Updated 1/11/2020

T1: Data models and query languages
L2: SQL (continued)

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp20)

https://northeastern-datalab.github.io/cs7240/sp20/

Version 1/10/2020

Topic 1: Data models and query languages

- **Lecture 1 (Tue 1/7):** Course introduction, SQL
  - Introduction, SQL (a refresher)
- **Lecture 2 (Fri 1/10):** Relational algebra, calculus & Codd's theorem
- **Lecture 3 (Tue 1/14):** Information theory & normal forms
- **Lecture 4 (Fri 1/17):** Alternative data models

Pointers to relevant concepts & supplementary material:

- SQL refresher: [SAMS'12], [Cow'03] Ch3 & Ch5, [Complete'14] Ch6
- first-order logic, relational calculus, relational algebra, Datalog, Codd's theorem: [Alice'95] Ch3 & Ch4 & Ch24.1, [Cow'03] Ch4, [Complete'14] Ch2 & Ch5
- normal forms and their information-theoretic justification: [Complete'14] Ch3, [Lee'87], [Arenas, Libkin'05]
- alternative data models and NoSQL: [Hellerstein, Stonebraker'05], [Sadalage, Fowler'12], [Harrison'16]

Topic 2: Complexity of query evaluation

- **Lecture 4 (Fri 1/17):** Conjunctive queries 1
- **Lecture 5 (Tue 1/21):** Conjunctive queries 1
- **Lecture 6 (Fri 1/24): (A1 due)** Query containment & minimization
- **Lecture 7 (Tue 1/28):** Acylic queries
- **Lecture 8 (Fri 1/31):** Cyclic queries 1
- **Lecture 9 (Tue 2/4):** Cyclie queries 2
- **Lecture 10 (Fri 2/7): (A2 due)** Top-k

Pointers to relevant concepts & supplementary material:

- semantics of conjunctive queries, connection to constraint satisfaction problems (CSPs) and homomorphisms, data vs. query complexity: [Kolaitis, Vardi'00], [Koutris'19] L1,
- query containment, query minimization, adsorption: [Koutris'19] L2
- acyclic joins: query hypergraph, Yannakakis algorithm, GYO reduction, dynamic programming, algebraic semirings, ranked enumeration: [Alice] Ch6.4, [Koutris'19] L4, [Tziavelis+'19]
- cyclic joins: tree & hypertree decomposition, fractional hypertree width, AGM bound, worst-case optimal join algorithms, optimal algorithms, submodular width and multiple decompositions: [AGM'13], [NPRR'18], [KNR'17], [KNS'17]
- top-k: [Roughgarden'10], [Ilyas+08], [Rahul, Tao'19], [Tziavelis+'19]

Scribe sign-up sheet

Please sign-up to be the scribe for 3-5 lectures (depending on the number of students in class) by first copying this template below and create one single "student answer" in Piazza that all students can then edit (https://trunkuserguide.screenstepslive.com/s/5891/m/18197/l/195294-how-do-students-respond-to-other-student-s-questions-in-piazza)

Notice that we may have lectures that continue the topic from a previous lecture. In those cases it is best for the scribes to work together on a single document for two lectures (we will discuss those situations in class).

L1: 1/7
L2: 1/10
L3: 1/14
L4: 1/17
L5: 1/21
L6: 1/24
L7: 1/28
L8: 1/31
L9: 2/4
L10: 2/7

L11: 2/11
L12: 2/14
L13: 2/18
L14: 2/21
L15: 2/25
L16: 2/28

Spring break

L17: 3/10
L18: 3/13: Midterm (no lecture)
L19: 3/17
L20: 3/20

L21: 3/24
L22: 3/27
L23: 3/31
L24: 4/3
L25: 4/7
L26: 4/10
L27: 4/14: Project presentations (no lecture)
L28: 4/17: Project presentations (no lecture)

#pin

logistics

edit · good question | 0

Updated Just now by Wolfgang Gatterbauer

the instructors' answer, *where instructors collectively construct a single answer*

Click to start off the wiki answer

followup discussions *for lingering questions and comments*

Start a new followup discussion

Compose a new followup discussion

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

Subqueries = Nested queries

We only focus on nestings in the WHERE clause, which is the most expressive type of nesting

Outer block

Inner block

```
SELECT  ...
FROM    ...
WHERE   ...
   (SELECT  ...
      FROM    ...
      WHERE   ...  )
```

- We can nest queries because SQL is compositional:
  - Everything (inputs / outputs) is represented as multisets
  - the output of one query can thus be used as the input to another (nesting)
  - Subqueries return relations
- This is extremely powerful!

Subqueries in WHERE

What do these queries compute?

R

a
1
2

U

a
2
3
4

```
SELECT  a
FROM    R
WHERE   a IN
        (SELECT * from U)
```

⇨ ?

```
SELECT  a
FROM    R
WHERE   a < ANY
        (SELECT * from U)
```

⇨ ?

```
SELECT  a
FROM    R
WHERE   a < ALL
        (SELECT * from U)
```

⇨ ?

Subqueries in WHERE

R

a
1
2

U

a
2
3
4

What do these queries compute?

```
SELECT   a
FROM     R
WHERE    a IN
         (SELECT * from U)
```

a
2

Since 2 is in the set (bag) (2, 3, 4)

```
SELECT   a
FROM     R
WHERE    a < ANY
         (SELECT a from U)
```

a
1
2

Since 1 and 2 are < than at least one ("any") of 2, 3 or 4

```
SELECT   a
FROM     R
WHERE    a < ALL
         (SELECT * from U)
```

a
1

Since 1 is < than each ("all") of 2, 3, and 4

Correlated subqueries

- In previous cases, the nested subquery in the inner select block could be entirely evaluated before processing the outer select block.
- This is no longer the case for correlated nested queries.
- Whenever a condition in the WHERE clause of a nested query references some column of a table declared in the outer query, the two queries are said to be correlated.
- The nested query is then evaluated once for each tuple (or combination of tuples) in the outer query.

Correlated subquery (existential)

Product (pname, price, cid)
Company (cid, cname, city)

Existential quantifiers ∃

Q: Find all companies that make some products with price < 25!

Using IN:

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    C.cid IN ( 1, 2 )
```

cid	CName	City
1	GizmoWorks	Oslo
2	Canon	Osaka
3	Hitachi	Kyoto

PName	Price	cid
Gizmo	$19.99	1
Powergizmo	$29.99	1
SingleTouch	$14.99	2
MultiTouch	$203.99	3

Correlated subquery (existential)

Product (pname, price, cid)
Company (cid, cname, city)

Existential quantifiers ∃

Q: Find all companies that make some products with price < 25!

Using IN:

"Set membership"

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    C.cid IN ( SELECT  P.cid
                    FROM    Product P
                    WHERE   P.price < 25)
```

cid	CName	City
1	GizmoWorks	Oslo
2	Canon	Osaka
3	Hitachi	Kyoto

PName	Price	cid
Gizmo	$19.99	1
Powergizmo	$29.99	1
SingleTouch	$14.99	2
MultiTouch	$203.99	3

Correlated subquery (existential)

Product (pname, price, cid)
Company (cid, cname, city)

Existential quantifiers ∃

Q: Find all companies that make <u>some</u> products with price < 25!

EXISTS is true iff the subquery's result is not empty

Using EXISTS:

"Test for empty relations"

```
SELECT  DISTINCT C.cname
FROM    Company C
WHERE   EXISTS ( SELECT  *
                 FROM    Product P
                 WHERE   C.cid = P.cid
                   and   P.price < 25)
```

Correlated subquery

cid	CName	City
1	GizmoWorks	Oslo
2	Canon	Osaka
3	Hitachi	Kyoto

PName	Price	cid
Gizmo	$19.99	1
Powergizmo	$29.99	1
SingleTouch	$14.99	2
MultiTouch	$203.99	3

Correlated subquery (existential)

Product (pname, price, cid)
Company (cid, cname, city)

Existential quantifiers ∃

Q: Find all companies that make some products with price < 25!

Using ANY (also some):

"Set comparison"

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    25 > ANY ( SELECT  price
                    FROM    Product P
                    WHERE   P.cid = C.cid)
```

cid	CName	City
1	GizmoWorks	Oslo
2	Canon	Osaka
3	Hitachi	Kyoto

PName	Price	cid
Gizmo	$19.99	1
Powergizmo	$29.99	1
SingleTouch	$14.99	2
MultiTouch	$203.99	3

Correlated subquery

Correlated subquery (existential)

Product (pname, price, cid)
Company (cid, cname, city)

Existential quantifiers ∃

Q: Find all companies that make some products with price < 25!

Now, let's unnest:

```
SELECT  DISTINCT C.cname
FROM    Company C, Product P
WHERE   C.cid = P.cid
   and  P.price < 25
```

cid	CName	City
1	GizmoWorks	Oslo
2	Canon	Osaka
3	Hitachi	Kyoto

PName	Price	cid
Gizmo	$19.99	1
Powergizmo	$29.99	1
SingleTouch	$14.99	2
MultiTouch	$203.99	3

Existential quantifiers are easy ! ☺

Correlated subquery (universal)

315

Product (pname, price, cid)
Company (cid, cname, city)

Universal quantifiers ∀

Q: Find all companies that make only products with price < 25!

same as:

Q: Find all companies for which all products have price < 25!

Universal quantifiers are more complicated ! ☹
(Think about the companies that should not be returned)

Correlated subquery (exist not -> universal)

Q: Find all companies that make <u>only</u> products with price < 25!

1. Find the other companies: i.e. they have some product ≥ 25!

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    C.cid IN (  SELECT   P.cid
                     FROM     Product P
                     WHERE    P.price >= 25)
```

2. Find all companies s.t. all their products have price < 25!

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    C.cid NOT IN ( SELECT   P.cid
                        FROM     Product P
                        WHERE    P.price >= 25)
```

Correlated subquery (exist not -> universal)


315

Product (pname, price, cid)
Company (cid, cname, city)

Universal quantifiers ∀

Q: Find all companies that make only products with price < 25!

Using NOT EXISTS:

```
SELECT  DISTINCT C.cname
FROM    Company C
WHERE   NOT EXISTS ( SELECT  *
                     FROM    Product P
                     WHERE   C.cid = P.cid
                        and  P.price >= 25)
```

Correlated subquery (exist not -> universal)

315

Product (pname, price, cid)
Company (cid, cname, city)

Universal quantifiers ∀

Q: Find all companies that make only products with price < 25!

Using ALL:

```
SELECT   DISTINCT C.cname
FROM     Company C
WHERE    25 > ALL (   SELECT   price
                      FROM     Product P
                      WHERE    P.cid = C.cid)
```

SQLlite does not support "ALL" ☹

A natural question

- How can we unnest the universal quantifier query ?

Queries that must be nested

- Definition: A query Q is monotone if:
  - Whenever we add tuples to one or more of the tables…
  - … the answer to the query cannot contain fewer tuples
- Fact: all unnested queries are monotone
  - Proof: using the "nested for loops" semantics
- Fact: Query with universal quantifier is not monotone
  - Add one tuple violating the condition. Then "all" returns fewer tuples
- Consequence: we cannot unnest a query with a universal quantifier

Outline: SQL (a refresher)

- SQL
  - Schema and keys
  - Joins
  - Aggregates and grouping
  - Nested queries (Subqueries)
  - Understanding nested queries

The sailors database

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Figure 5.1 An Instance *S*3 of Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Figure 5.2 An Instance *R*2 of Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Figure 5.3 An Instance *B*1 of Boats

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 1

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
      ( SELECT R.sid
        FROM Reserves R
        WHERE R.bid IN
             ( SELECT B.bid
               FROM Boats B
               WHERE B.color = 'red'))
```

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 2

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
       ( SELECT R.sid
       FROM Reserves R
       WHERE R.bid not IN
            ( SELECT B.bid
            FROM Boats B
            WHERE B.color = 'red'))
```

They must have reserved at least one boat in another color

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 3

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have not reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
       ( SELECT R.sid
       FROM Reserves R
       WHERE R.bid IN
            ( SELECT B.bid
            FROM Boats B
            WHERE B.color = 'red'))
```

They can have reserved 0 or more boats in another color, but must not have reserved any red boat

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

More nested Queries 4

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved only red boats

Q: Find the names of sailors who have not reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
      ( SELECT R.sid
      FROM Reserves R
      WHERE R.bid not IN
           ( SELECT B.bid
           FROM Boats B
           WHERE B.color = 'red'))
```

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 5

Sailors (<u>sid</u>, sname, rating, age)
Reserves (<u>sid, bid, day</u>)
Boats (<u>bid</u>, bname, color)

340

= Find the names of sailors who have reserved all red boats

Q: Find the names of sailors so there is no red boat that is not reserved by him.

```
SELECT S.sname
FROM Sailors S
WHERE not exists
      ( SELECT B.bid
      FROM Boats B
      WHERE B.color = 'red'
      AND not exists
            ( SELECT R.bid
            FROM Reserves R
            WHERE R.bid = B.bid
            AND R.sid = S.sid))
```

To understand semantics of nested queries, think of a nested loops evaluation: For each Sailors tuple, check the qualification by computing the subquery

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

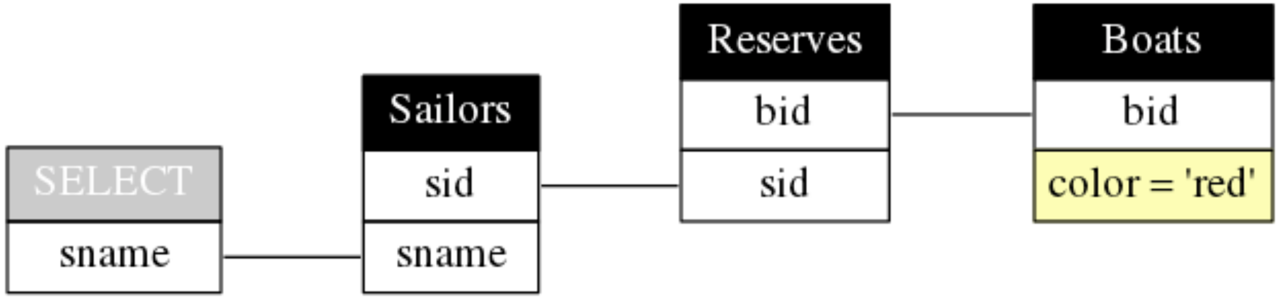
Once more: 1

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
      ( SELECT R.sid
      FROM Reserves R
      WHERE R.bid IN
           ( SELECT B.bid
           FROM Boats B
           WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

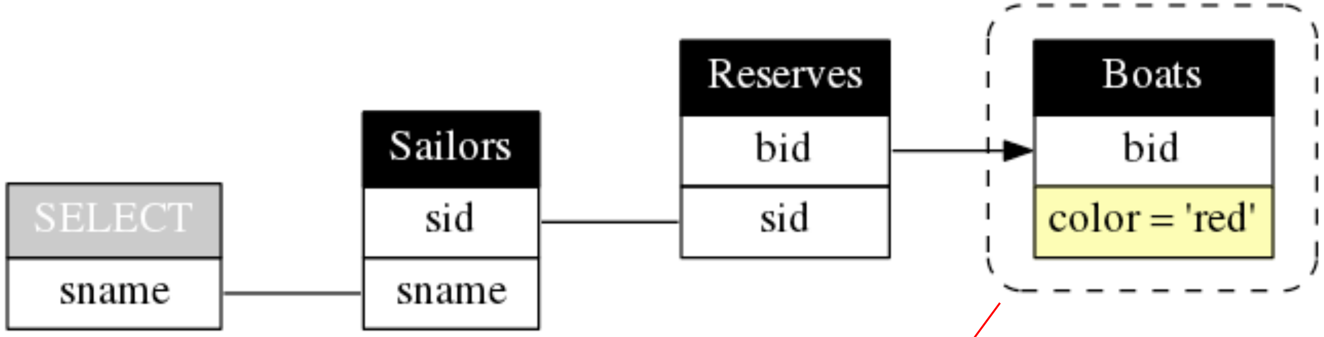
Once more: 2

Sailors (<u>sid</u>, sname, rating, age)
Reserves (<u>sid, bid, day</u>)
Boats (<u>bid</u>, bname, color)

340

Q: Find the names of sailors who have reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN
      ( SELECT R.sid
        FROM Reserves R
        WHERE R.bid not IN
             ( SELECT B.bid
               FROM Boats B
               WHERE B.color = 'red'))
```



Dashed lines represent not exists ∄

They must have reserved at least one boat in another color

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

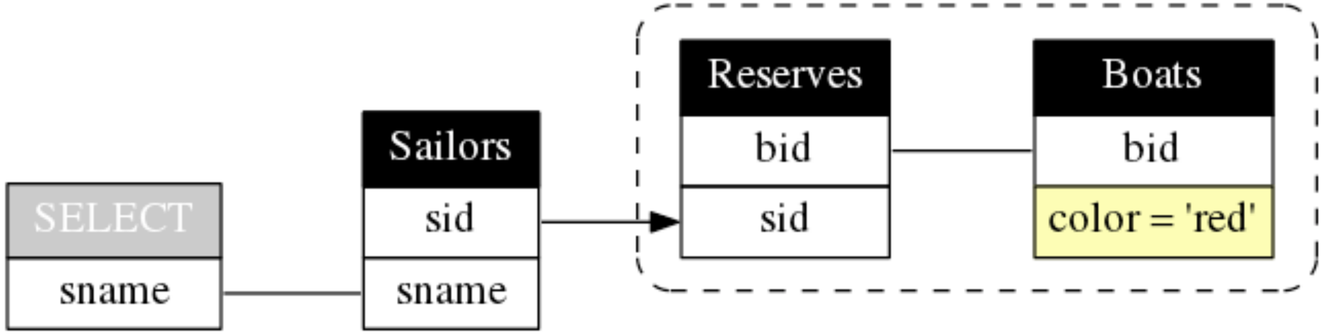
Once more: 3

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

Q: Find the names of sailors who have not reserved a red boat.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
      ( SELECT R.sid
        FROM Reserves R
        WHERE R.bid IN
             ( SELECT B.bid
               FROM Boats B
               WHERE B.color = 'red'))
```



They can have reserved 0 or more boats in another color, but must not have reserved any red boat

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 4

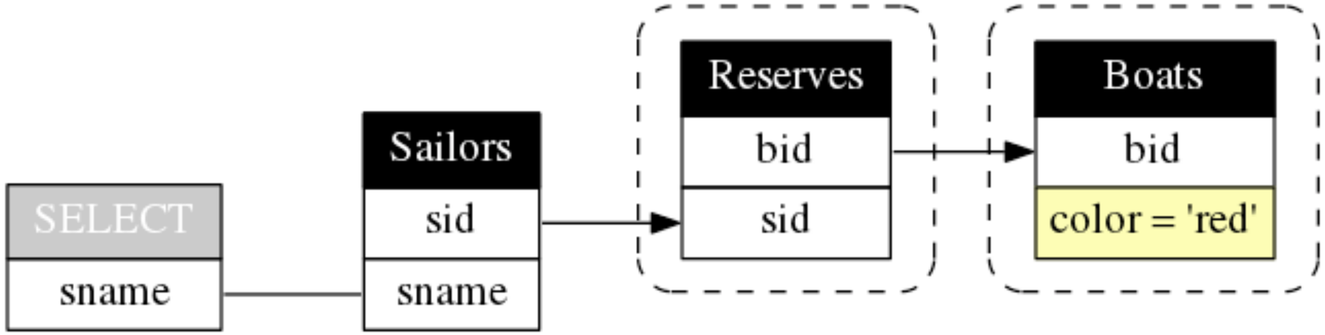
Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved only red boats

Q: Find the names of sailors who have not reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
        ( SELECT R.sid
          FROM Reserves R
          WHERE R.bid not IN
                ( SELECT B.bid
                  FROM Boats B
                  WHERE B.color = 'red'))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

Once more: 4

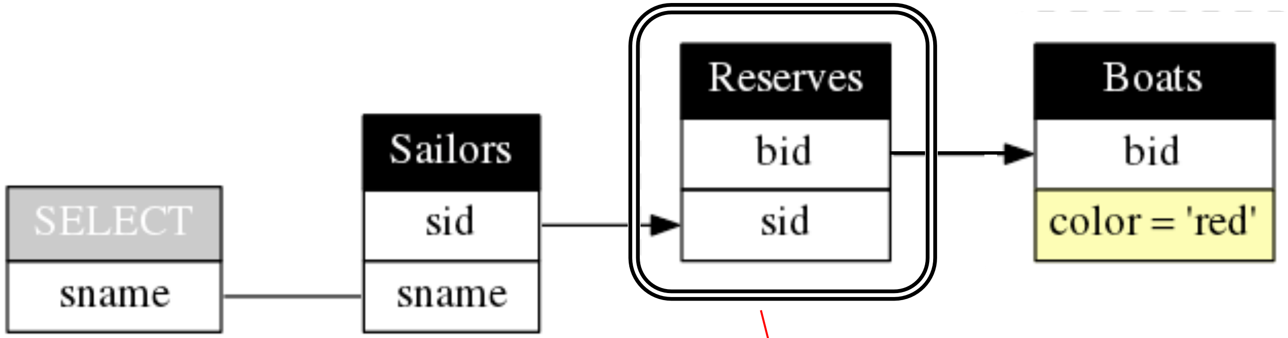
Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

340

= Find the names of sailors who have reserved only red boats

Q: Find the names of sailors who have not reserved a boat that is not red.

```
SELECT S.sname
FROM Sailors S
WHERE S.sid not IN
      ( SELECT R.sid
        FROM Reserves R
        WHERE R.bid not IN
             ( SELECT B.bid
               FROM Boats B
               WHERE B.color = 'red'))
```



Double lines represent for all ∀

Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

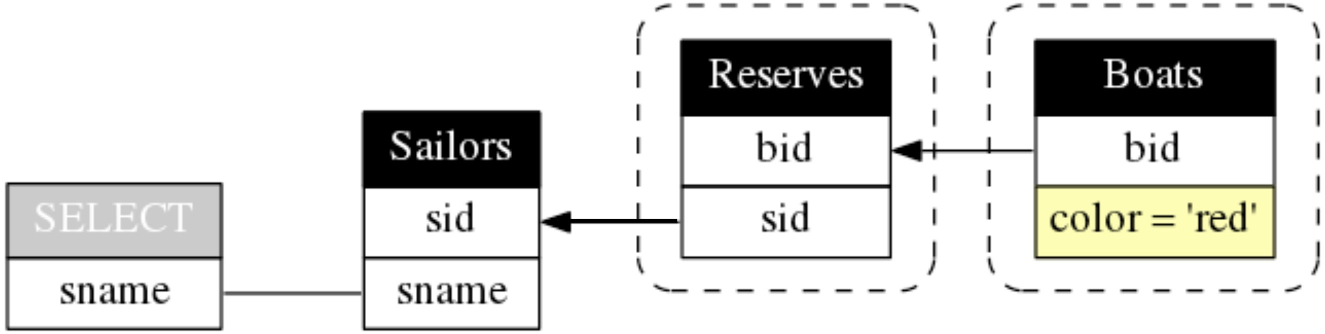
Once more: 5

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

= Find the names of sailors who have reserved all red boats

Q: Find the names of sailors so there is no red boat that is not reserved by him.

```
SELECT S.sname
FROM Sailors S
WHERE not exists
        ( SELECT B.bid
        FROM Boats B
        WHERE B.color = 'red'
        AND not exists
              ( SELECT R.bid
              FROM Reserves R
              WHERE R.bid = B.bid
              AND R.sid = S.sid))
```



Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

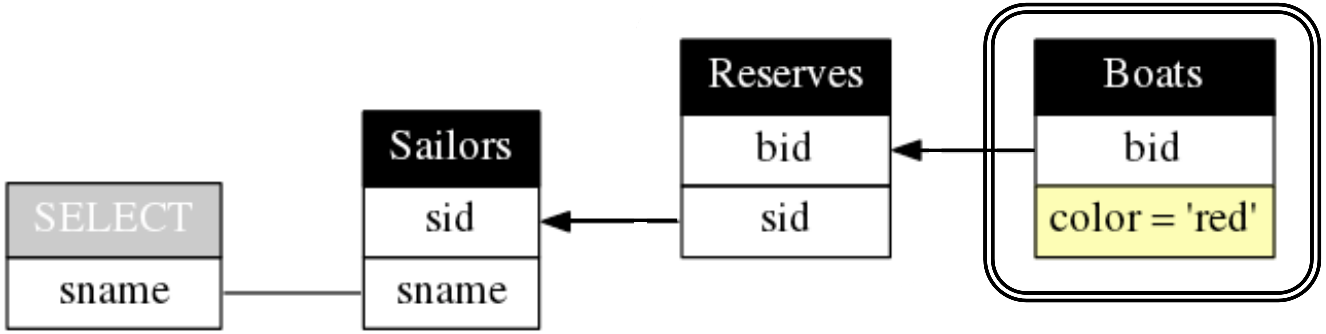
Once more: 5

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

= Find the names of sailors who have reserved all red boats

Q: Find the names of sailors so there is no red boat that is not reserved by him.

```
SELECT S.sname
FROM Sailors S
WHERE not exists
      ( SELECT B.bid
        FROM Boats B
        WHERE B.color = 'red'
        AND not exists
            ( SELECT R.bid
              FROM Reserves R
              WHERE R.bid = B.bid
              AND R.sid = S.sid))
```



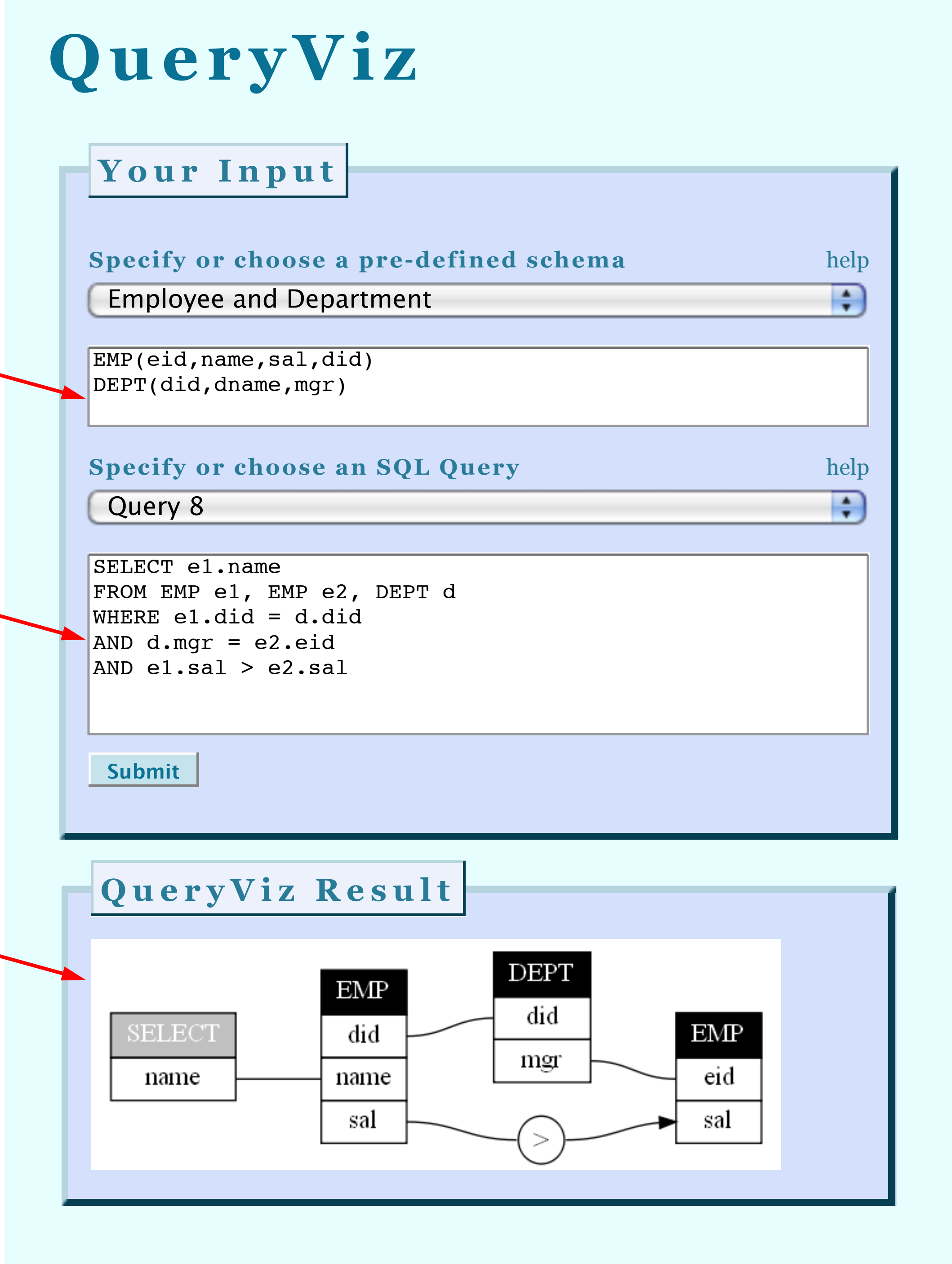
Query from: Ramakrishnan, Gehrke: Database management systems, 2nd ed (2000)

http://queryviz.com

Input: Schema

Input Query

Output: Visualization



http://queryviz.com/online

http://www.youtube.com/watch?v=kVFnQRGAQls

The person/bar/drinks example (formerly drinkers/bars/beers, courtesy Jeff Ullman)

331

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

Challenge: write these in SQL.
Solutions: http://queryviz.com/online/

Find persons that frequent some bar that serves some drink they like.

Find persons that frequent only bars that serve some drink they like.

Find persons that frequent some bar that serves only drinks they like.

Find persons that frequent only bars that serve only drinks they like.
(= Find persons who like all drinks that are served in all the bars they visit.)
(= Find persons for which there does not exist a bar they frequent that serves a drink they do not like.)

The person/bar/drinks example (formerly drinkers/bars/beers, courtesy Jeff Ullman)

Likes(person, drink)
Frequents(person, bar)
Serves(bar, drink)

Challenge: write these in SQL.
Solutions: http://queryviz.com/online/

Find persons that frequent <u>some</u> bar that serves <u>some</u> drink they like.

$x: \quad \exists y.\ \exists z.\ Frequents(x, y) \land Serves(y,z) \land Likes(x,z)$

Find persons that frequent <u>only</u> bars that serve <u>some</u> drink they like.

$x: \quad \forall y.\ Frequents(x, y) \Rightarrow (\exists z.\ Serves(y,z) \land Likes(x,z))$

Find persons that frequent <u>some</u> bar that serves <u>only</u> drinks they like.

$x: \quad \exists y.\ Frequents(x, y) \land \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$

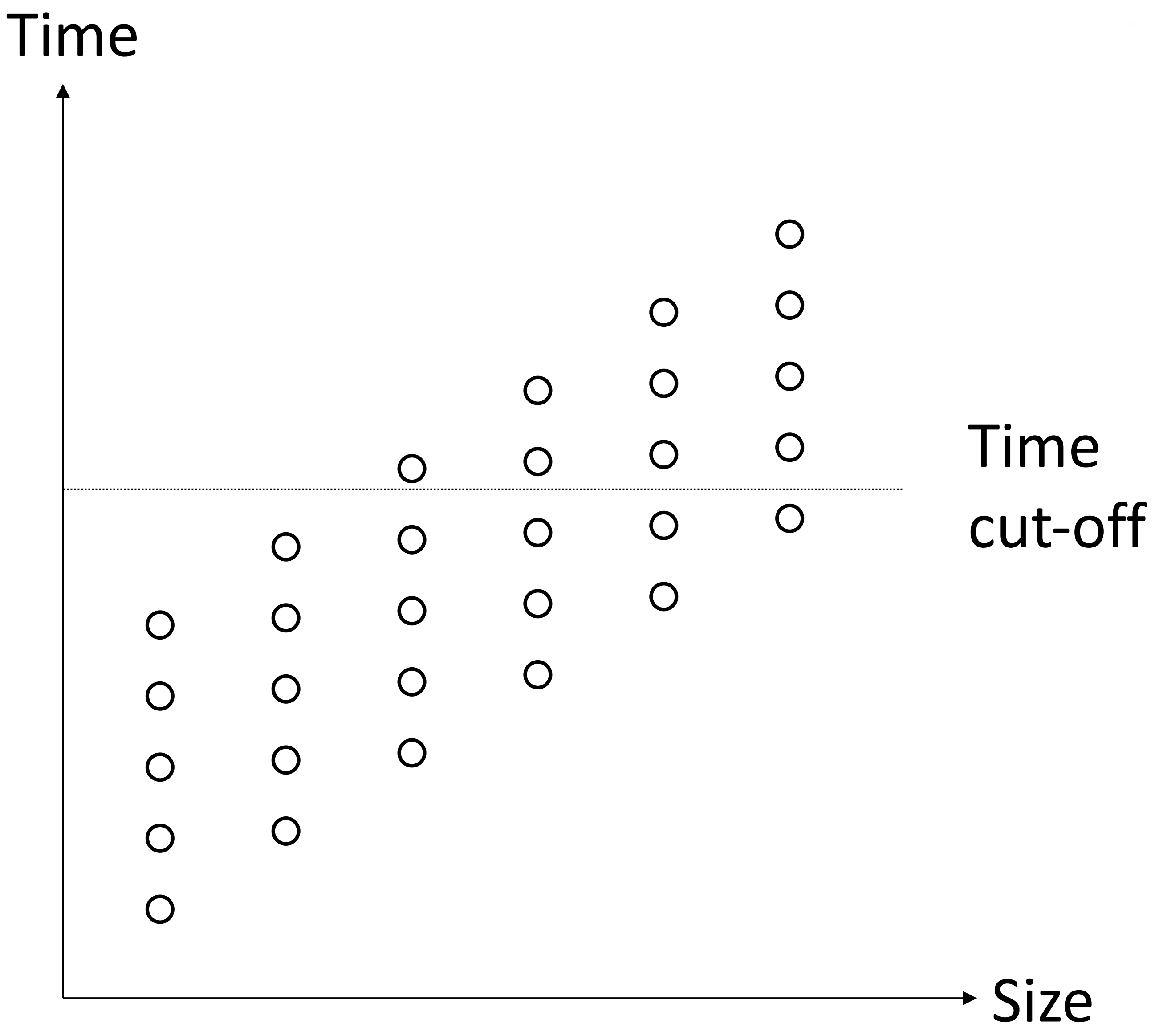
Find persons that frequent <u>only</u> bars that serve <u>only</u> drinks they like.
(= Find persons who like all drinks that are served in all the bars they visit.)
(= Find persons for which there does not exist a bar they frequent that serves a drink they do not like.)

$x: \quad \forall y.\ Frequents(x, y) \Rightarrow \forall z.(Serves(y,z) \Rightarrow Likes(x,z))$
$x: \quad \nexists y.\ Frequents(x, y) \land (\exists z.Serves(y,z) \land \nexists z2.\ Likes(x,2z))$

Revisiting our question from last time

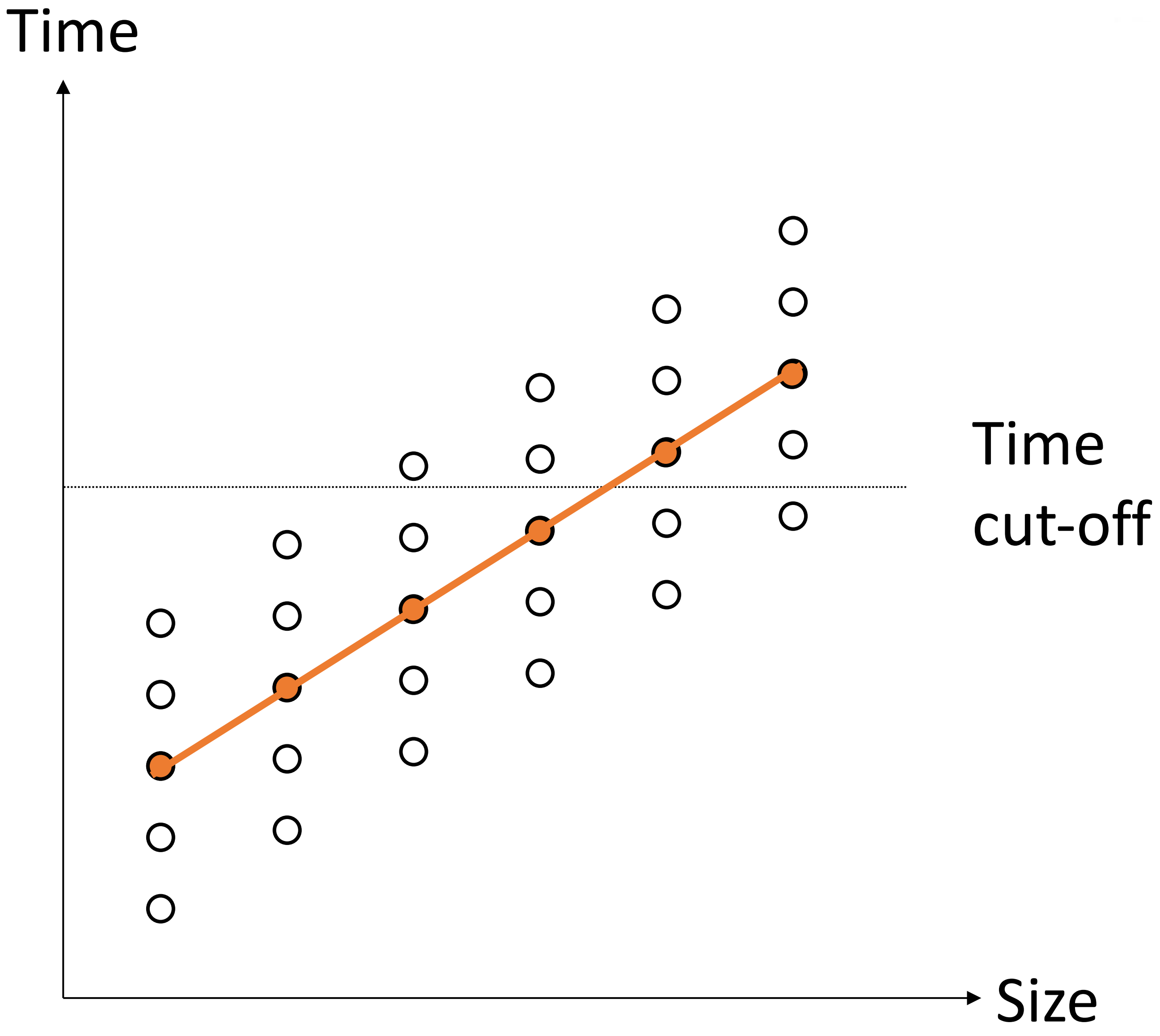
How to deal with cut-offs when binning

- 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

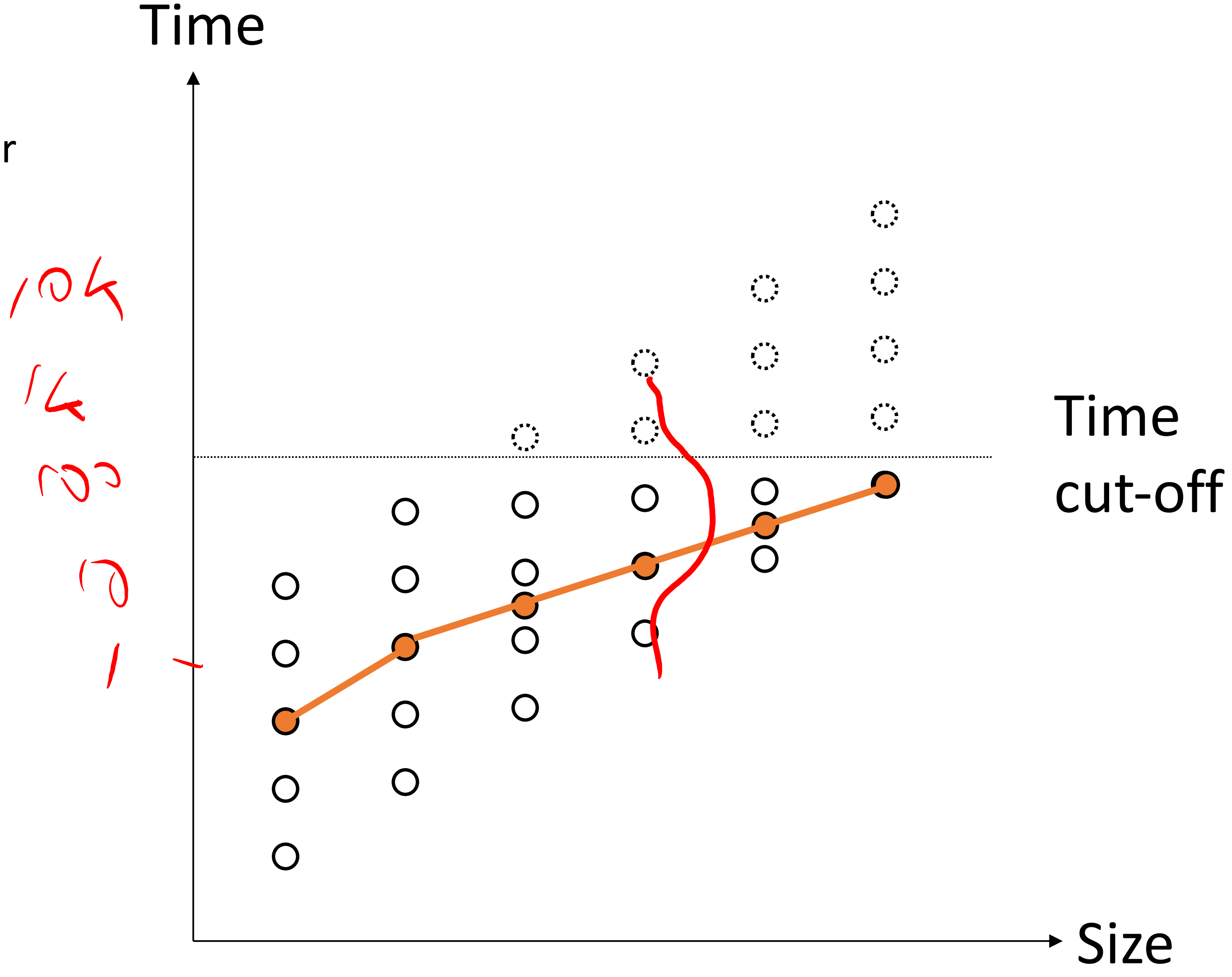
- Here is what the aggregate would look like like if we could get all points and then aggregated for each size



How to deal with cut-offs when binning

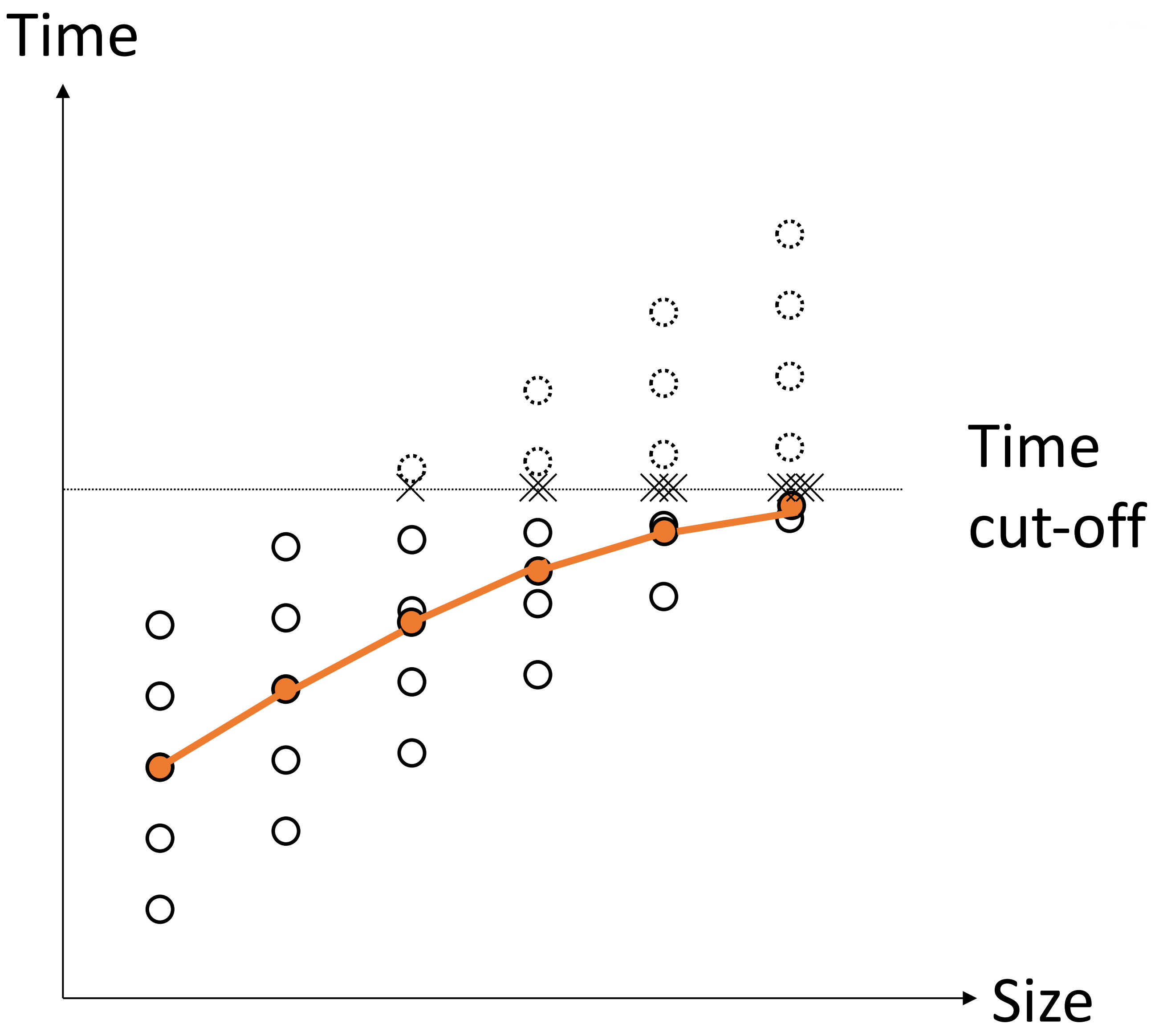
- Here is what happens if we throw away all those points that take longer than the cut-off, and only average over the "seen points"

- What would you do?



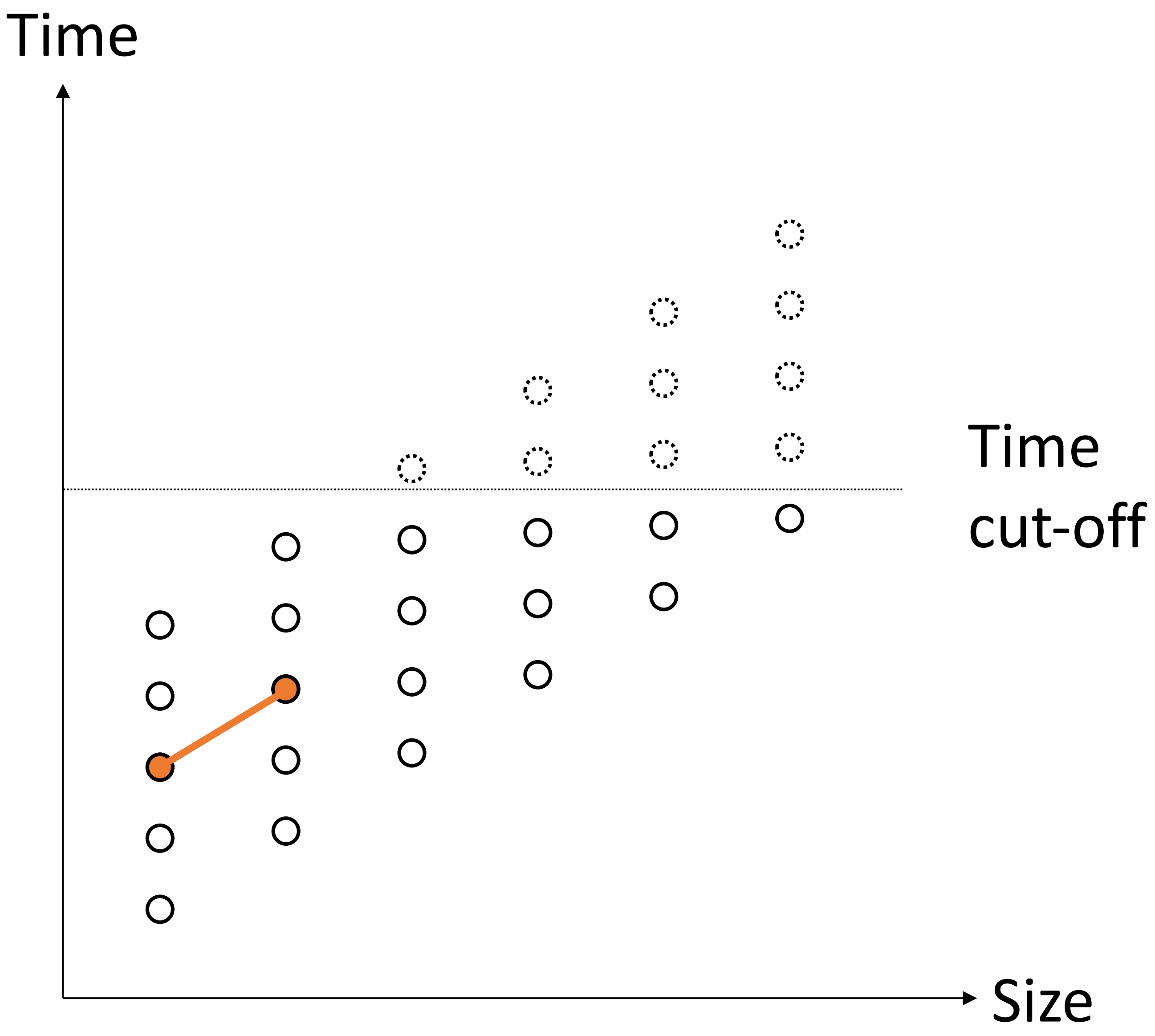
How to deal with cut-offs when binning

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



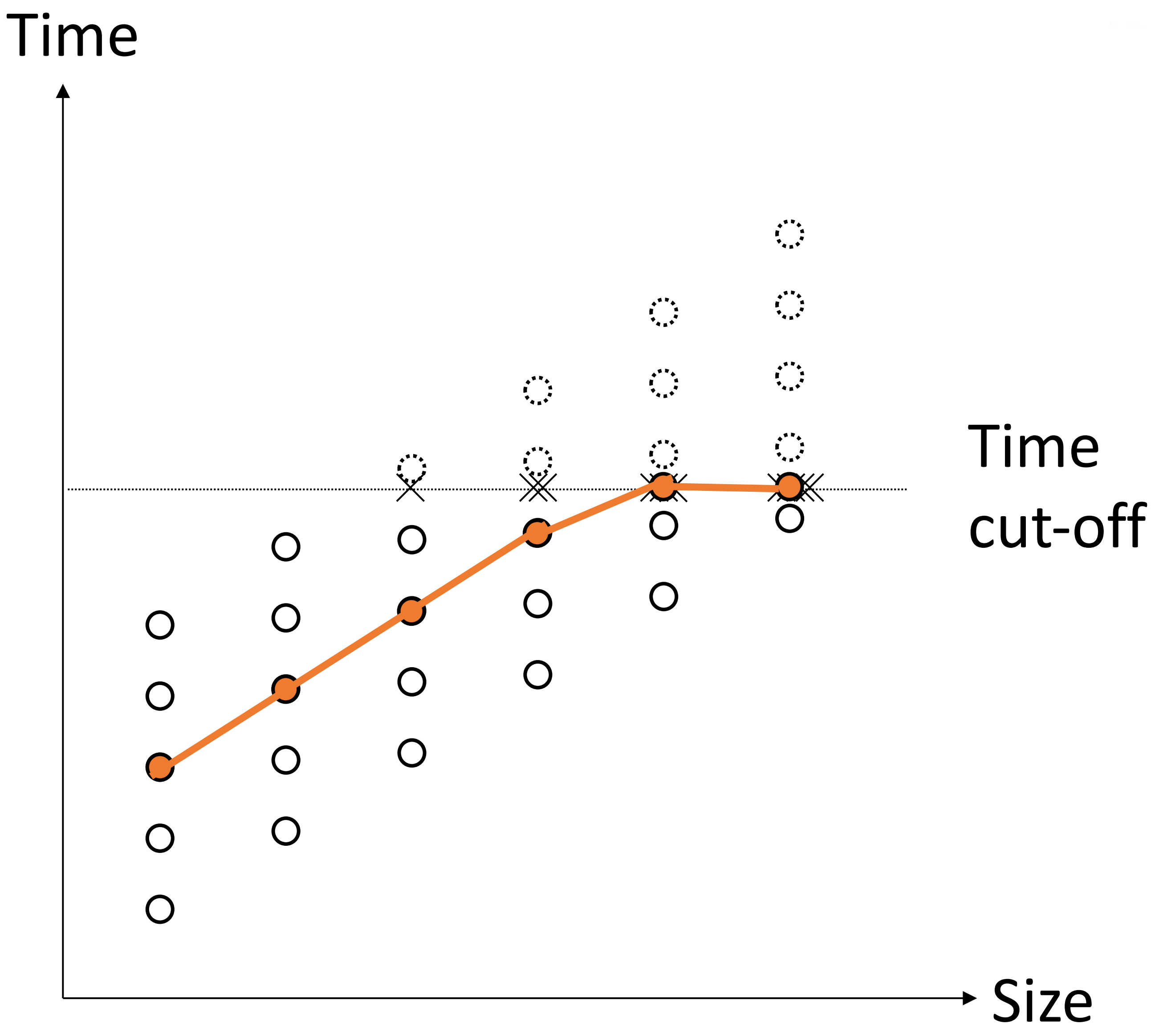
How to deal with cut-offs when binning

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



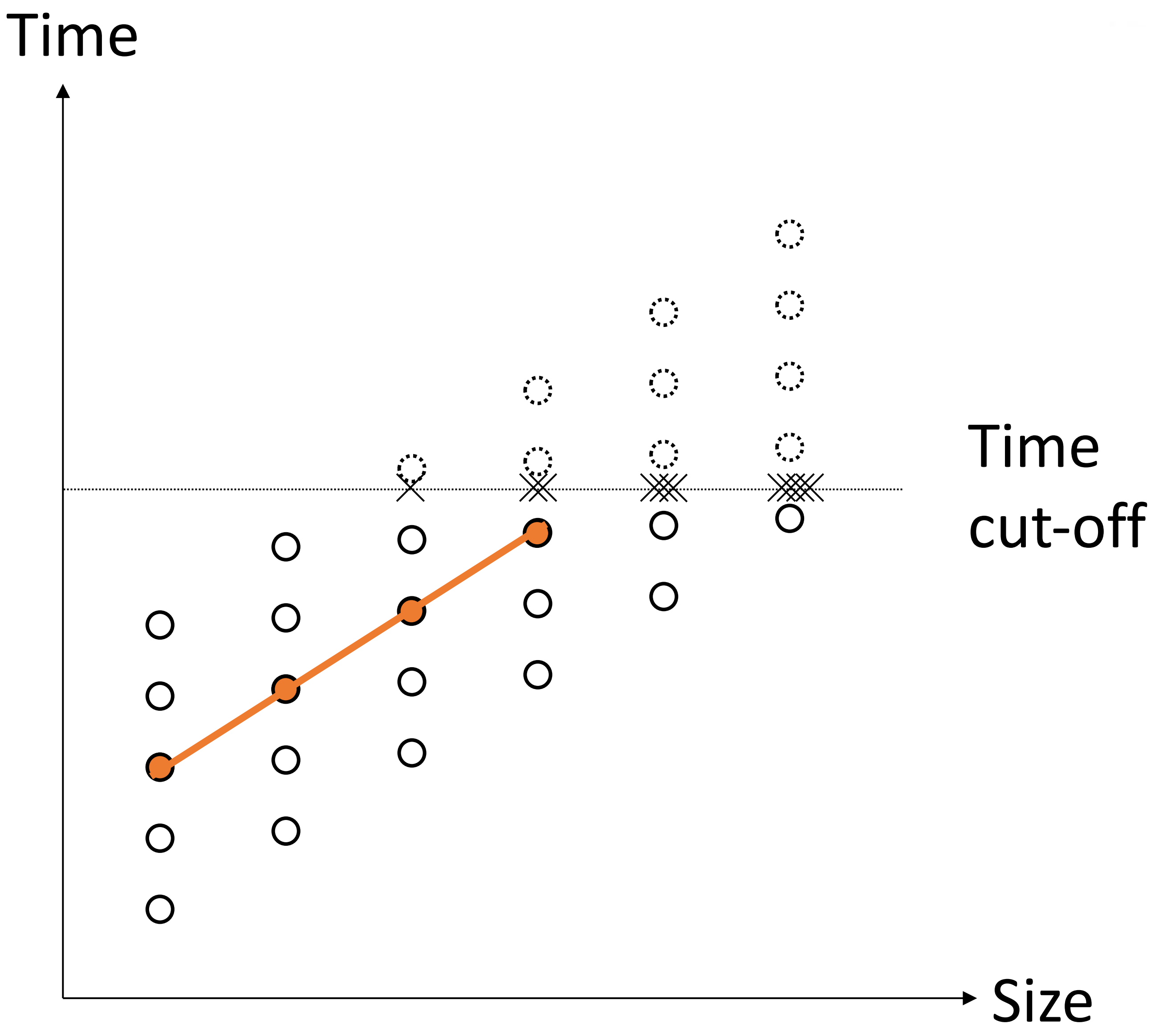
How to deal with cut-offs when binning

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



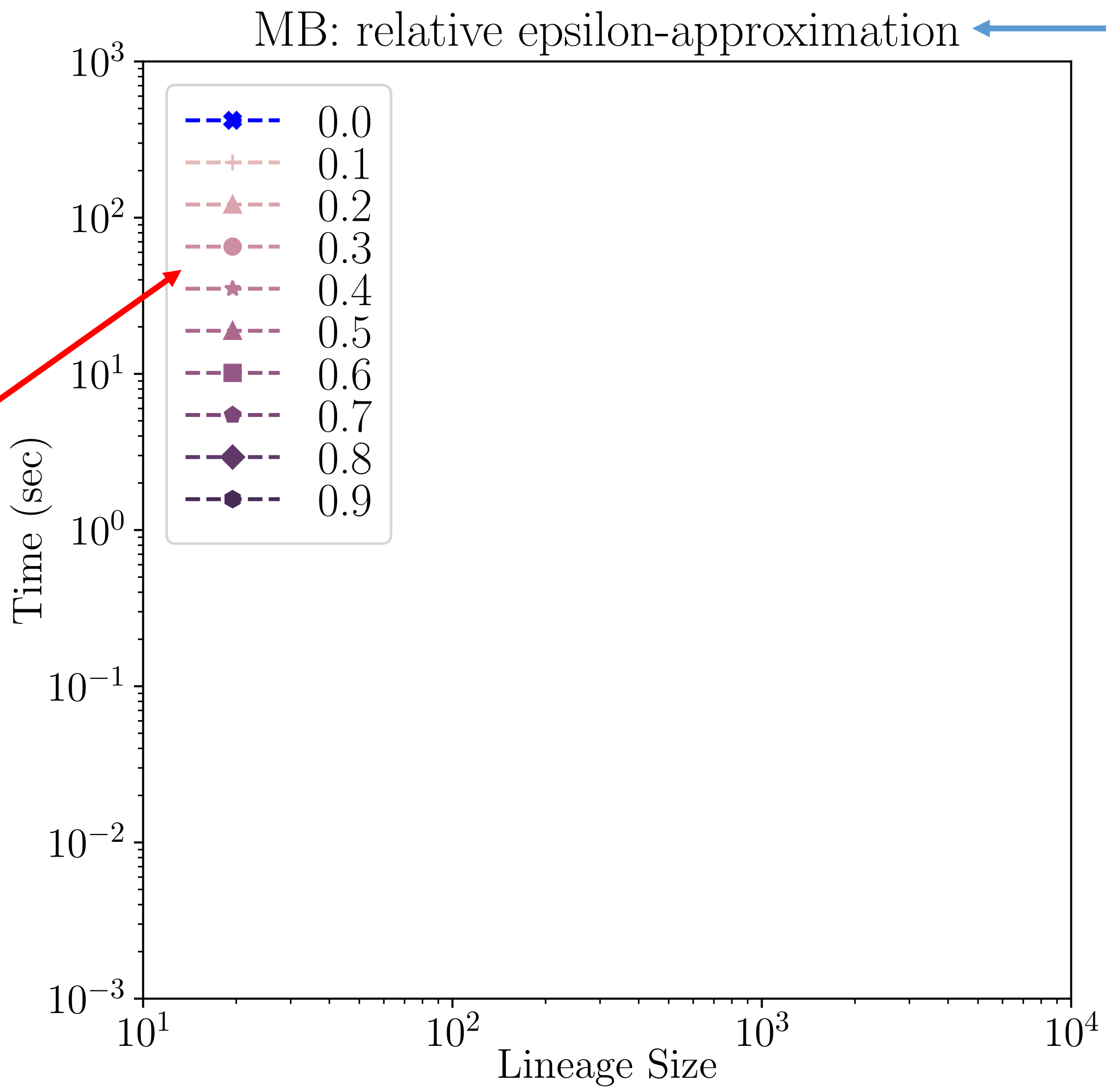
How to deal with cut-offs when binning

- 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 2019]

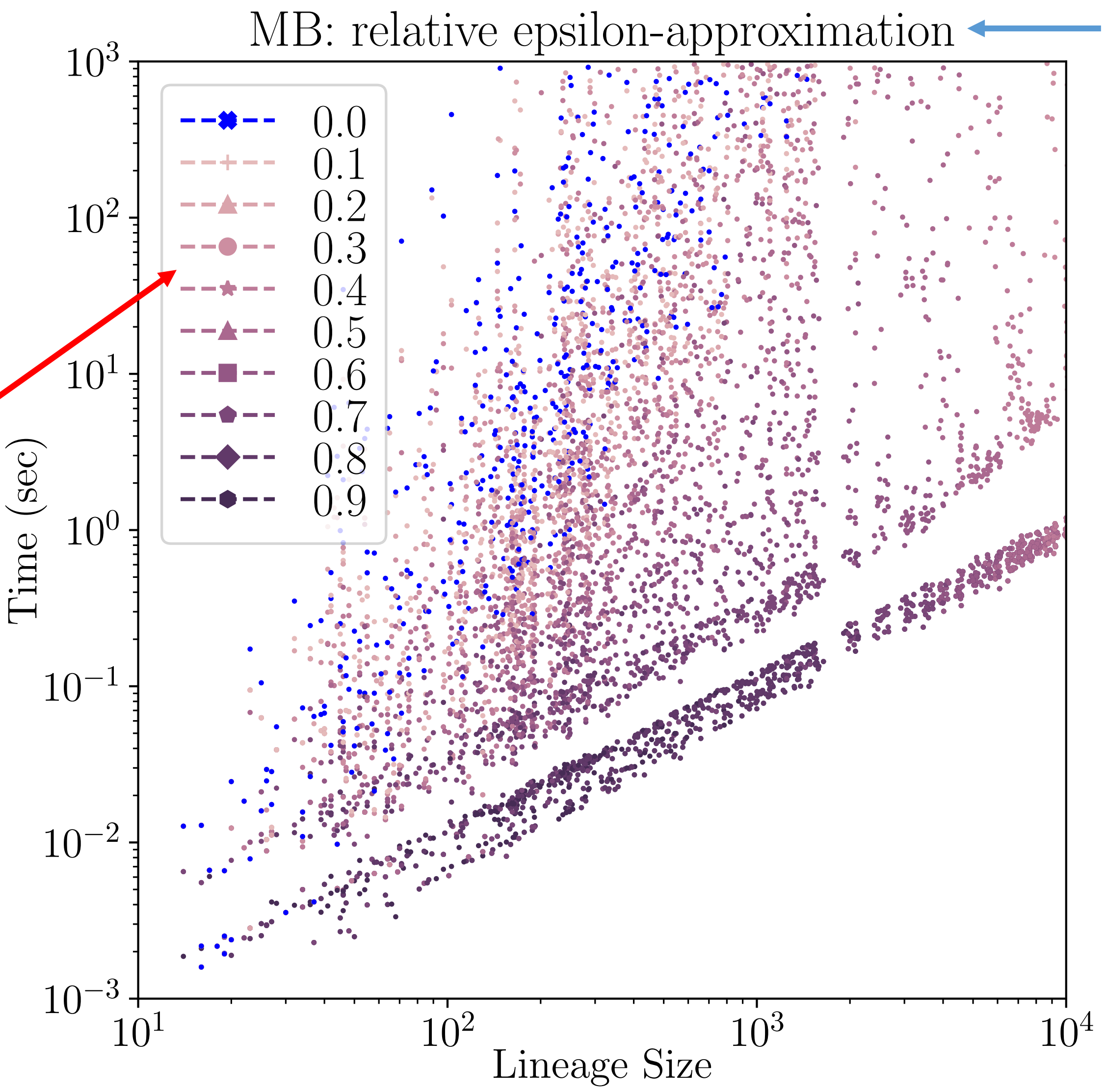
MB (prior): model-based
10 random bounds



Some type of error guarantees (smaller is better)

notice the log scale!

Example: Experiments figures from [Van der Heuvel+ SIGMOD 2019]

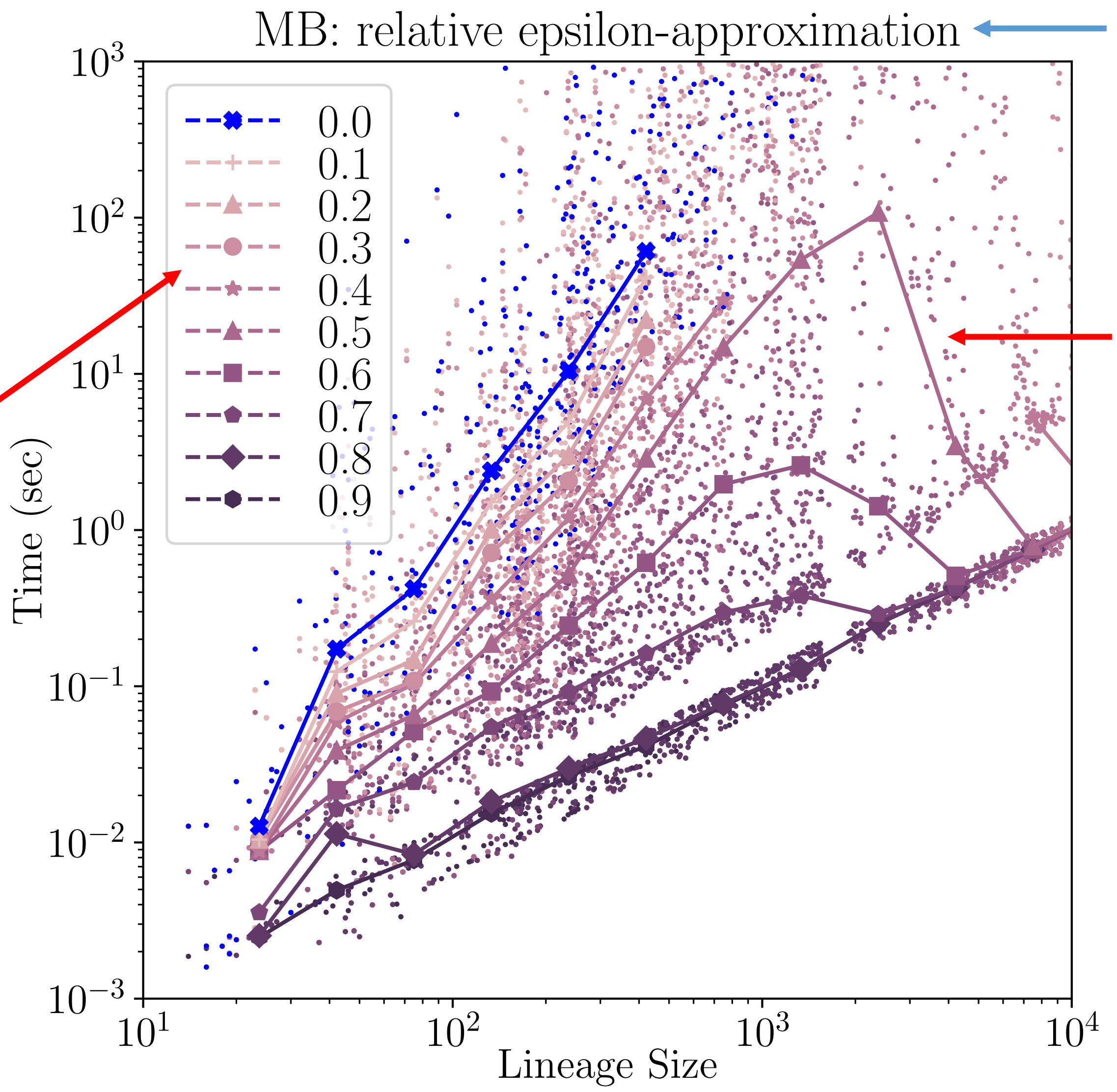


MB (prior): model-based
10 random bounds

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

notice the log scale!

Example: Experiments figures from [Van der Heuvel+ SIGMOD 2019]



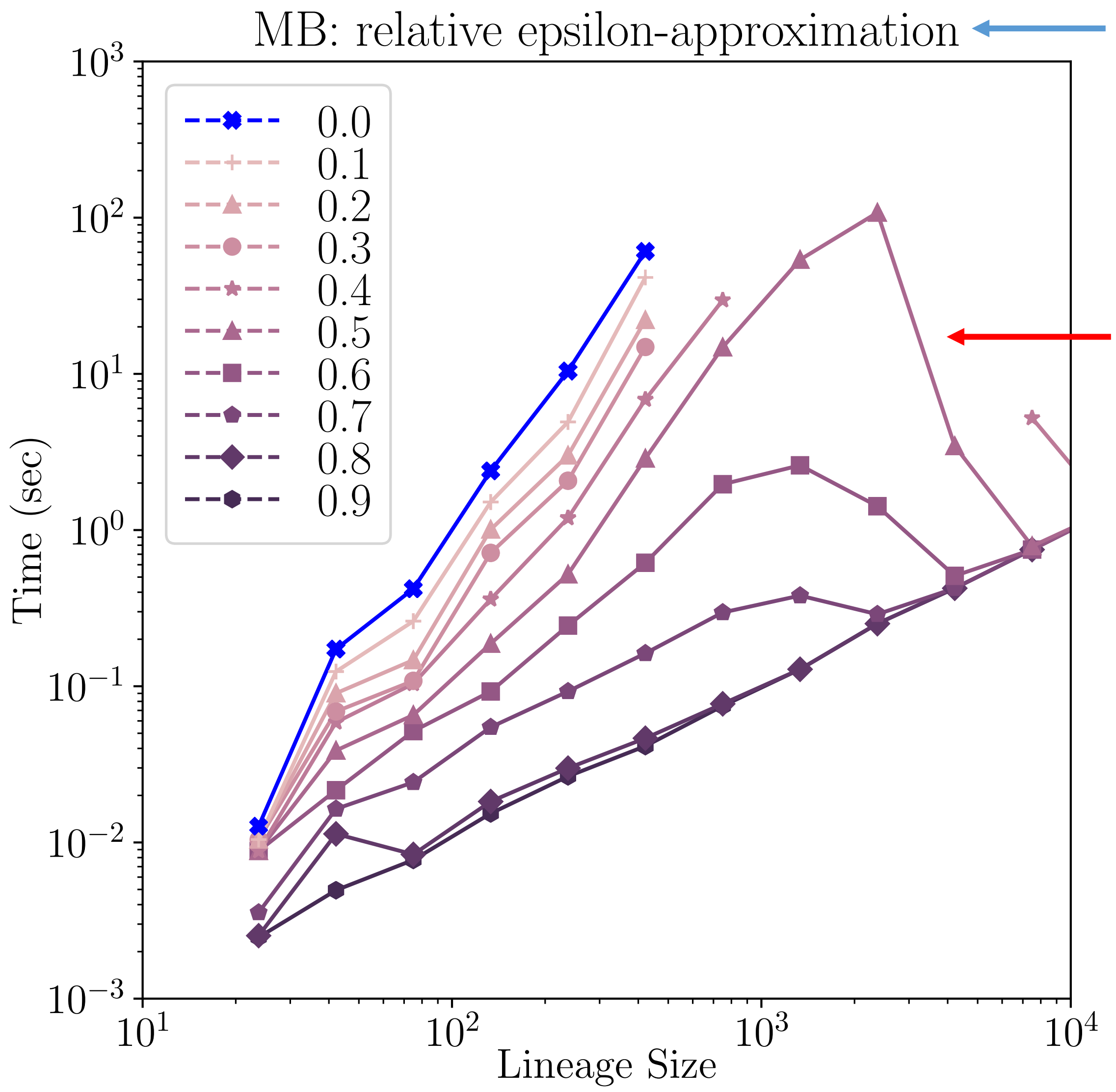
MB (prior): model-based
10 random bounds

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

Some type of error guarantees (smaller is better)

notice the log scale!

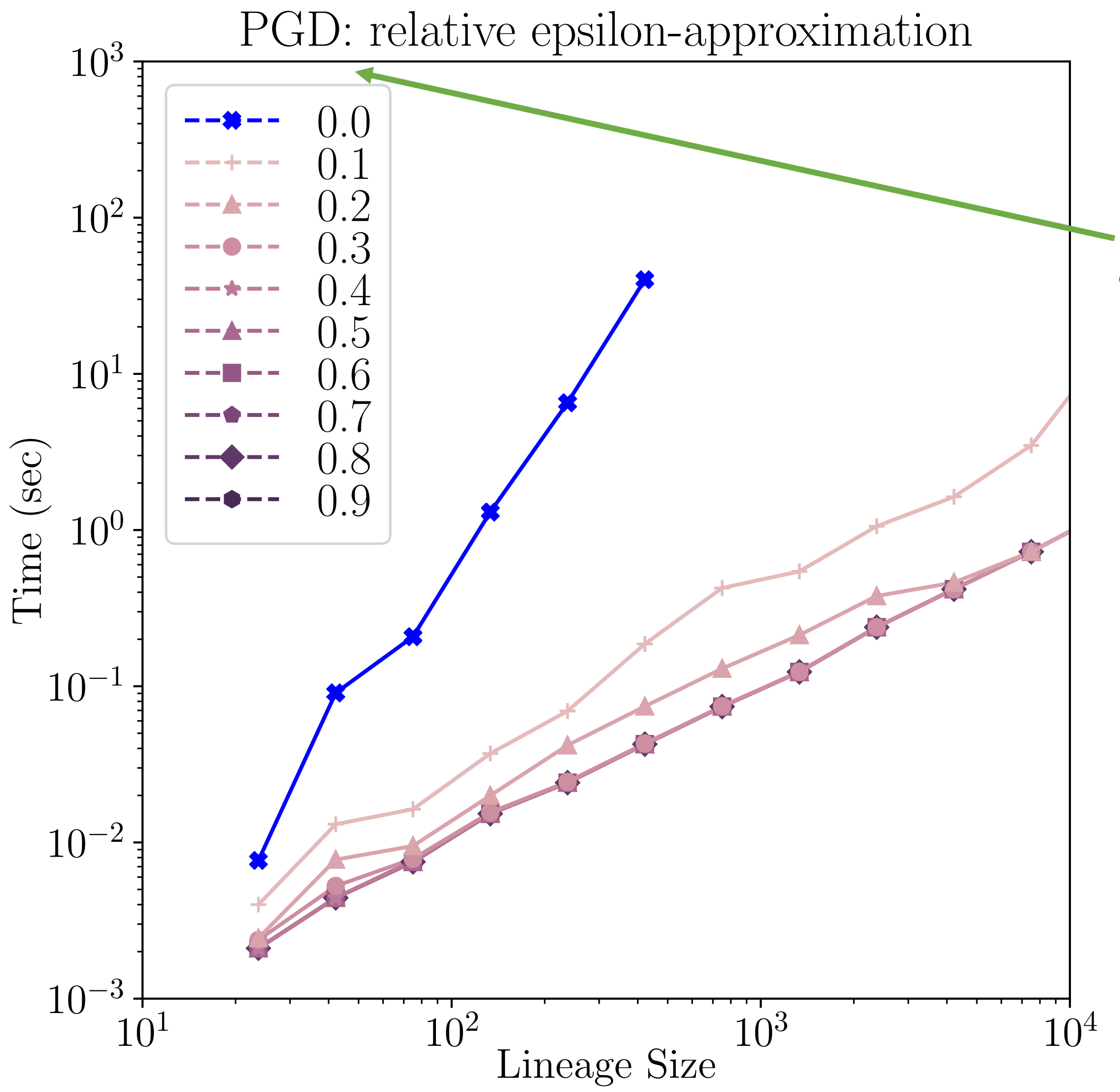
Example: Experiments figures from [Van der Heuvel+ SIGMOD 2019]



MB (prior): model-based
10 random bounds

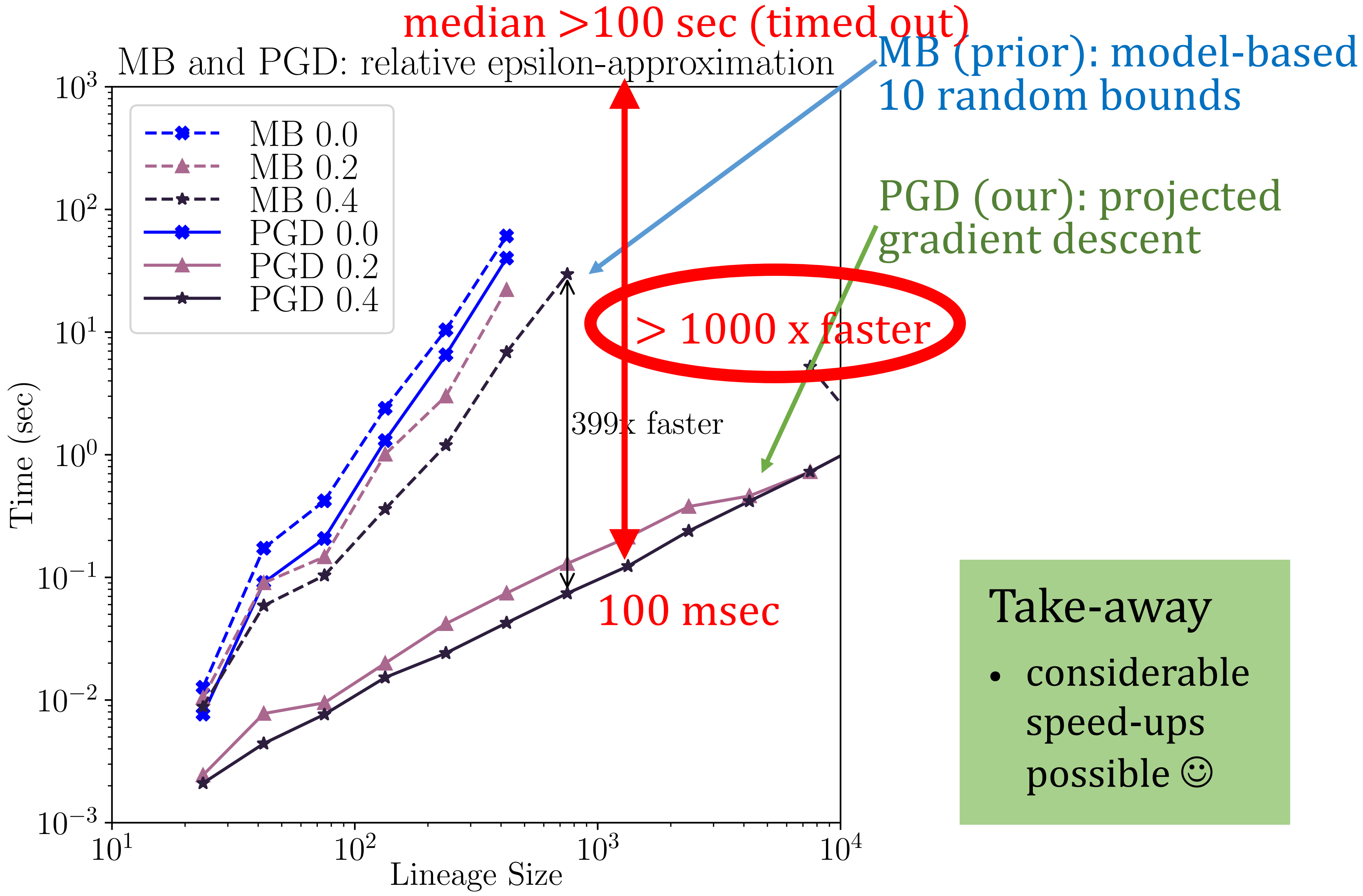
Median time to reach a certain error guarantee for fixed lin. size

Example: Experiments figures from [Van der Heuvel+ SIGMOD 2019]



PGD (our): projected gradient descent

Example: Experiments figures from [Van der Heuvel+ SIGMOD 2019]



Take-away

- considerable speed-ups possible ☺