Updated 1/28/2022

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

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp22)

https://northeastern-datalab.github.io/cs7240/sp22/

1/25/2022

Pre-class conversations

- Last class recapitulation
- Any questions on class procedures?
  - Hybrid: leave feedback after class today if you have suggestions
  - Latex: an additional reference on the website
  - Piazza: could you all post a line?
  - Class scribes: Example "minimum examples" today in class

- today:
  - SQL continued (with connections to optimization and algebra)
  - perhaps start of logic

PRELIMINARY

- **Lecture 1 (Tue 1/18):** Course introduction, SQL, PostgreSQL setup, SQL Activities
- **Lecture 2 (Fri 1/21):** SQL
- **Lecture 3 (Tue 1/25):** SQL, Logic & Relational Calculus
- **Lecture 4 (Fri 1/28):** Logic & Relational Calculus
- **Lecture 5 (Tue 2/1):** Relational Algebra & Codd's Theorem
- **Lecture 6 (Fri 2/4):** Relational Algebra & Codd's Theorem
- **Lecture 7 (Tue 2/8):** Datalog & Recursion,
- **Lecture 8 (Fri 2/11):** Datalog & Recursion,
- **Lecture 9 (Tue 2/15):** Alternative Data Models

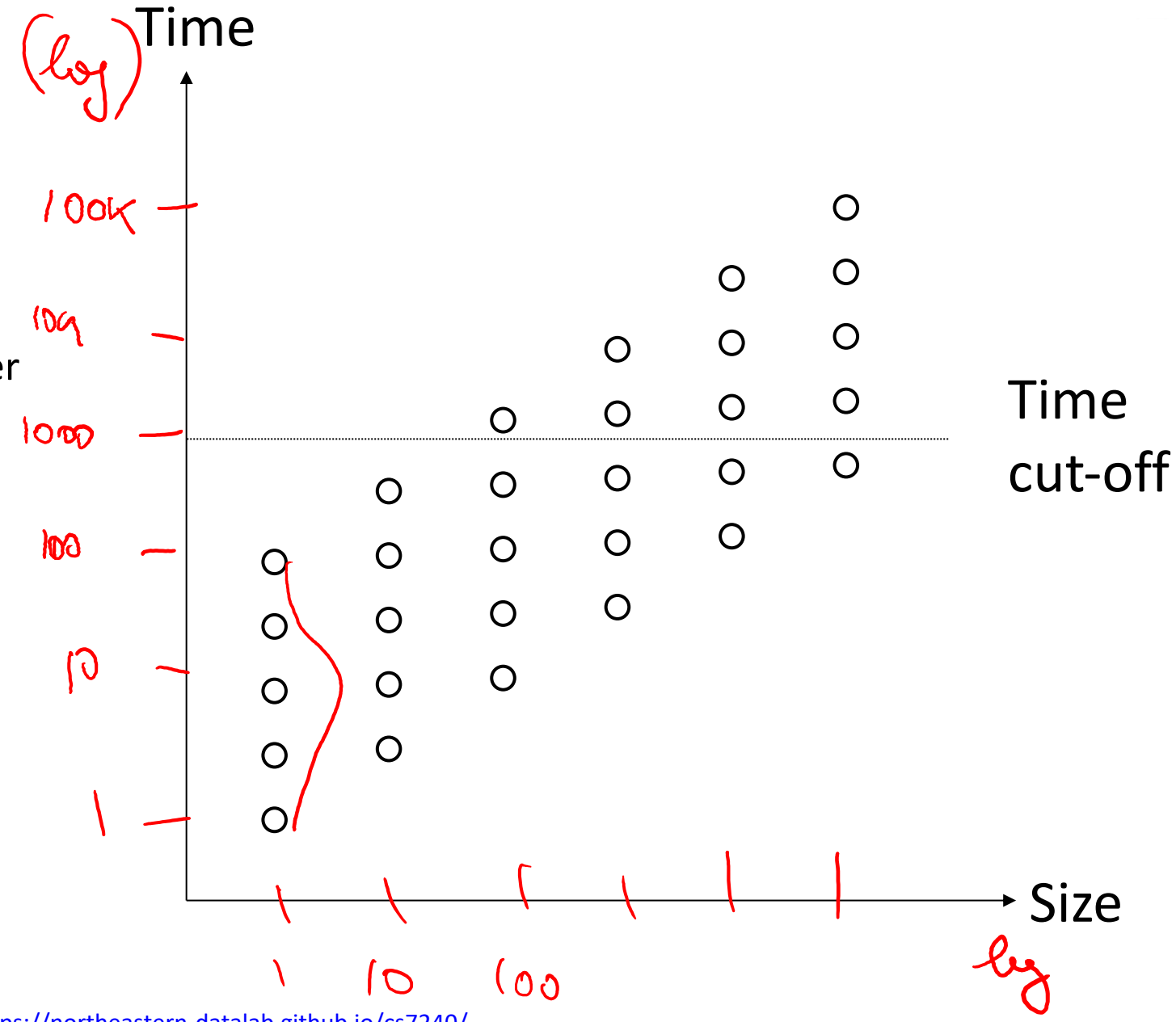
Pointers to relevant concepts & supplementary material:

- **Unit 1. SQL**: [SAMS'12], [CS3200'18] [Cow'03] Ch3 & Ch5, [Complete'08] Ch6, [Silberschatz+'20] Ch3.8
- **Unit 2. Logic & Relational Calculus**: First-Order Logic (FOL), relational calculus (RC): [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+'20] Ch27.1 & Ch27.2, [Alice'95] Ch3.1-3.3 & Ch4.2 & Ch4.4 & Ch5.3-5.4, [Barker-Plummer+'11] Ch11
- **Unit 3. Relational Algebra & Codd's Theorem**: Relational Algebra (RA), Codd's theorem: [Cow'03] Ch4.2, [Complete'08] Ch2.4 & Ch5.1-5.2, [Elmasri, Navathe'15] Ch8, [Silberschatz+'20] Ch2.6, [Alice'95] Ch4.4 & Ch5.4
- **Unit 4. Datalog & Recursion**: Datalog, recursion, Stratified Datalog with negation, Datalog evaluation strategies, Stable Model semantics, Answer Set Programming (ASP): [Complete'08] Ch5.3, [Cow'03] Ch 24, [Koutris'19] L9 & L10, [G., Suciu'10]
- **Unit 5. Alternative Data Models**: NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Revisiting our question from first class

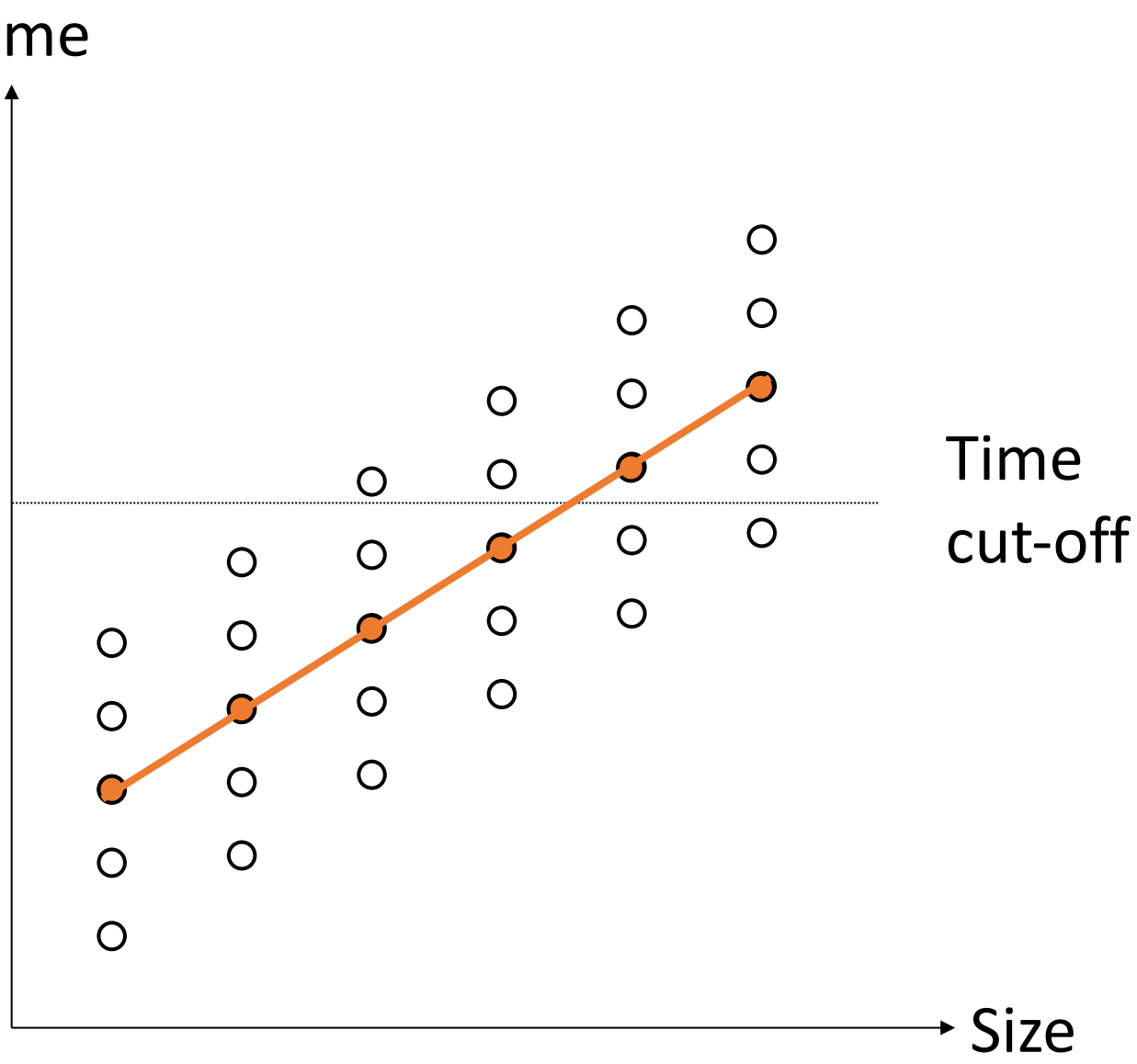
Question: How to deal with cut-offs when binning

- 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



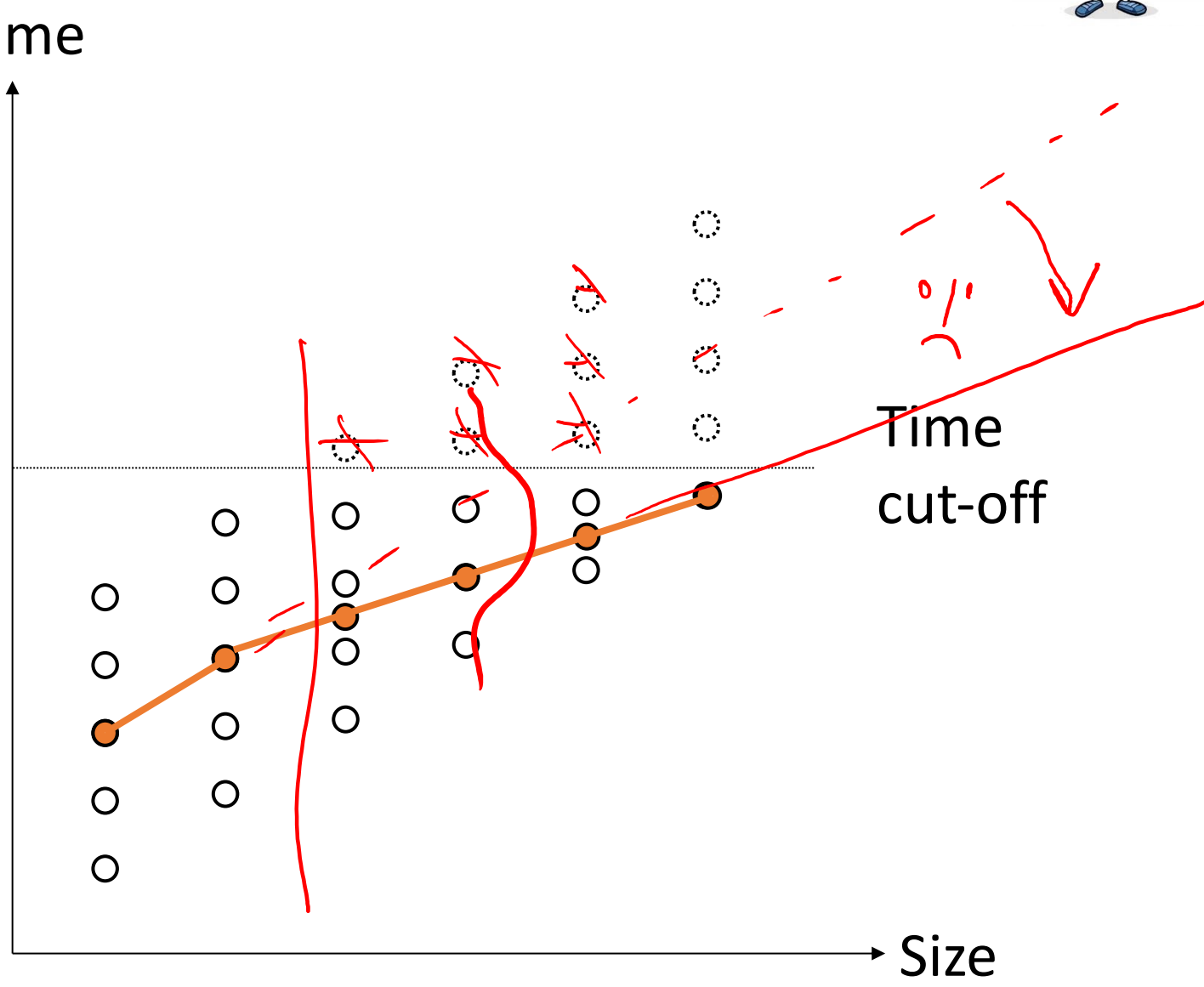
Question: How to deal with cut-offs when binning

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



Question: 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"

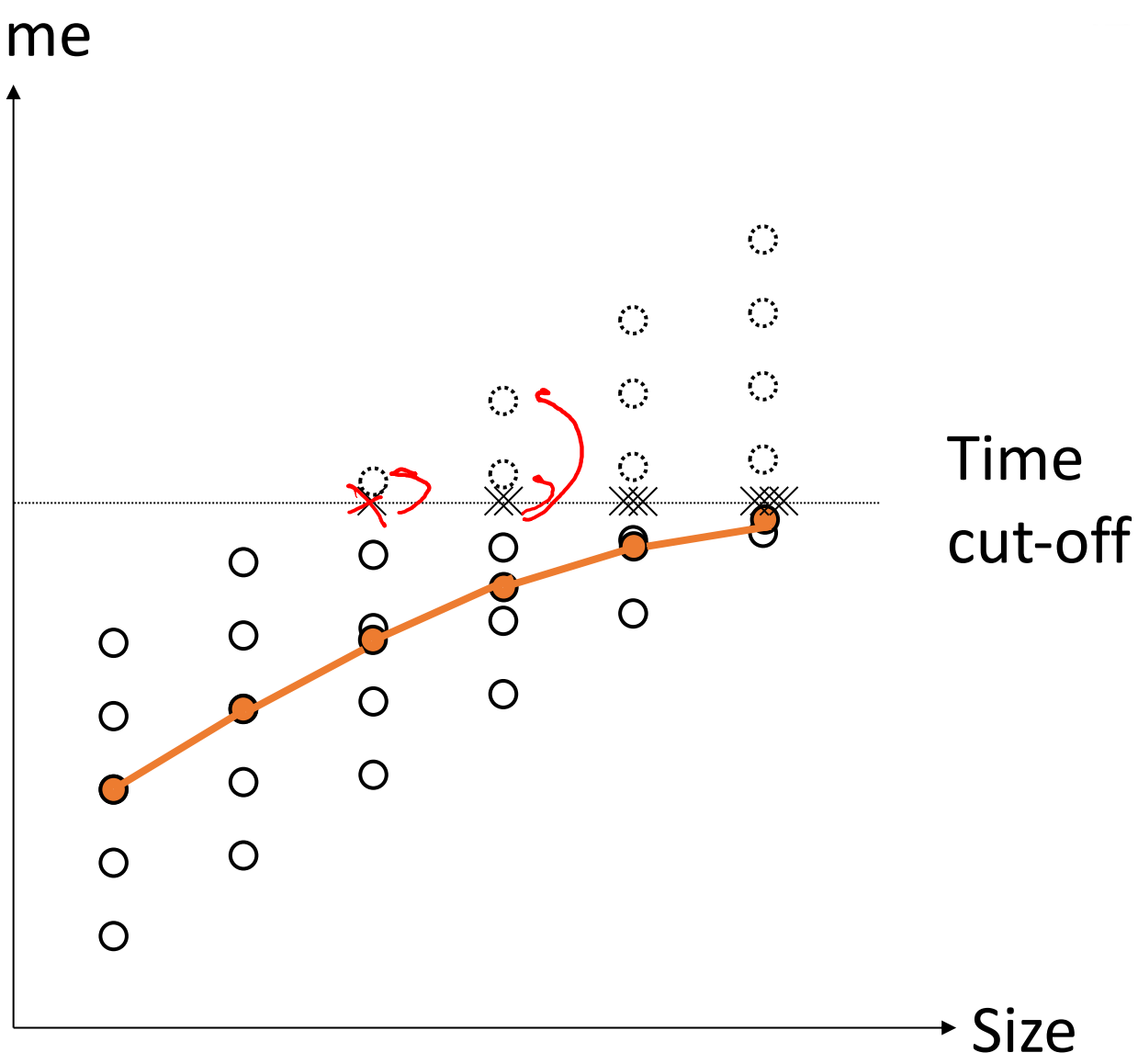


What would you do ?

We will discuss next class

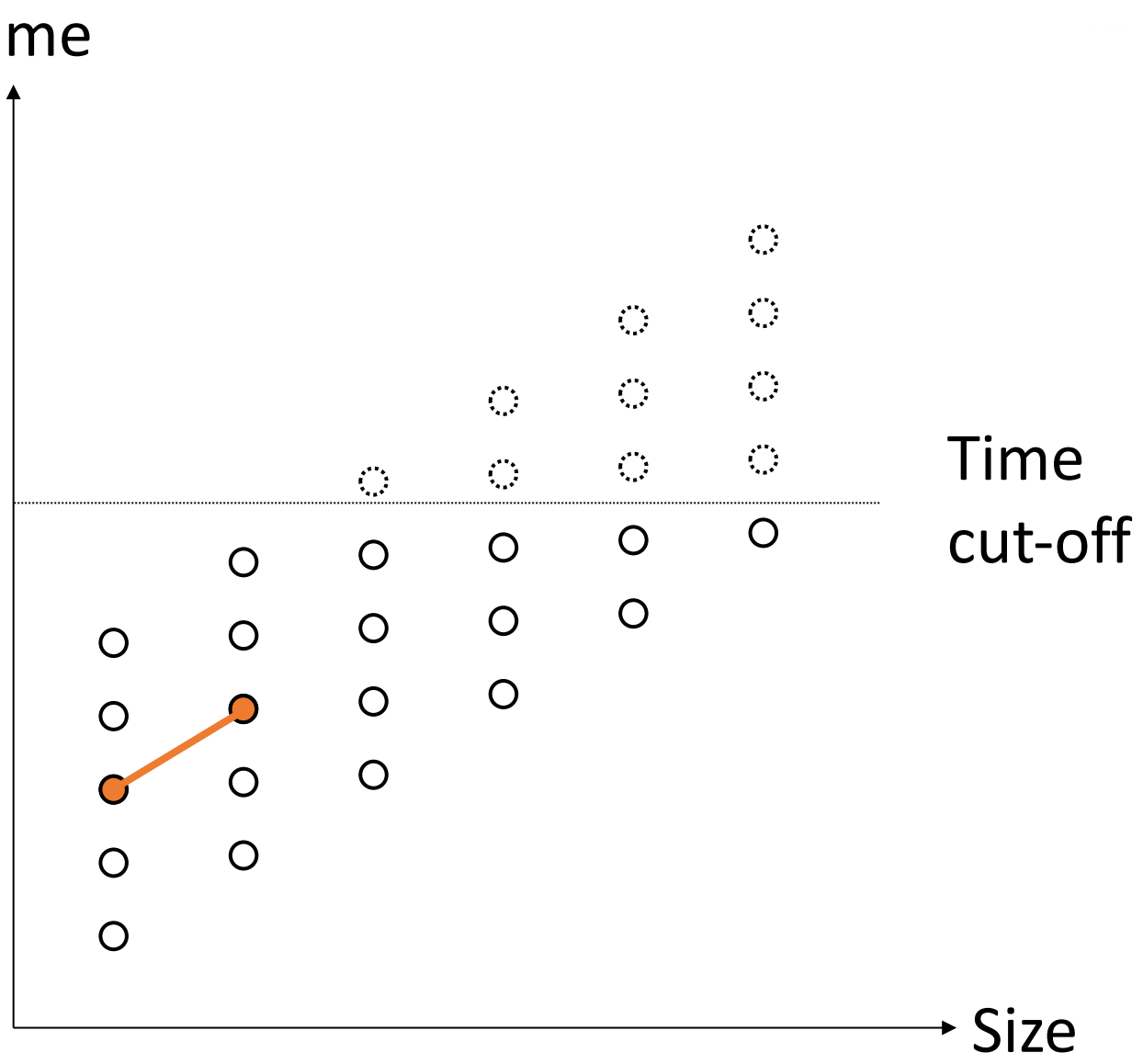
How to deal with cut-offs when binning: Option 1

- Option 1: Here is what happens if we cut the points off and still use the points, and then average



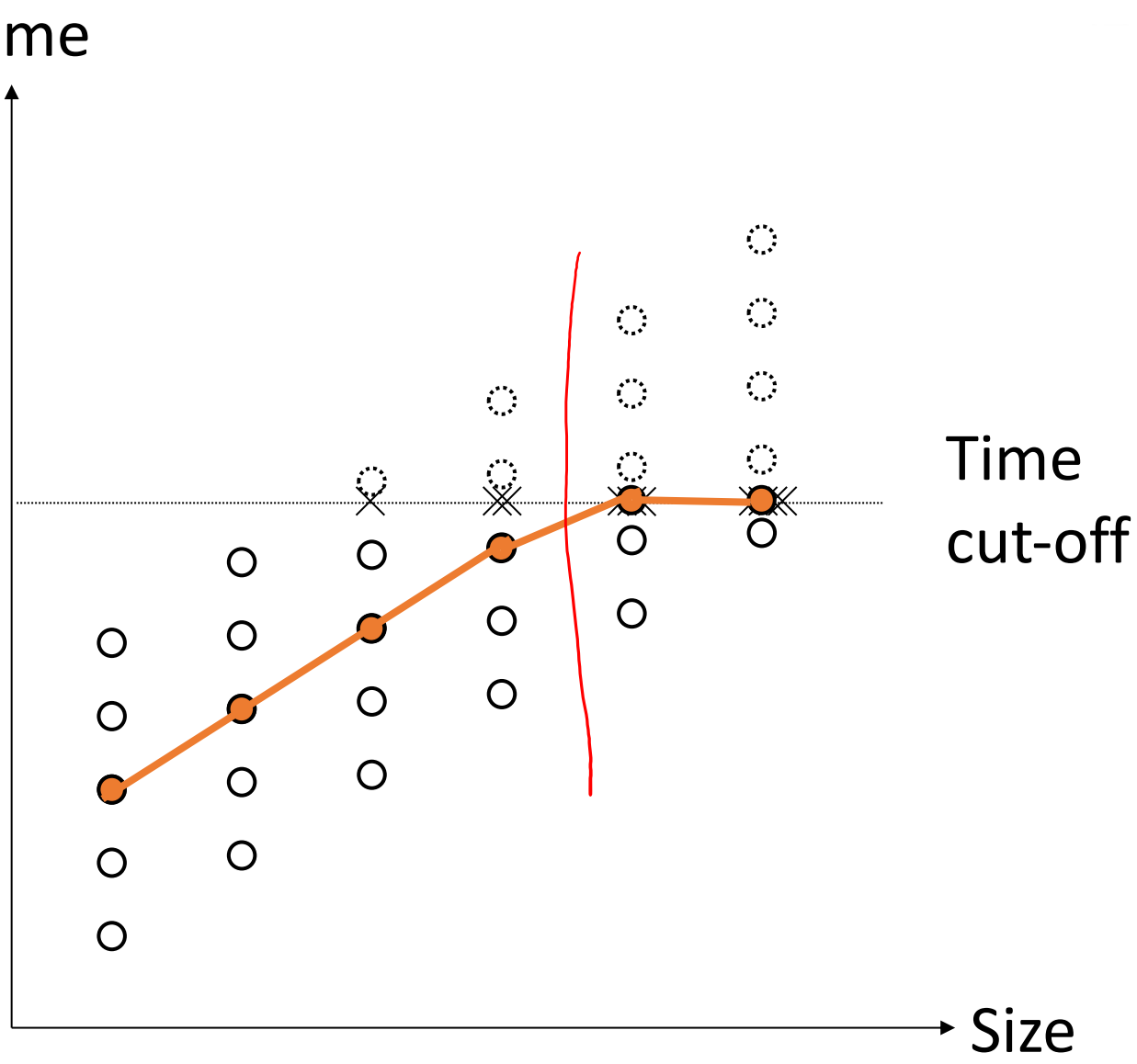
How to deal with cut-offs when binning: Option 2

- Option 2: Here is what we can do if we *only* use those sizes for which all experiments finish in time



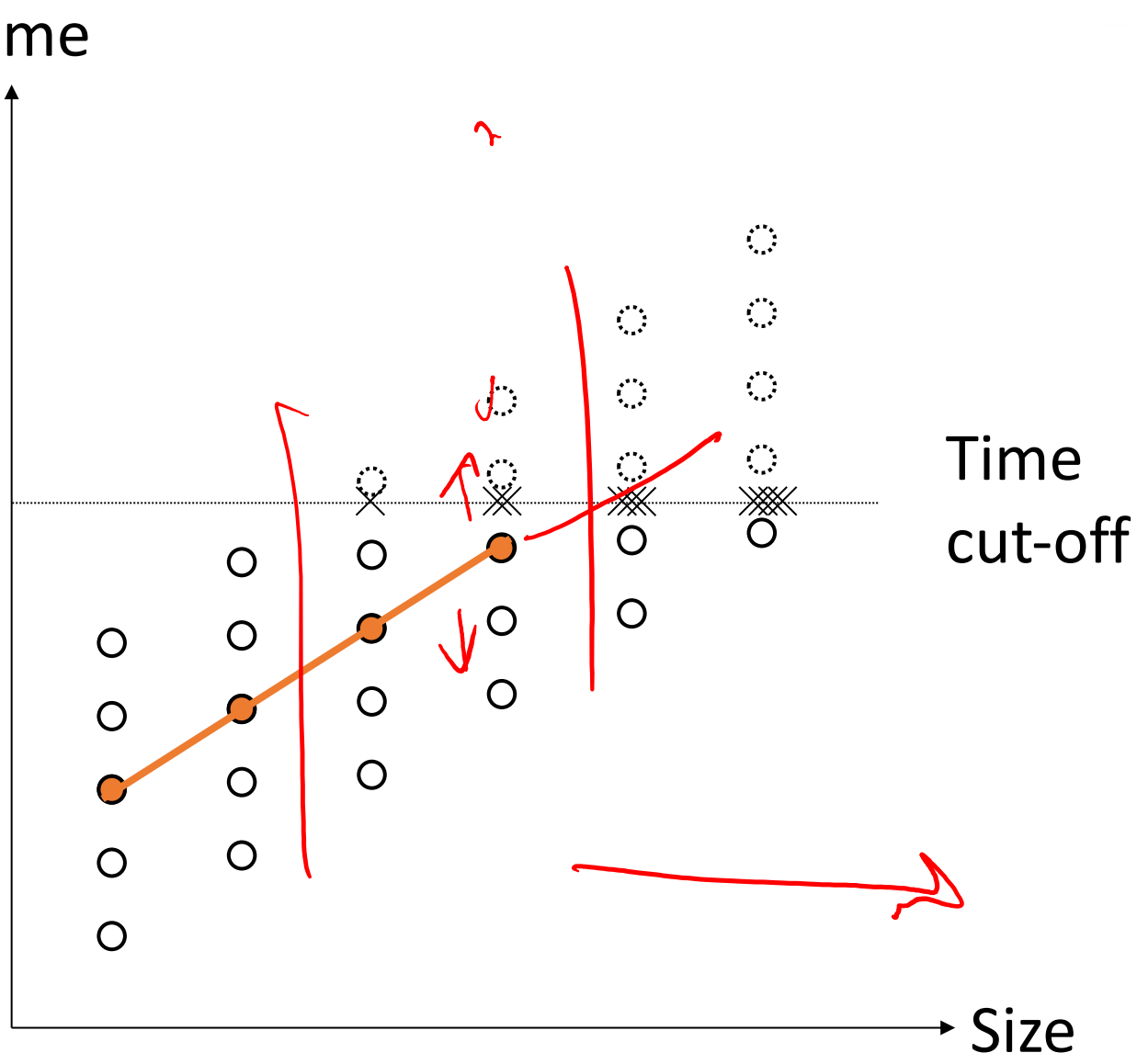
How to deal with cut-offs when binning: Option 3

- Option 3: Here is what happens if we take the median over all seen and cut-off points



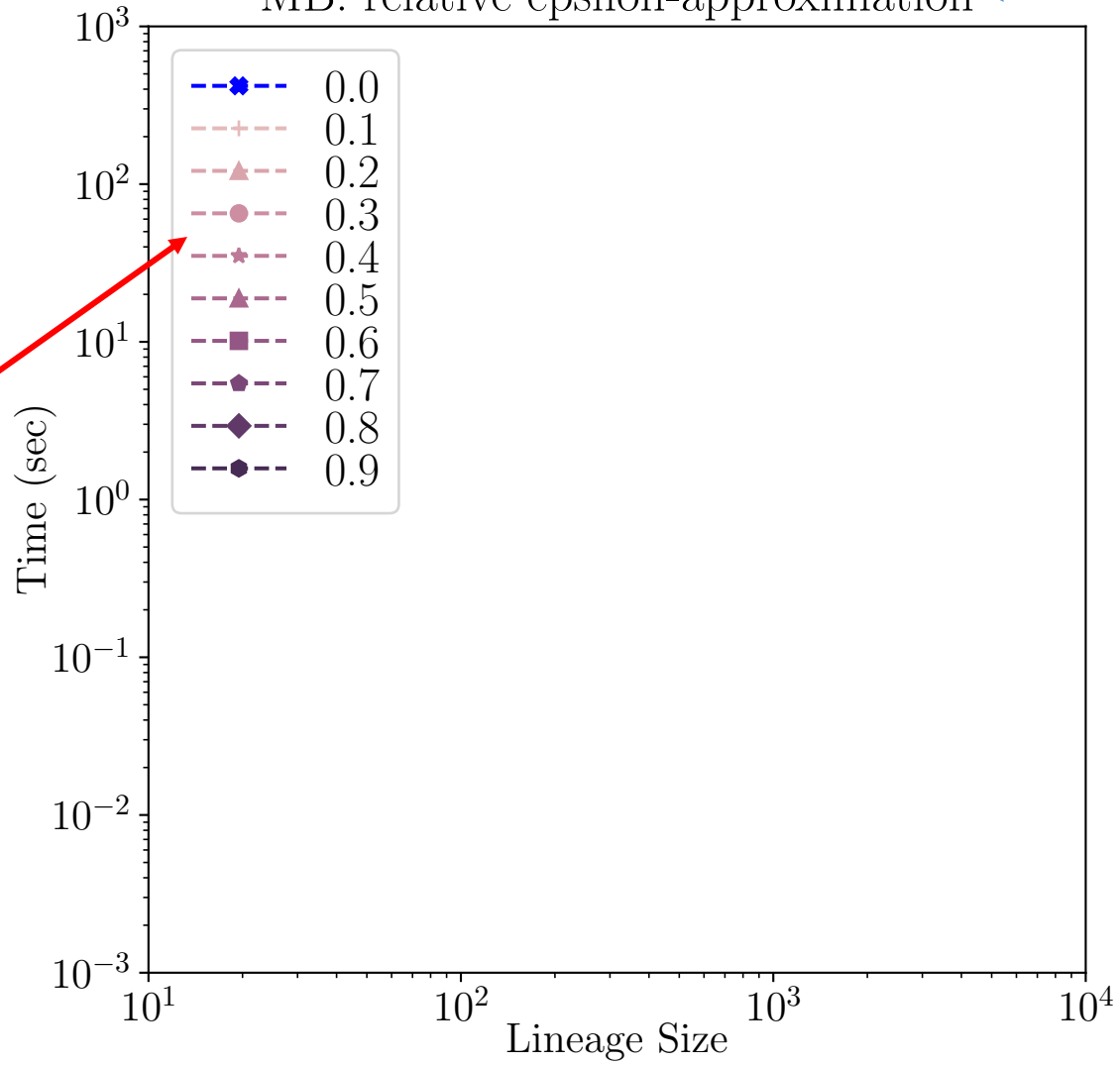
How to deal with cut-offs when binning: Suggestion

- Suggestion: Here is what happens if we take the median over all seen and cut-off points, as long as there are <50% cut-off points



Notice the informal "semantics" of median: If more points are "above you" then you are pulled by their number, not by their distance (in contrast to average where distance is kind of a weight)

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



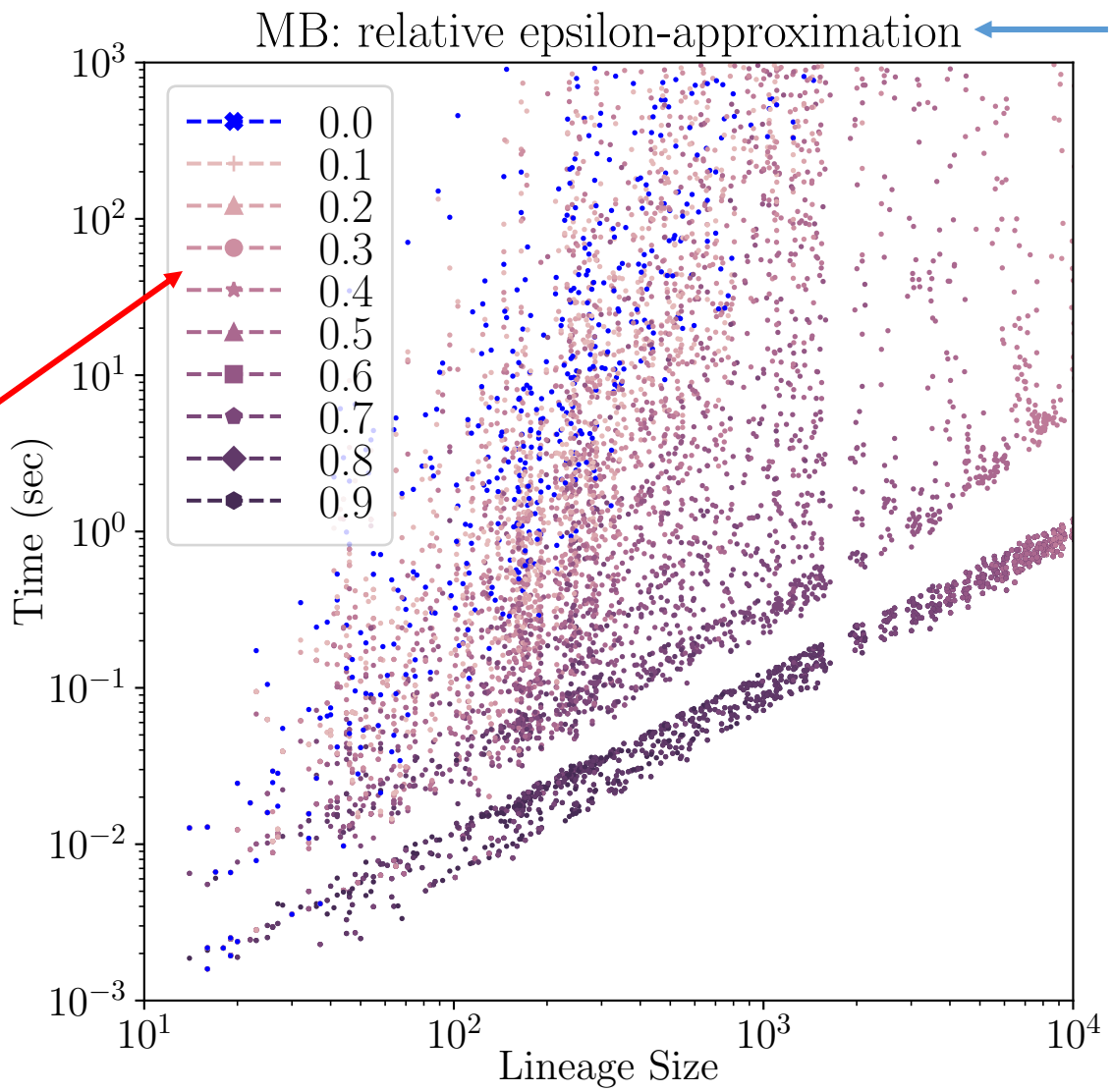
MB (prior): model-based
10 random bounds

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]



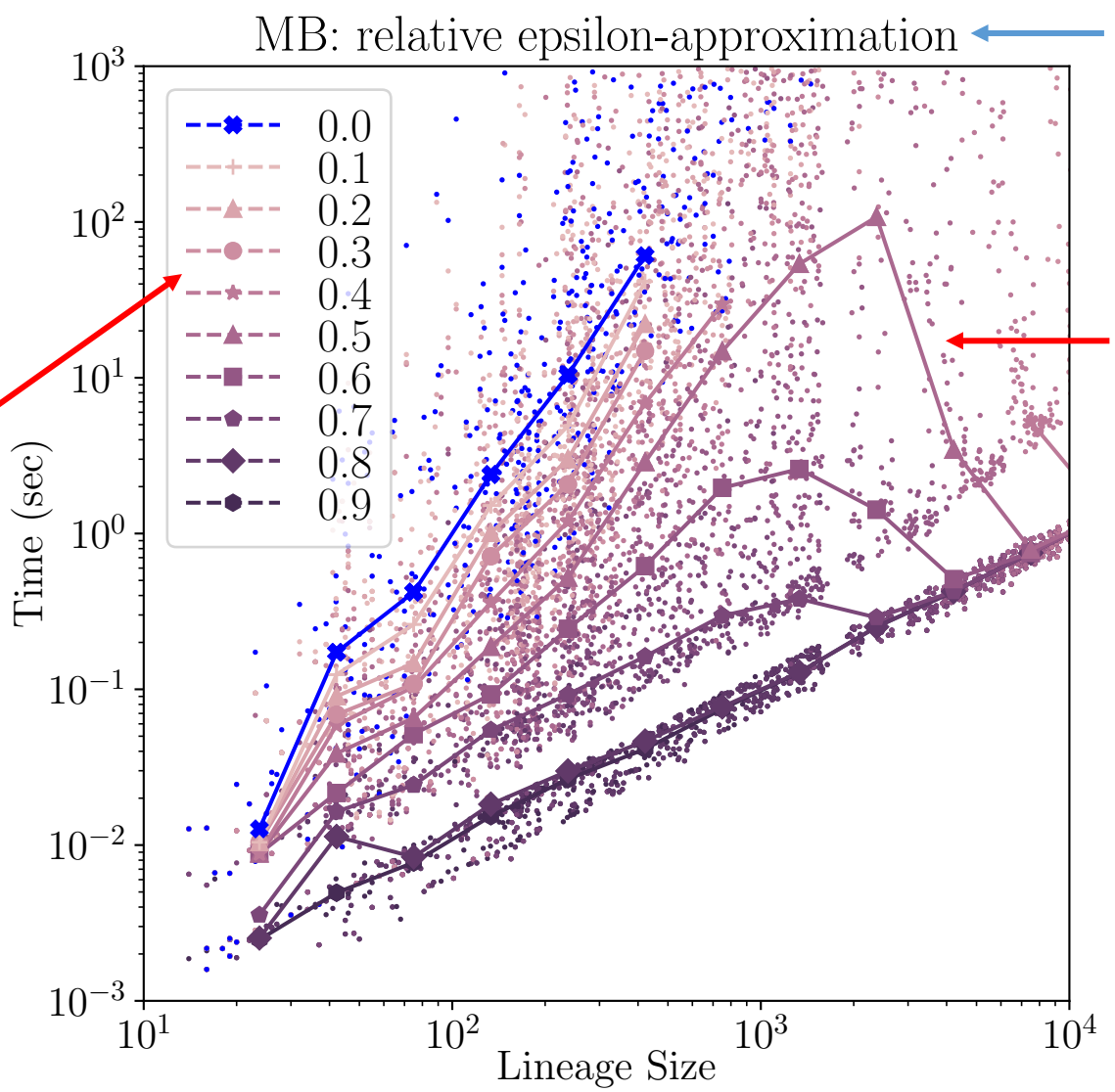
MB (prior): model-based
10 random bounds

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

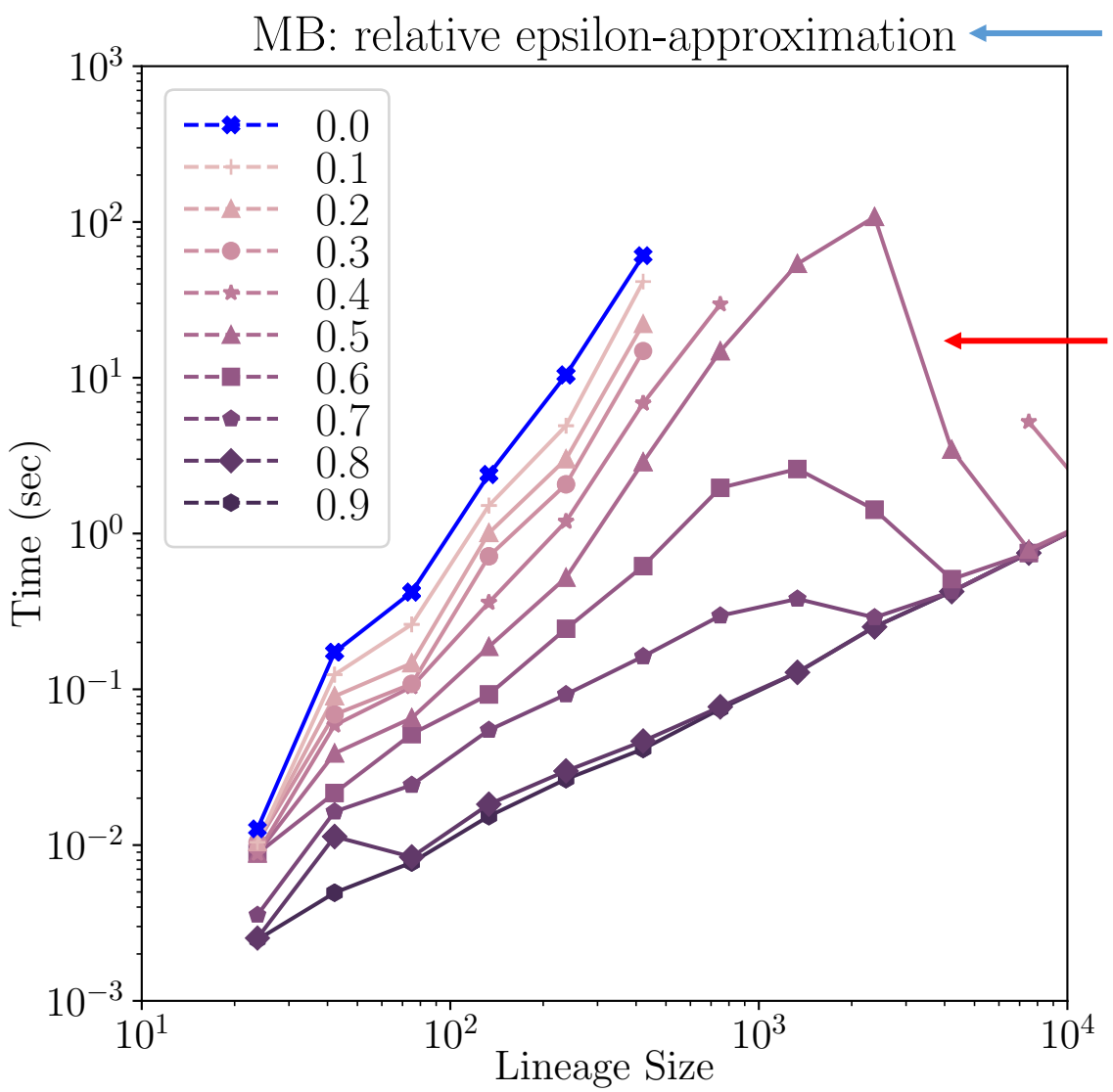
Median time to reach a certain error guarantee for fixed lin. size

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]

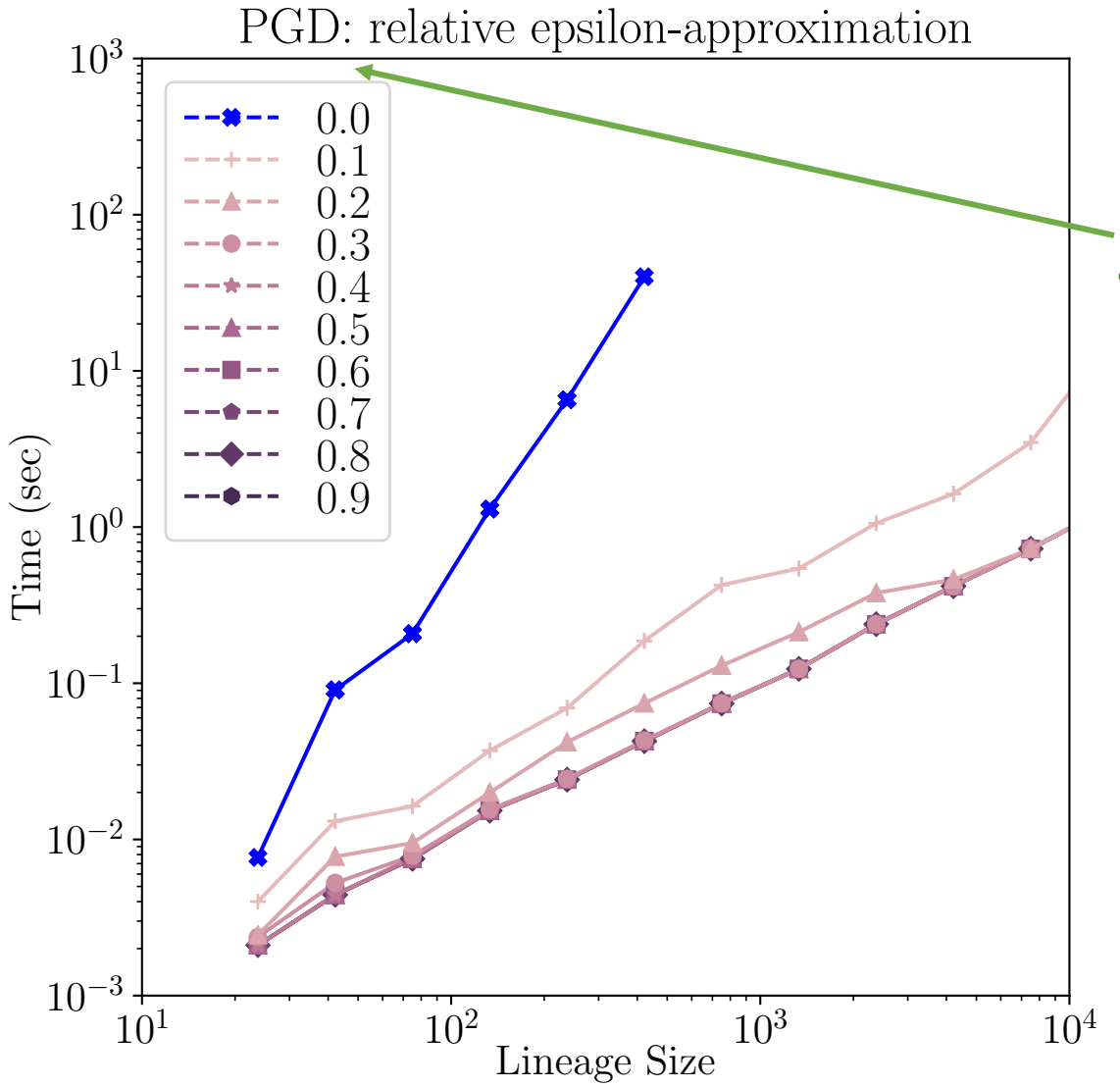


MB (prior): model-based
10 random bounds

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

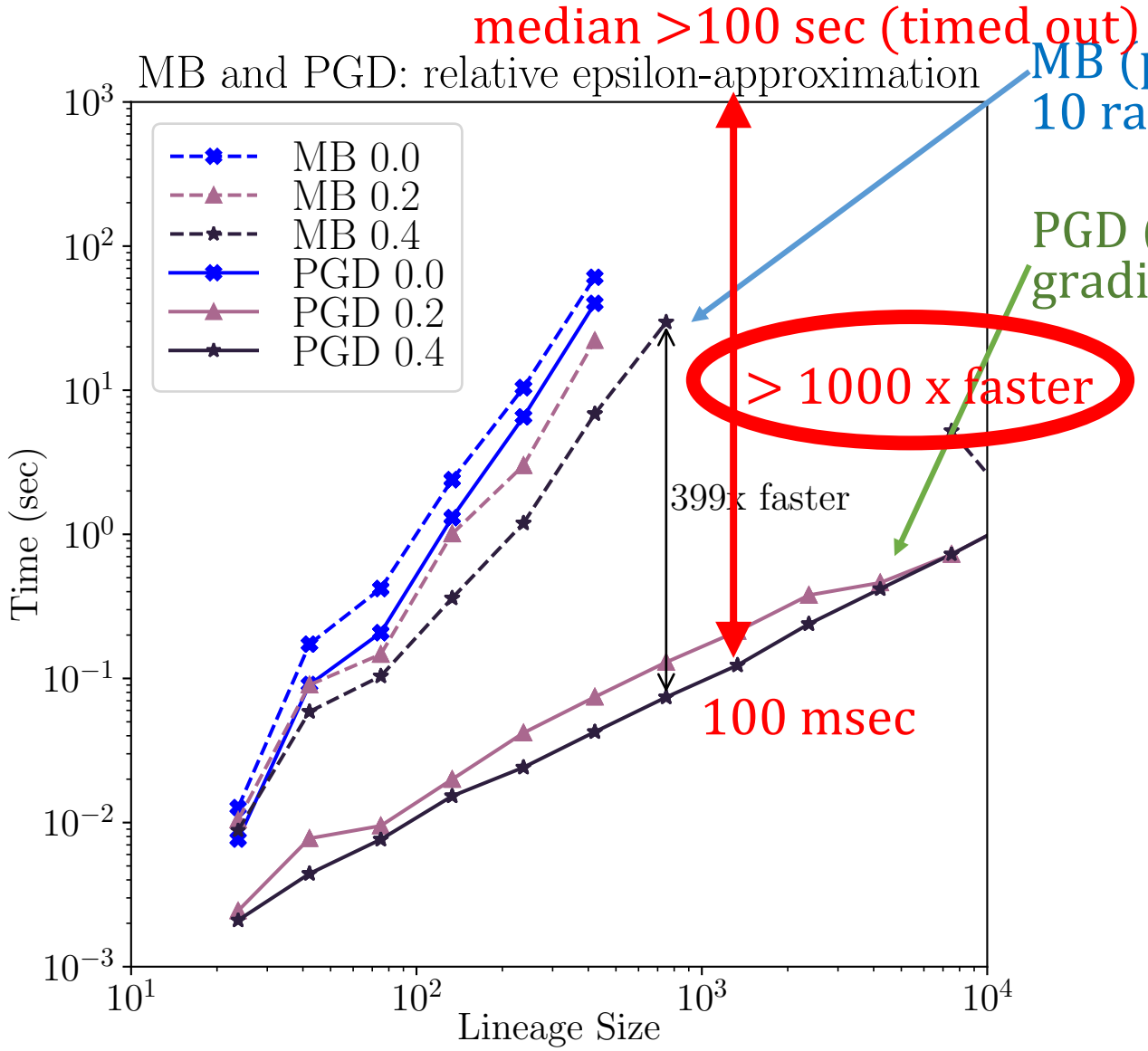
Example: Experiments figures from Van der Heuvel+ [SIGMOD'19]



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

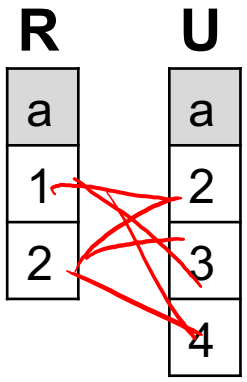
Take-away

- considerable speed-ups possible ☺

Outline: SQL

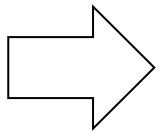
- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k

Theta joins



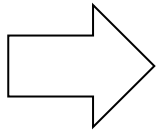
What do these queries compute?

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a < U.a
```



?

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a >= U.a
```



?

A **Theta-join** allows for arbitrary comparison relationships (such as ≥).
An **equijoin** is a theta join using the equality operator.

Theta joins

305

R

a
1
2

U

a
2
3
4

What do these queries compute?

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a < U.a
```

⇒

a	b
1	2
1	3
1	4
2	3
2	4

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a >= U.a
```

⇒ ?

A **Theta-join** allows for arbitrary comparison relationships (such as ≥).
An **equijoin** is a theta join using the equality operator.

Theta joins

R
a
1
2

U
a
2
3
4

What do these queries compute?

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a < U.a
```

a	b
1	2
1	3
1	4
2	3
2	4

∪

```
SELECT   R.a, U.a as b
FROM     R, U
WHERE    R.a >= U.a
```

a	b
2	2

×

Think about these two queries as a partition of the Cartesian product

A **Theta-join** allows for arbitrary comparison relationships (such as ≥).
An **equijoin** is a theta join using the equality operator.

Outline: SQL

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & Outer joins
  - Top-k

3-valued logic example

- Three logicians walk into a bar. The bartender asks: "Do all of you want a drink?"
- The 1st logician says: "I don't know."
- The 2nd logician says: "I don't know."
- The 3rd logician says: "Yes!"

What is going on here ?

Nulls in SQL

- Whenever we don't have a value, we can put a NULL
- Can mean many things, e.g.:

?

Nulls in SQL

- Whenever we don't have a value, we can put a NULL
- Can mean many things, e.g.:
  - Value exists but is unknown
  - Value not applicable
- The schema specifies for each attribute if it can be NULL (nullable attribute) or not
- Lots of ongoing research on NULLs

Join Illustration

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

An "inner join":

```
SELECT *
FROM   English, French
WHERE  eid = fid
```

etext	eid	fid	ftext
One	1	1	Un
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz

Same as:

```
SELECT *
FROM   English JOIN French
ON     eid = fid
```

"JOIN" same as "INNER JOIN"

Join Illustration

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Null also sometimes just shown as empty

How do we get a join with the full data ?

```
SELECT *
FROM   English JOIN French
ON     eid = fid
```

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

Join Illustration

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

"FULL JOIN"
same as
"FULL OUTER JOIN"

```
SELECT *
FROM   English FULL JOIN French
ON     English.eid = French.fid
```

~~SELECT * FROM English JOIN French ON eid = fid~~

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

Null also sometimes just shown as empty

Join Illustration

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

```
SELECT *
FROM   English LEFT JOIN French
ON     English.eid = French.fid
```

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz

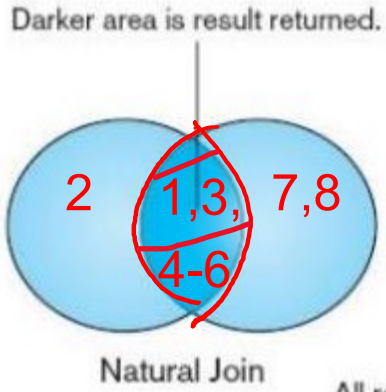
Join Illustration

English

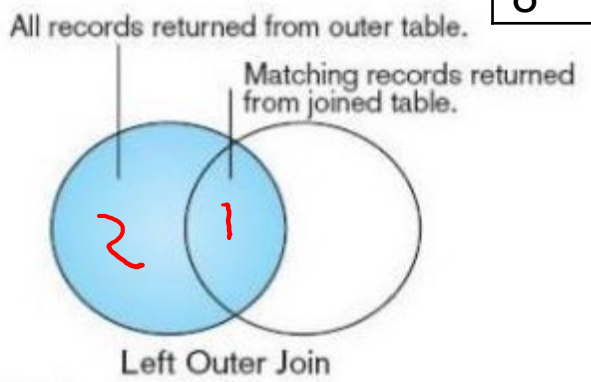
eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

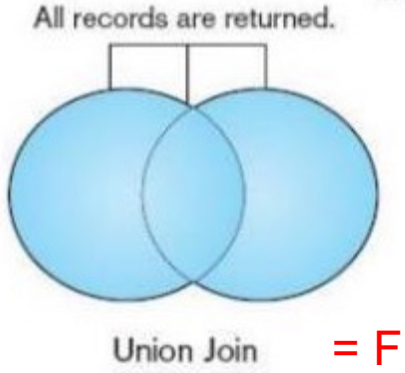
fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit



= (INNER) JOIN



= LEFT (OUTER) JOIN

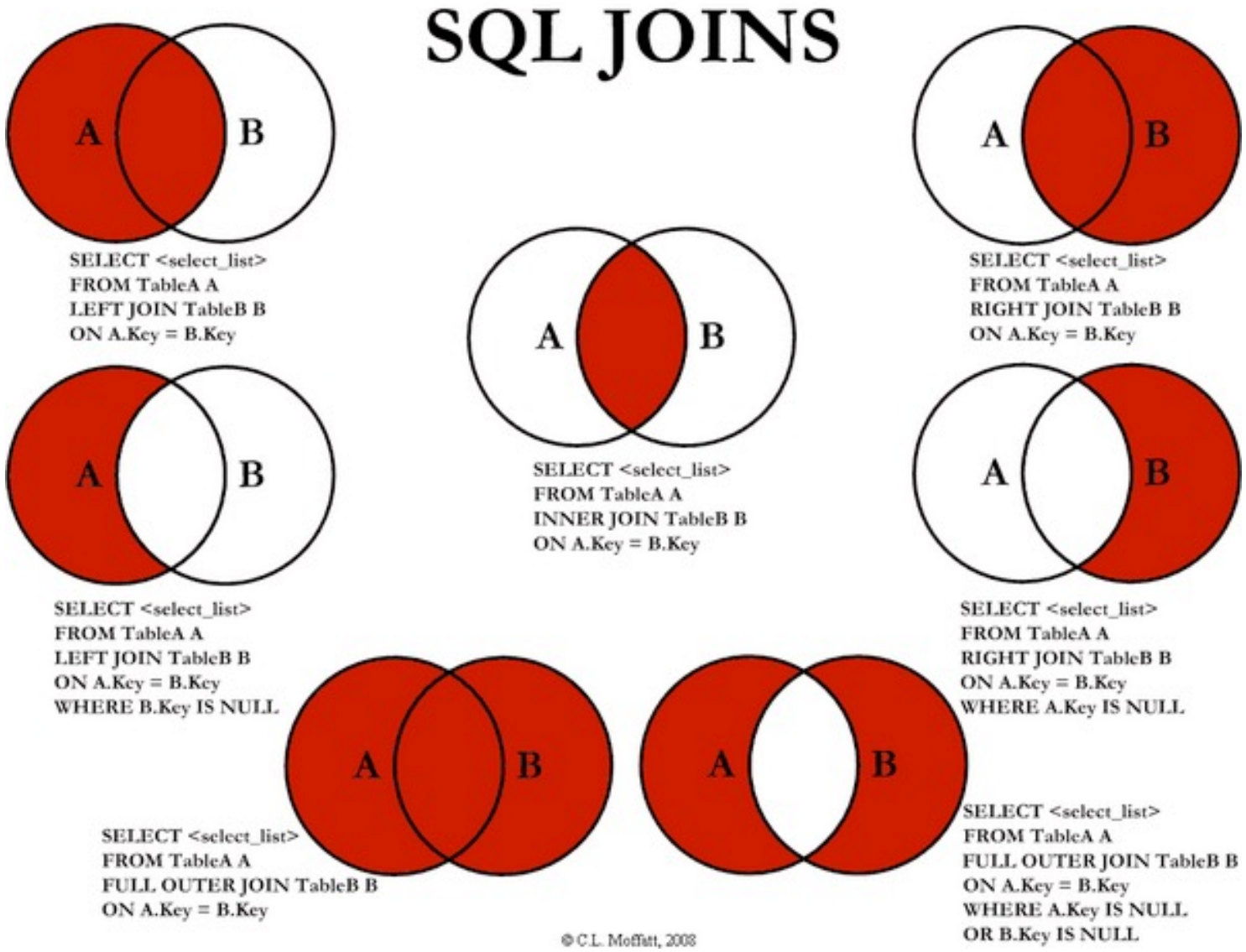


= FULL (OUTER) JOIN

Source: Fig. 7-2, Hoffer et al., Modern Database Management, 10ed ed, 2011.

Detailed Illustration with Examples (follow the link)

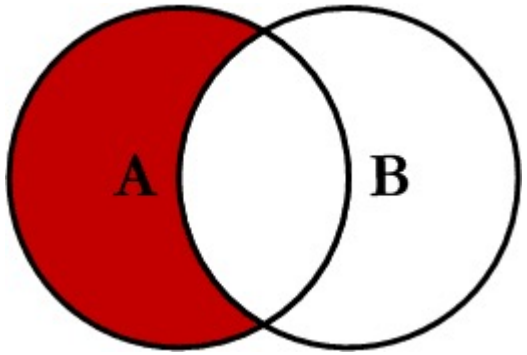
also called "anti-join"



Check this web page for illustrating examples

Source: http://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Let's practice anti-joins




```
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key = B.key
WHERE B.key IS NULL
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

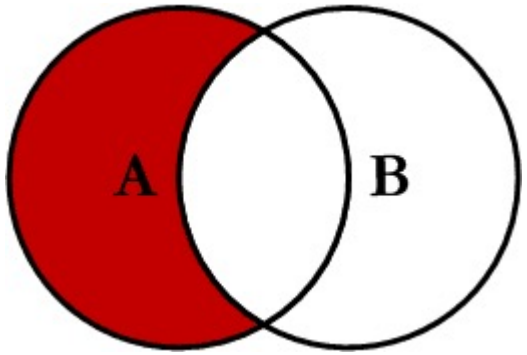
French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

?

Let's practice anti-joins




```
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key = B.key
WHERE B.key IS NULL
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

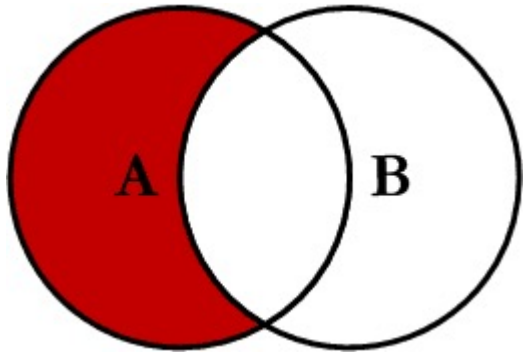
Results

eText	eid
Two	2

How to write in SQL?

?

Let's practice anti-joins




```
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key = B.key
WHERE B.key IS NULL
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

eText	eid
Two	2

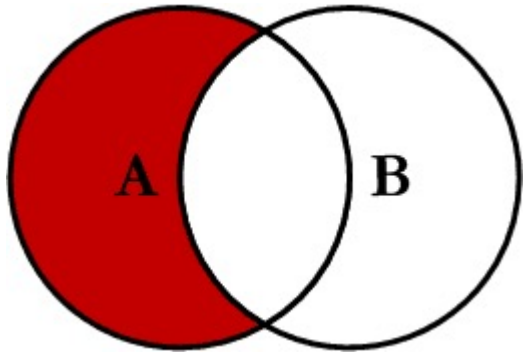
How to write in SQL?

```
SELECT eText, eid
FROM English
LEFT JOIN French
ON eid = fid
WHERE fid IS NULL
```

Any alternative?

?

Let's practice anti-joins




```
SELECT <select_list>
FROM A
LEFT JOIN B
ON A.key = B.key
WHERE B.key IS NULL
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

eText	eid
Two	2

How to write in SQL?

```
SELECT eText, eid
FROM English
LEFT JOIN French
ON eid = fid
WHERE fid IS NULL
```

Any alternative?

```
SELECT *
FROM English
WHERE eid NOT IN
   (SELECT fid
   FROM French)
```

Semi-joins: kind of the anti-anti-joins…


English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

eText	eid
One	1
Three	3
Four	4
Five	5
Six	6

What do we have to change to these queries to get the tuples in English that have a partner in French?

?

```
SELECT eText, eid
FROM English
LEFT JOIN French
ON eid = fid
WHERE fid IS NULL
```

```
SELECT *
FROM English
WHERE eid NOT IN
   (SELECT fid
   FROM French)
```

Semi-joins: kind of the anti-anti-joins…


English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

eText	eid
One	1
Three	3
Four	4
Five	5
Six	6

What do we have to change to these queries to get the tuples in English that have a partner in French?

What if fid is not a key?

?

```
SELECT eText, eid
FROM English
LEFT JOIN French
ON eid = fid
WHERE fid IS NOT NULL
```

```
SELECT *
FROM English
WHERE eid IN
   (SELECT fid
   FROM French)
```

Semi-joins: kind of the anti-anti-joins…


English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

Results

eText	eid
One	1
Three	3
Four	4
Five	5
Six	6

What do we have to change to these queries to get the tuples in English that have a partner in French?

What if fid is not a key?

DISTINCT

```
SELECT eText, eid
FROM English
LEFT JOIN French
ON eid = fid
WHERE fid IS NOT NULL
```

```
SELECT *
FROM English
WHERE eid IN
   (SELECT fid
   FROM French)
```

Another look at Outer Joins

```
SELECT  *
FROM    English FULL JOIN French
ON      English.eid = French.fid
```

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

361

FULL JOIN can be written as union of inner join with anti-joins

?

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

Another look at Outer Joins


```
SELECT  *
FROM    English FULL JOIN French
ON      English.eid = French.fid
```

```
SELECT   etext,eid, fid, ftext
FROM     English Inner JOIN French
ON       English.eid = French.fid
UNION ALL
SELECT   etext, eid, NULL, NULL
FROM     English
WHERE    NOT EXISTS(
         SELECT *
         FROM French
         WHERE eid=fid)
UNION ALL
SELECT   NULL, NULL, fid, ftext
FROM     French
WHERE    NOT EXISTS(
         SELECT *
         FROM English
         WHERE eid=fid)
```

anti-join

English

eText	eid
One	1
Two	2
Three	3
Four	4
Five	5
Six	6

French

fid	fText
1	Un
3	Trois
4	Quatre
5	Cinq
6	Siz
7	Sept
8	Huit

etext	eid	fid	ftext
One	1	1	Un
Two	2	NULL	NULL
Three	3	3	Trois
Four	4	4	Quatre
Five	5	5	Cinq
Six	6	6	Siz
NULL	NULL	7	Sept
NULL	NULL	8	Huit

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)
```

?

Also see use of COALESCE across programming languages: https://en.wikipedia.org/wiki/Null_coalescing_operator

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,
C(x,y,z)=C(x,C(y,z))=C(C(x,y),z)

Result

M.a	N.a	b

? ?

```
SELECT COALESCE(1, NULL)
```
⇨ 1

```
SELECT COALESCE(NULL, 3)
```
⇨ 3

```
SELECT COALESCE(1, 2)
```
⇨ 1

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

Also see use of COALESCE across programming languages: https://en.wikipedia.org/wiki/Null_coalescing_operator

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,
C(x,y,z)=C(x,C(y,z))=C(C(x,y),z)

Result

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

```
SELECT COALESCE(1, NULL)
```
⇨ 1

```
SELECT COALESCE(NULL, 3)
```
⇨ 3

```
SELECT COALESCE(1, 2)
```
⇨ 1

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

Also see use of COALESCE across programming languages: https://en.wikipedia.org/wiki/Null_coalescing_operator

Coalesce, Natural Outer Join, Union

M

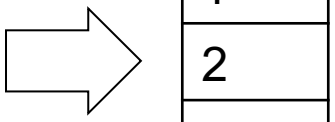
a
1
2

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)

Quick recap: Commutativity & Associativity

Multiplication

$3 \bullet 2 \bullet 4 = 24$

?

Multiplication is associative ☺

Matrix multiplication

$$\begin{bmatrix} 2 & 3 \end{bmatrix} \bullet \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bullet \begin{bmatrix} 3 \\ 1 \end{bmatrix} =$$

Quick recap: Commutativity & Associativity

Multiplication

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

Matrix multiplication

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

Order of operations can be exchanged:

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

Multiplication is associative ☺

?

and commutative ☺

Quick recap: Commutativity & Associativity

Multiplication

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

Order of operations can be exchanged:

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

Multiplication is associative ☺

Order of operands can be exchanged:

$$4 \cdot 2$$

and commutative ☺

Matrix multiplication

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

?

Quick recap: Commutativity & Associativity

Multiplication

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

Order of operations can be exchanged:

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

Multiplication is associative ☺

Order of operands can be exchanged:

$4 \cdot 2$

and commutative ☺

Matrix multiplication

$$\left(\begin{bmatrix}1&1\\2&3\end{bmatrix} \cdot \begin{bmatrix}1&2\\3&4\end{bmatrix}\right) \cdot \begin{bmatrix}3\\1\end{bmatrix} = \begin{bmatrix}18\\49\end{bmatrix}$$

$$\begin{bmatrix}4&6\\11&16\end{bmatrix}$$

$$\begin{bmatrix}1&1\\2&3\end{bmatrix} \cdot \left(\begin{bmatrix}1&2\\3&4\end{bmatrix} \cdot \begin{bmatrix}3\\1\end{bmatrix}\right) = \begin{bmatrix}18\\49\end{bmatrix}$$

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

Matrix multipl. is associative ☺

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

#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

The power of associativity

Option 1: $\left(\begin{bmatrix}1&1\\2&3\end{bmatrix} \cdot \begin{bmatrix}1&2\\3&4\end{bmatrix}\right) \cdot \begin{bmatrix}3\\1\end{bmatrix} = \begin{bmatrix}18\\49\end{bmatrix}$

Option 2: $\begin{bmatrix}1&1\\2&3\end{bmatrix} \cdot \left(\begin{bmatrix}1&2\\3&4\end{bmatrix} \cdot \begin{bmatrix}3\\1\end{bmatrix}\right) = \begin{bmatrix}18\\49\end{bmatrix}$

Which option would you choose to evaluate this matrix multiplication ?

The power of associativity

Option 1: $\left(\begin{bmatrix}1&1\\2&3\end{bmatrix} \bullet \begin{bmatrix}1&2\\3&4\end{bmatrix}\right) \bullet \begin{bmatrix}3\\1\end{bmatrix} = \begin{bmatrix}18\\49\end{bmatrix}$

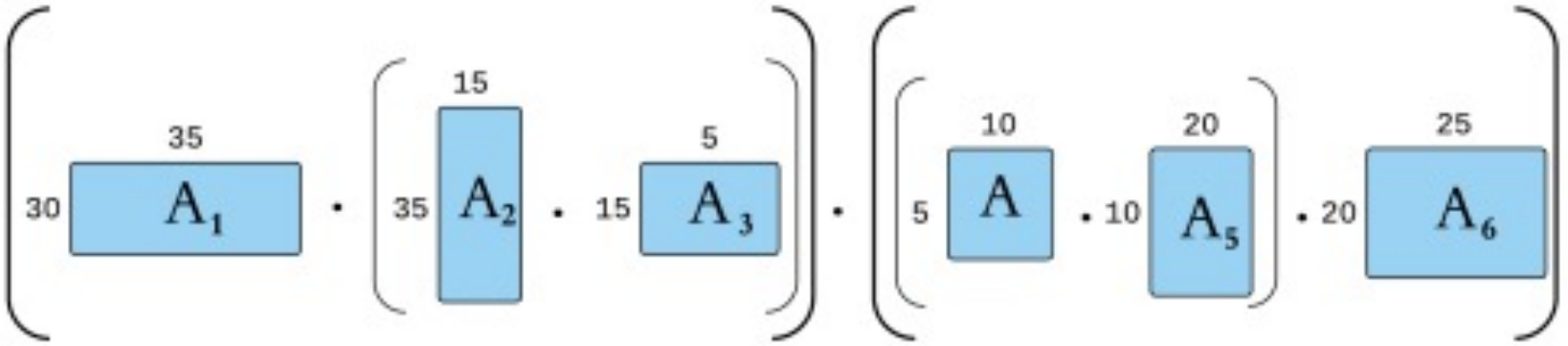
$\begin{bmatrix}4&6\\11&16\end{bmatrix}$

Option 2: $\begin{bmatrix}1&1\\2&3\end{bmatrix} \bullet \left(\begin{bmatrix}1&2\\3&4\end{bmatrix} \bullet \begin{bmatrix}3\\1\end{bmatrix}\right) = \begin{bmatrix}18\\49\end{bmatrix}$

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

Matrix chain multiplication

Given n matrices, what is the optimal sequence to multiply them? ?

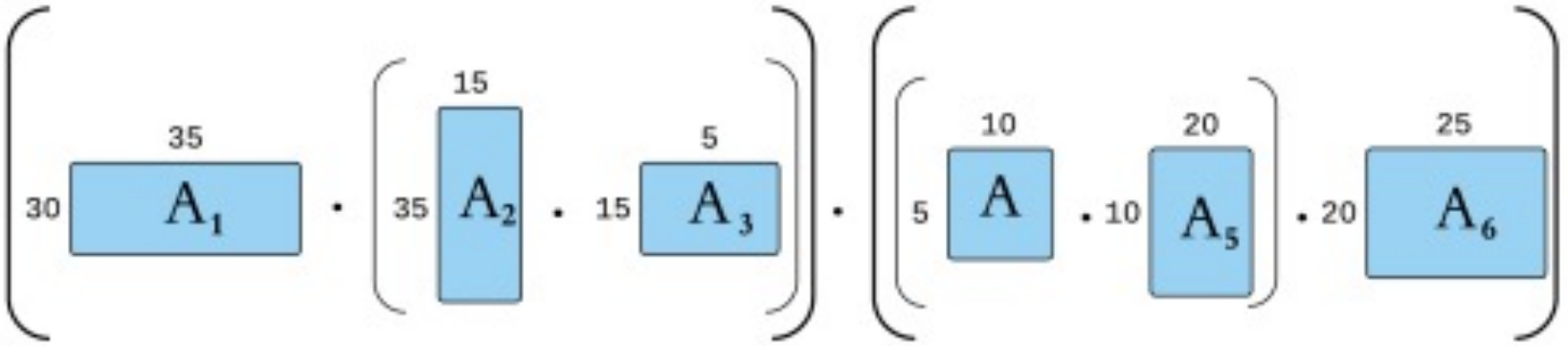


This is an example optimal factorization. What is its cost? ?

See also https://en.wikipedia.org/wiki/Catalan_number , https://en.wikipedia.org/wiki/Matrix_chain_multiplication , https://en.wikipedia.org/wiki/Matrix_multiplication#Associativity
Source figure: https://bruceoutdoors.wordpress.com/2015/11/24/matrix-chain-multiplication-with-c-code-part-3-extracting-the-sequence/
Wolfgang Gatterbauer. Principles of scalable data management: https://northeastern-datalab.github.io/cs7240/

Matrix chain multiplication

Given n matrices, what is the optimal sequence to multiply them? ?



This is an example optimal factorization. What is its cost? ?

MinCost: (30*35*5 + (35*15*5)) + 30*5*25 + (5*10*20) + 5*20*25)

Nave method: all possible way to place closed parentheses: "Catalan numbers"

Via Dynamic programming: $O(n^3)$

Best known: O(n log n)

C_n is the number of different ways $n + 1$ factors can be completely parenthesized (or the number of ways of associating n applications of a binary operator, as in the matrix chain multiplication problem). For $n = 3$, for example, we have the following five different parenthesizations of four factors:

((ab)c)d (a(bc))d (ab)(cd) a((bc)d) a(b(cd))

See also https://en.wikipedia.org/wiki/Catalan_number , https://en.wikipedia.org/wiki/Matrix_chain_multiplication , https://en.wikipedia.org/wiki/Matrix_multiplication#Associativity
Source figure: https://bruceoutdoors.wordpress.com/2015/11/24/matrix-chain-multiplication-with-c-code-part-3-extracting-the-sequence/

Commutativity & Associativity

Outer joins

R

A	B
1	2

S

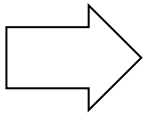
B	C
2	3

T

A	C
4	5

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

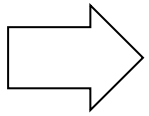
Result



?

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

Result



?

Commutativity & Associativity


R

A	B
1	2

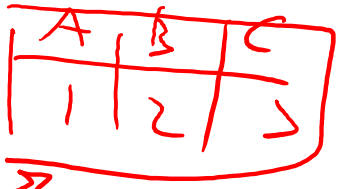
S

B	C
2	3

T

A	C
4	5

Outer joins

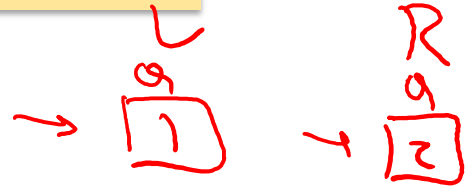


```
SELECT A, B, C
FROM (R
NATURAL FULL JOIN S)
NATURAL FULL JOIN T
```

Result

A	B	C
1	2	3
4	NULL	5

```
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

A	B	C
1	2	3
4	L	5

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

Result

A	B	C
1	2	3
4	NULL	5

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

Result

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

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

Example: Data Sources on Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

Accommodations

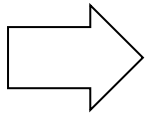
Country	City	Hotel	Stars
Canada	Toronto	Plaza	4
Canada	London	Ramada	3
Bahamas	Nassau	Hilton	

Sites

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

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

Example: Data Sources on Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

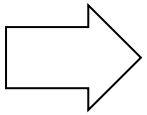
Accommodations

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 (Accommodations
NATURAL FULL JOIN Climates)
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

Example: Data Sources on 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 (Accommodations
NATURAL FULL JOIN Climates)
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

Example: Data Sources on 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 (Accommodations
NATURAL FULL JOIN Climates)
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

Example: Data Sources on Tourist Information

Climates

Country	Climate
Canada	diverse
Bahamas	tropical
UK	temperate

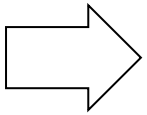
Accommodations

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 Accommodations
NATURAL FULL JOIN (Climates
NATURAL FULL JOIN Sites)
```



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

Example: Data Sources on 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 Accommodations
NATURAL FULL JOIN (Climates
NATURAL FULL JOIN Sites)
```

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

Full disjunction

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! We may discuss later in class in more detail (or skip this year)

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

Full disjunction: definition

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	

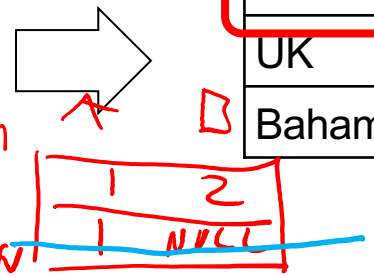
Sites

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

- Two tuples (max one from each relation) are join consistent if they agree on common attributes, e.g. t_1/t_2, t_3/t_4. A set of tuples is join consistent if every pair is join consistent.
- Set of tuples (max one from each relation) is connected if the schema is connected, thus share attributes
- A tuple is in the Full disjunction if it is the inner join from tuples that are connected, join consistent, and there is no superset with both conditions (related to "subsumption").

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		



SUBSUMTION

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

- SQL
  - Schema, keys, referential integrity
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Theta Joins
  - Nulls & 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

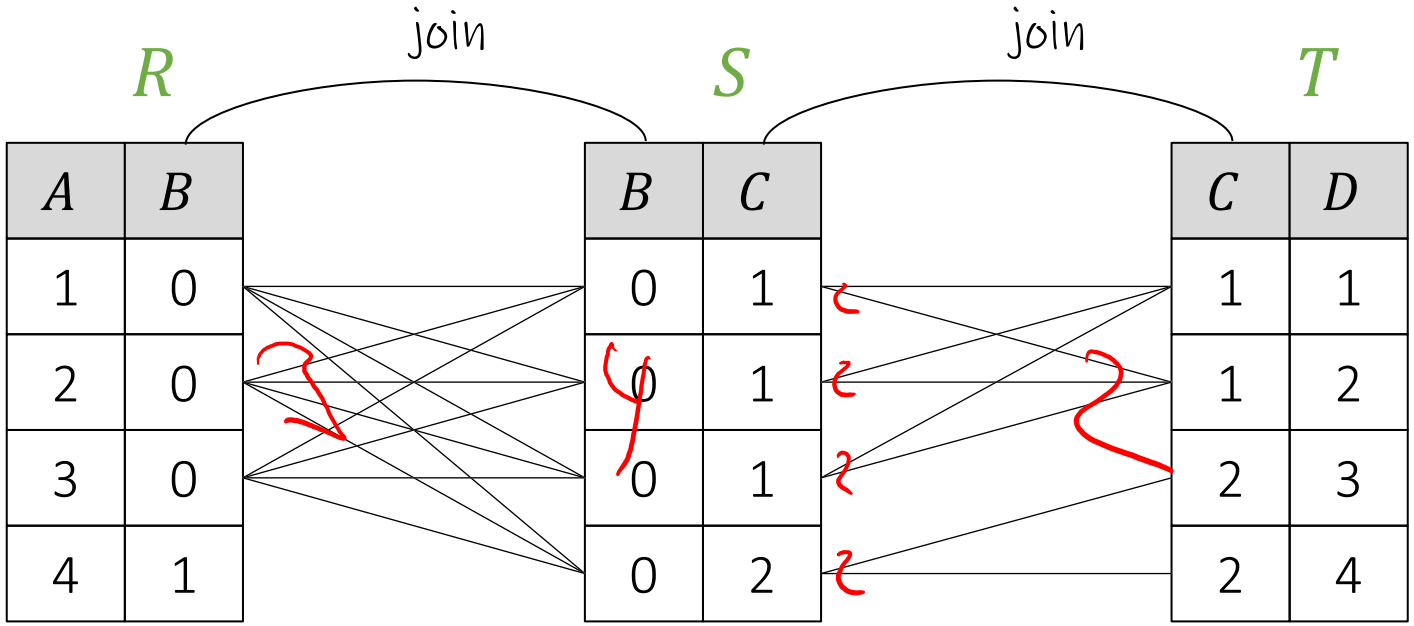
```
select   A, R.B, S.C, D

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



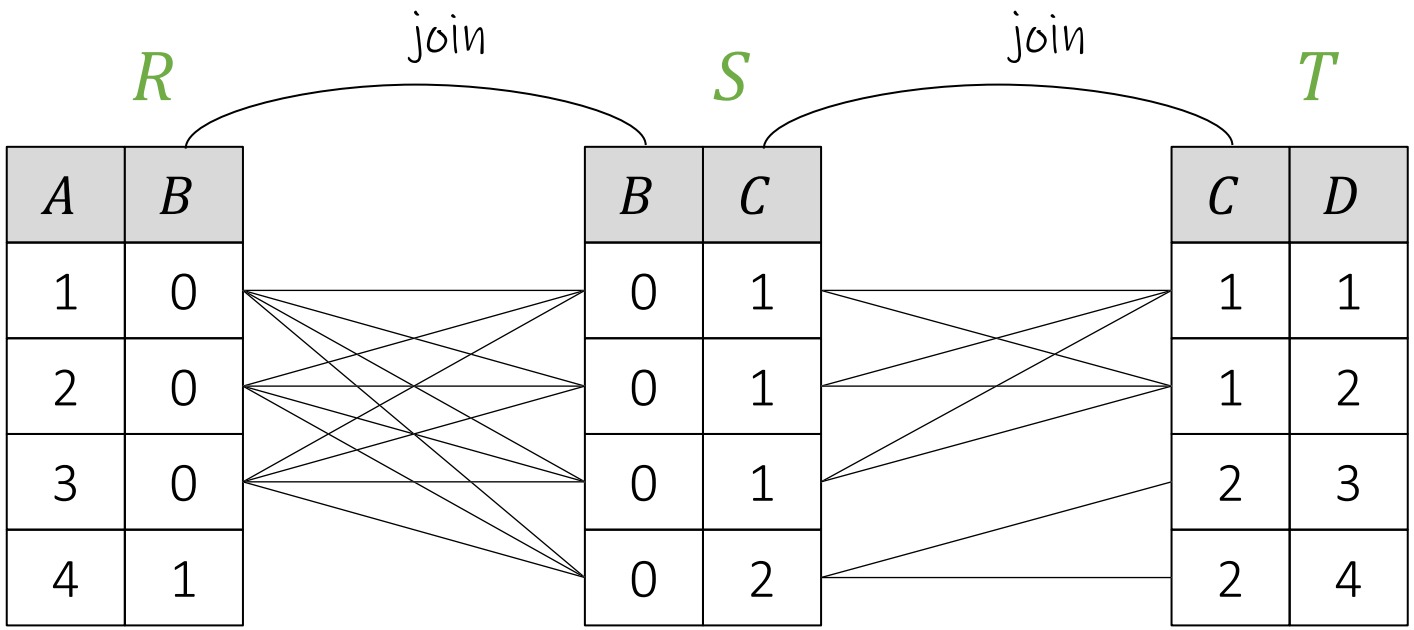

```
select   A, R.B, S.C, D

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



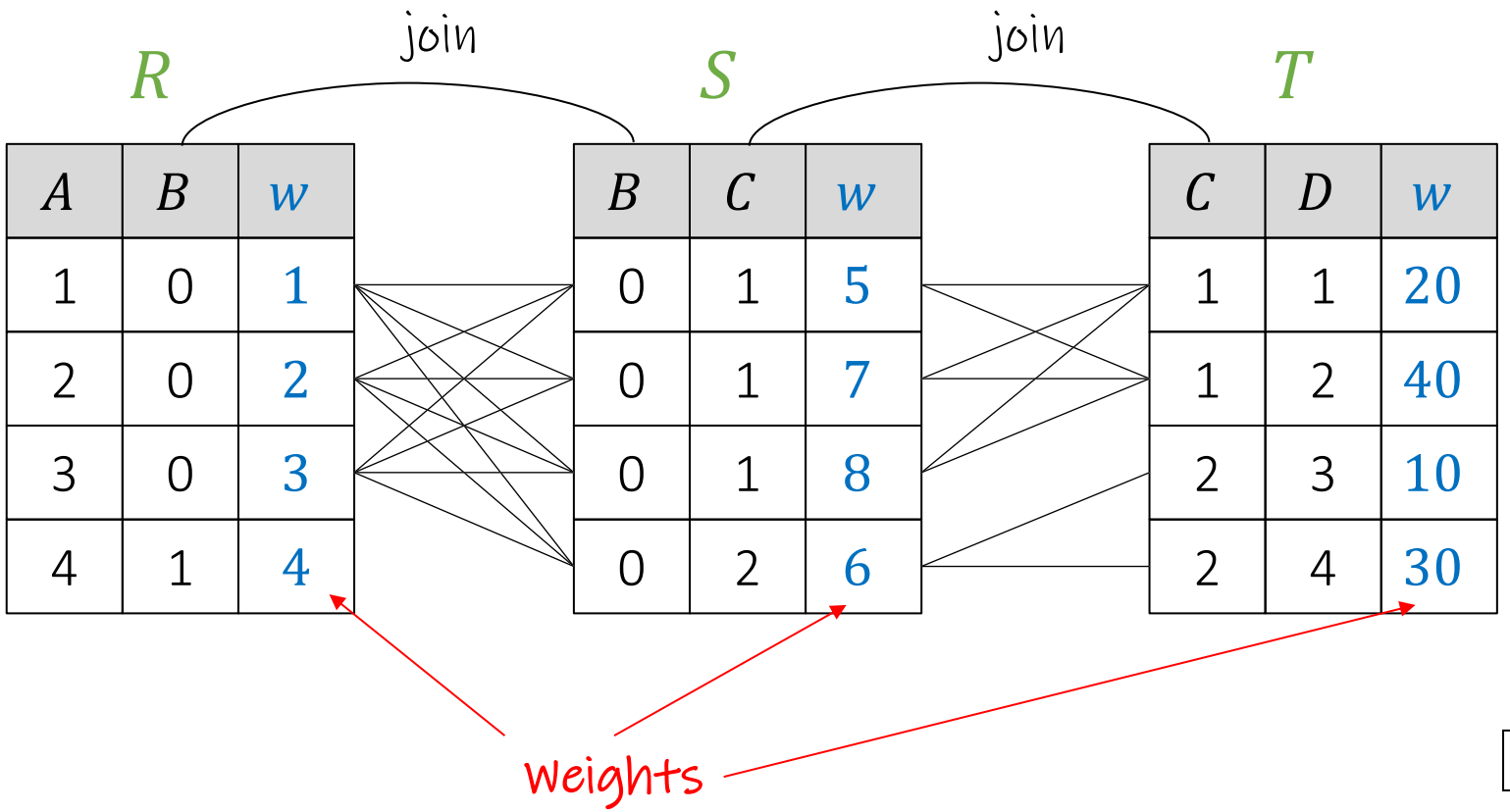

```
select   A, R.B, S.C, D

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?

?