIN T
```

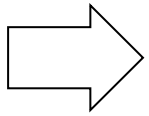
Result



?

```
SELECT  A, B, C
FROM R
NATURAL FULL JOIN (S
NATURAL FULL JOIN T)
```

Result



?

Commutativity & Associativity


Outer joins

R

A	B
1	2

S

B	C
2	3

T

A	C
4	5

SUBSUMTION

A	B
1	2
1	NULL

A	B	C
1	2	3
4	⊥	5

```
SELECT A, B, C
FROM (R
NATURAL FULL JOIN S)
NATURAL FULL JOIN T
```

Result

A	B	C
4	NULL	5
1	2	3

```
SELECT  A, B, C
FROM R
NATURAL FULL JOIN (S
NATURAL FULL JOIN T)
```

Result

?

Commutativity & Associativity

Outer joins

R

A	B
1	2

S

B	C
2	3

T

A	C
4	5

```
SELECT A, B, C
FROM (R
NATURAL FULL JOIN S)
NATURAL FULL JOIN T
```

Result

A	B	C
4	NULL	5
1	2	3

```
SELECT  A, B, C
FROM R
NATURAL FULL JOIN (S
NATURAL FULL JOIN T)
```

Result

A	B	C
4	NULL	5
NULL	2	3
1	2	NULL

Thus outer joins are not associative! (but they are commutative)

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

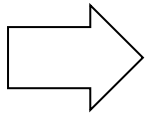
Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Accommodations)
NATURAL FULL JOIN Sites
```

Result



?

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

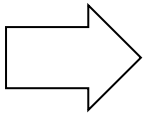
Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Accommodations)
NATURAL FULL JOIN Sites
```



Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto	diverse	Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada					Mount Logan
UK	London				Buckingham
UK	London				Hyde Park
UK		temperate			
Bahamas	Nassau	Tropical	Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

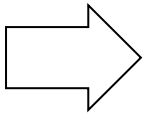
Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Accommodations)
NATURAL FULL JOIN Sites
```



Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto	diverse	Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada					Mount Logan
UK	London				Buckingham
UK	London				Hyde Park
UK		temperate			
Bahamas	Nassau	Tropical	Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Accommodations)
NATURAL FULL JOIN Sites
```

Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto	diverse	Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada					Mount Logan
UK	London				Buckingham
UK	London				Hyde Park
UK		temperate			
Bahamas	Nassau	Tropical	Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Sites)
NATURAL FULL JOIN Accommodations
```

Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto		Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada		diverse			Mount Logan
UK	London	temperate			Buckingham
UK	London	temperate			Hyde Park
Bahamas		Tropical			
Bahamas	Nassau		Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM (Climates
NATURAL FULL JOIN Sites)
NATURAL FULL JOIN Accommodations
```

Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto		Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada		diverse			Mount Logan
UK	London	temperate			Buckingham
UK	London	temperate			Hyde Park
Bahamas		Tropical			
Bahamas	Nassau		Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Data Sources – Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

Country	City	Site
Canada	London	Air show
Canada		Mount Logan
UK	London	Buckingham
UK	London	Hyde Park

```
SELECT  *
FROM FULL DISJUNCTION(Climates,
Accommodations, Sites)
```

FD: variation of the join operator that maximally combines join consistent tuples from connected relations, while preserving all information in the relations.

Not available in SQL! Will discuss later in class in more detail.

Result

Country	City	Climate	Hotel	Stars	Site
Canada	Toronto	diverse	Plaza	4	
Canada	London	diverse	Ramada	3	Air Show
Canada		diverse			Mount Logan
UK	London	temperate			Buckingham
UK	London	temperate			Hyde Park
Bahamas	Nassau	tropical	Hilton		

Example from: Cohen, Fadida, Kanza, Kimelfeld, Sagiv. "Full Disjunctions: Polynomial-Delay Iterators in Action", VLDB 2006. http://vldb.org/conf/2006/p739-cohen.pdf

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Outer joins
  - Top-k

Sorting & Top-*k* evaluation with SQL

R

A	B
1	0
2	0
3	0
4	1

S

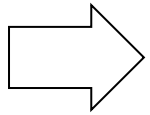
B	C
0	1
0	1
0	1
0	2

T

C	D
1	1
1	2
2	3
2	4

```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

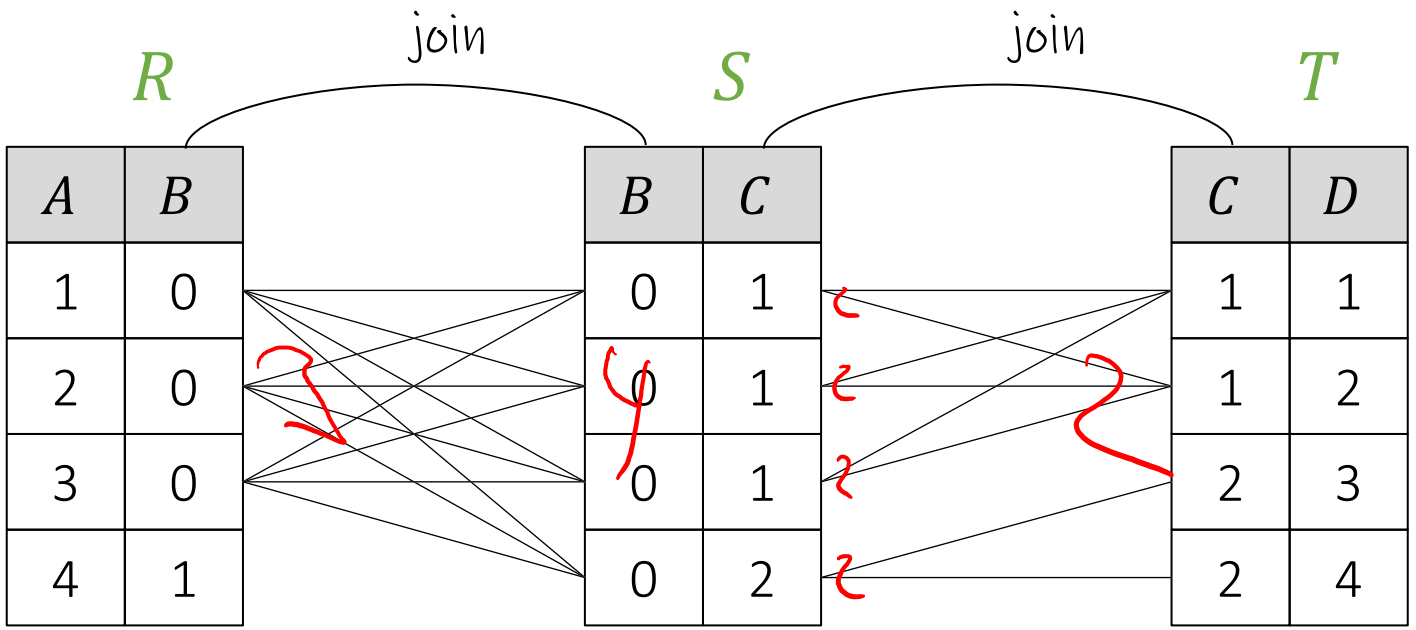
Result



How many results do we get?

?

Sorting & Top-*k* evaluation with SQL



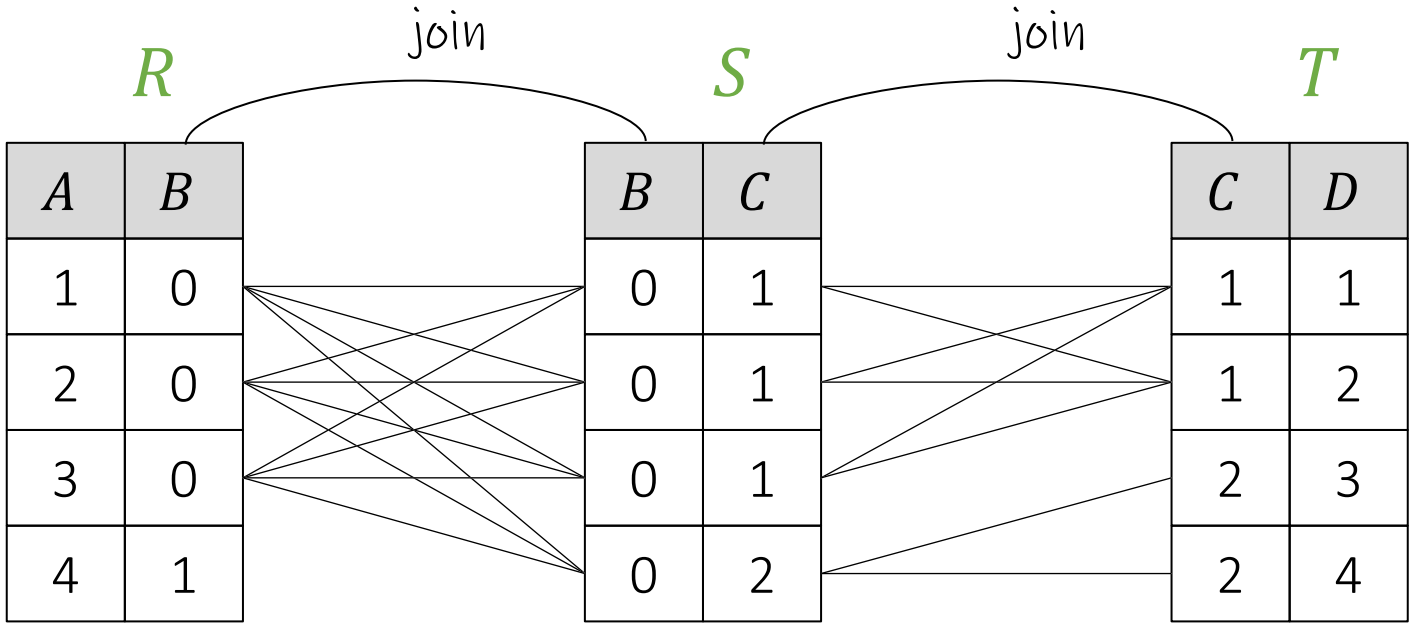

```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

Result

How many results do we get?

?

Sorting & Top-*k* evaluation with SQL



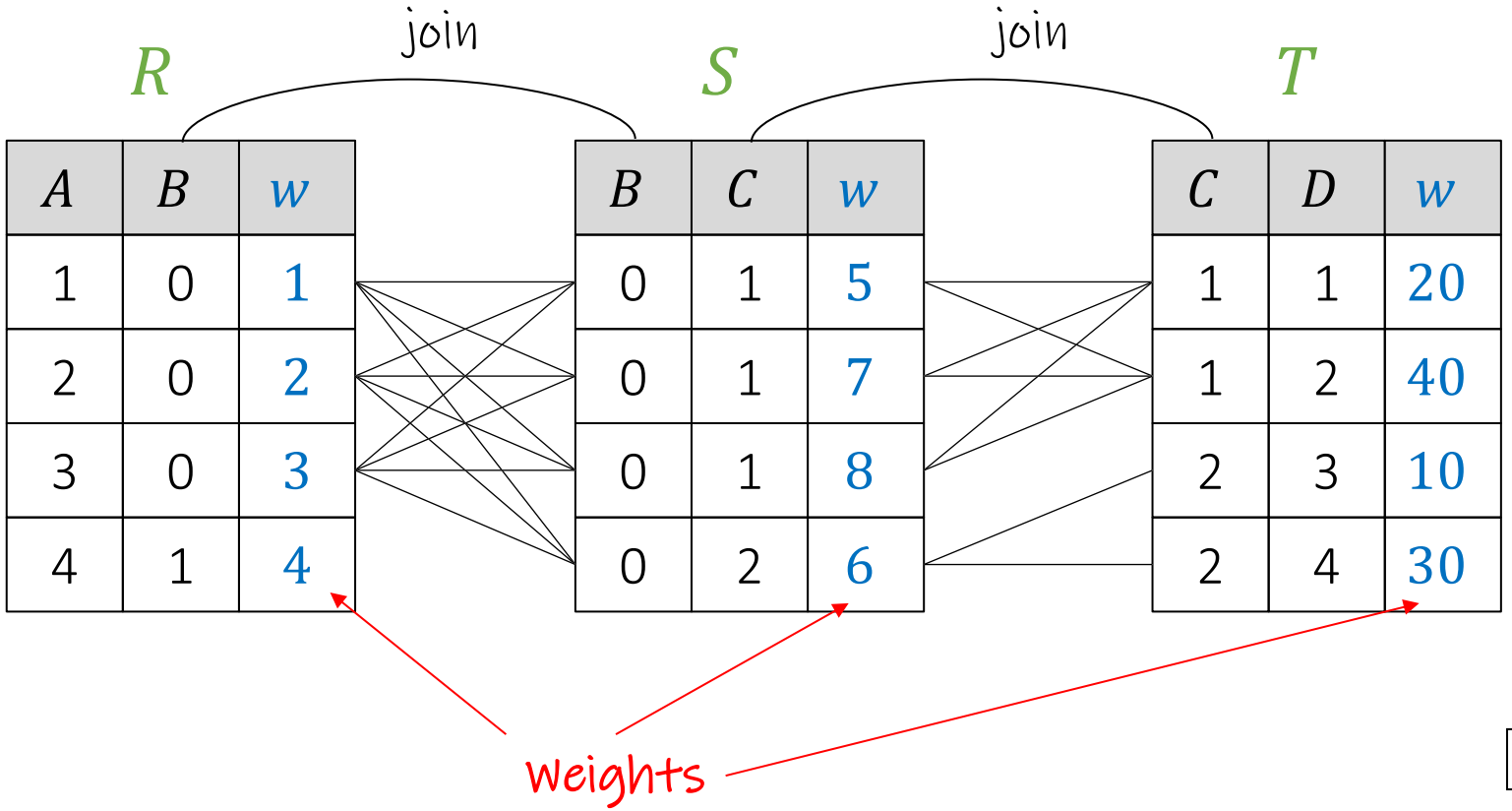

```
select   A, R.B, S.C, D

from     R, S, T
where    R.B=S.B and S.C=T.C
```

24 total results

Result

A	B	C	D
1	0	2	3
2	0	2	3
3	0	2	3
1	0	1	1
2	0	1	1
3	0	1	1
1	0	1	1
...	...	...	...

Sorting & Top-*k* evaluation with SQL




```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
```

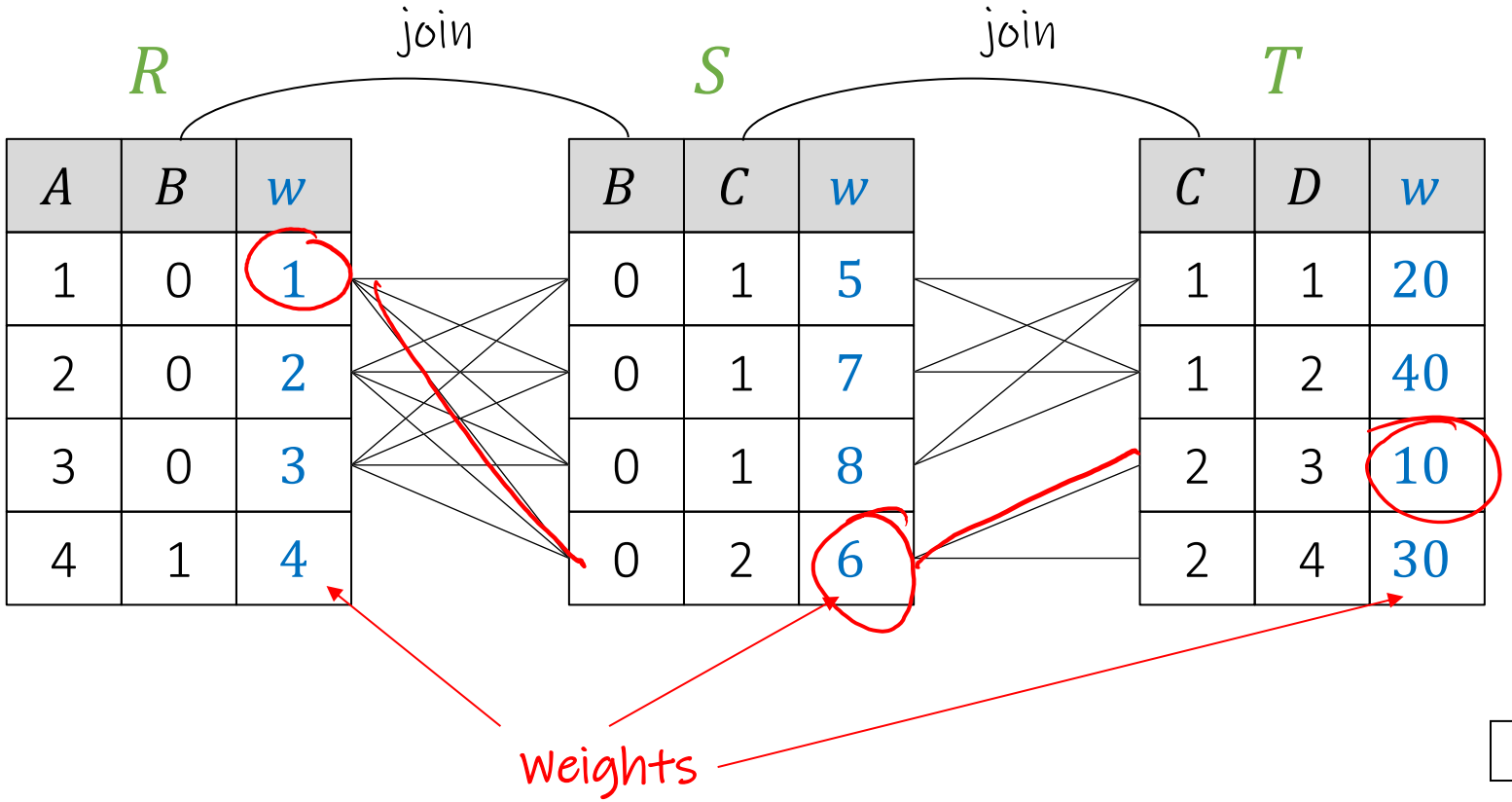
Result

A	B	C	D	weight

What do we get now?

?

Sorting & Top-*k* evaluation with SQL



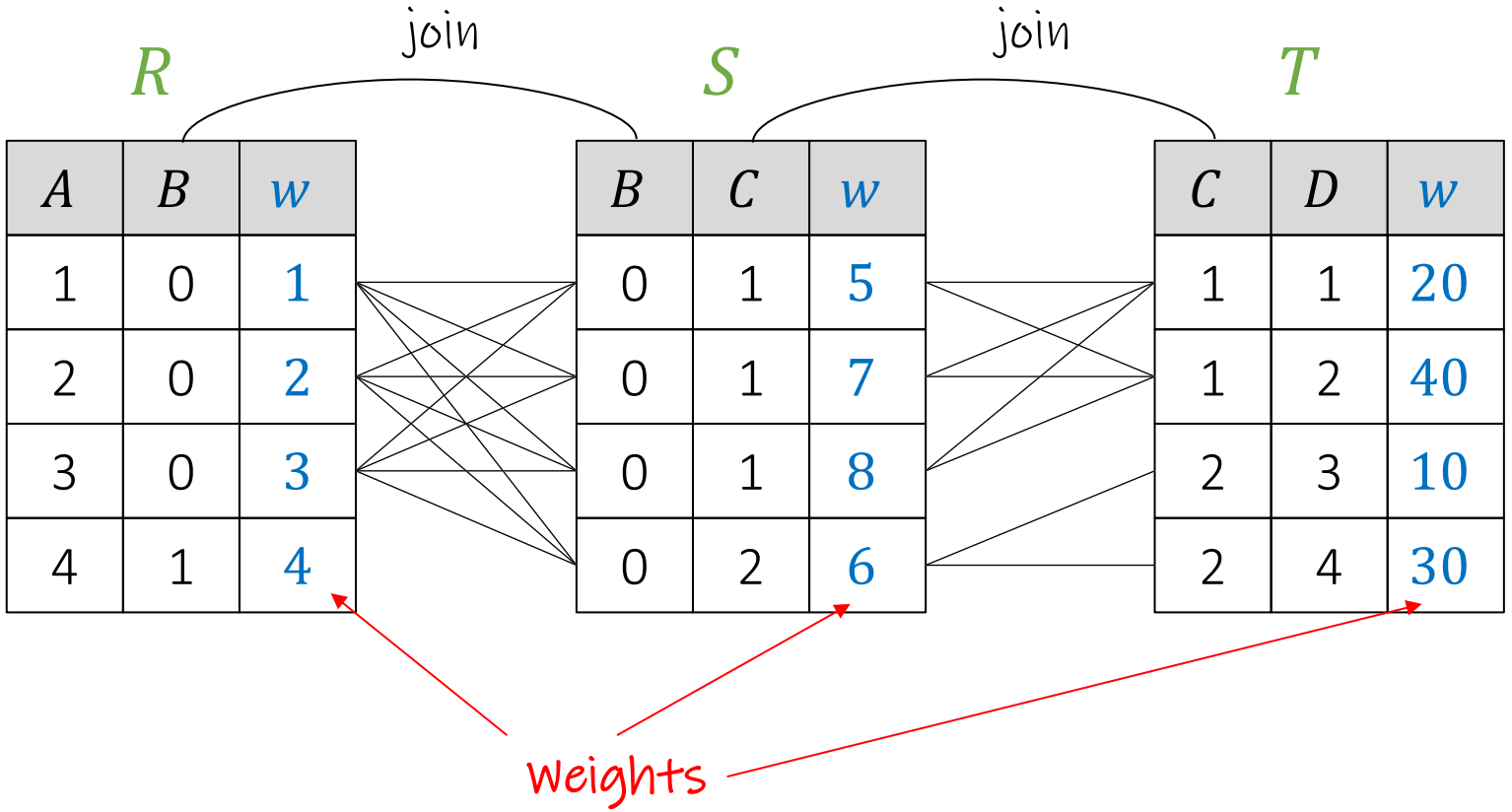

```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
```

Return all 24 results in order of sum of weights

Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

Sorting & Top-*k* evaluation with SQL



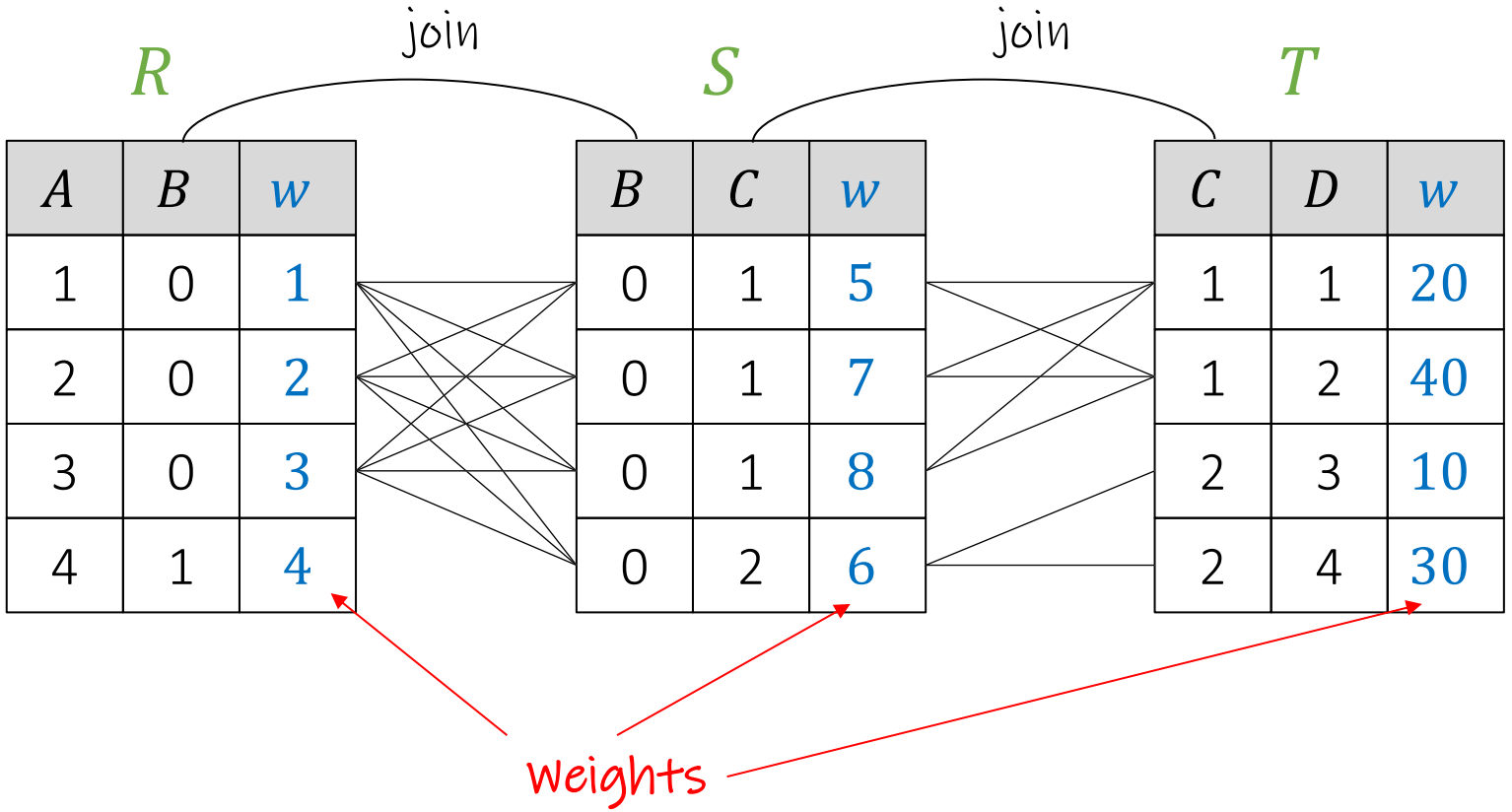

```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
limit    6
```

What do we get now?

Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

Sorting & Top-*k* evaluation with SQL

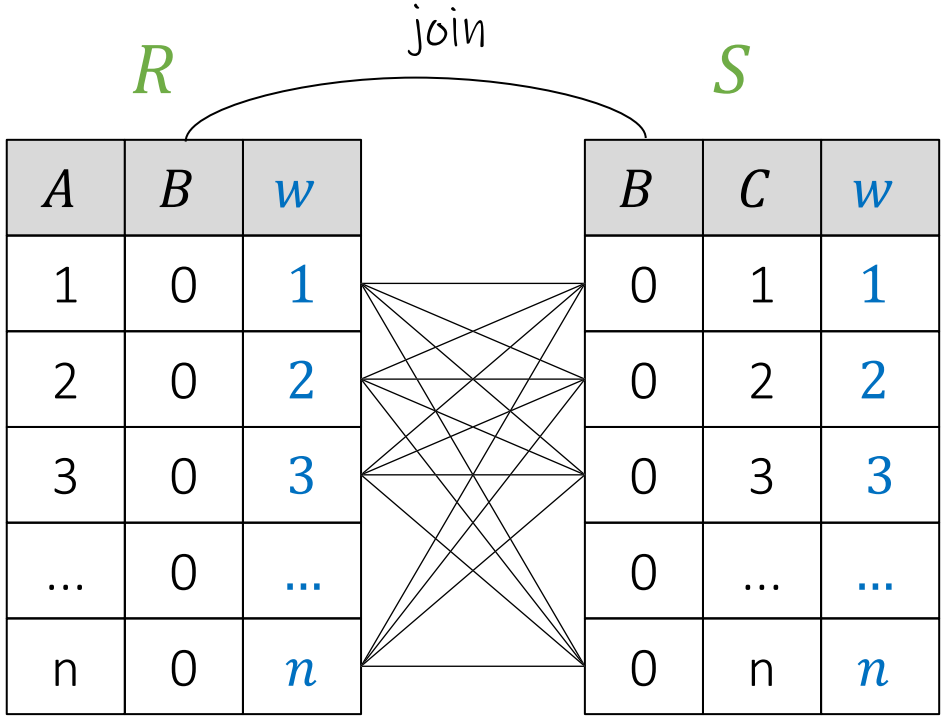



```
select   A, R.B, S.C, D,
         R.w + S.w + T.w as weight
from     R, S, T
where    R.B=S.B and S.C=T.C
order by weight ASC
limit    6
```

Result

A	B	C	D	weight
1	0	2	3	17
2	0	2	3	18
3	0	2	3	19
1	0	1	1	26
2	0	1	1	27
3	0	1	1	28
1	0	1	1	28
...	...	...	...	...

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
select   A, R.B, S.C,
         R.w + S.w as weight
from     R, S
where    R.B=S.B
order by weight ASC
limit    1
```

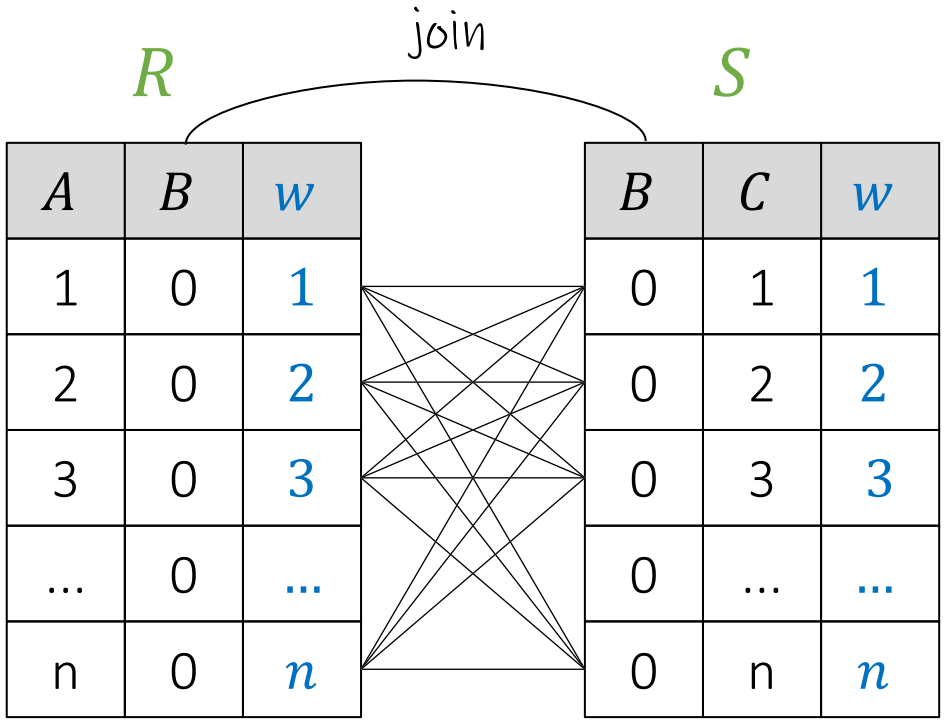
Result

A	B	C	weight
1	0	1	2
1	0	2	3
2	0	1	3
3	0	1	4
2	0	2	4
1	0	3	4
4	0	1	5
…	…	…	…
n	0	n	n*n

Can you see any possible problem of this query as n gets bigger?

?

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
select   A, R.B, S.C,
         R.w + S.w as weight
from     R, S
where    R.B=S.B
order by weight ASC
limit    1
```

Result

A	B	C	weight
1	0	1	2
1	0	2	3
2	0	1	3
3	0	1	4
2	0	2	4
1	0	3	4
4	0	1	5
…	…	…	…
n	0	n	n*n

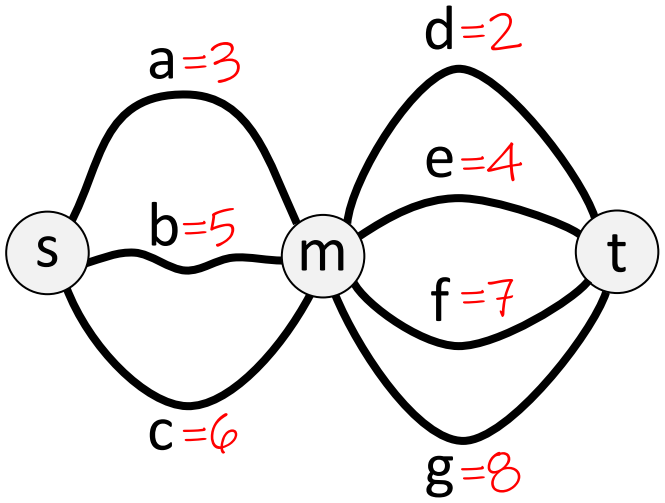
n^2 total results. But we are only interested in the top-1.

Problem: Database first calculates all n^2 results before sorting.

Question: is there any way to push the sorting behind the join?

?

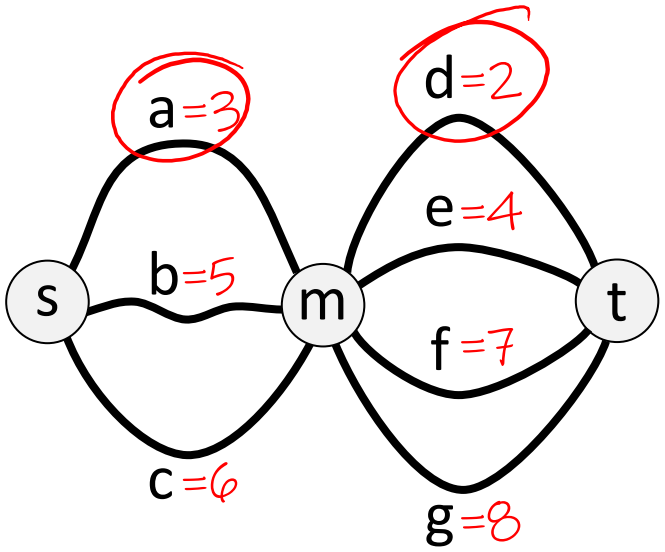
Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

?

Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

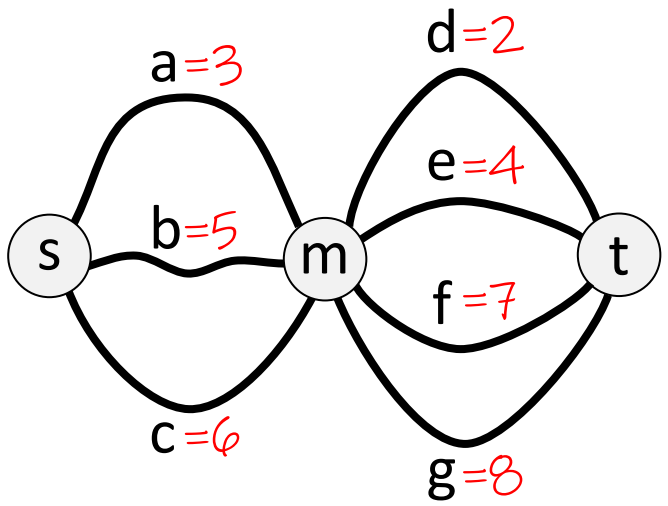
Answer: 5 = 3 + 2

min [a + d, a + e, a + f, a + g, ..., c + g]

min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

?

Digression: Distributivity = efficient factorization



What is the shortest path from s to t?

Answer: 5 = 3 + 2

Principle of optimality from Dynamic Programming: *irrespective of the initial state and decision, an optimal solution continues optimally from the resulting state*

min [a + d, a + e, a + f, a + g, ..., c + g]

min[3+2, 3+4, 3+7, 3+8, ..., 6+8]

= min [a, b, c] + min [d, e, f, g]

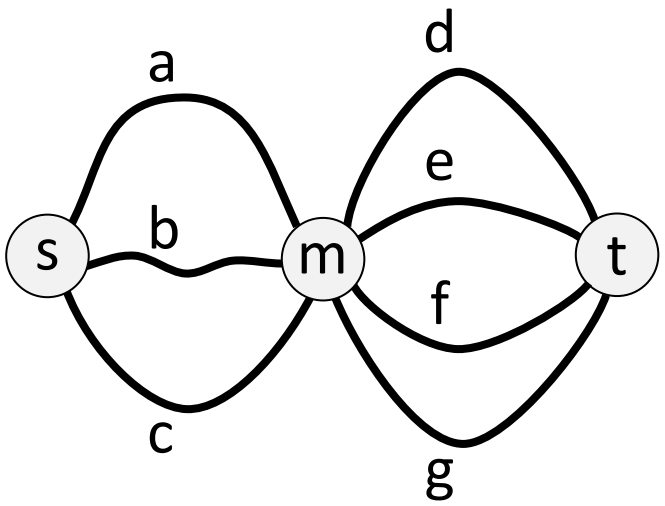
min[3,5,6] + min[2,4,7,8]

$(x+y)z = xz + yz$

min[x,y]+z = min[(x+z), (y+z)]

(+ distributes over min)

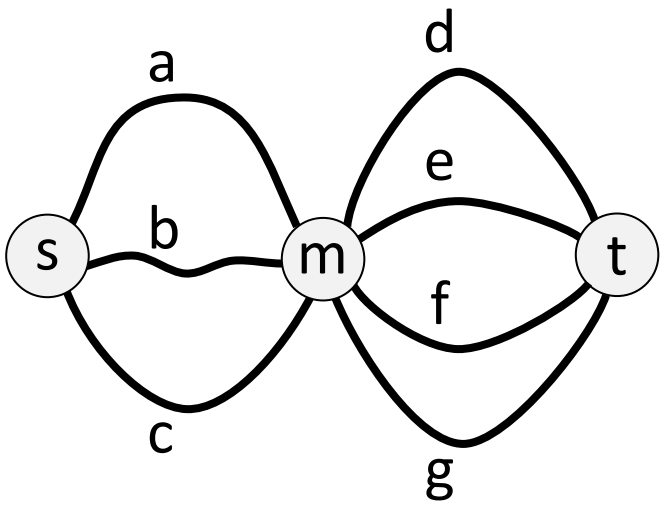
Digression: Distributivity = efficient factorization



How many paths are there from s to t?

?

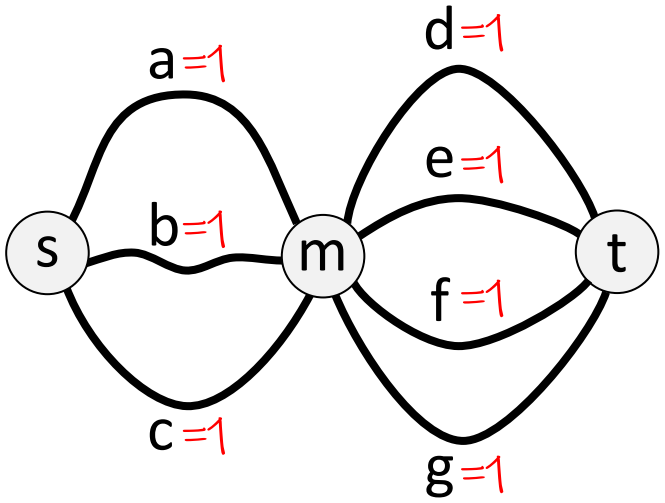
Digression: Distributivity = efficient factorization



How many paths are there from s to t?

Answer: $12 = 3 \cdot 4$

Digression: Distributivity = efficient factorization



How many paths are there from s to t?

Answer: 12 = 3 · 4

count [a·d, a·e, a · f, a · g, ..., c · g]

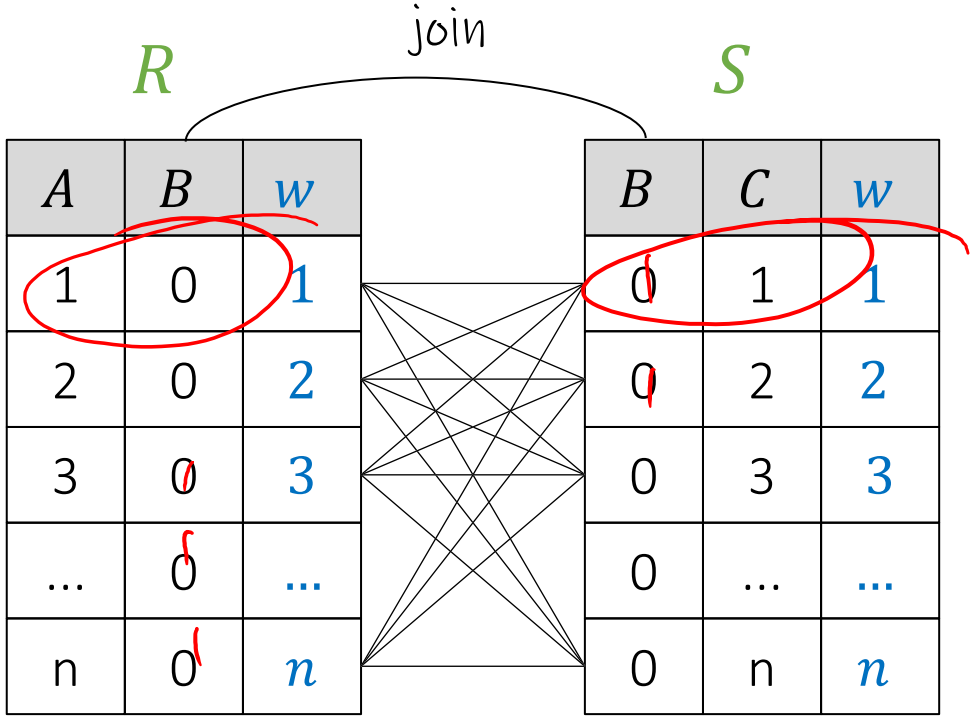
count[1·1, 1·1, 1·1, 1·1, ..., 1·1] (12)

= count [a, b, c] · count [d, e, f, g]

count[1,1,1] · count[1,1,1,1]

+[x,y] · z = +[x·z,y·z]
(· distributes over +)

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

```
-------------------------
-- Query 1
-------------------------
SELECT   A, R.B, S.C,
            R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

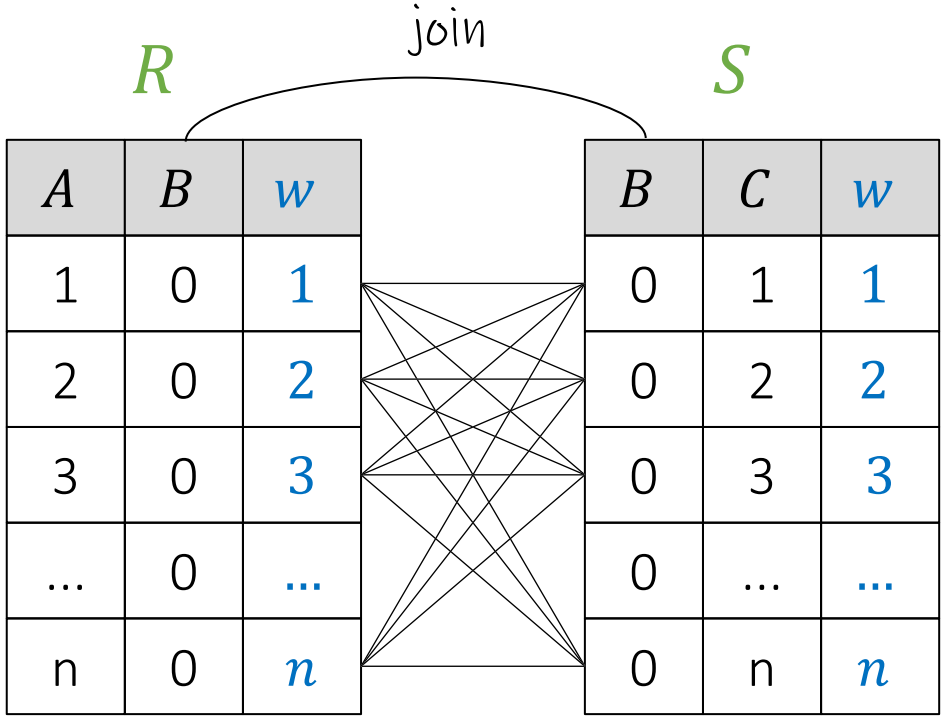
```
-------------------------
-- Query 2
-------------------------
```

?

n=1000: t_{Q1}= 0.88 sec t_{Q2}= ?

n=5000: t_{Q1}=18.6 sec t_{Q2}= ?

Top-k is evaluated inefficiently by modern DBMS's



A	B	w
1	0	1
2	0	2
3	0	3
…	0	…
n	0	n

B	C	w
0	1	1
0	2	2
0	3	3
0	…	…
0	n	n

Maximal intermediate result size is O(n) ☺
What is this algorithm called?

Dynamic programming

```
-------------------------
-- Query 1
-------------------------
SELECT   A, R.B, S.C,
           R.W + S.W as weight
FROM      R, S
WHERE    R.B=S.B
ORDER BY weight ASC
LIMIT    1;
```

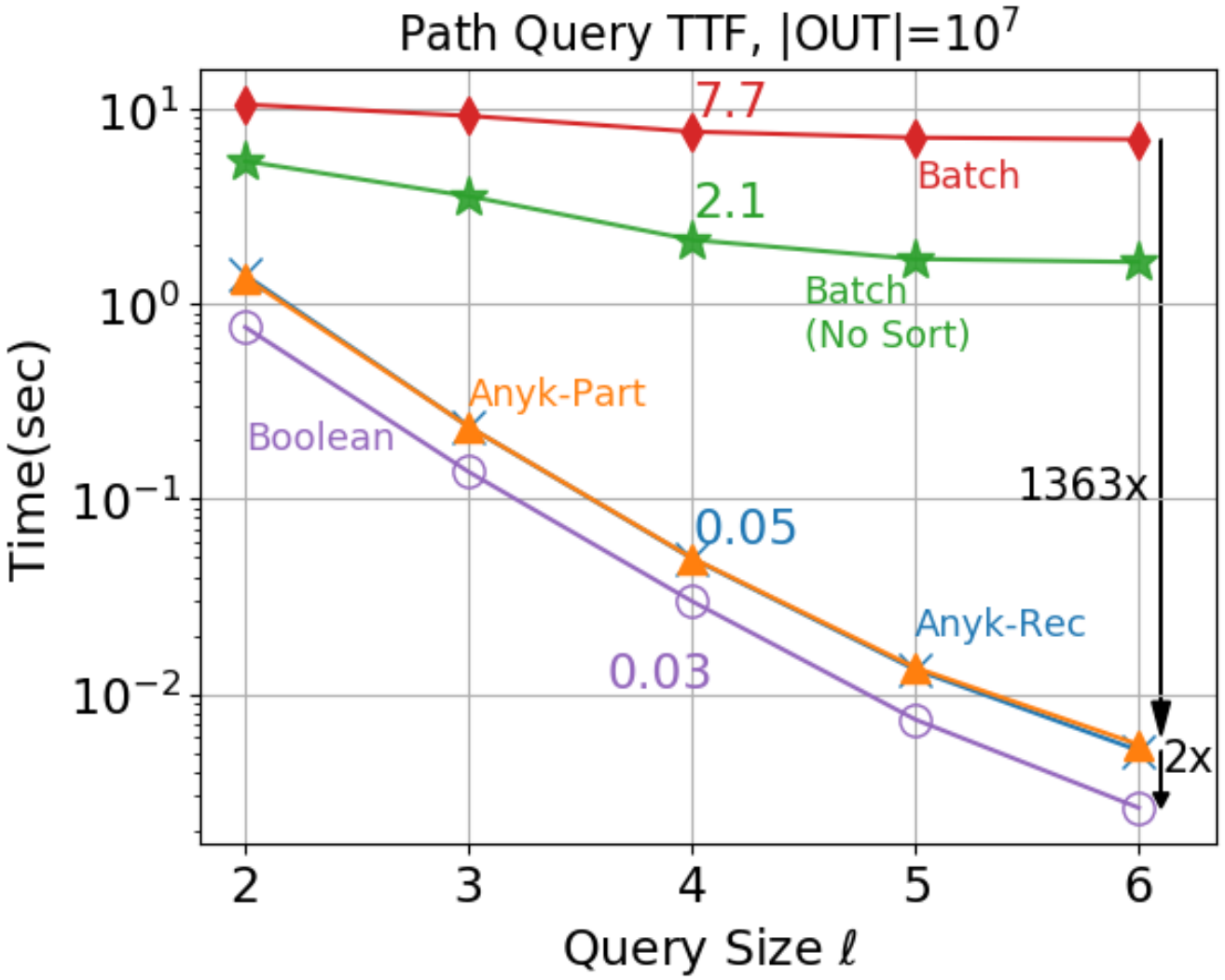
$O(n^2)$

```
-------------------------
-- Query 2
-------------------------
SELECT R.A, X.B, S.C, X.W as weight
FROM R, S,
     (SELECT T1.B, W1, W2, W1+W2 W
     FROM
          (SELECT B, MIN(W) W1
          FROM R
          GROUP BY B) T1,
          (SELECT B, MIN(W) W2
          FROM S
          GROUP BY B) T2
     WHERE T1.B = T2.B
     ORDER BY W ASC
     LIMIT 1) X
WHERE X.B = R.B
AND X.W1 = R.W
AND X.B = S.B
AND X.W2 = S.W
LIMIT   1;
```

$O(n)$

n=1000:	t_{Q1}= 0.88 sec	t_{Q2}=2 msec
n=5000:	t_{Q1}=18.6 sec	t_{Q2}=8 msec

Any-k: Faster and more versatile than Top-k



Path query with
constant size output
and increasing query size

Tziavalis+ [PVLDB'20], Tziavelis+ [SIGMOD'20 tutorial]

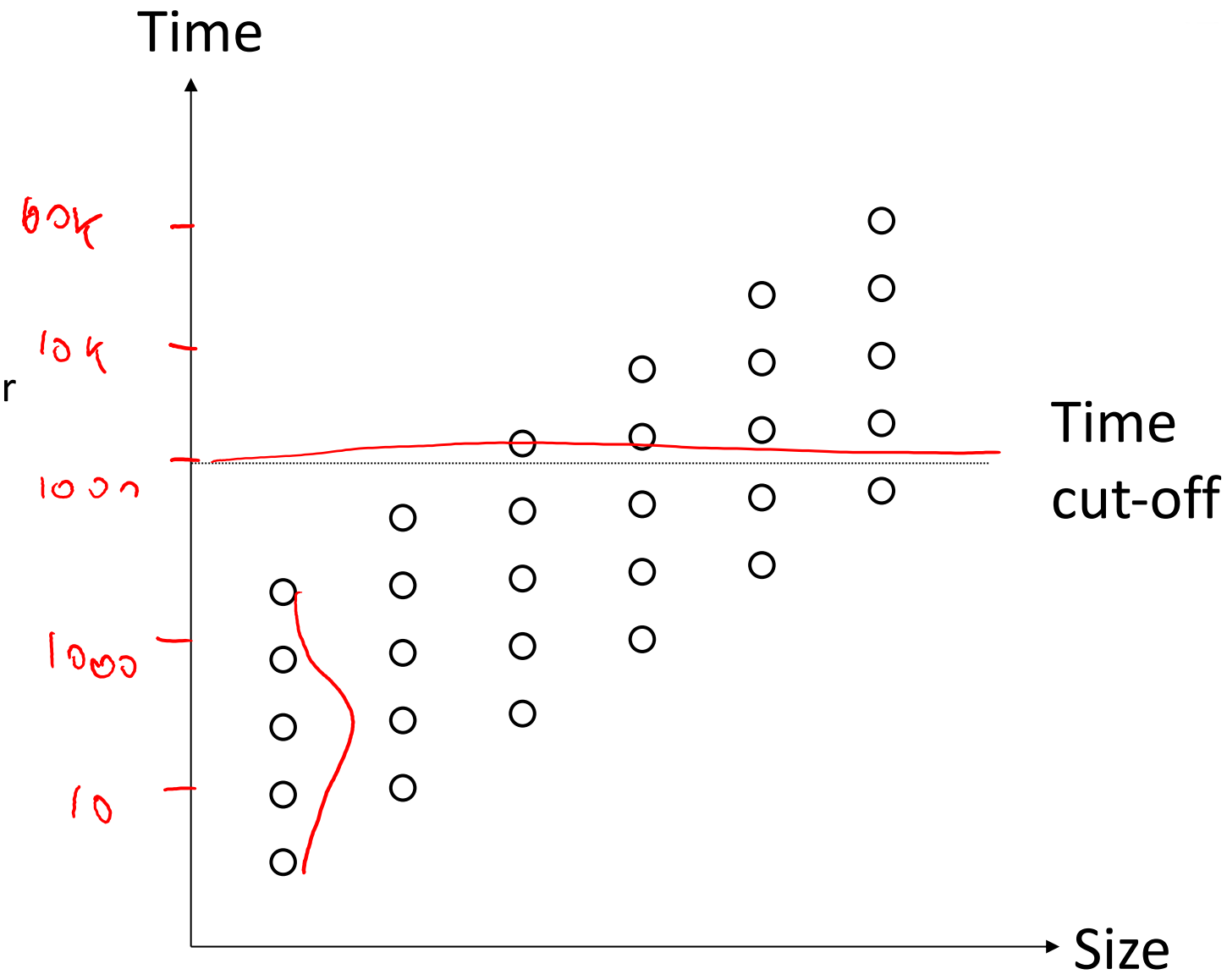
https://northeastern-datalab.github.io/anyk/
https://northeastern-datalab.github.io/topk-join-tutorial/
https://www.youtube.com/watch?v=KpUQayBuaQI&list=PL_72ERGKF6DR7kvGNwwjWlbpScKtGjt9R&index=2

Source: Tziavelis, Ajwani, Gatterbauer, Riedewald, Yang. Optimal Algorithms for Ranked Enumeration of Answers to Full Conjunctive Queries. PVLDB 2020.

Revisiting our question from first class

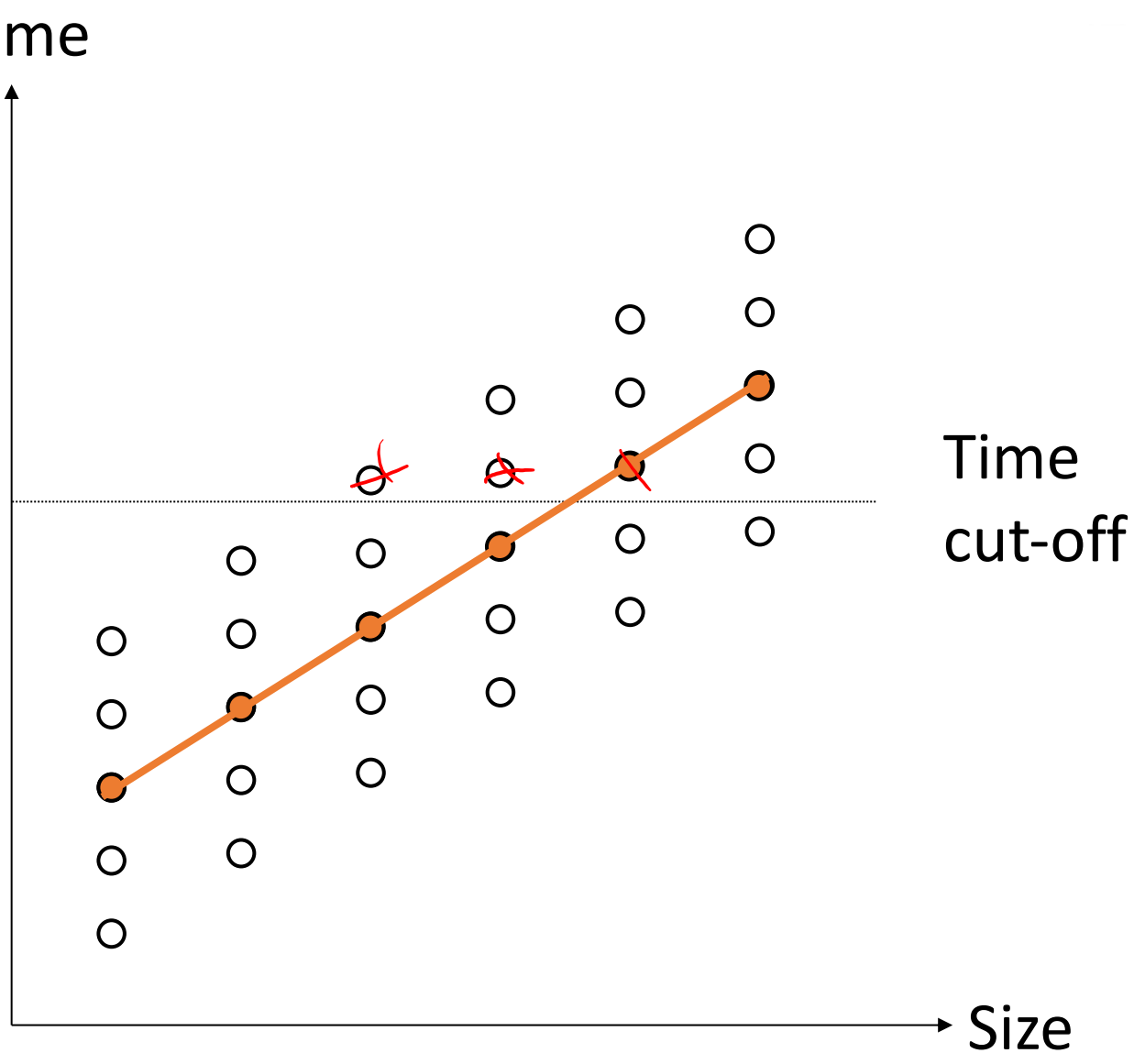
How to deal with cut-offs when binning

- These are the true points that you would get if you could run the experiments long enough.
  - Assume loglog scale
- However, we can't and thus in practice cut-off the experiments after some time.
- There is an overall trend, yet some variation for each experiment. We would still like to capture the trend with some smart aggregations



How to deal with cut-offs when binning

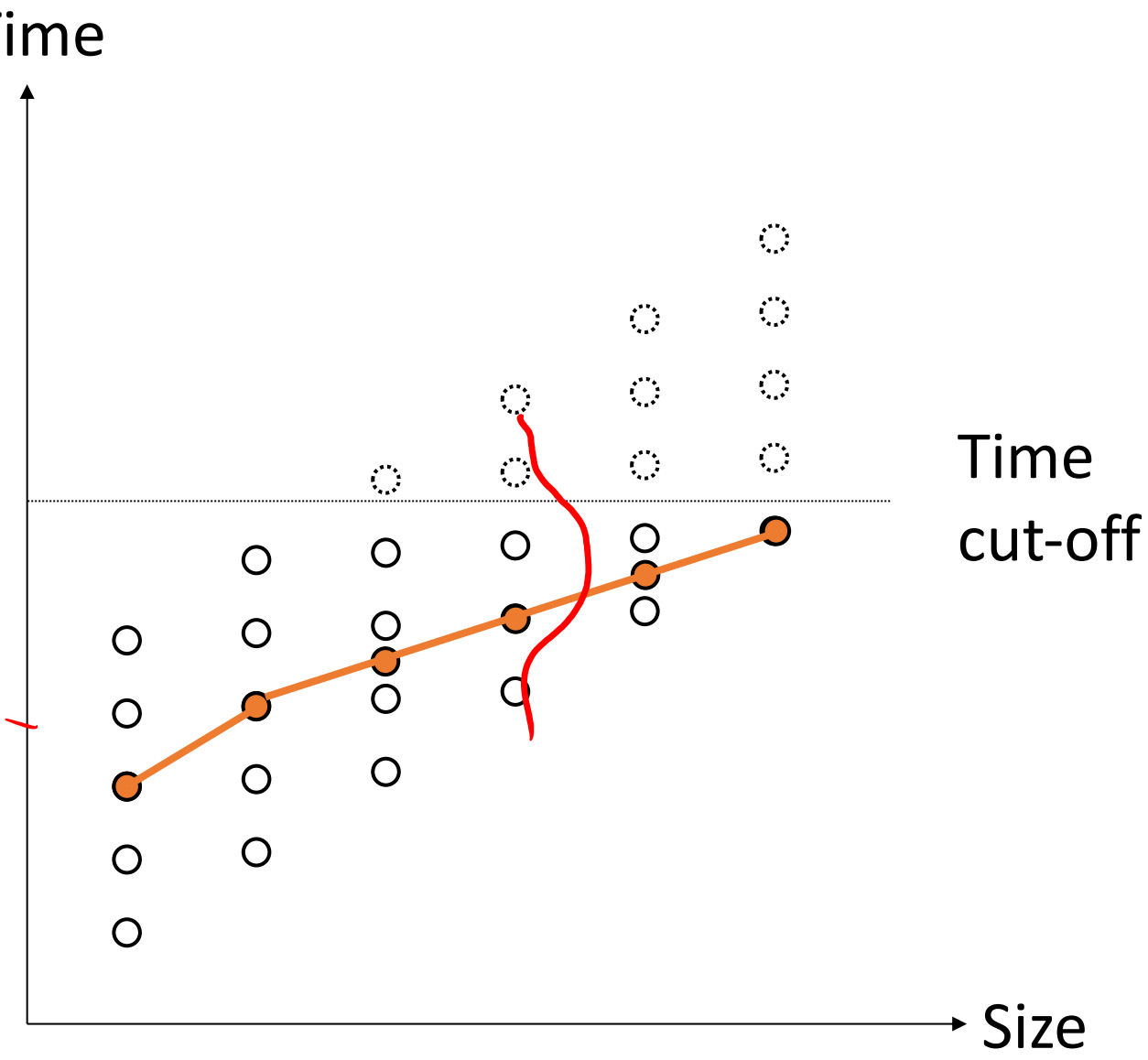
- Here is what the aggregate would look like like if we could get all points and then aggregated for each size



How to deal with cut-offs when binning

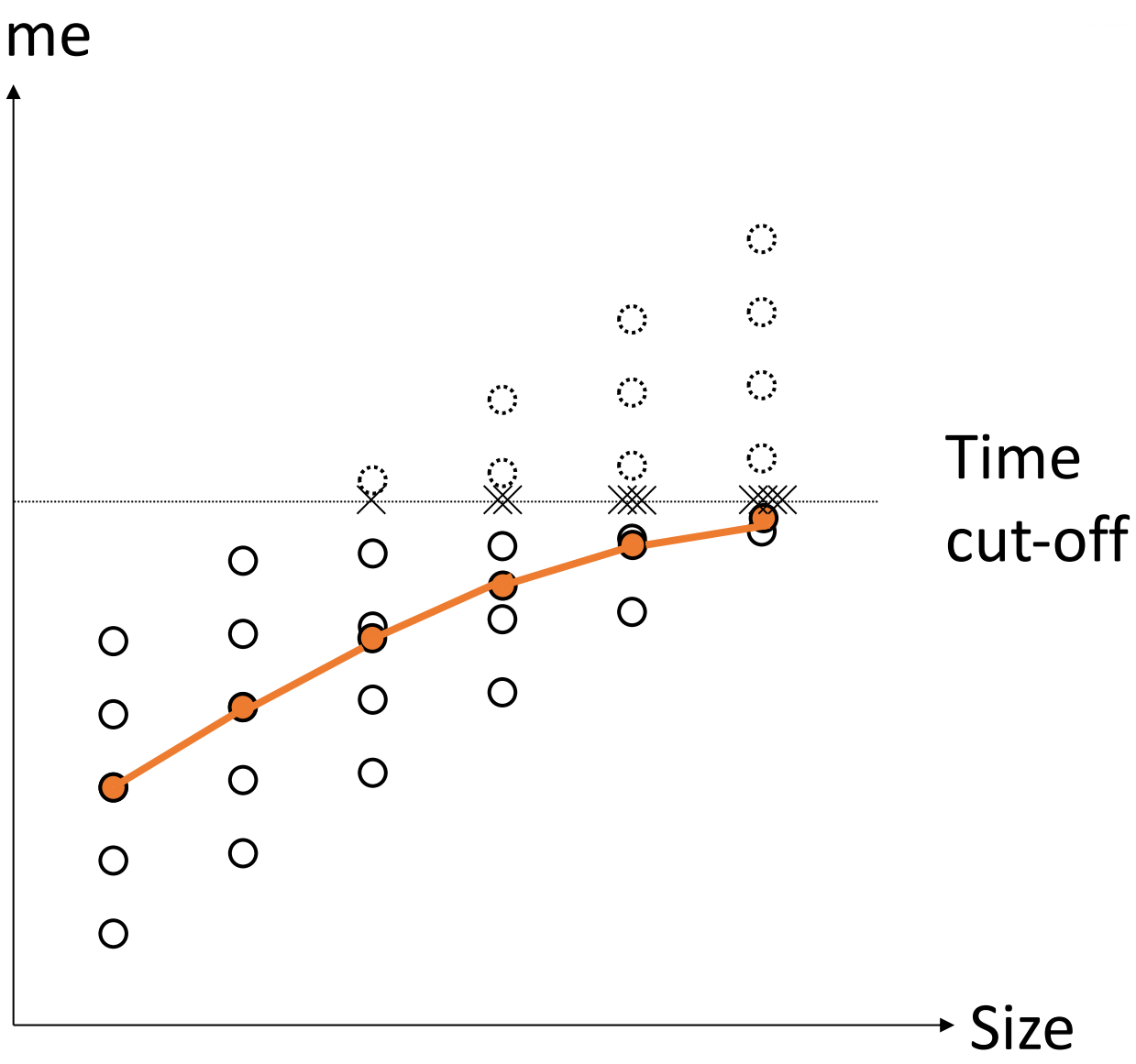
- Here is what happens if we throw away all those points that take longer than the cut-off, and only average over the "seen points"

- What would you do?



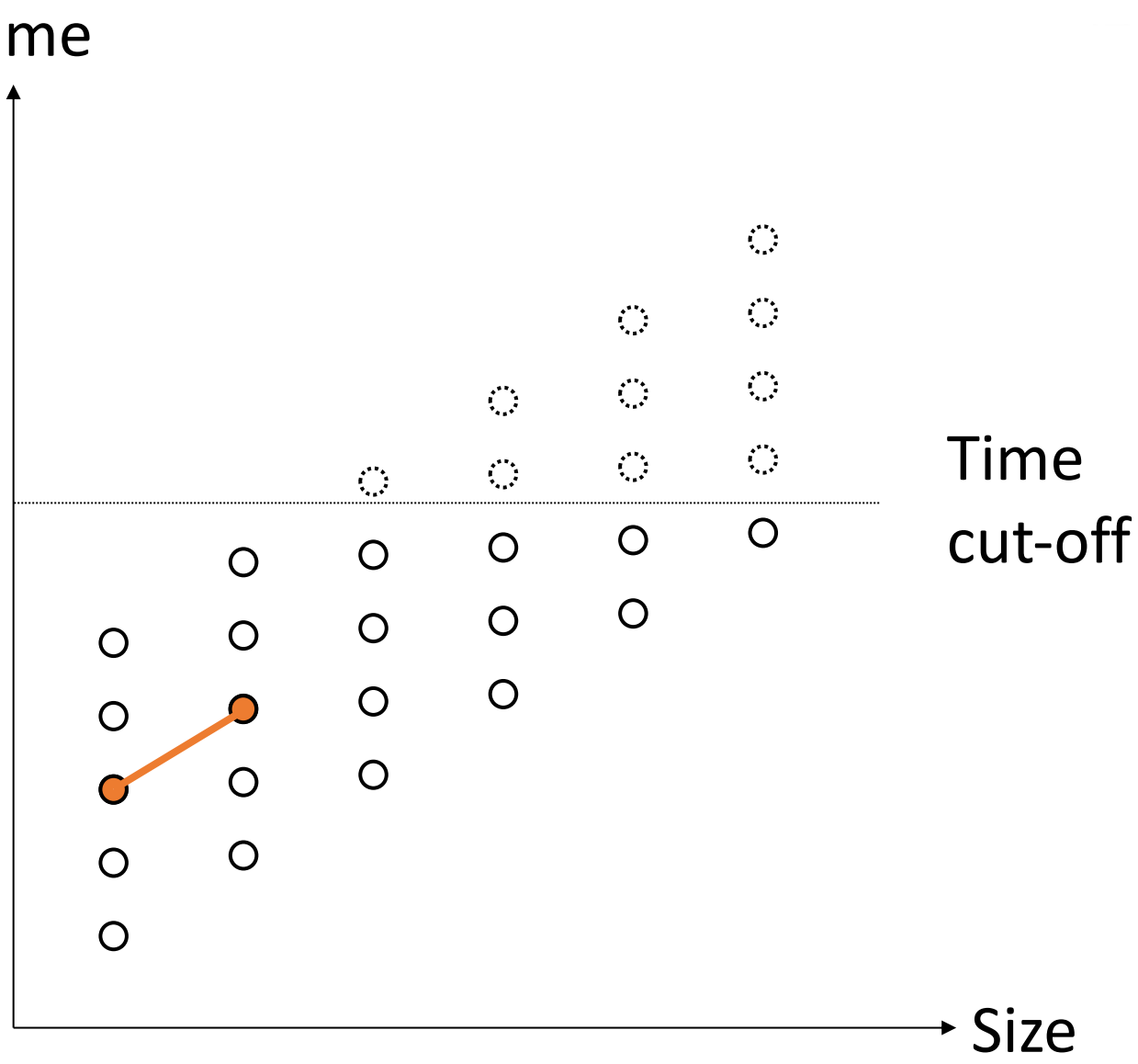
How to deal with cut-offs when binning

- Here is what happens if we cut the points off and still use the points, and then average



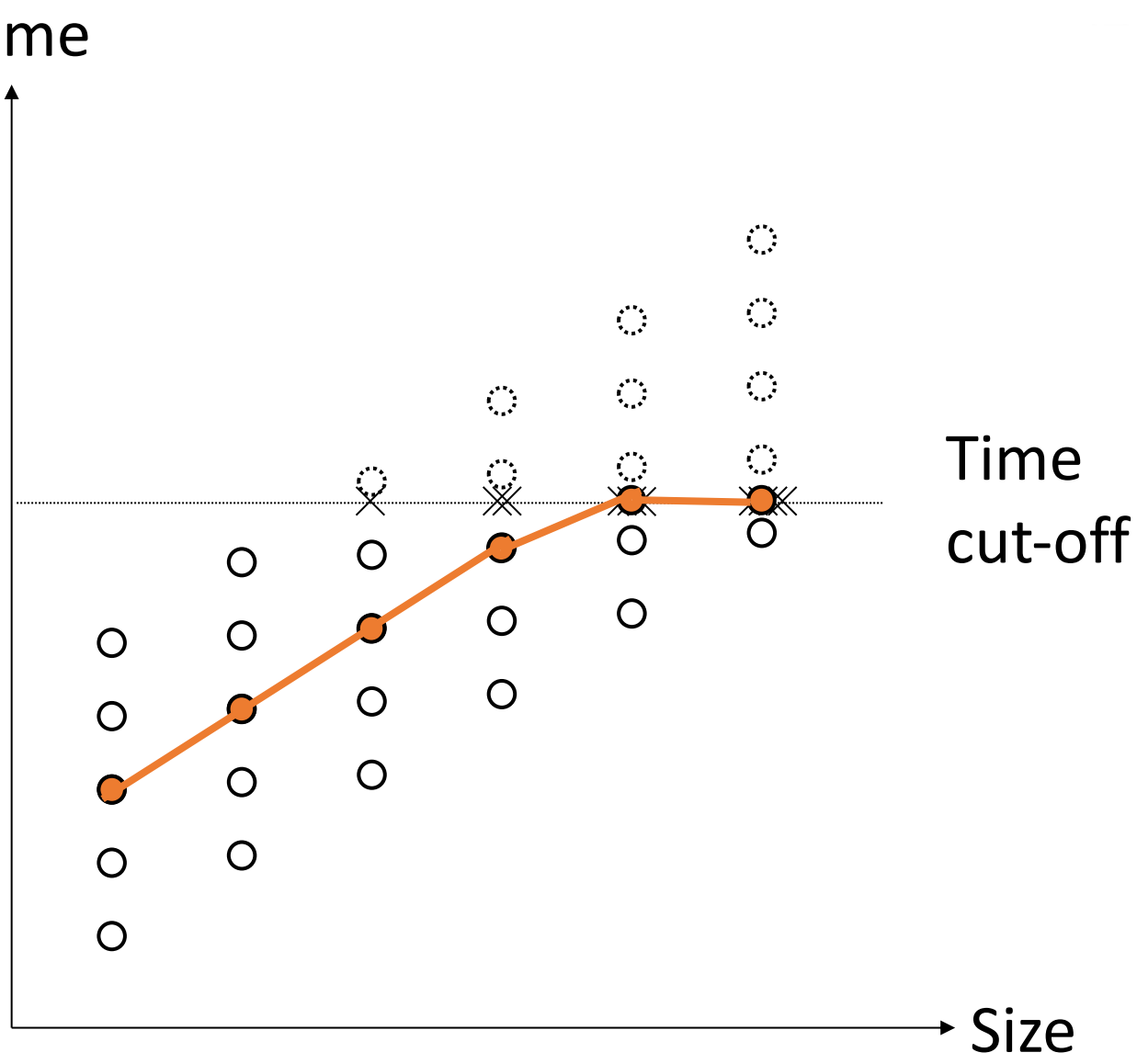
How to deal with cut-offs when binning

- Only use those sizes for which all experiments finish in time



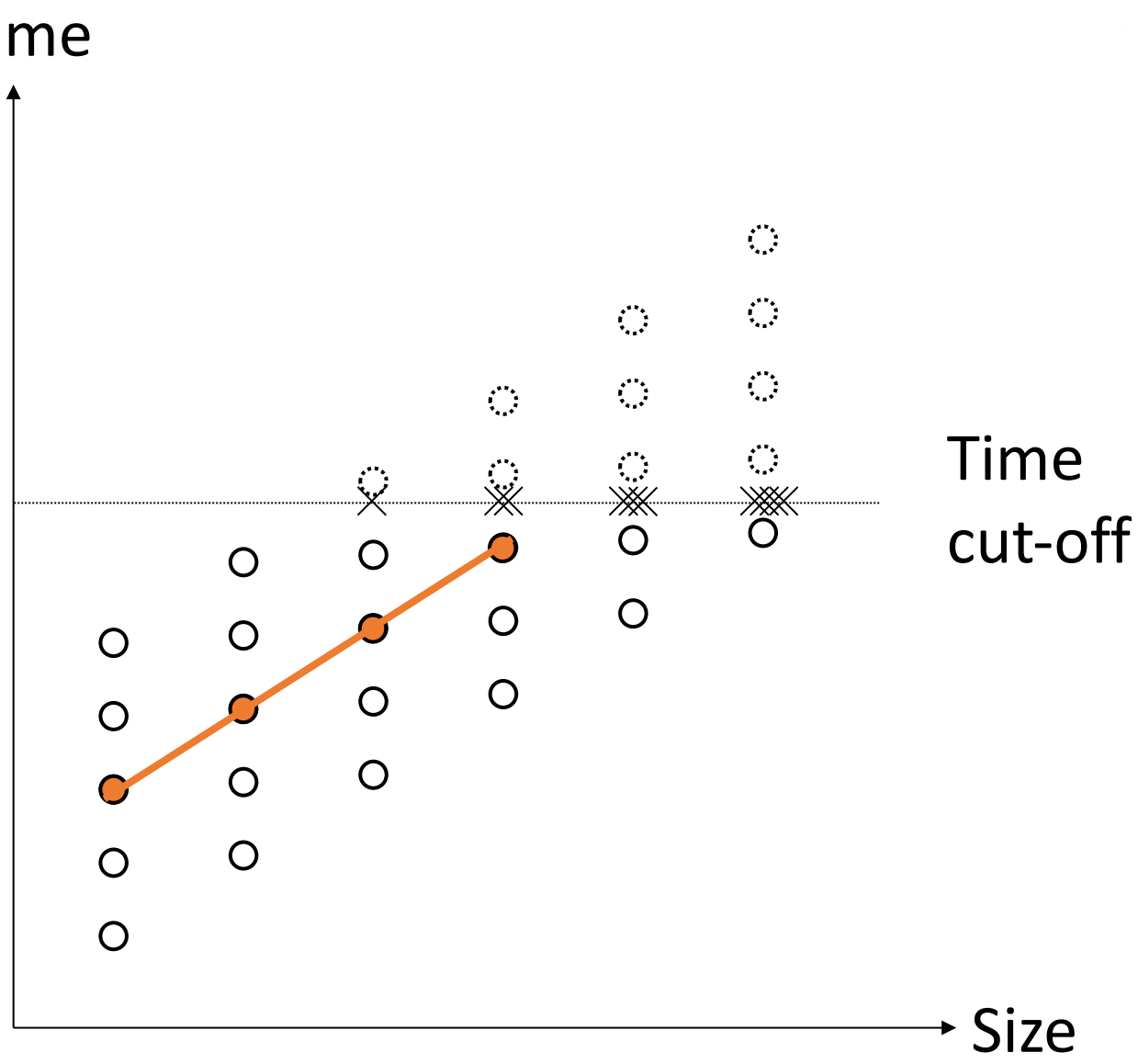
How to deal with cut-offs when binning

- Here is what happens if we take the median over all seen and cut-off points

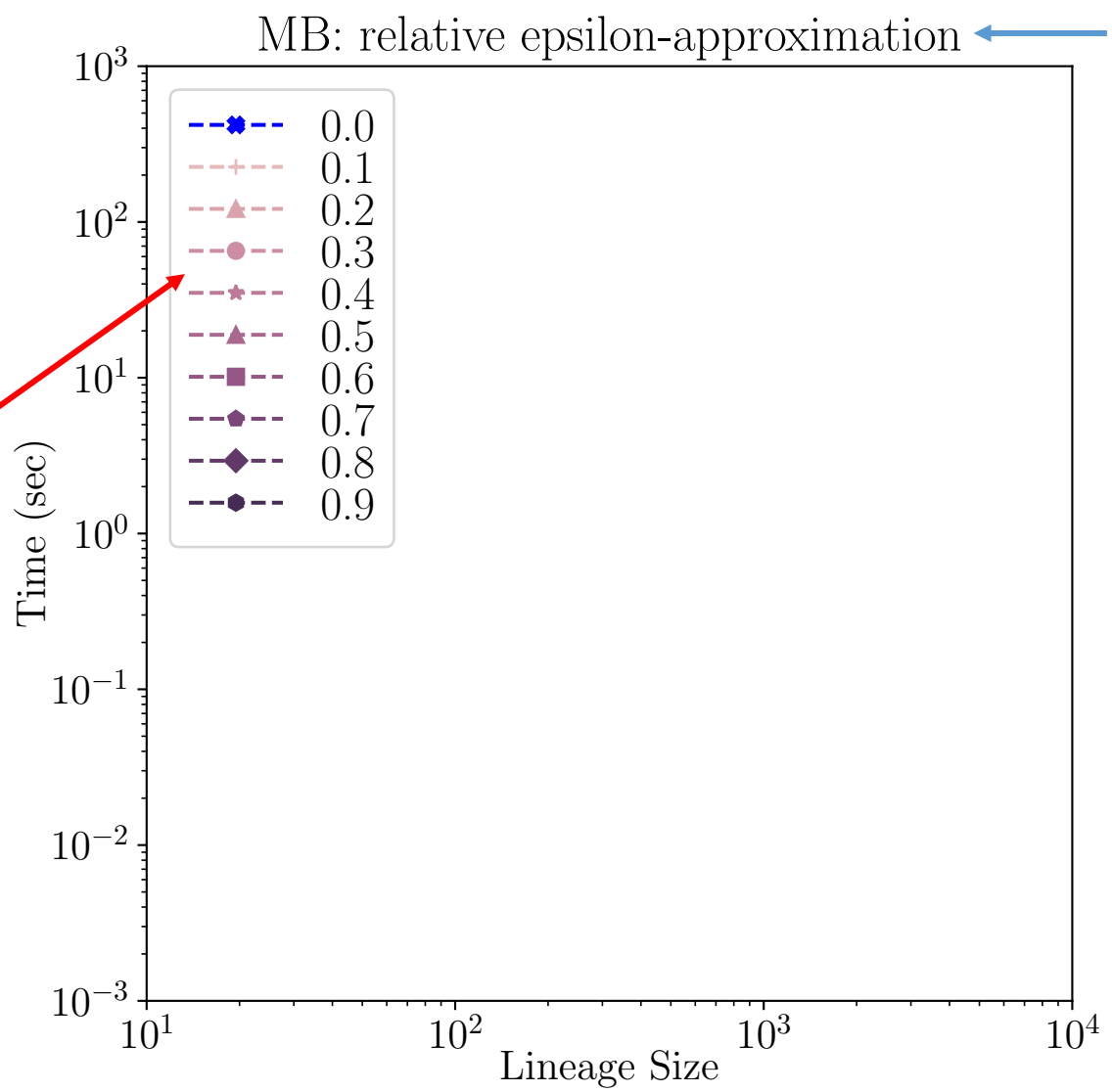


How to deal with cut-offs when binning

- Here is what happens if we take the median over all seen and cut-off points, as long as there are fewer cut-off points than actual points



Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



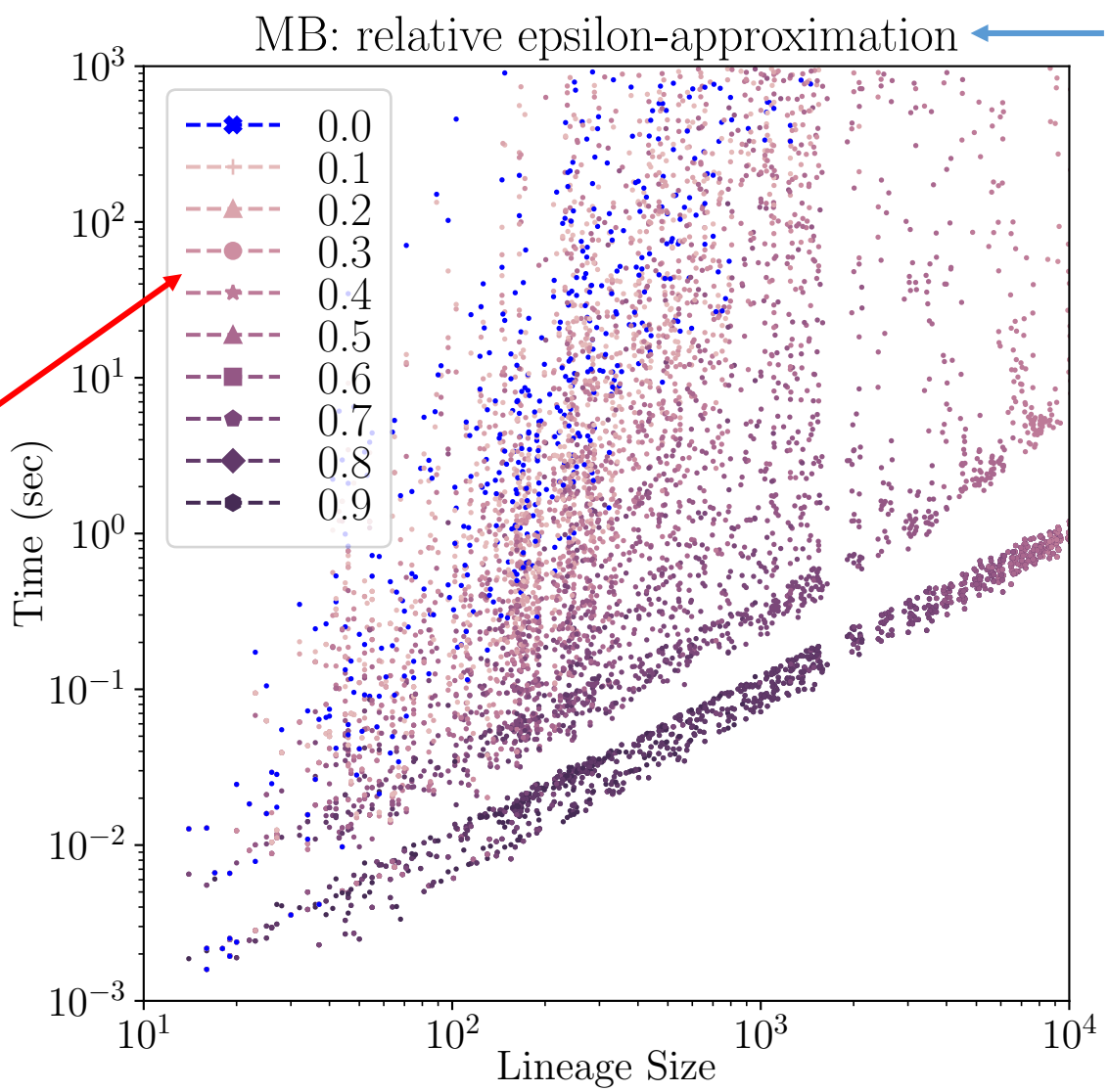
MB (prior): model-based
10 random bounds

Some type of error guaran-
tees (smaller is better)

notice the log scale!

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900

Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



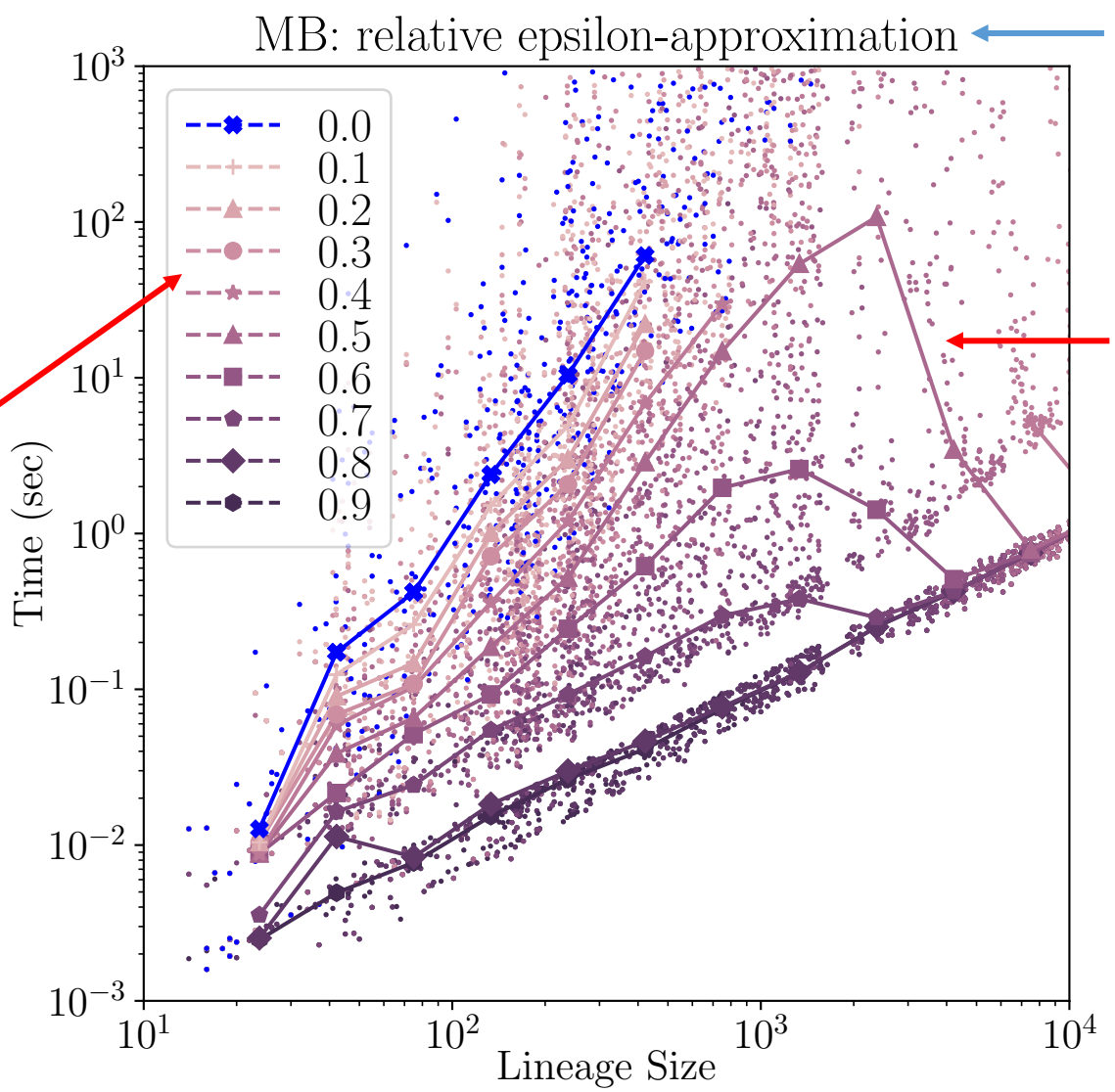
MB (prior): model-based
10 random bounds

Some type of error guaran-
tees (smaller is better)

notice the log scale!

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900

Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



MB (prior): model-based
10 random bounds

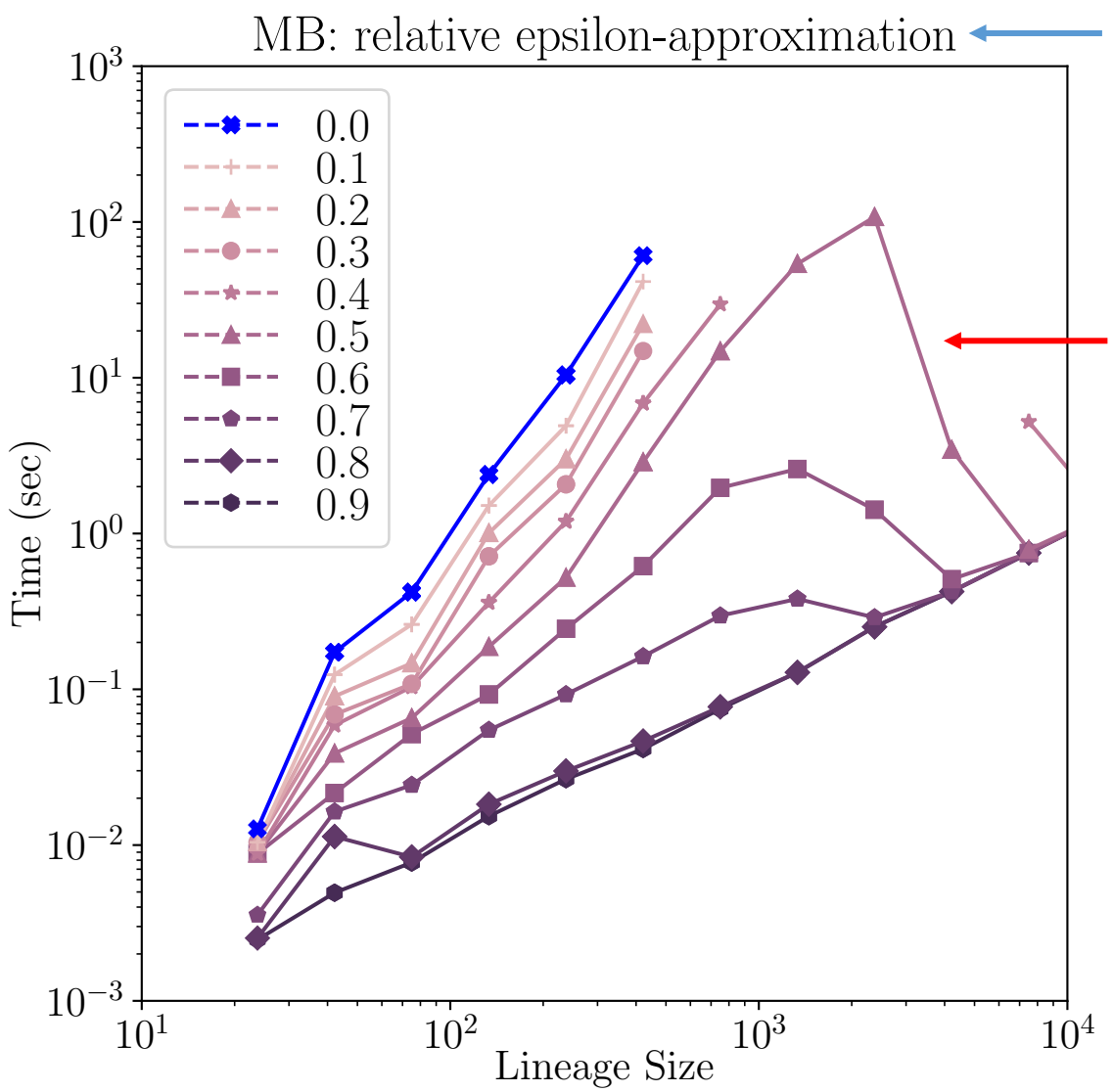
Median time to reach a certain error guarantee for fixed lin. size

Some type of error guaran-tees (smaller is better)

notice the log scale!

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900

Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]

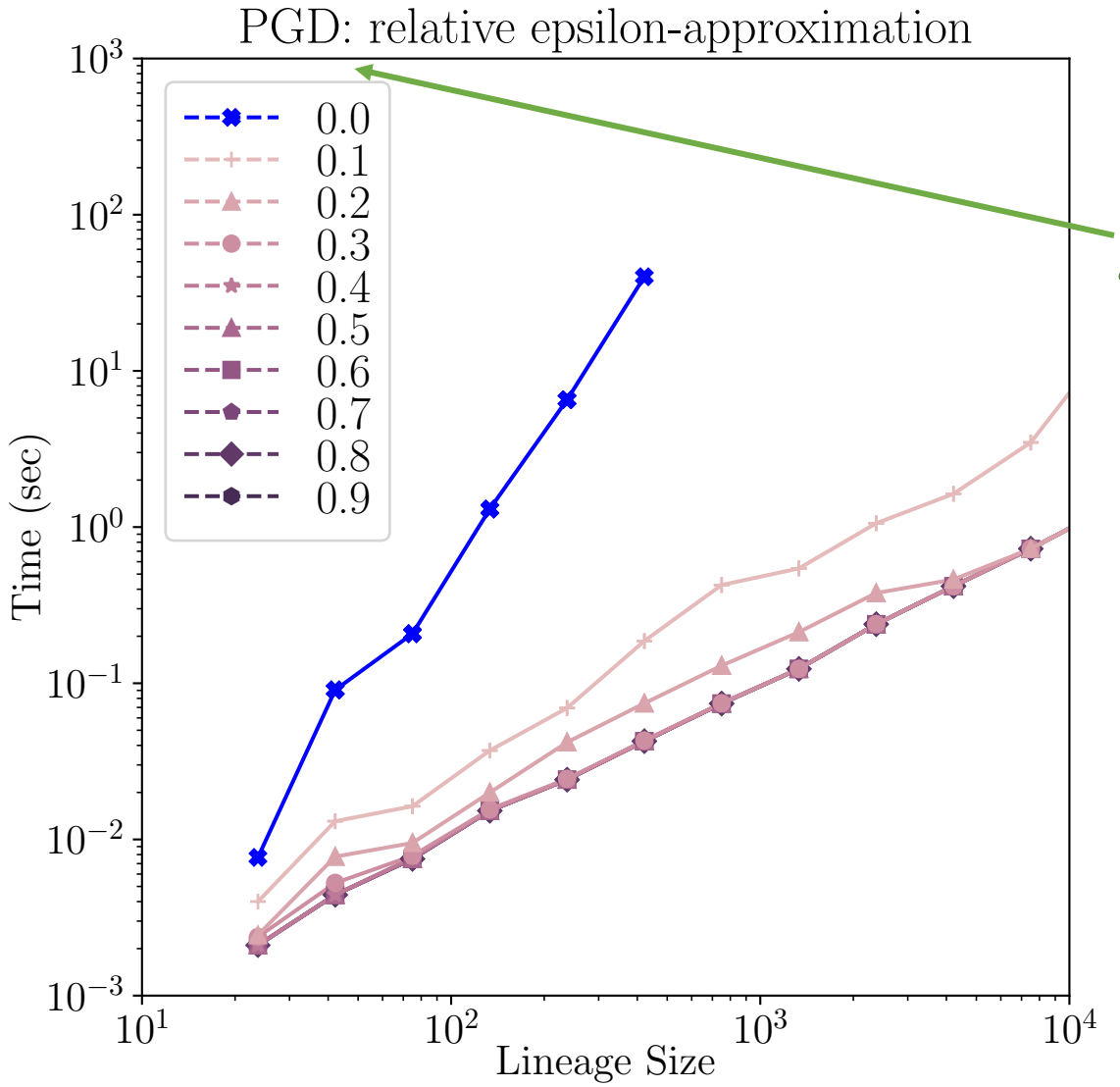


MB (prior): model-based
10 random bounds

Median time to reach a certain error guarantee for fixed lin. size

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900

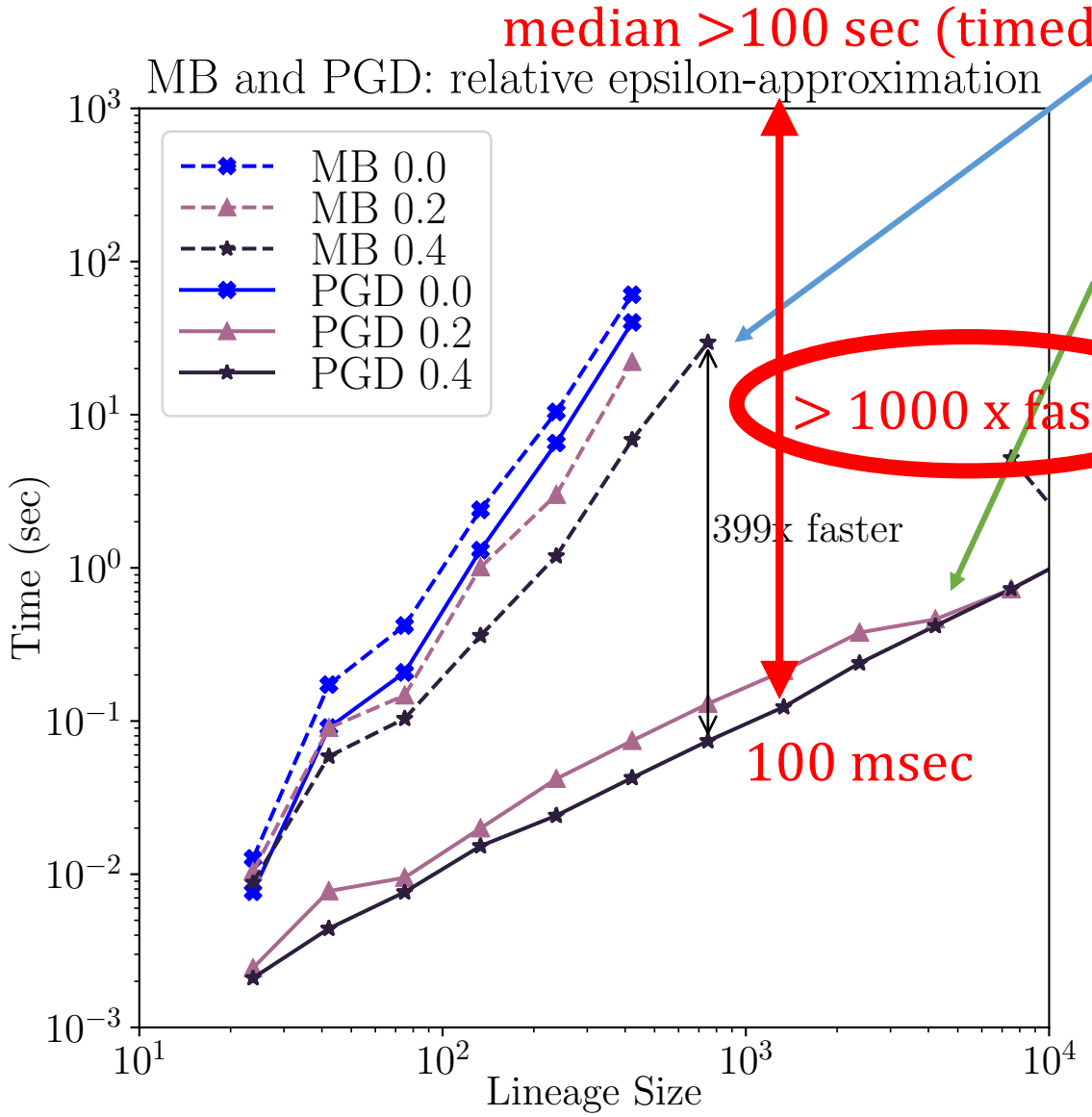
Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



PGD (our): projected gradient descent

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900

Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



MB (prior): model-based
10 random bounds

PGD (our): projected gradient descent

> 1000 x faster

Take-away

- considerable speed-ups possible ☺

Source: Van der Heuvel, Ivanov, Gatterbauer, Geerts, Theobald. Anytime approximation in probabilistic databases via scaled dissociations. SIGMOD 2019. https://doi.org/10.1145/3299869.3319900