Overview of Data Exploration Techniques

Stratos Idreos, Olga Papaemmanouil, Surajit Chaudhuri

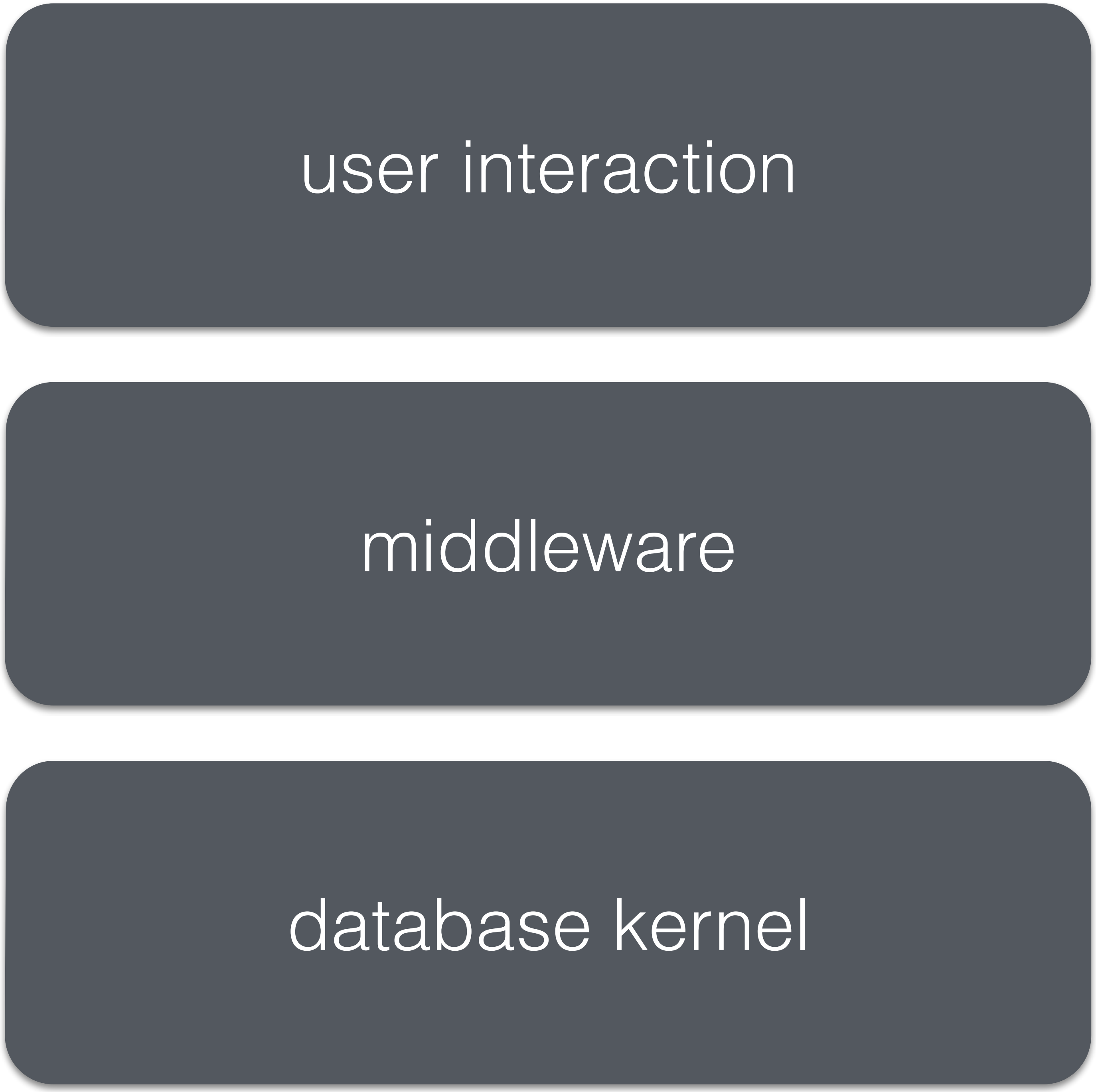
data exploration

not always sure
what we are looking for
(until we find it)

data has always been **big**

volume velocity variety veracity

content structure



user interaction

middleware

database kernel

user interaction

visualization

interfaces

prefetching

approximation

sampling

...

45 min

middleware

45 min

kernel

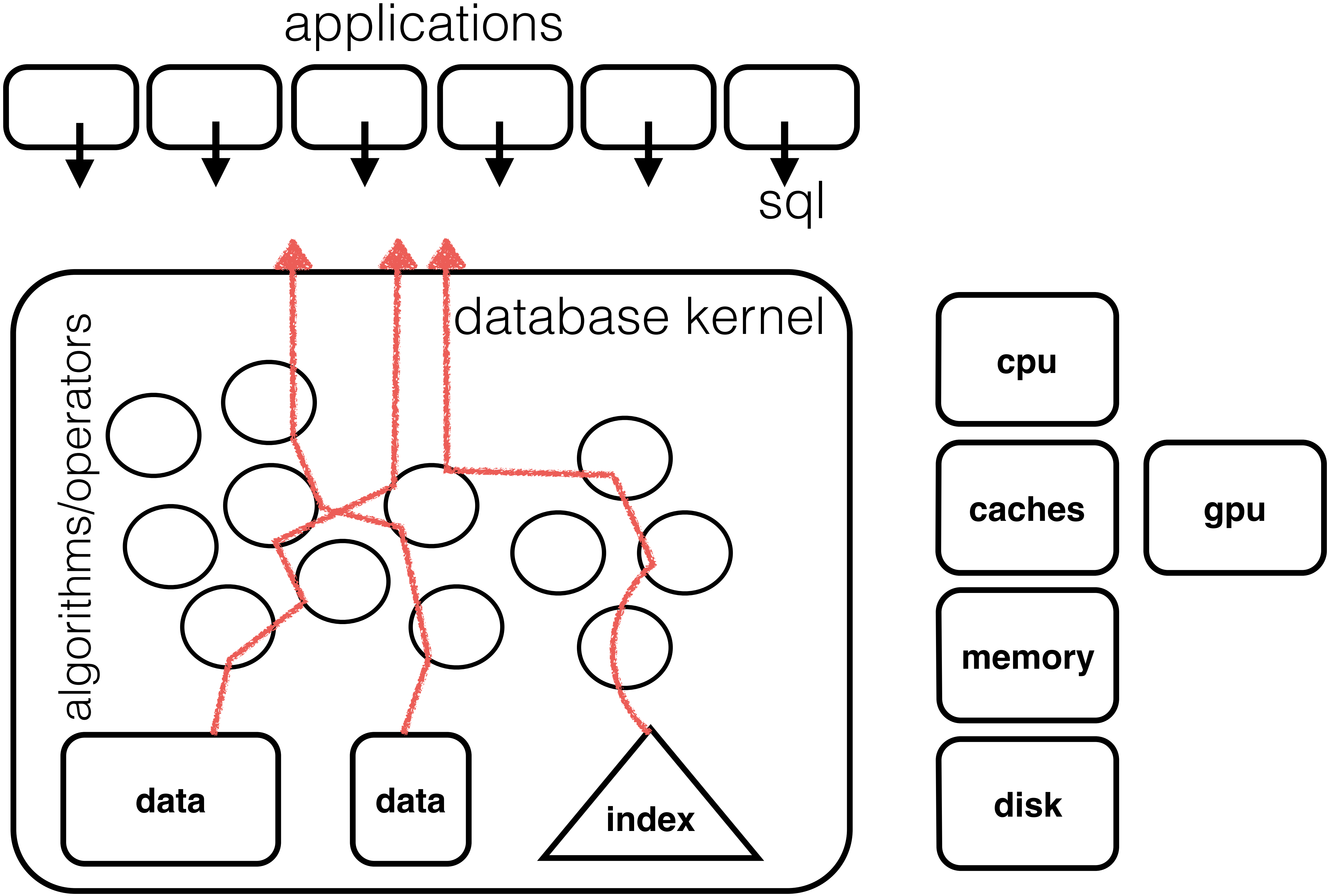
adaptive[
indexing,
loading,
storage]

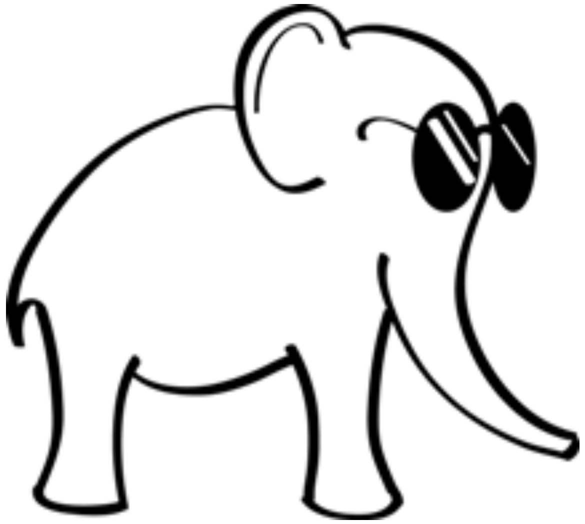
...

45 min

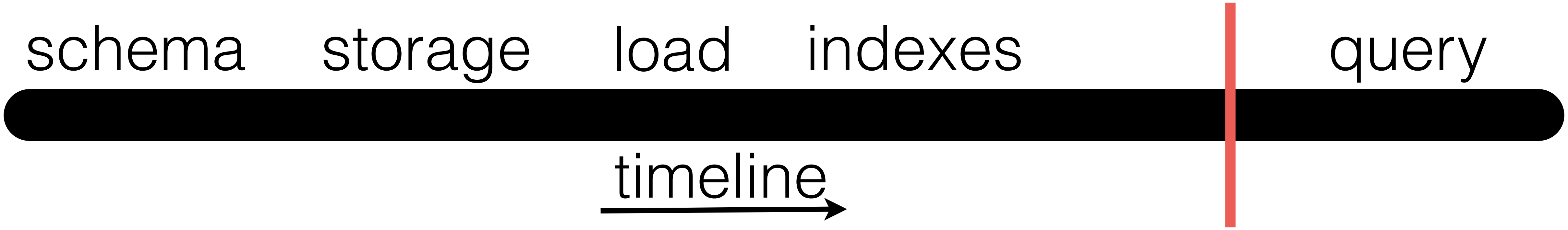
Part 3*

*for Part1 and 2 please look at the websites of the tutorial co-authors


applications
sql
database kernel
algorithms/operators
data
data
index
cpu
caches
gpu
memory
disk



too many preparation options
lead to complex installation

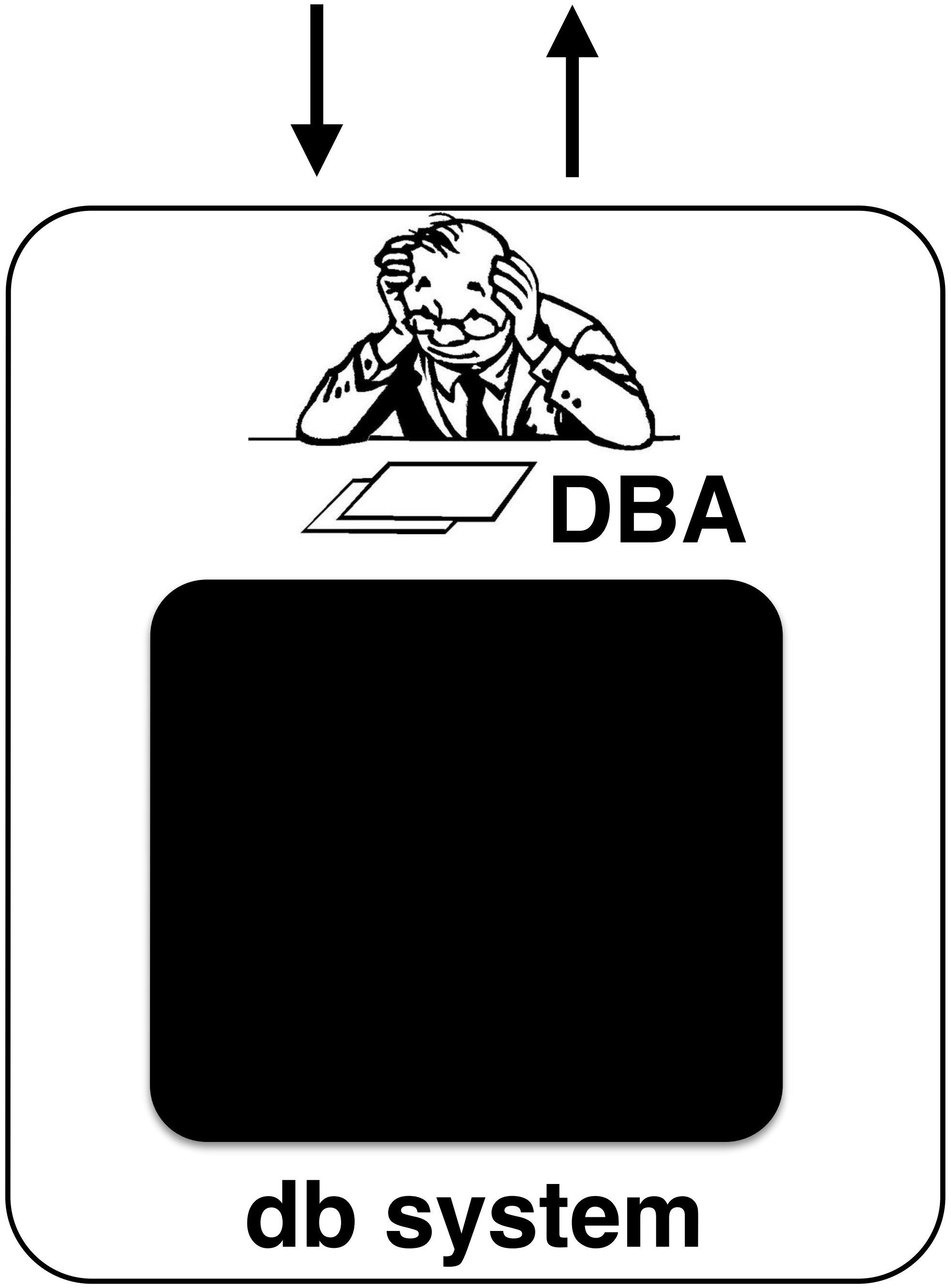


expert users - idle time - workload knowledge

users/applications

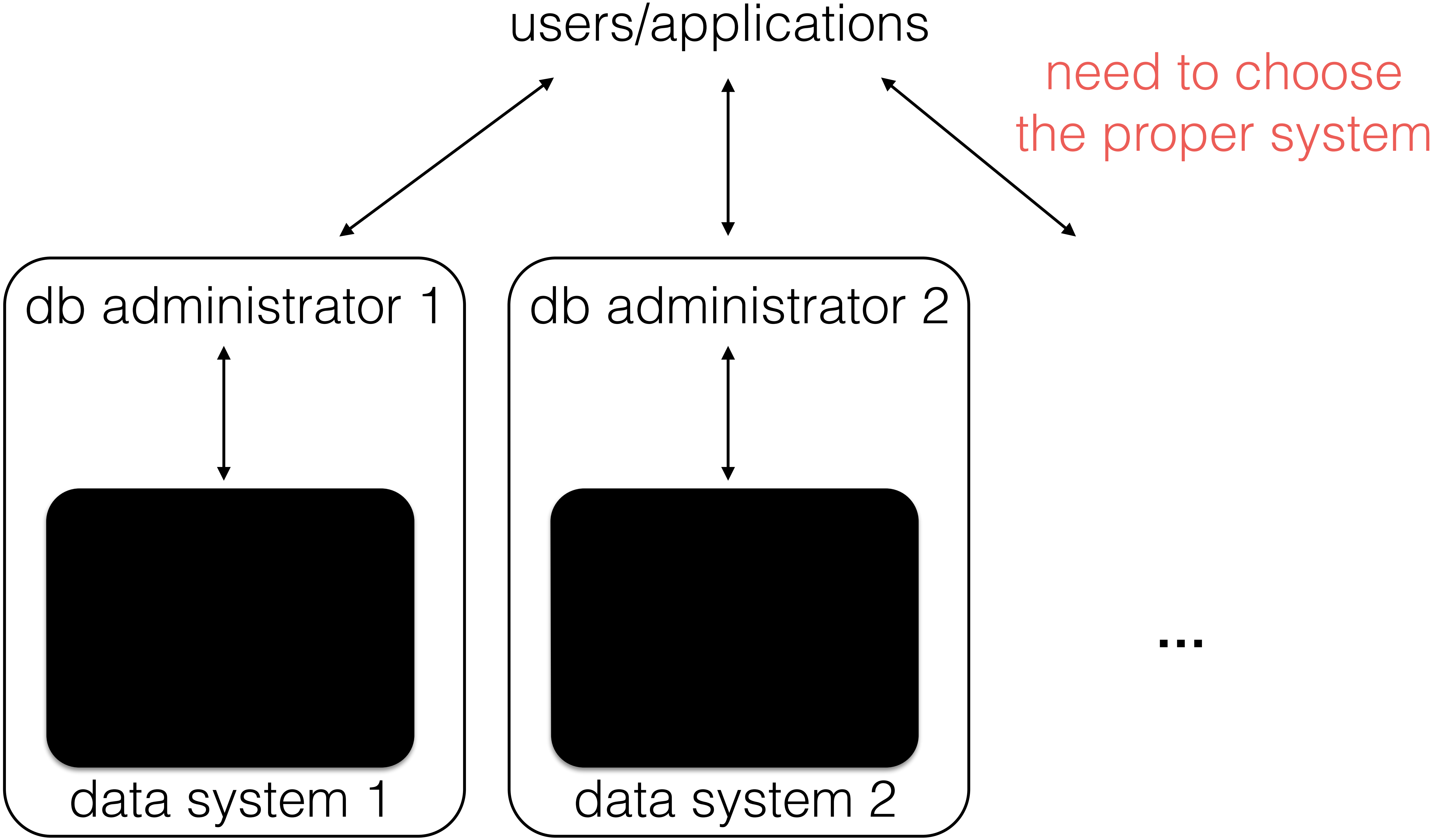
declarative interface

ask what you want



DBA

db system



users/applications

need to choose the proper system

db administrator 1

data system 1

db administrator 2

data system 2

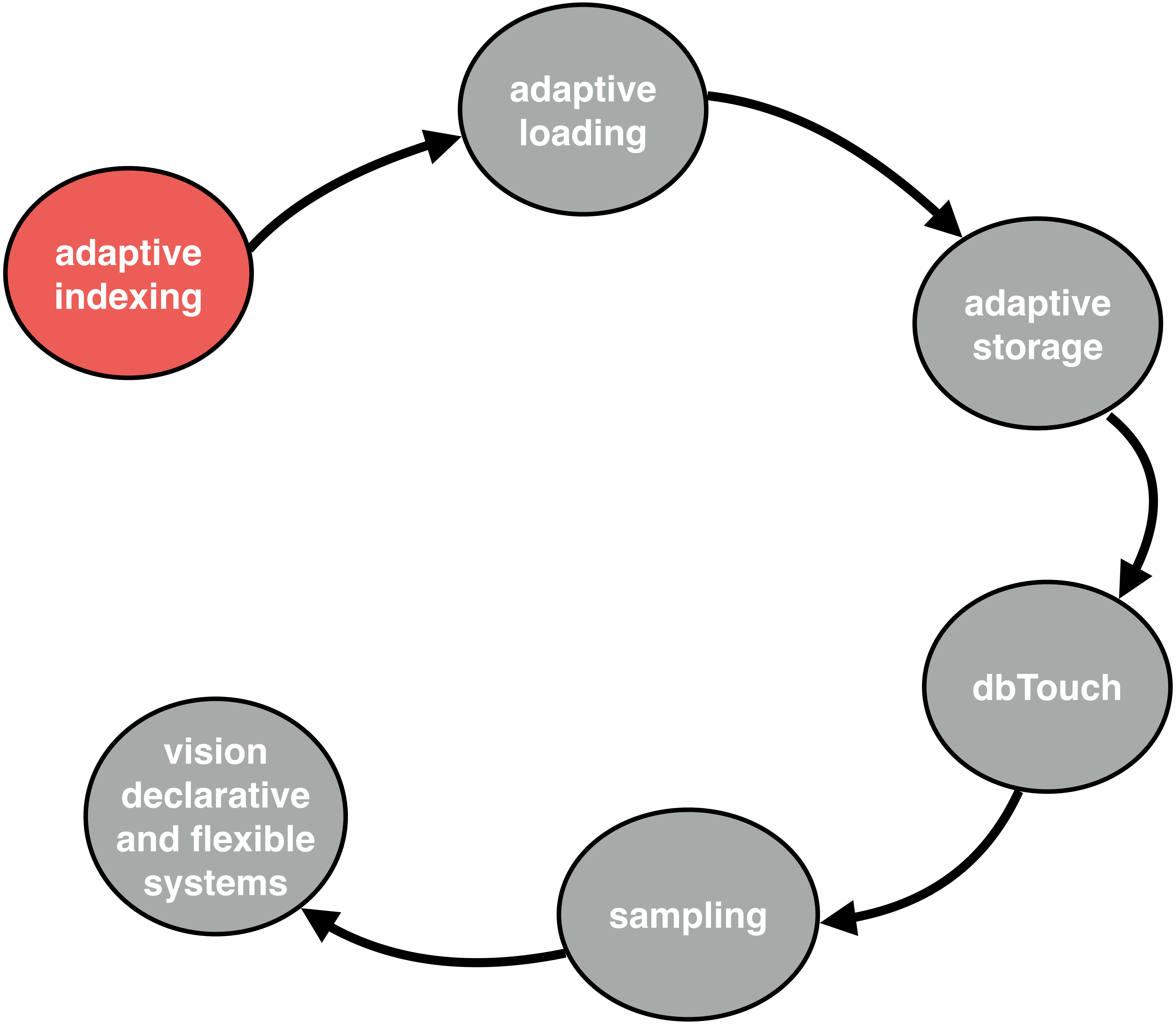
...

how can we prepare if we
do not know what we are up against?
(loading, indexing, storage, …)

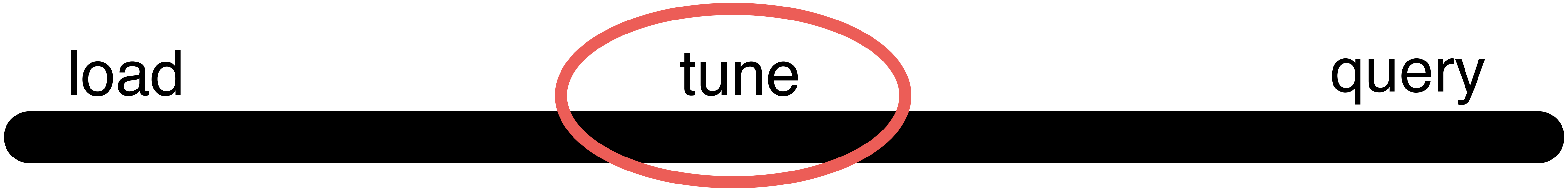
how can we prepare if we
do not know what we are up against?
(loading, indexing, storage, …)

data systems kernels tailored
for data exploration

no preparation - easy to use - fast


adaptive indexing
adaptive loading
adaptive storage
dbTouch
sampling
vision declarative and flexible systems

indexing



load

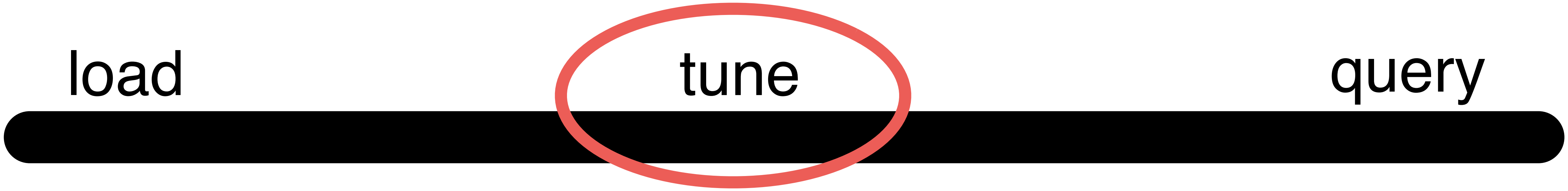
tune

query

tune= create proper indices offline

performance 10-100X

indexing



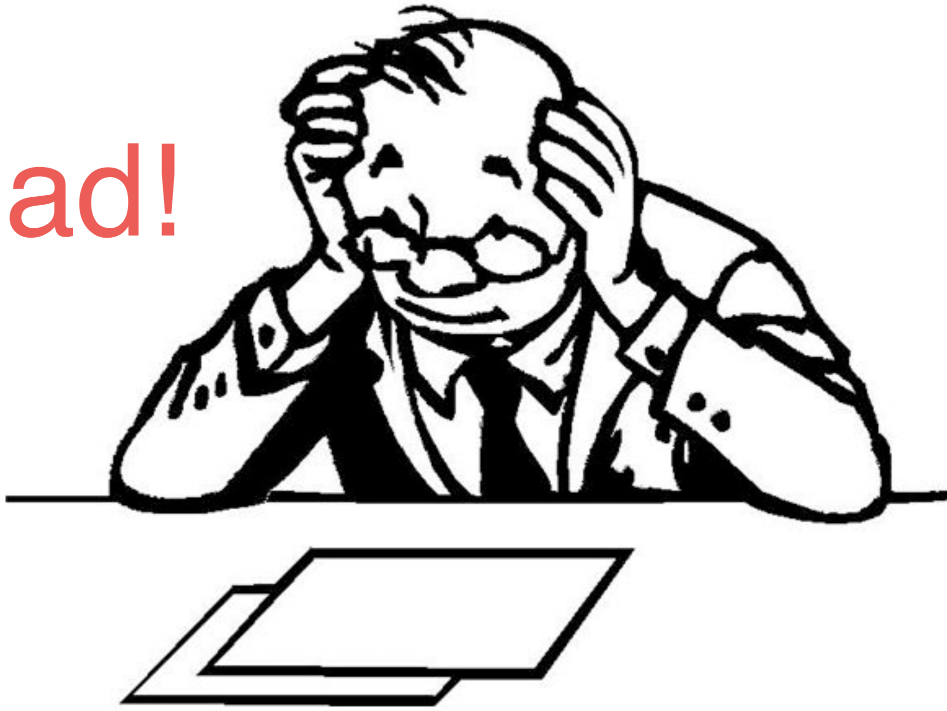
load tune query

tune= create proper indices offline

performance 10-100X

but it depends on the workload!

which indices to build?
on which data parts?
and when to build them?



big data V's volume velocity variety veracity

what can go wrong?

not enough space to index all data

not enough idle **time** to finish proper tuning

by the time we finish tuning, the **workload changes**

not enough money - energy - resources big data V's volume velocity variety veracity

what can go wrong?

not enough space to index all data

not enough idle **time** to finish proper tuning

by the time we finish tuning, the **workload changes**

not enough money - energy - resources

database cracking

database cracking

~~idle time~~

~~workload knowledge~~

~~external tools~~

~~human control~~

database cracking

auto-tuning database kernels

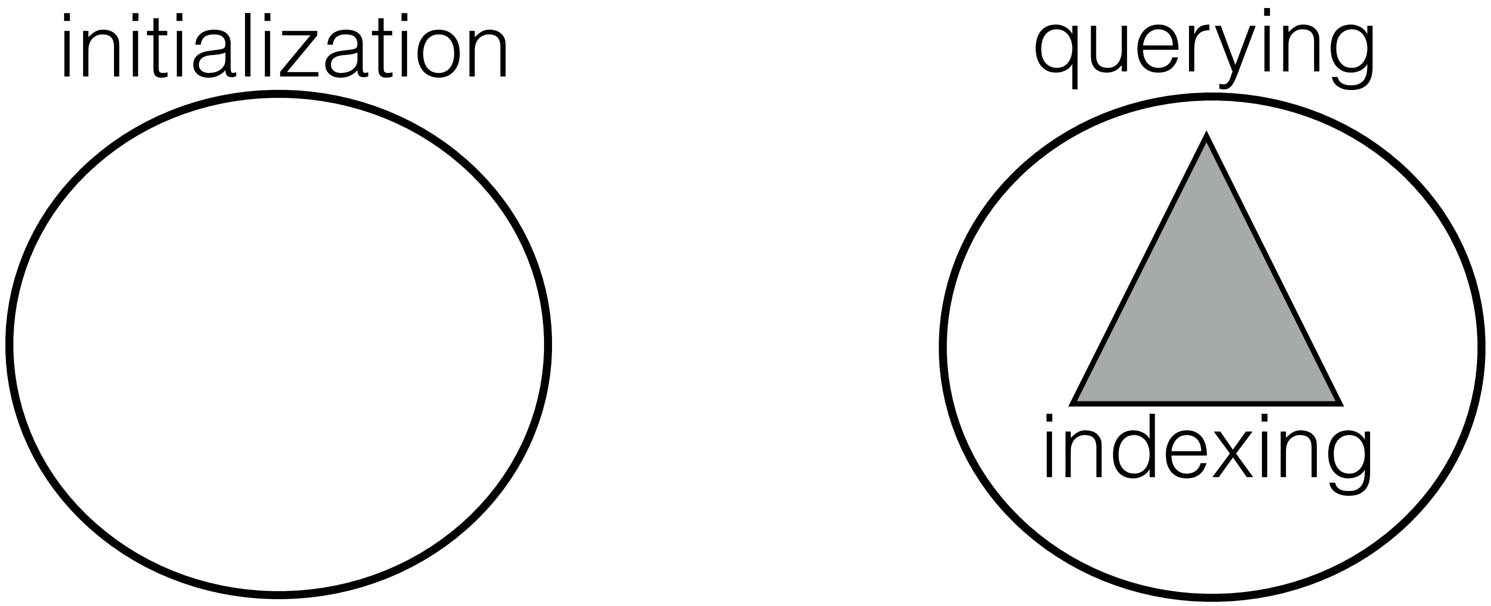
incremental, adaptive, partial indexing

~~idle time~~

~~workload knowledge~~

~~external tools~~

~~human control~~



initialization

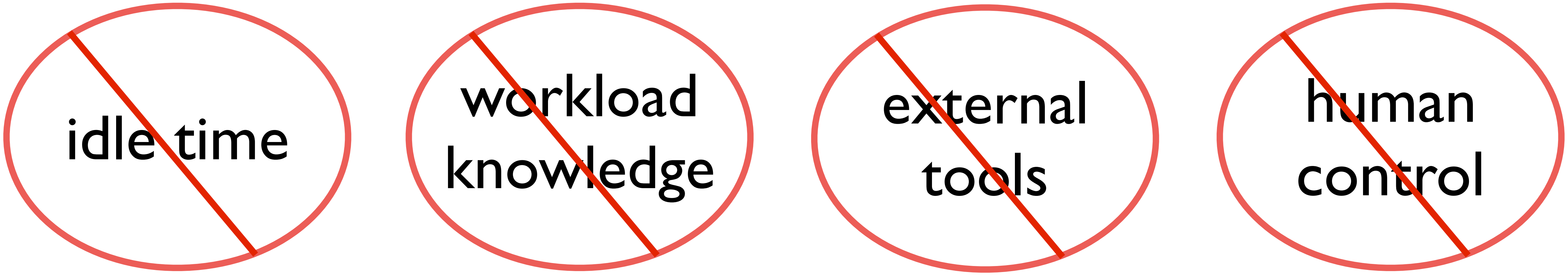
querying

indexing

database cracking

auto-tuning database kernels

incremental, adaptive, partial indexing

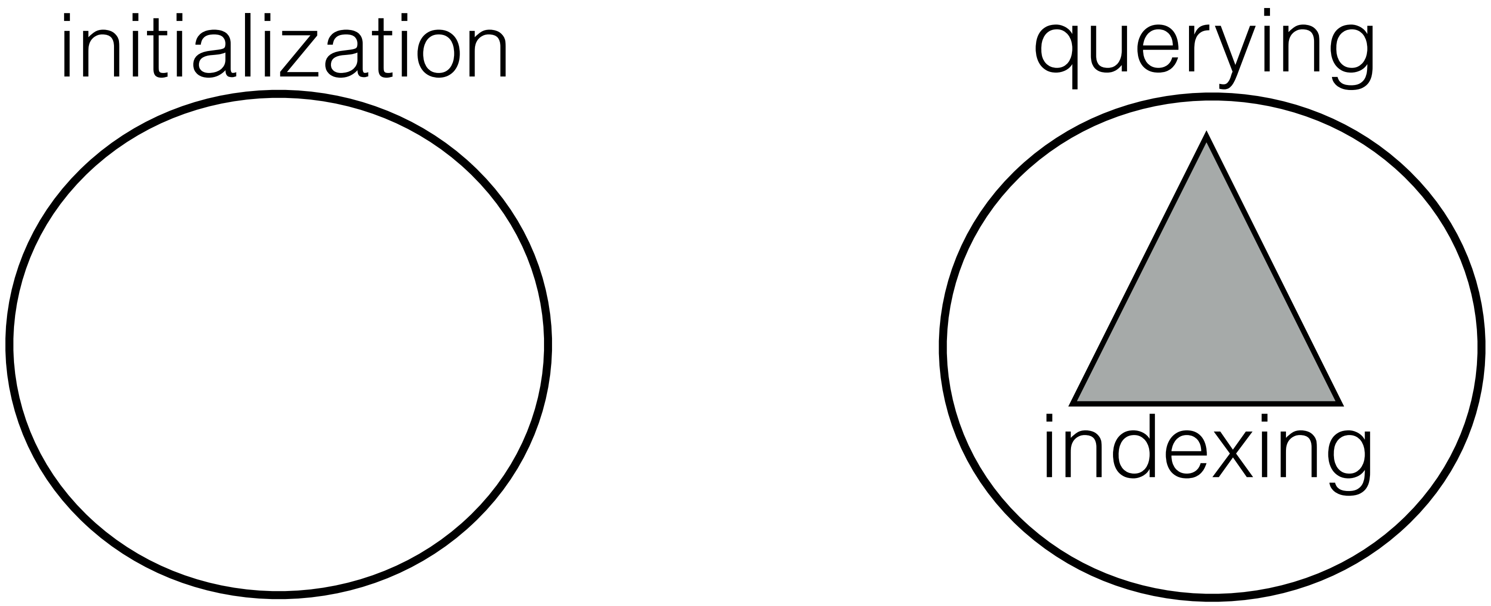


~~idle time~~

~~workload knowledge~~

~~external tools~~

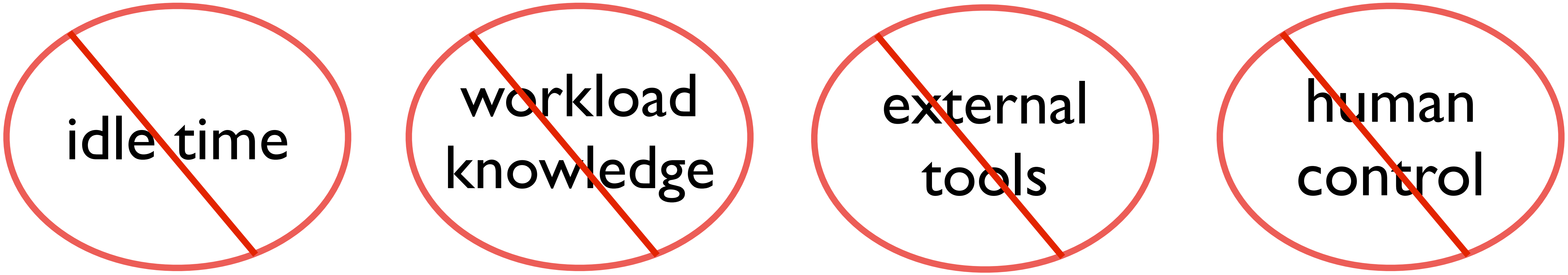
~~human control~~

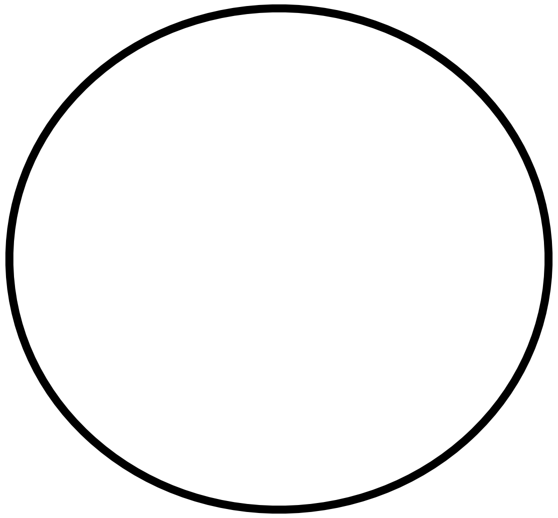


database cracking

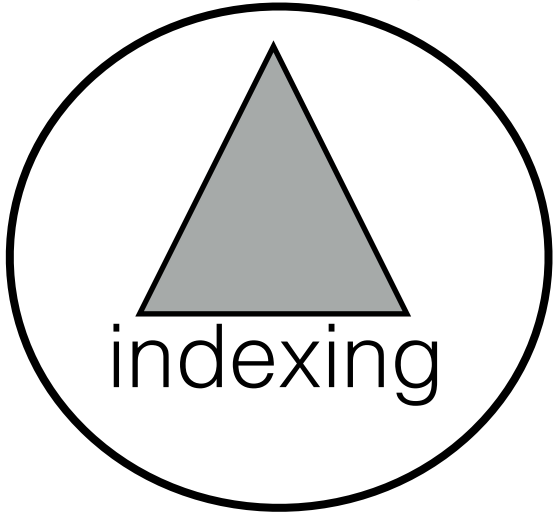
auto-tuning database kernels

incremental, adaptive, partial indexing





initialization



querying

indexing

database cracking

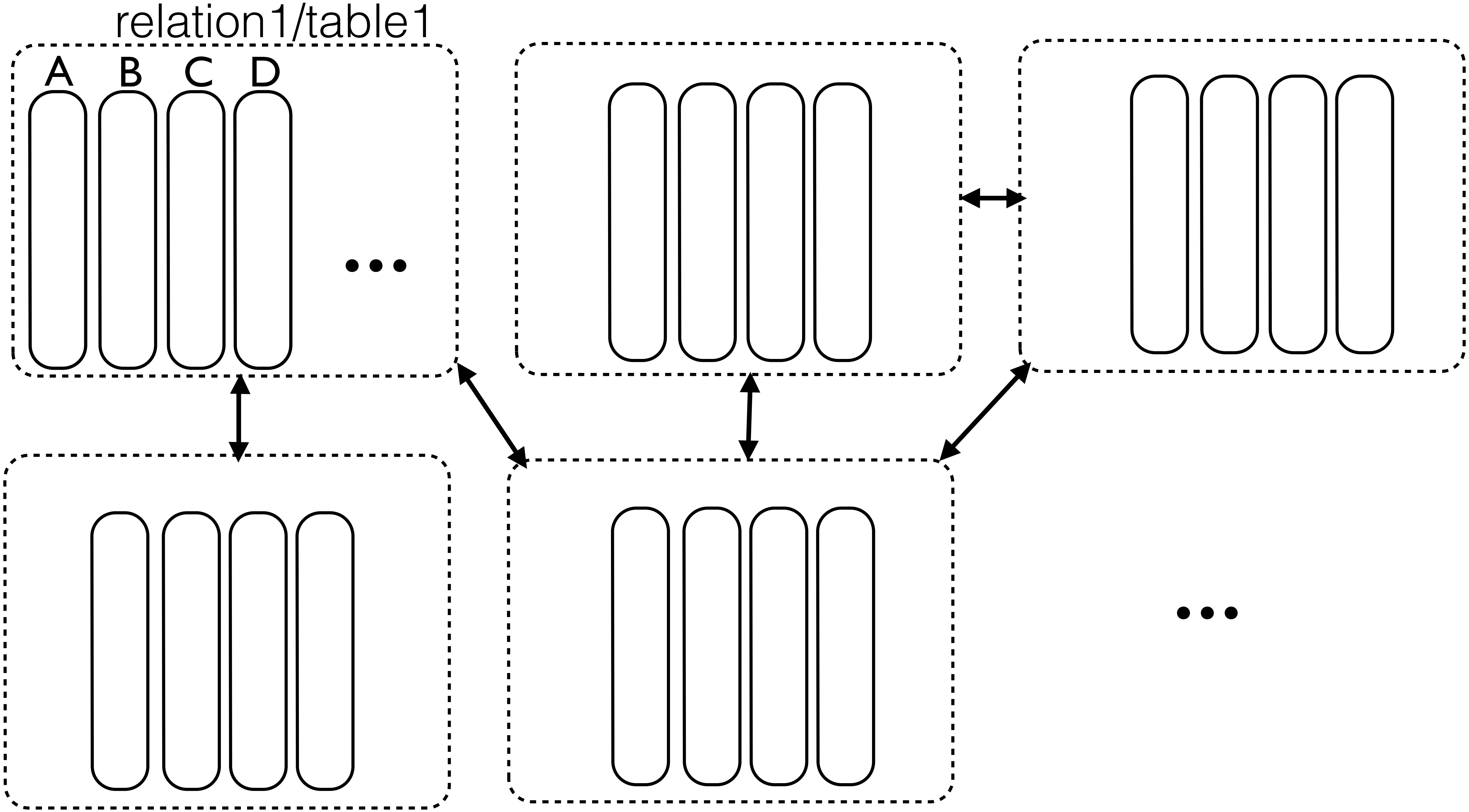
auto-tuning database kernels

incremental, adaptive, partial indexing

every query is treated as an advice on how data should be stored

column-store database

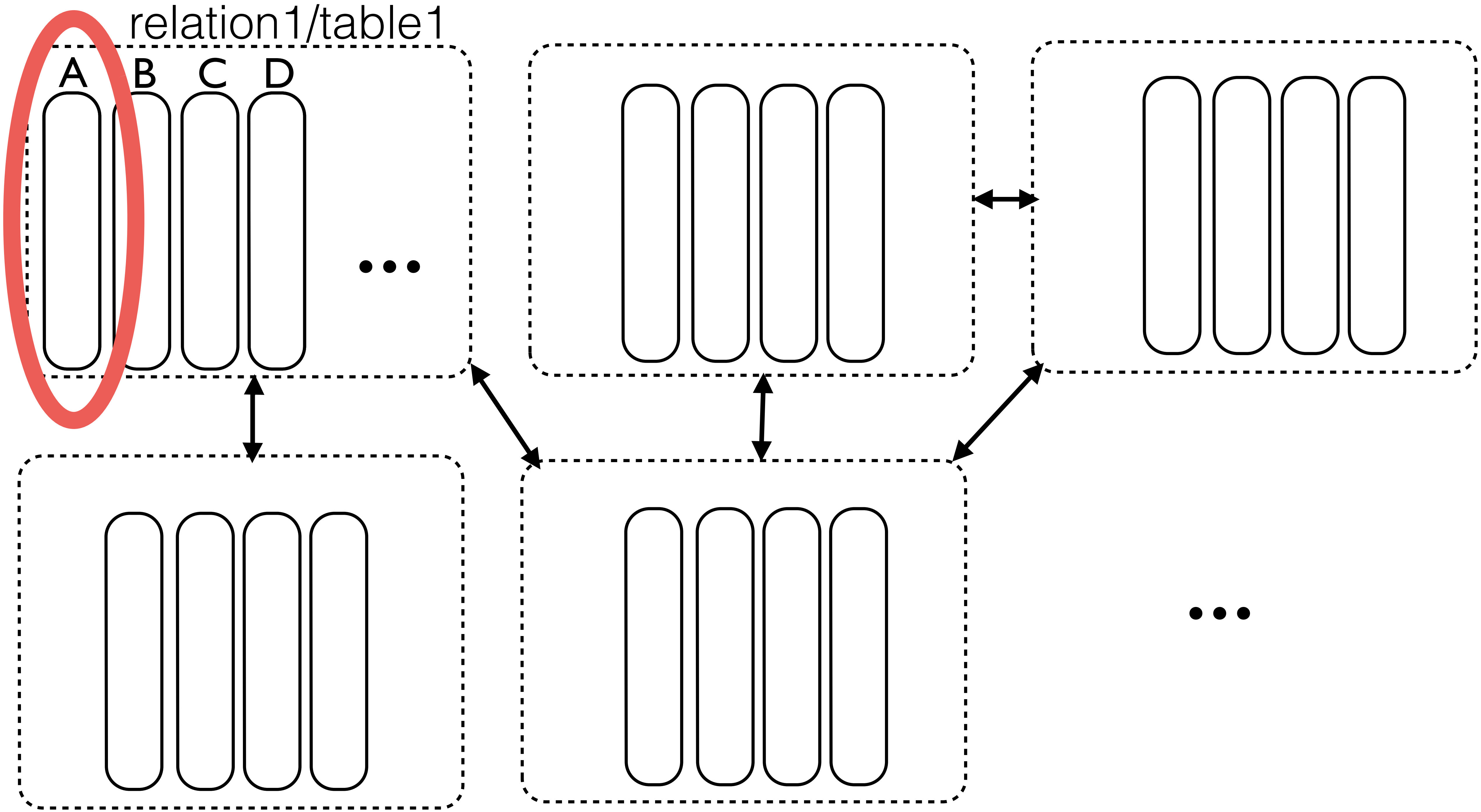
a fixed-width and dense array per attribute



Database Cracking CIDR 2007

column-store database

a fixed-width and dense array per attribute



Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

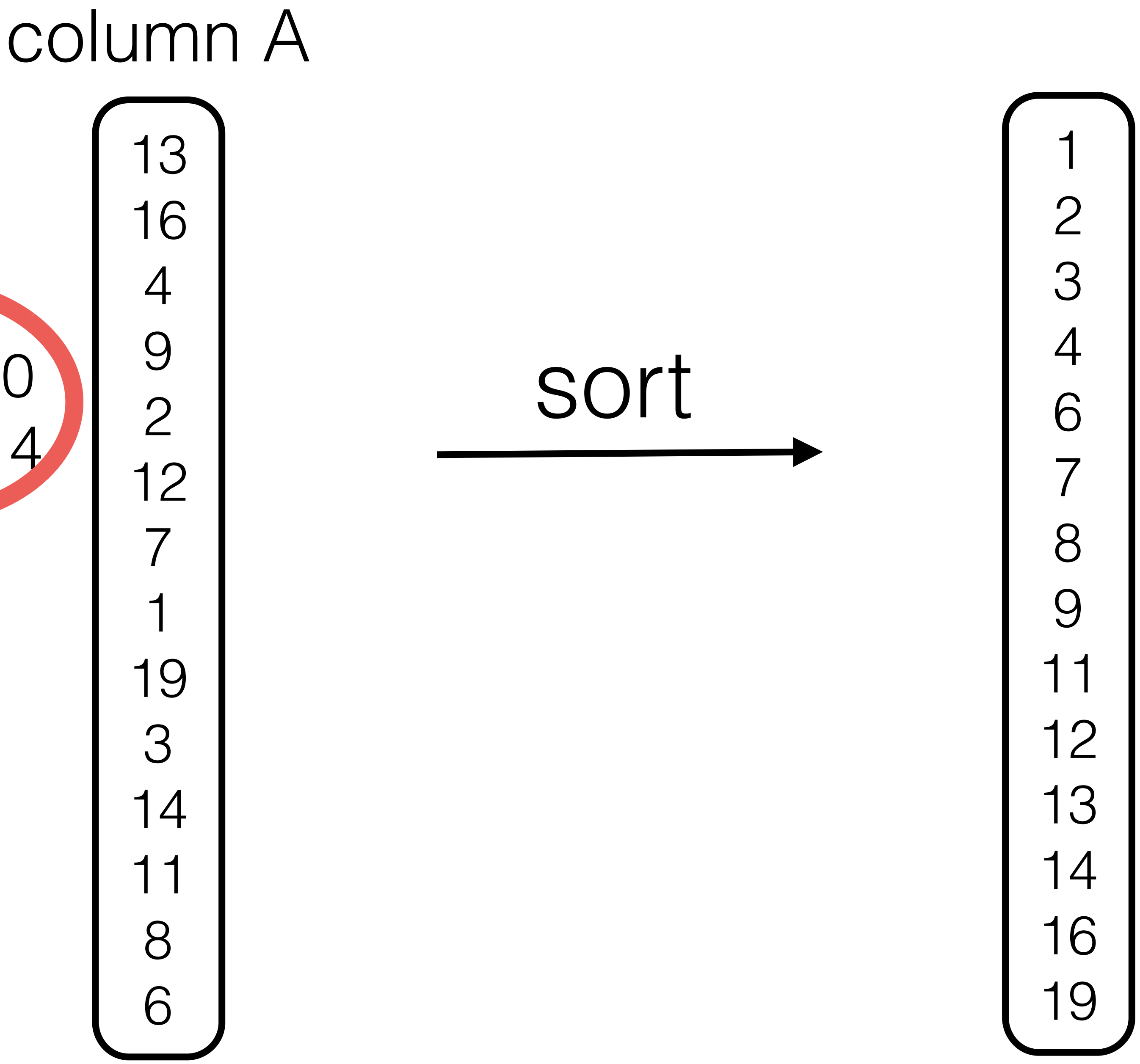
Q1:

```
select R.A
from R
where R.A>10
    and R.A<14
```

column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

```
Q1:
select R.A
from R
where R.A>10
   and R.A<14
```



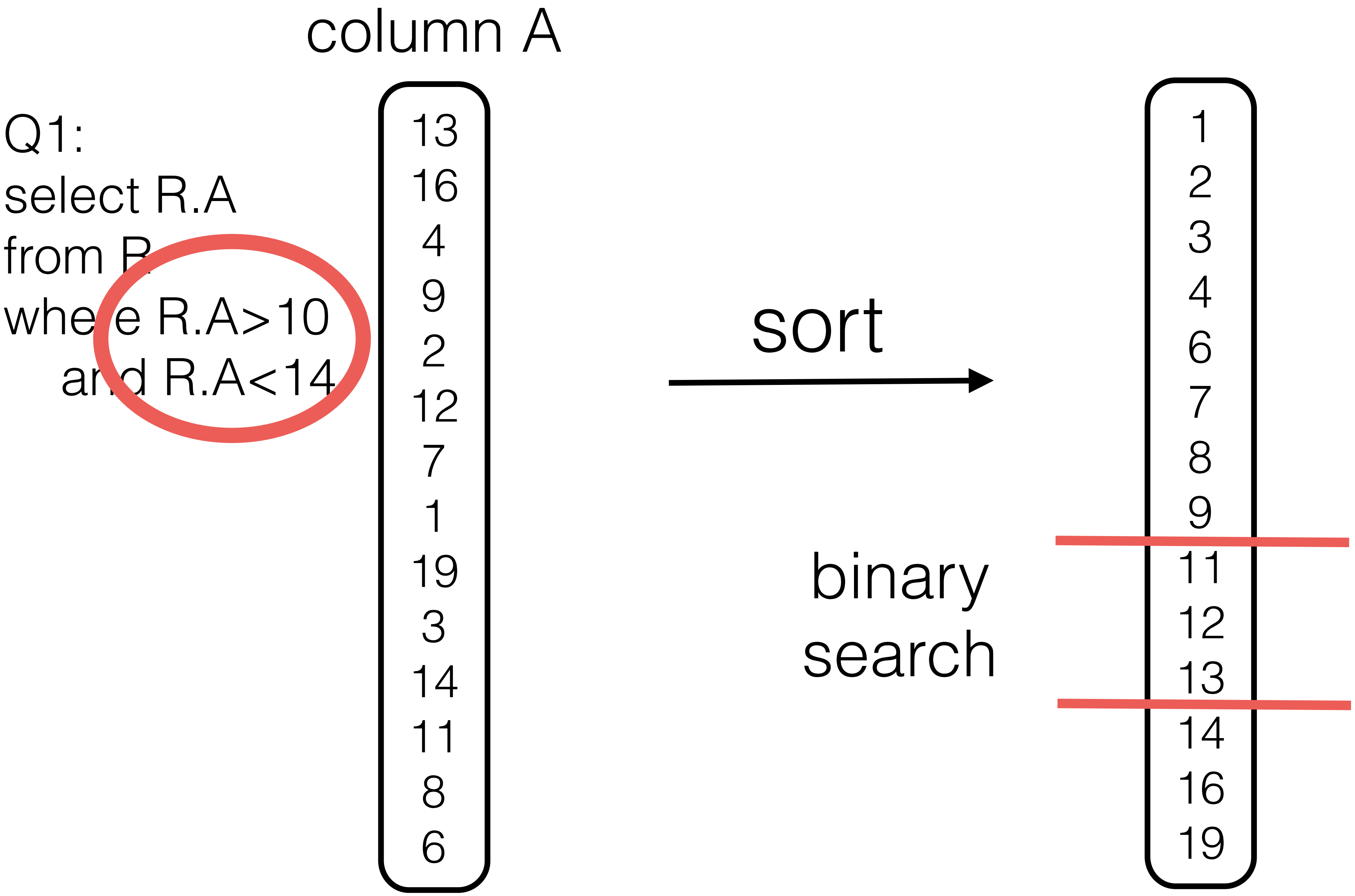
column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

sort

1
2
3
4
6
7
8
9
11
12
13
14
16
19

Database Cracking CIDR 2007



Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

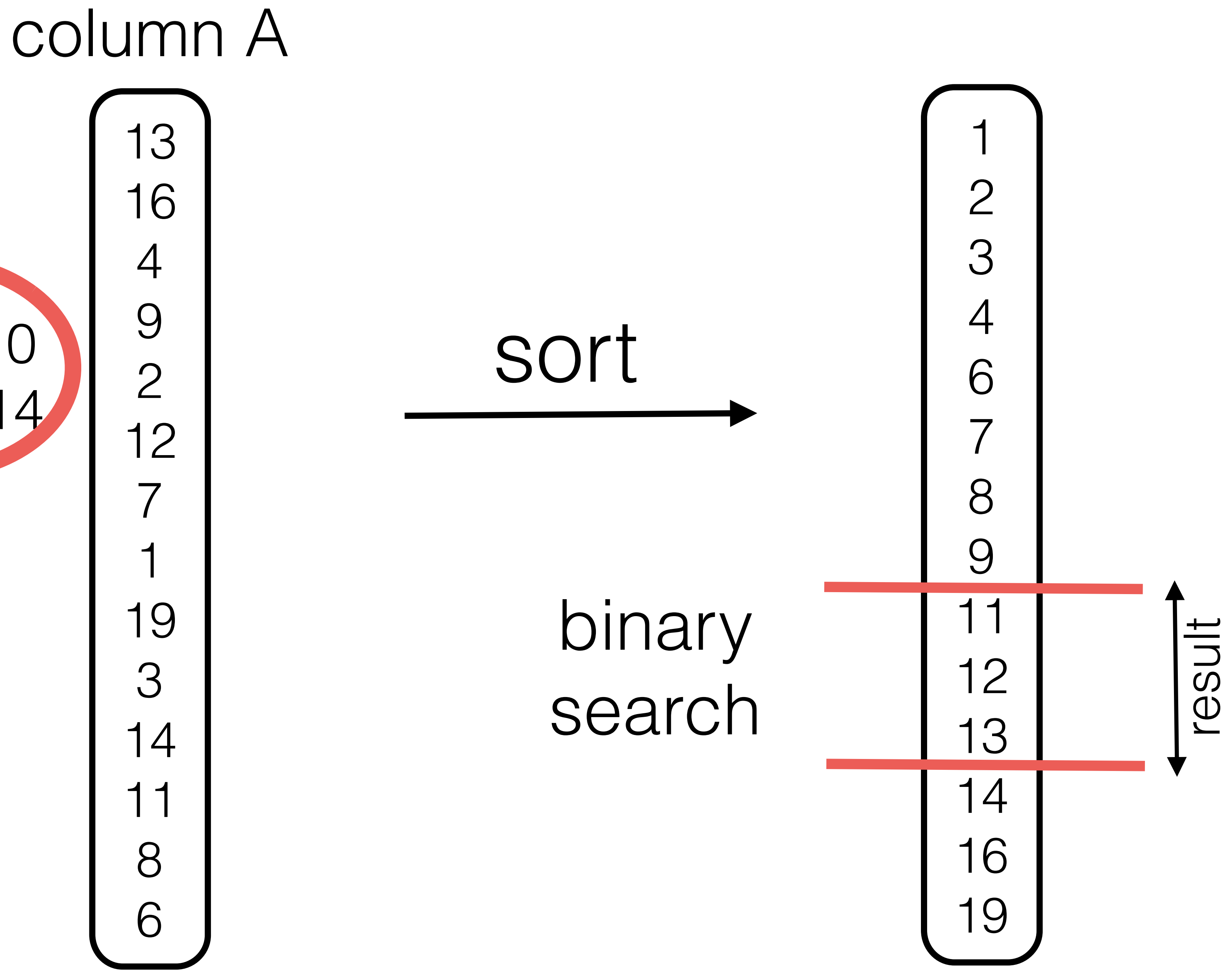
sort

1
2
3
4
6
7
8
9
11
12
13
14
16
19

binary search

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

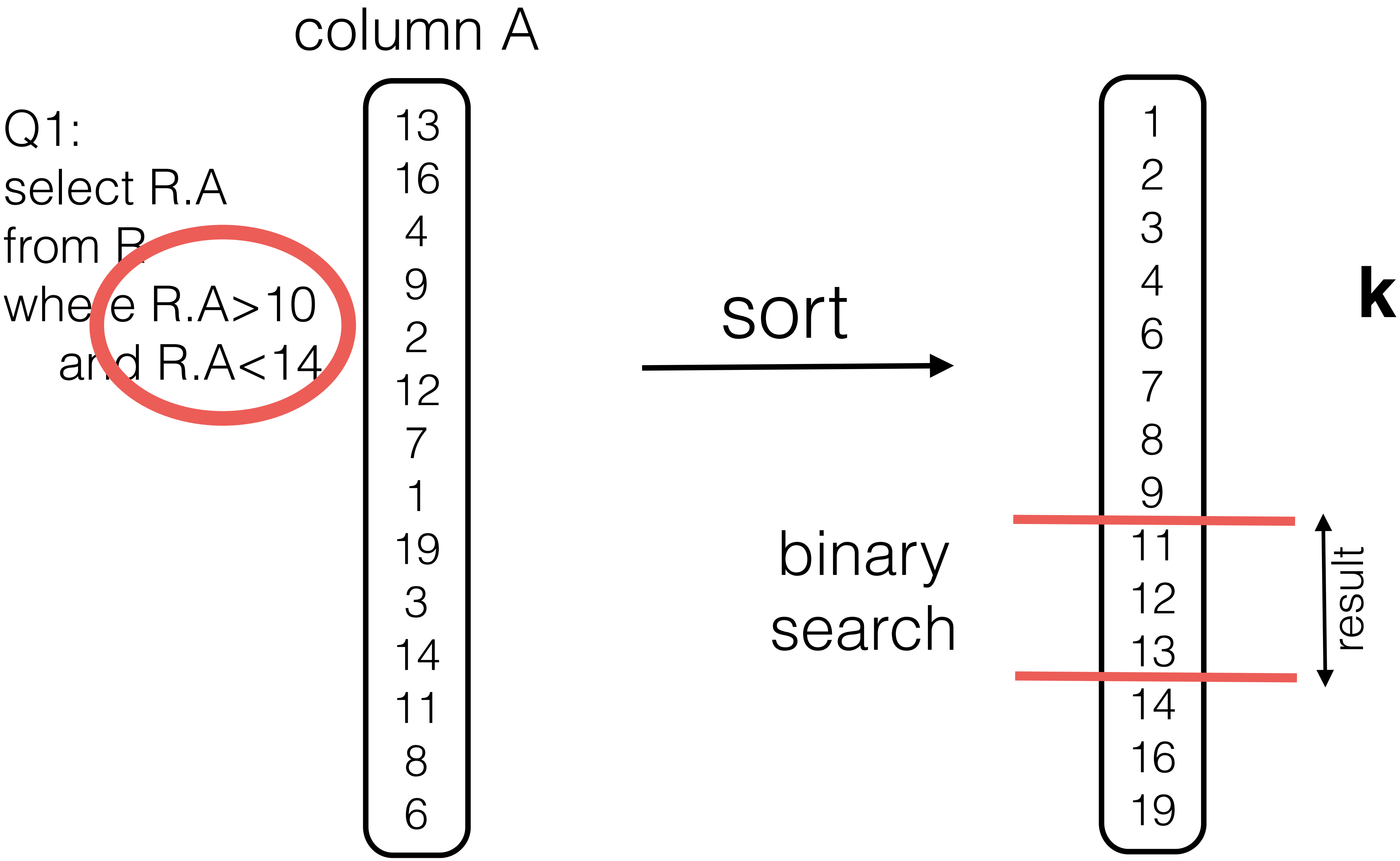
sort

binary search

1
2
3
4
6
7
8
9
11
12
13
14
16
19

result

Database Cracking CIDR 2007



Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

sort

1
2
3
4
6
7
8
9
11
12
13
14
16
19

binary search

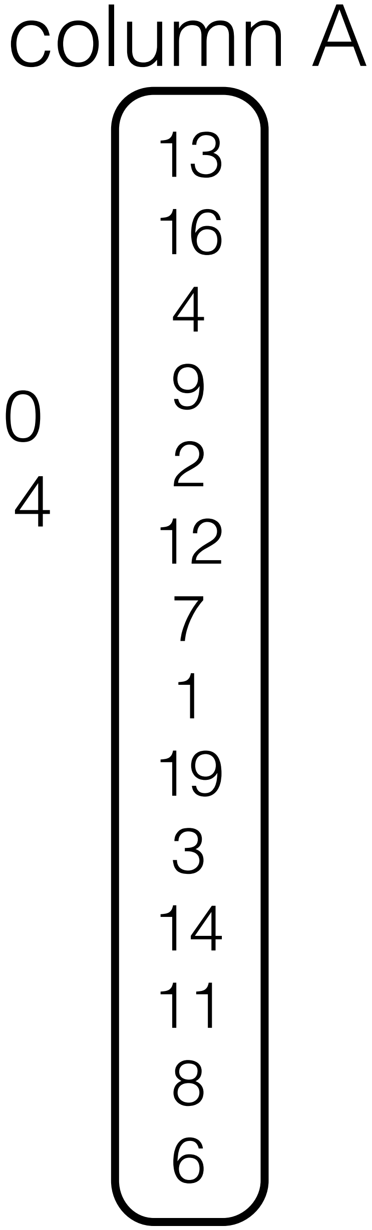
time

+

knowledge

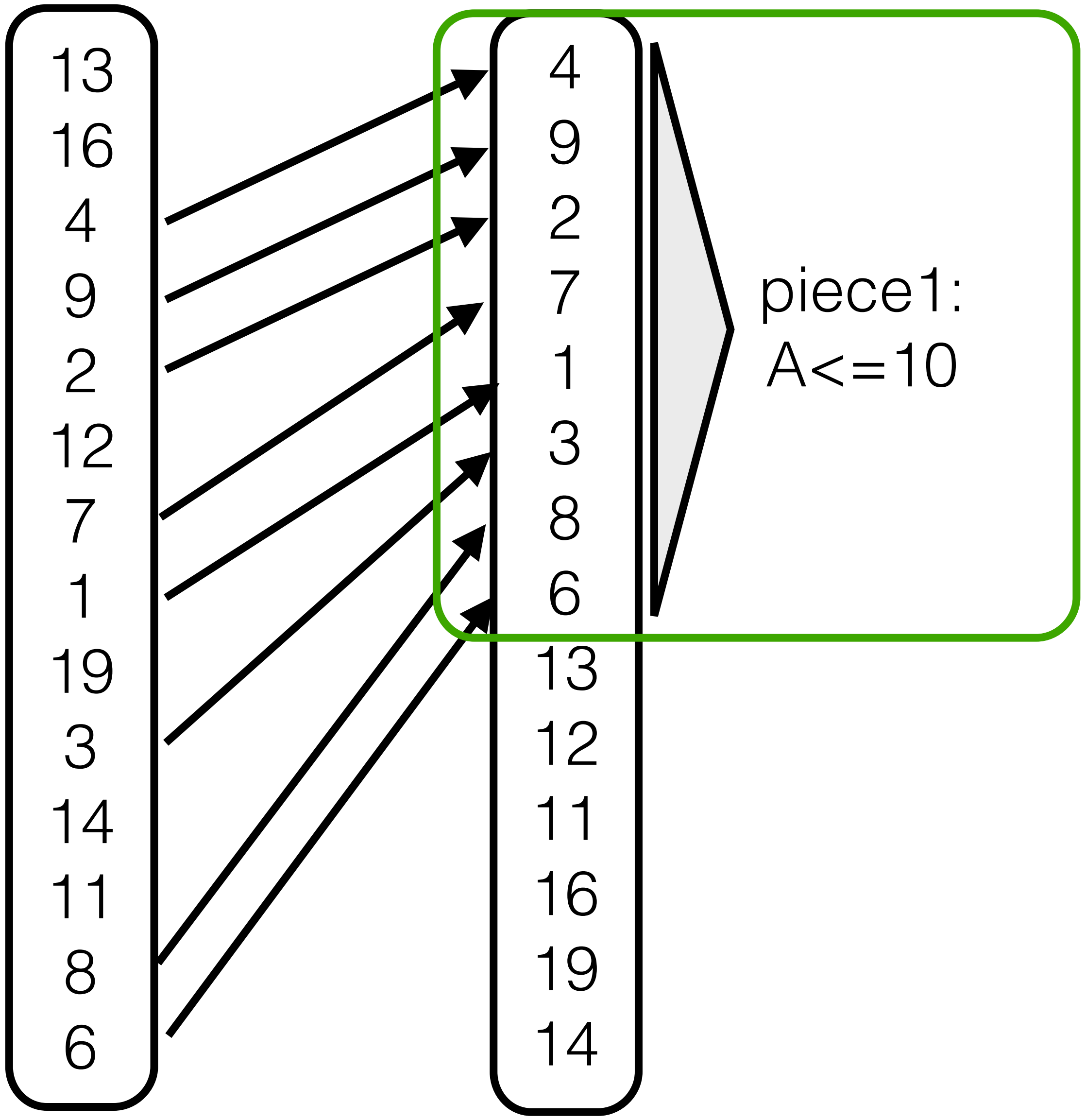
Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
    and R.A<14



Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

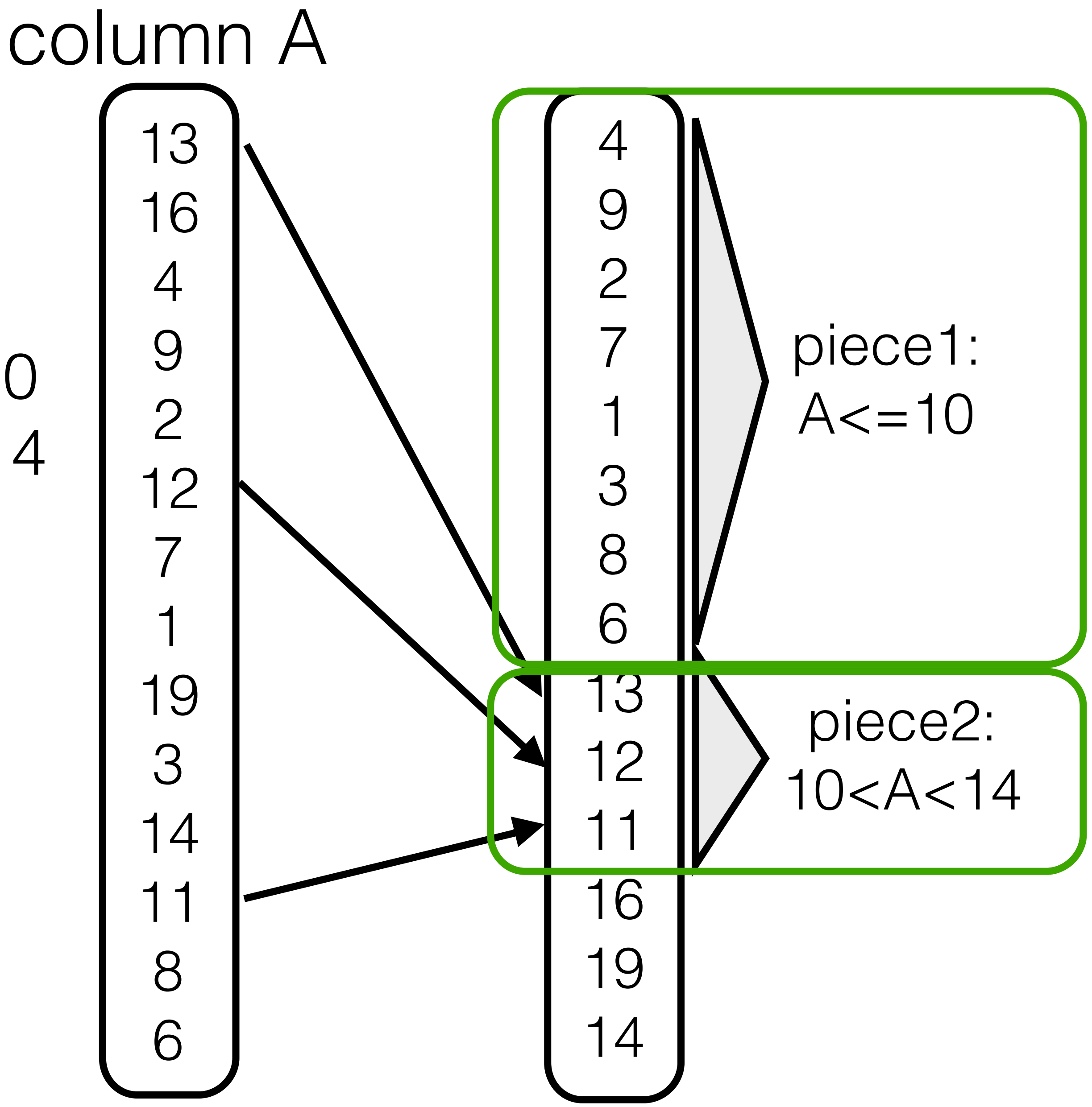


13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

Q1:
select R.A
from R
where R.A>10
and R.A<14



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

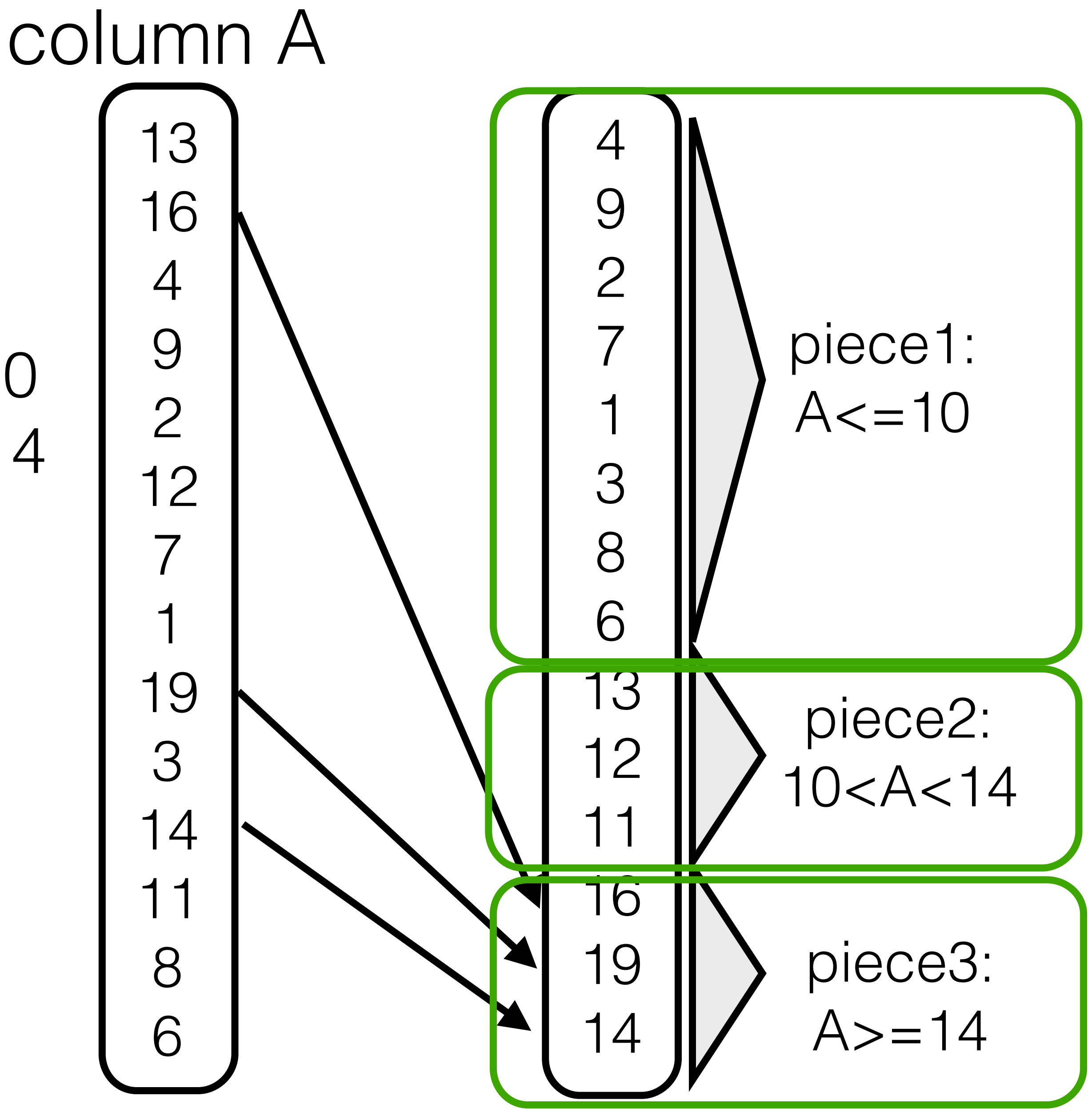
4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

piece2:
10<A<14

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

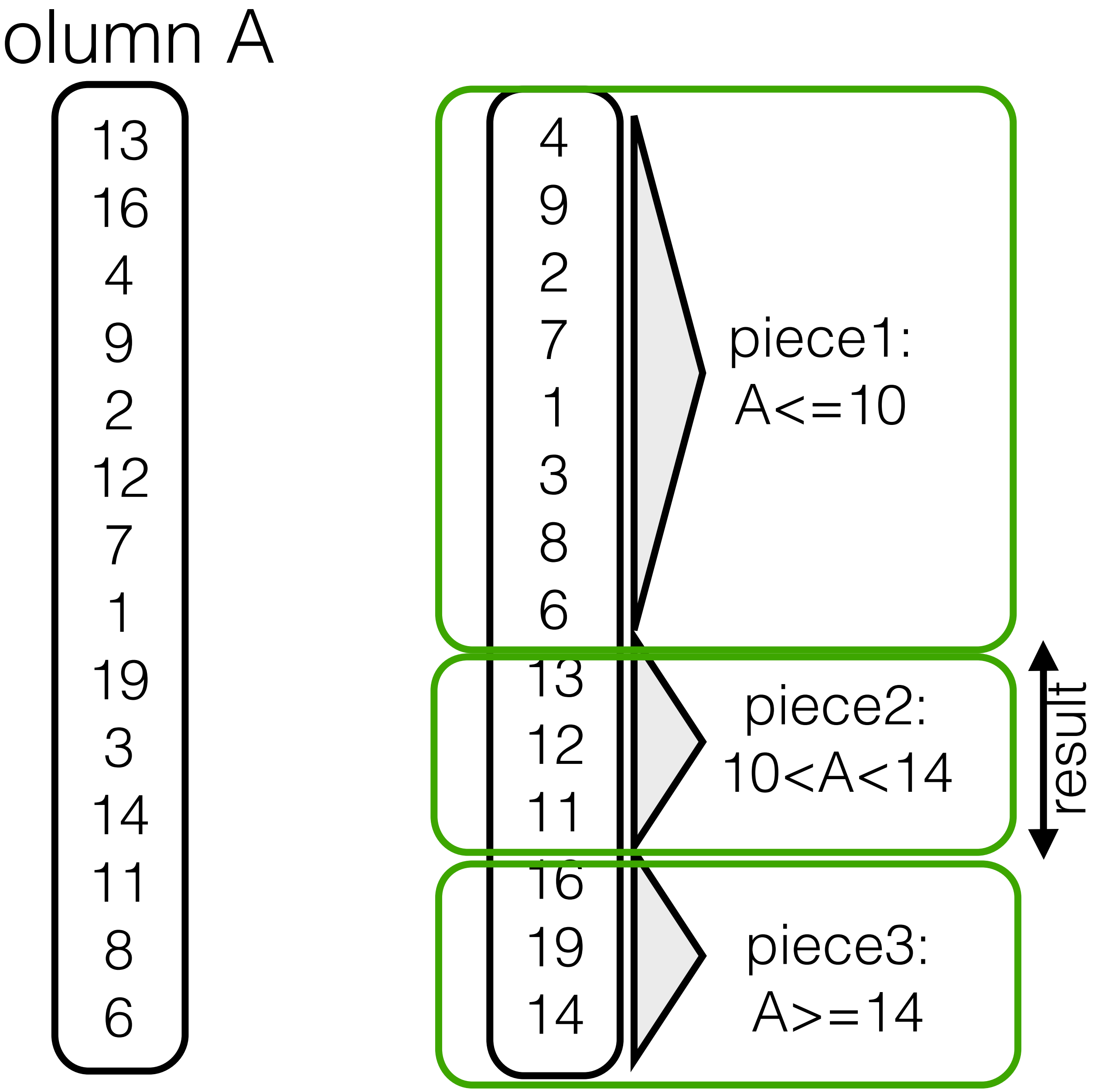
piece1:
A<=10

piece2:
10<A<14

piece3:
A>=14

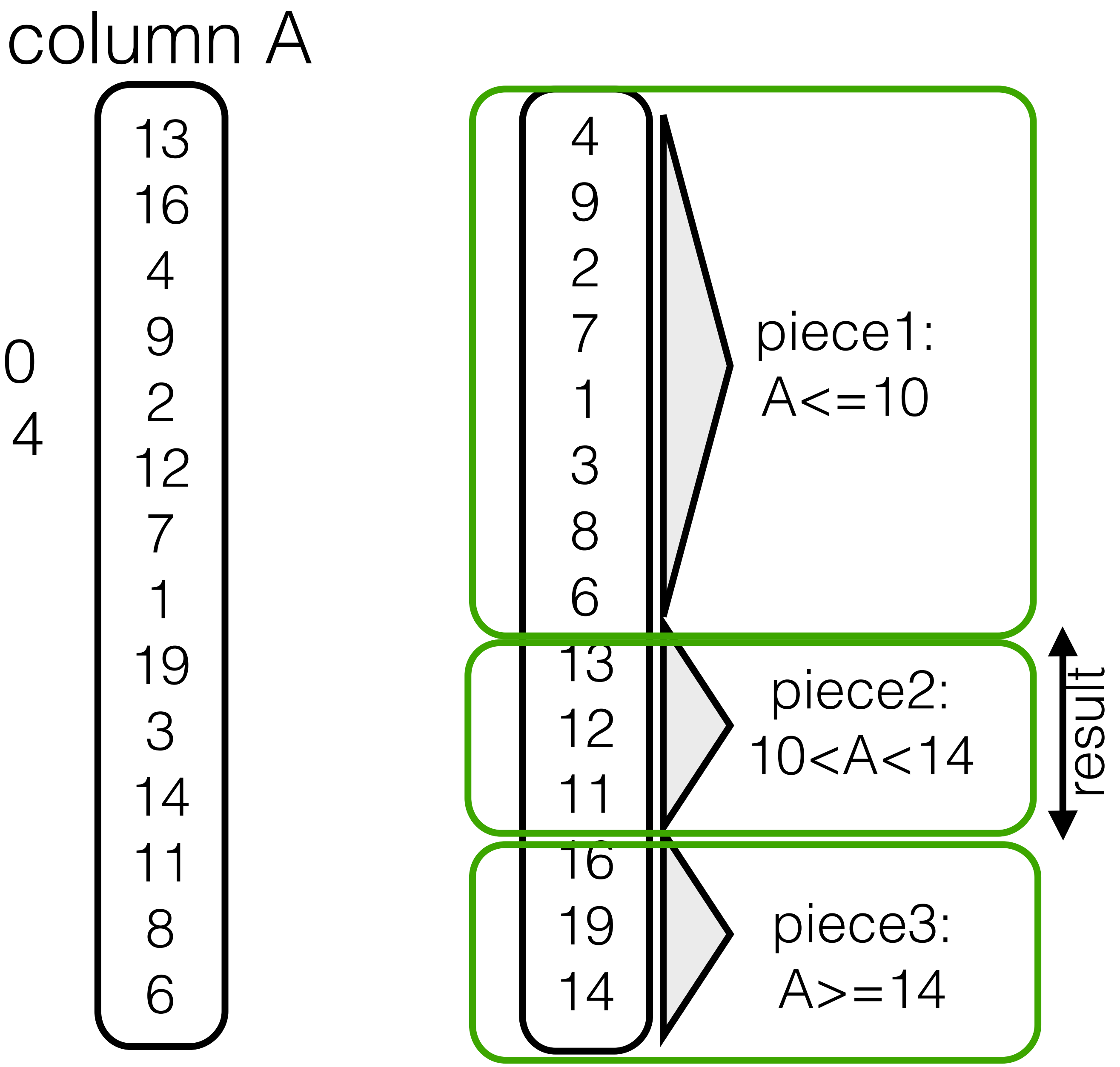
Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14



gain knowledge on how data is organized

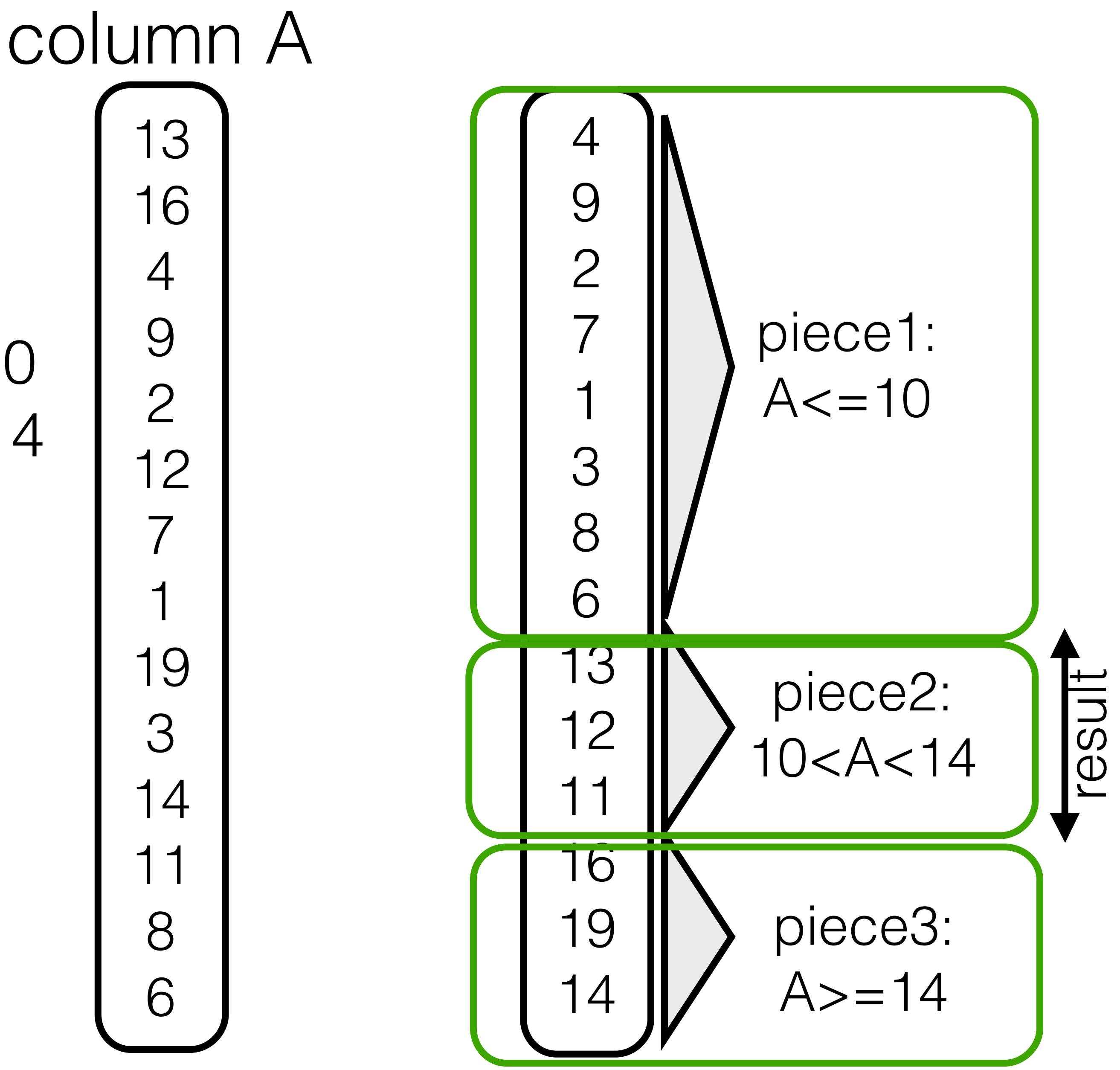
Q1:
select R.A
from R
where R.A>10
    and R.A<14



Database Cracking CIDR 2007

gain knowledge on how data is organized

Q1:
select R.A
from R
where R.A>10
and R.A<14

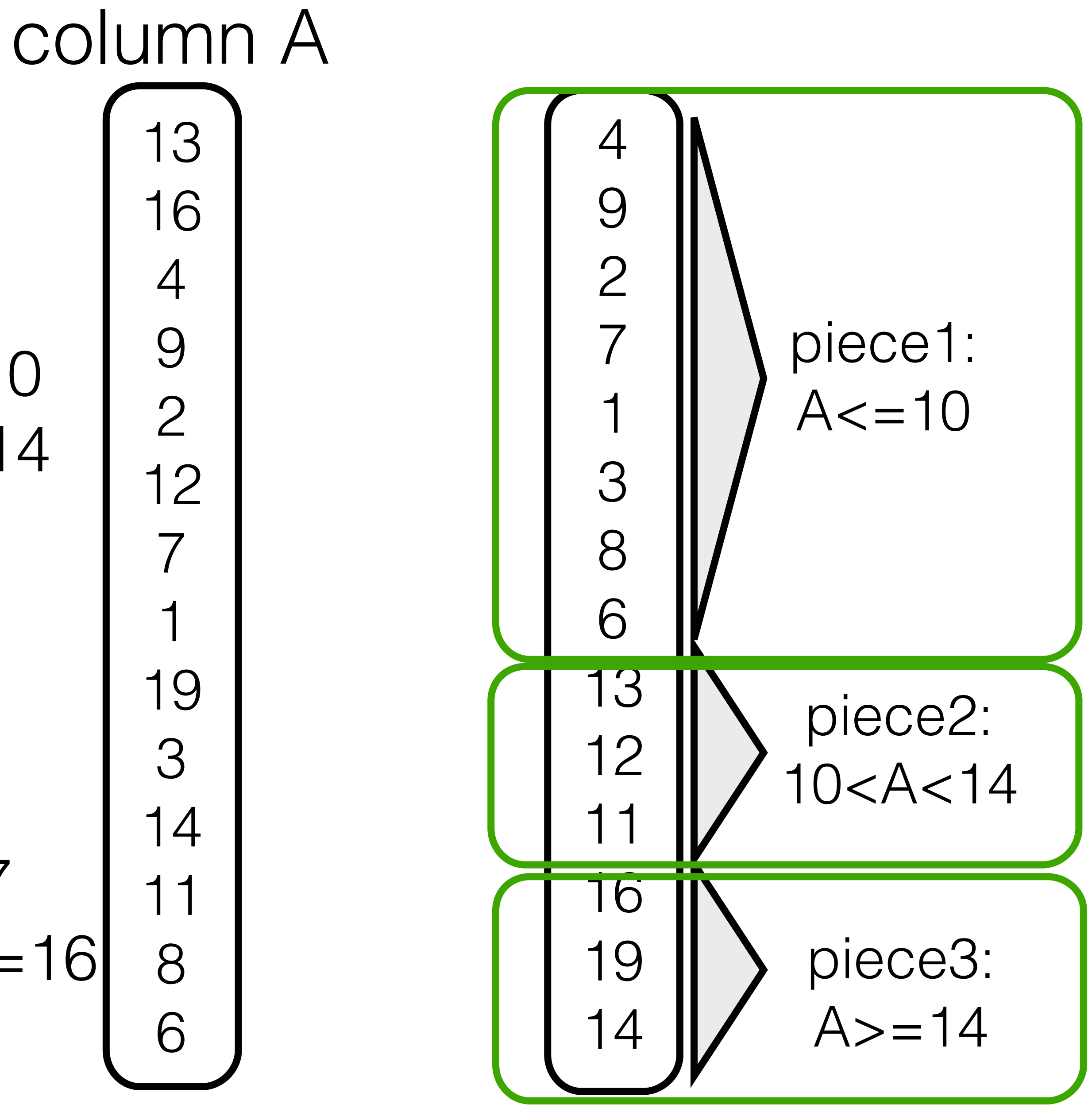


dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

piece2:
10<A<14

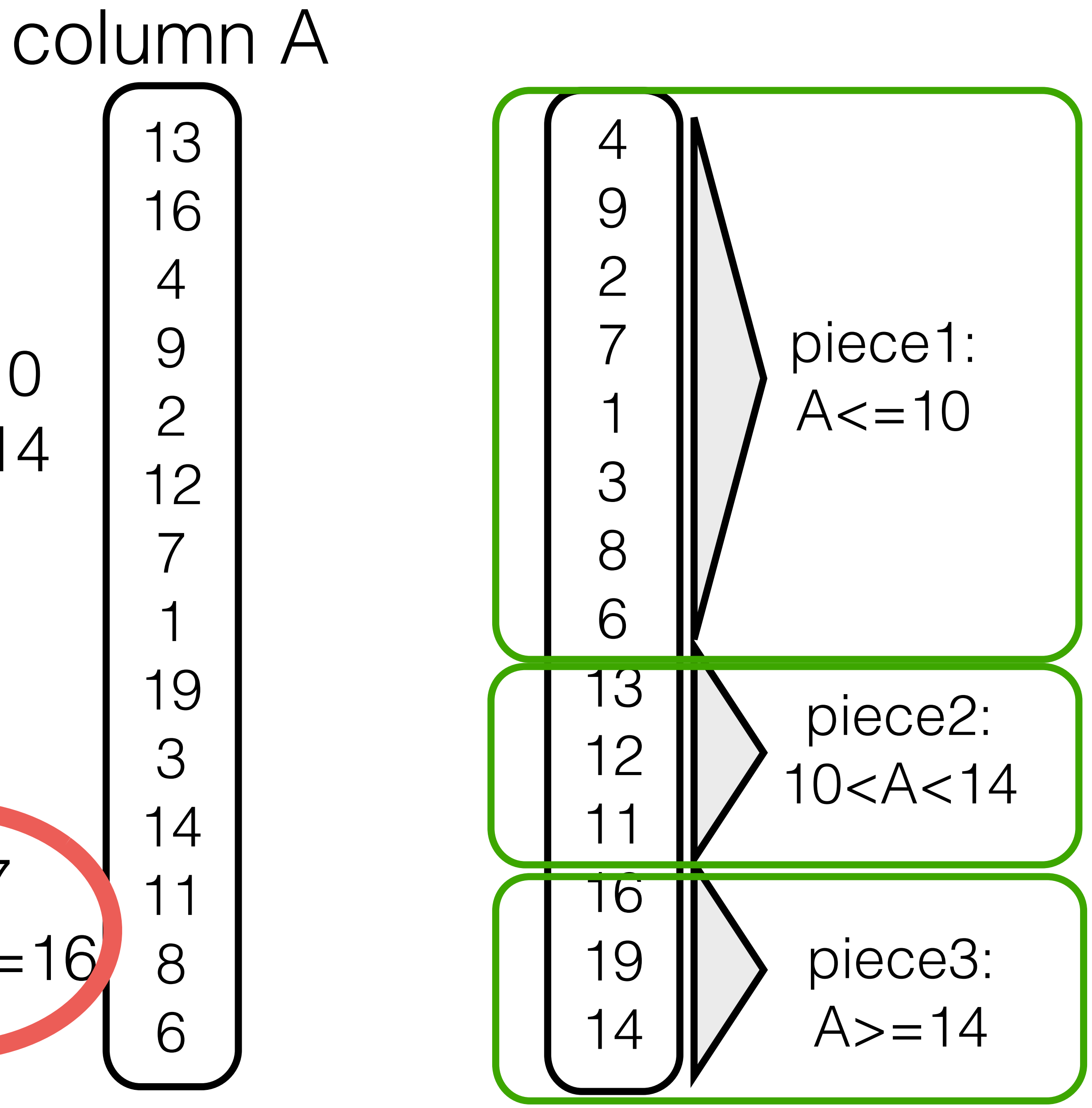
piece3:
A>=14

dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
    and R.A<14

Q2:
select R.A
from R
where R.A>7
    and R.A<=16

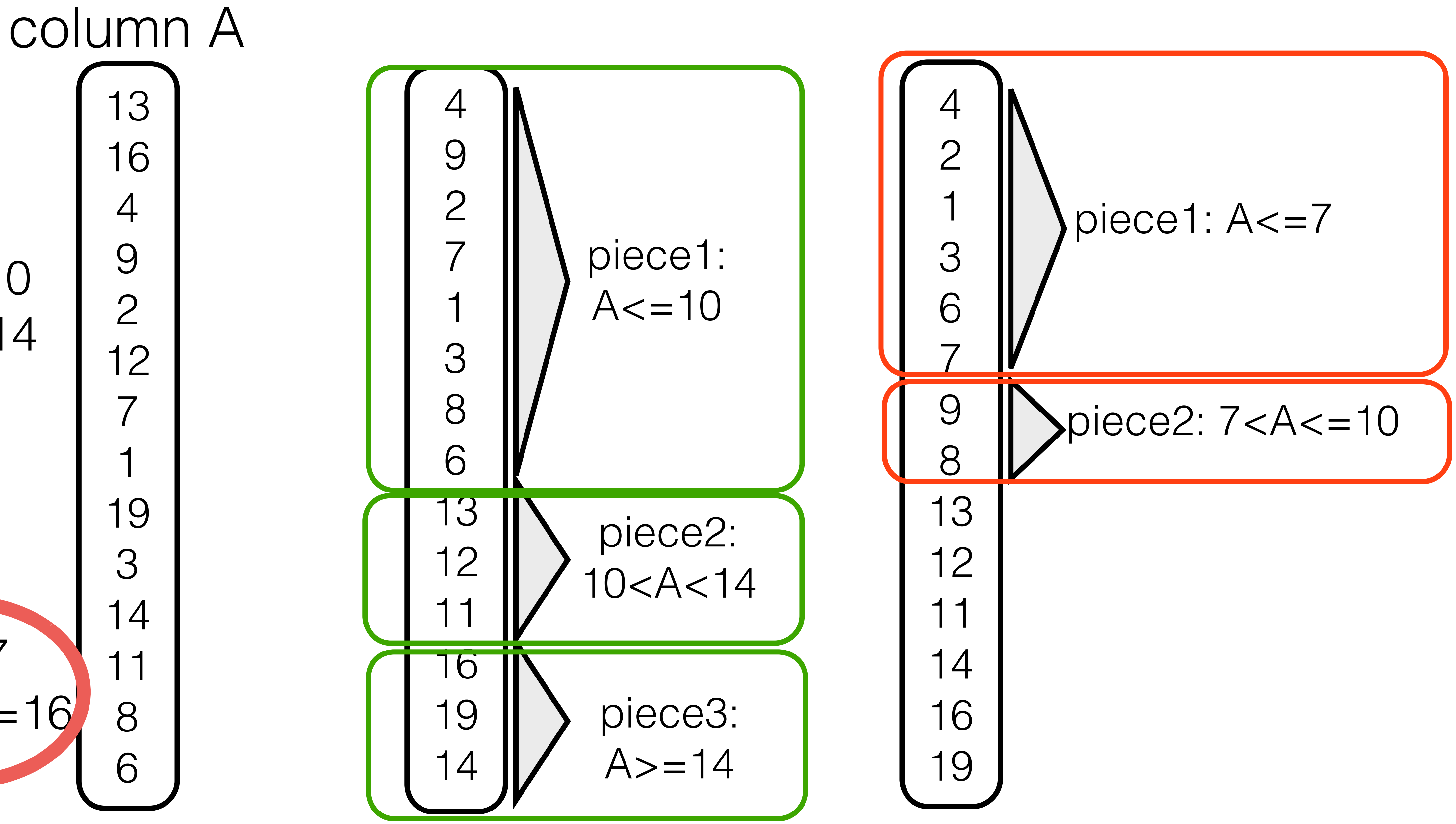


dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



column A		
13	4	4
16	9	2
4	2	1
9	7	3
2	1	6
12	3	7
7	8	9
1	6	8
19	13	13
3	12	12
14	11	11
11	16	14
8	19	16
6	14	19

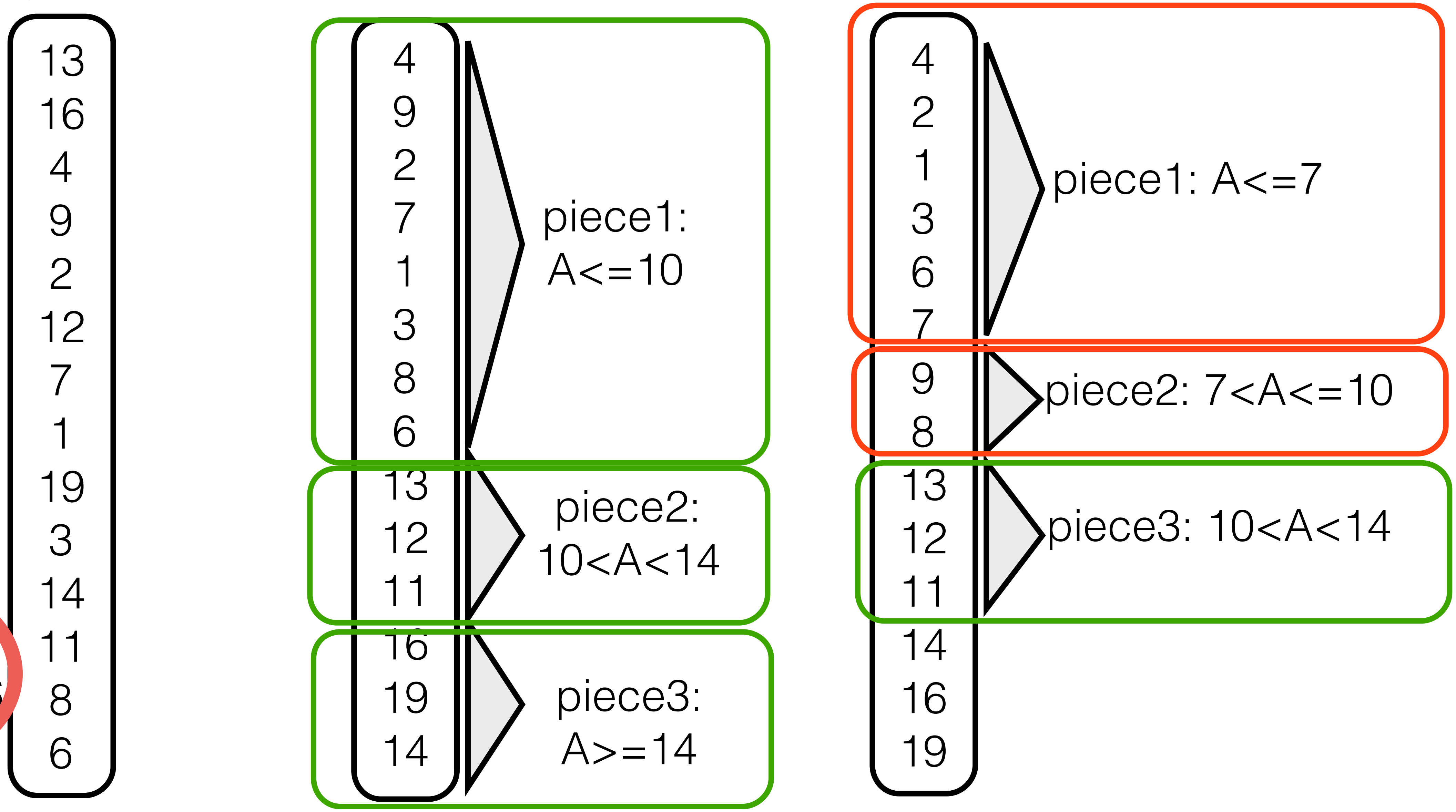
dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

column A

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

piece2:
10<A<14

piece3:
A>=14

4
2
1
3
6
7
9
8
13
12
11
14
16
19

piece1: A<=7

piece2: 7<A<=10

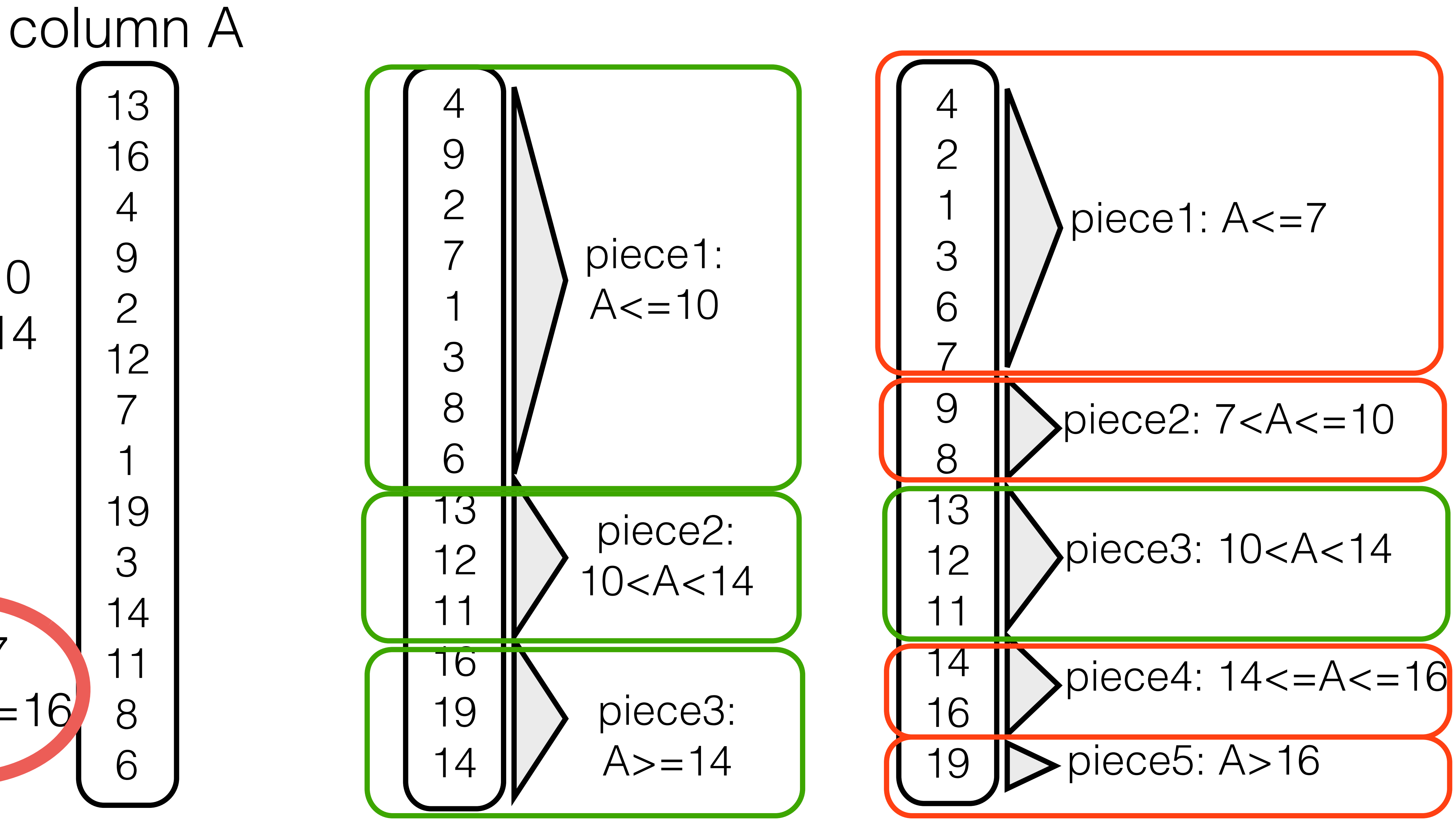
piece3: 10<A<14

dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

piece2:
10<A<14

piece3:
A>=14

4
2
1
3
6
7
9
8
13
12
11
14
16
19

piece1: A<=7

piece2: 7<A<=10

piece3: 10<A<14

piece4: 14<=A<=16

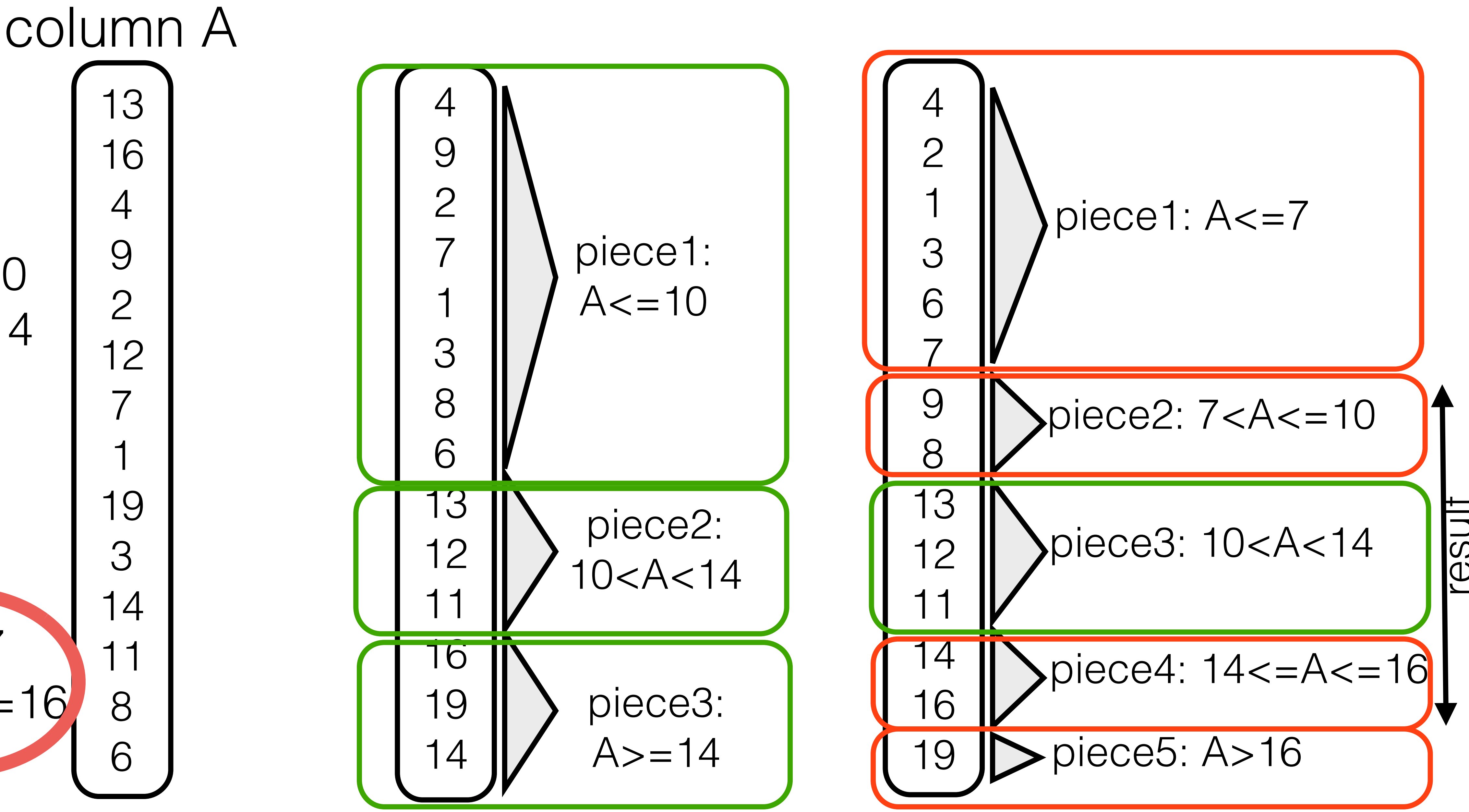
piece5: A>16

dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



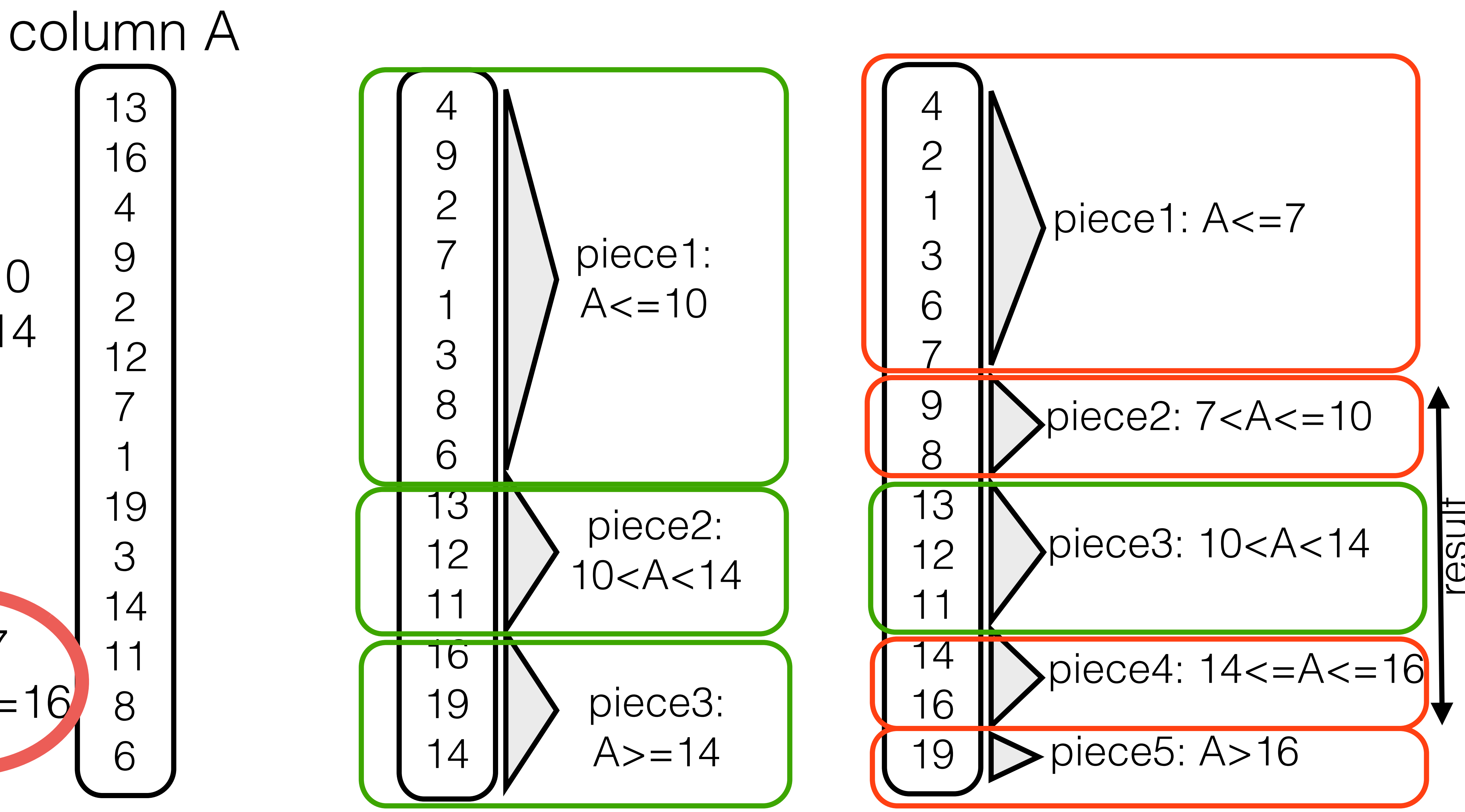
dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

the more we crack, the more we learn

Q1:
select R.A
from R
where R.A>10
and R.A<14

Q2:
select R.A
from R
where R.A>7
and R.A<=16



column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

4
9
2
7
1
3
8
6
13
12
11
16
19
14

piece1:
A<=10

piece2:
10<A<14

piece3:
A>=14

4
2
1
3
6
7
9
8
13
12
11
14
16
19

piece1: A<=7

piece2: 7<A<=10

piece3: 10<A<14

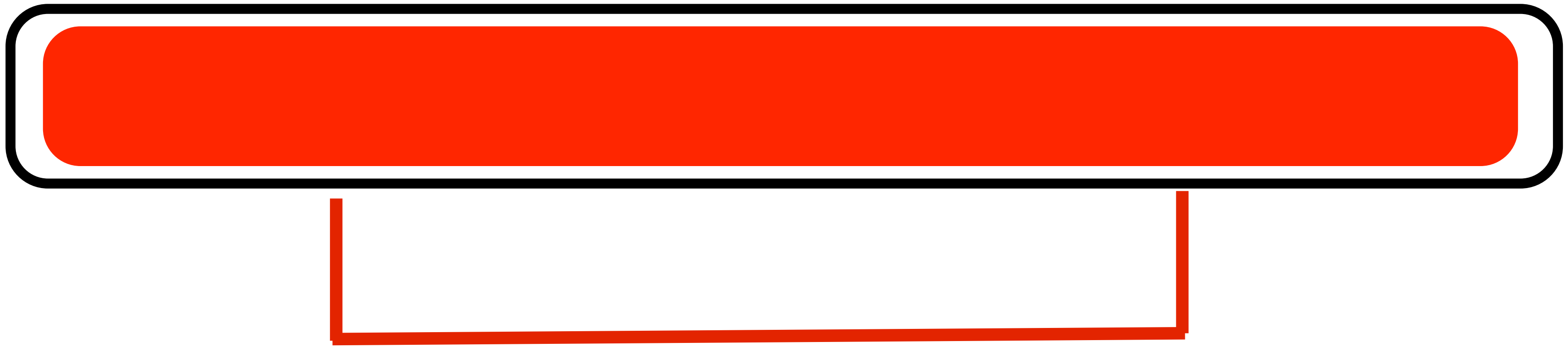
piece4: 14<=A<=16

piece5: A>16

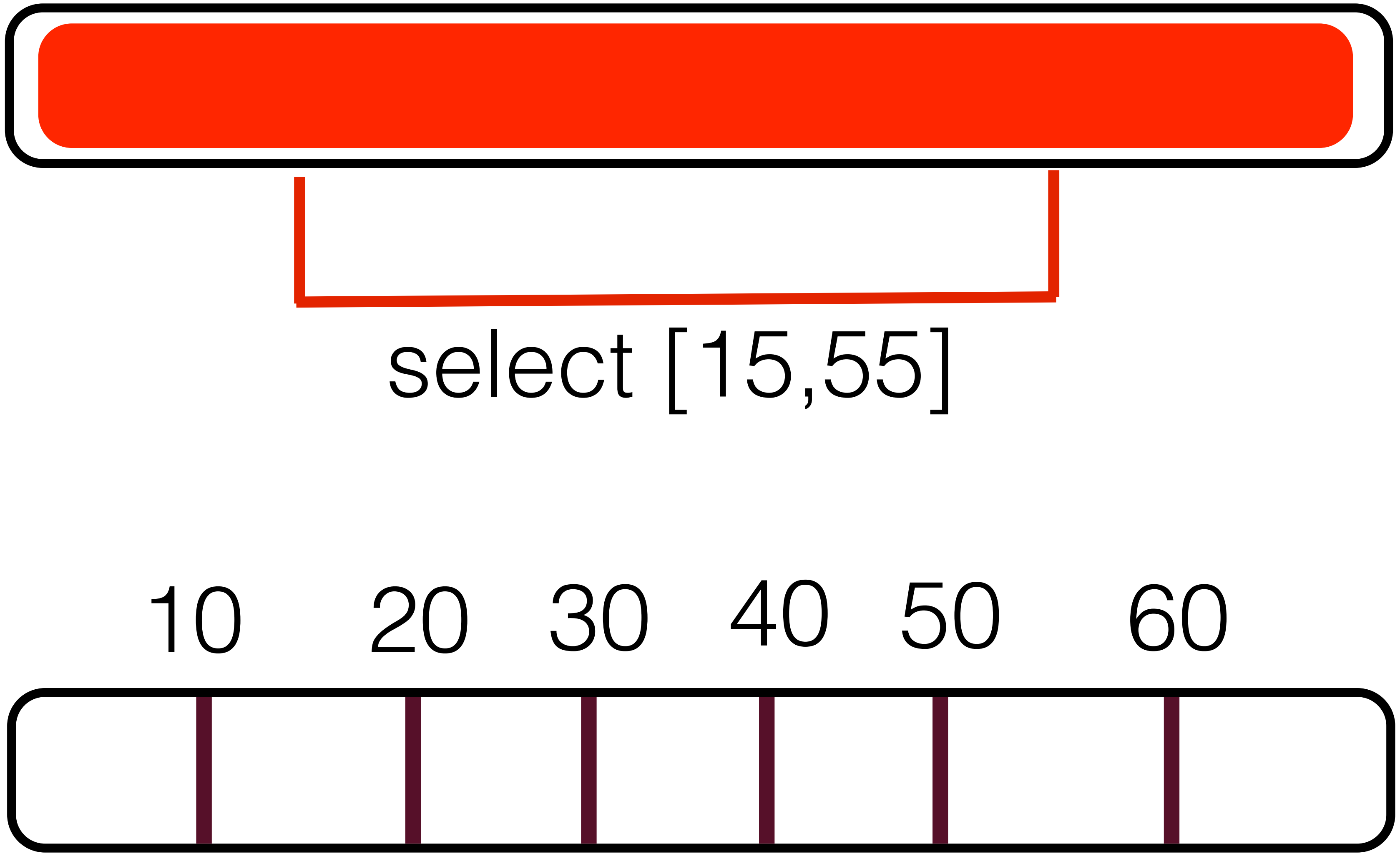
dynamically/on-the-fly within the select-operator

Database Cracking CIDR 2007

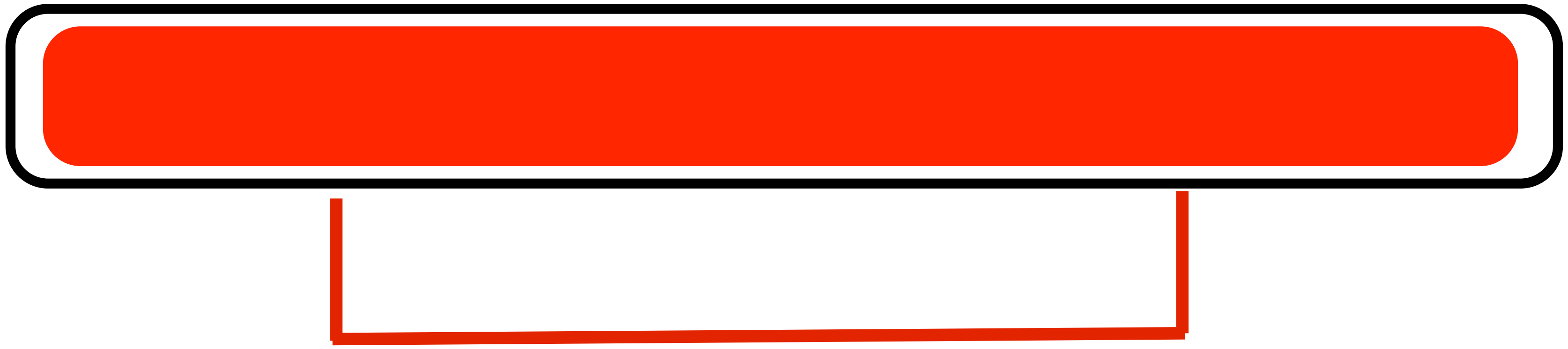
select [15,55]



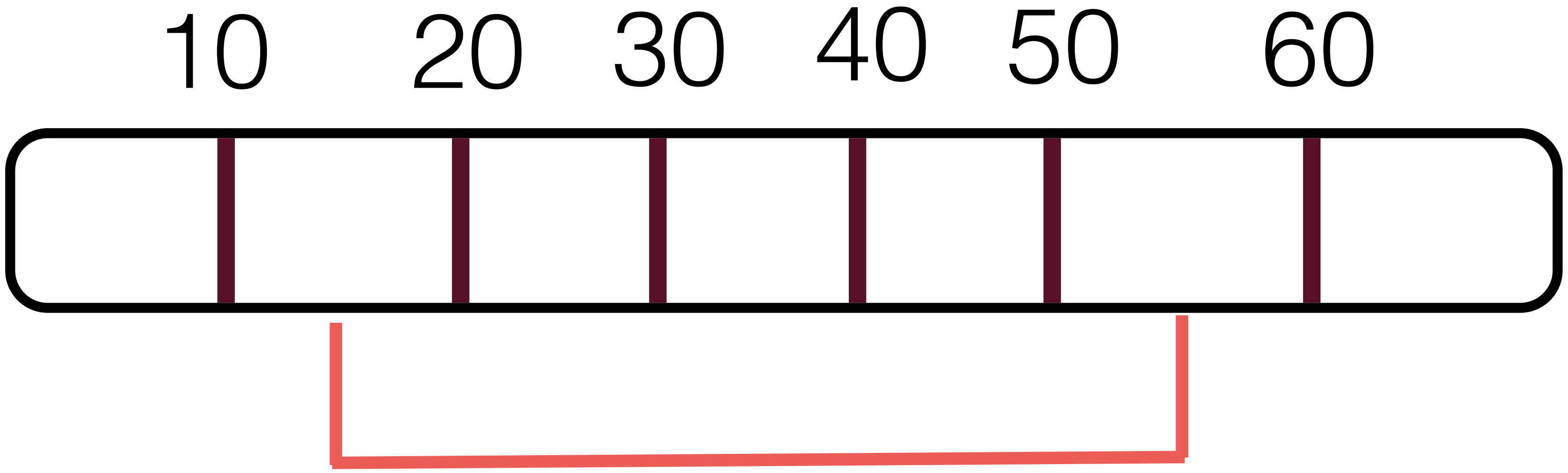
select [15,55]



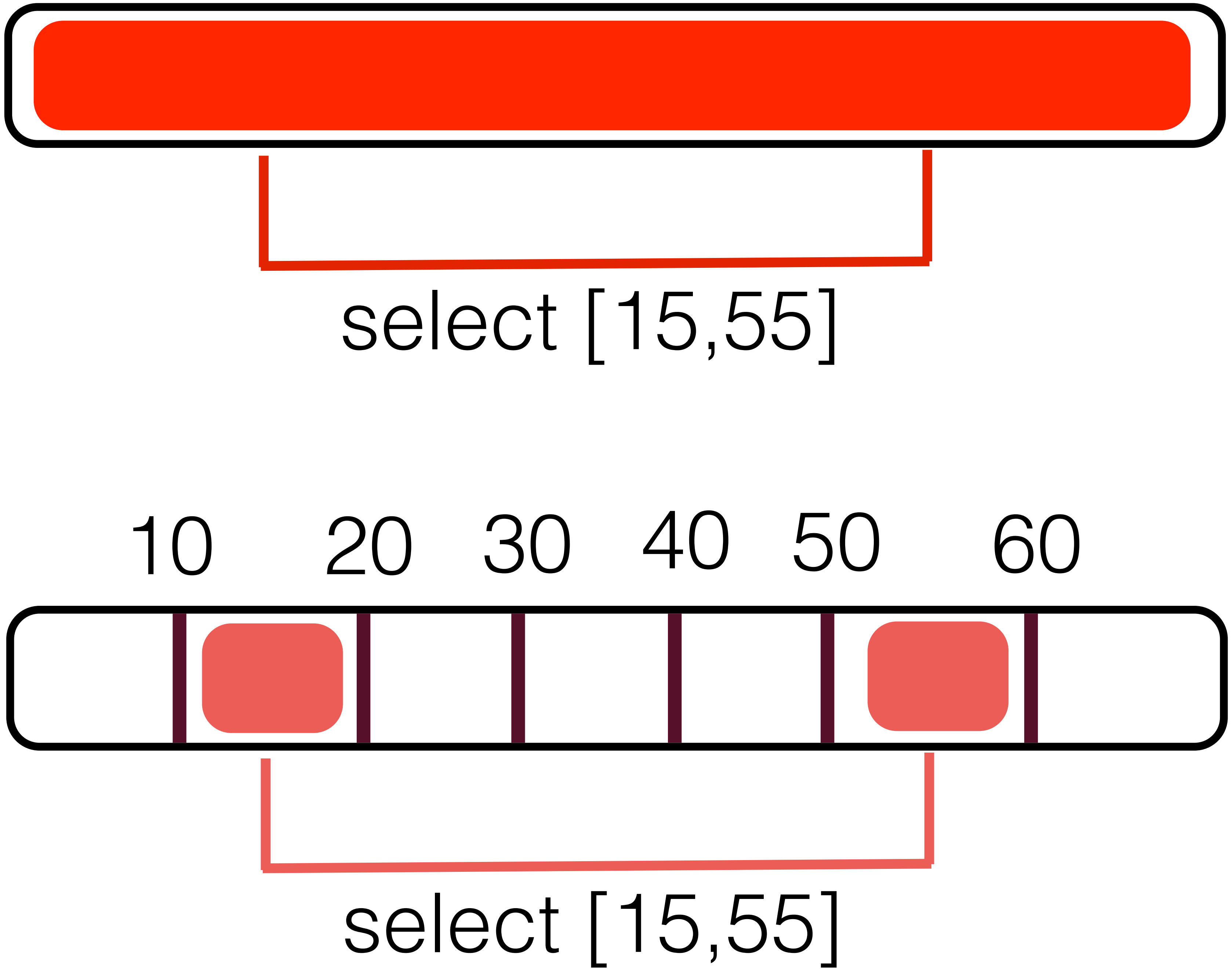
select [15,55]



select [15,55]



select [15,55]



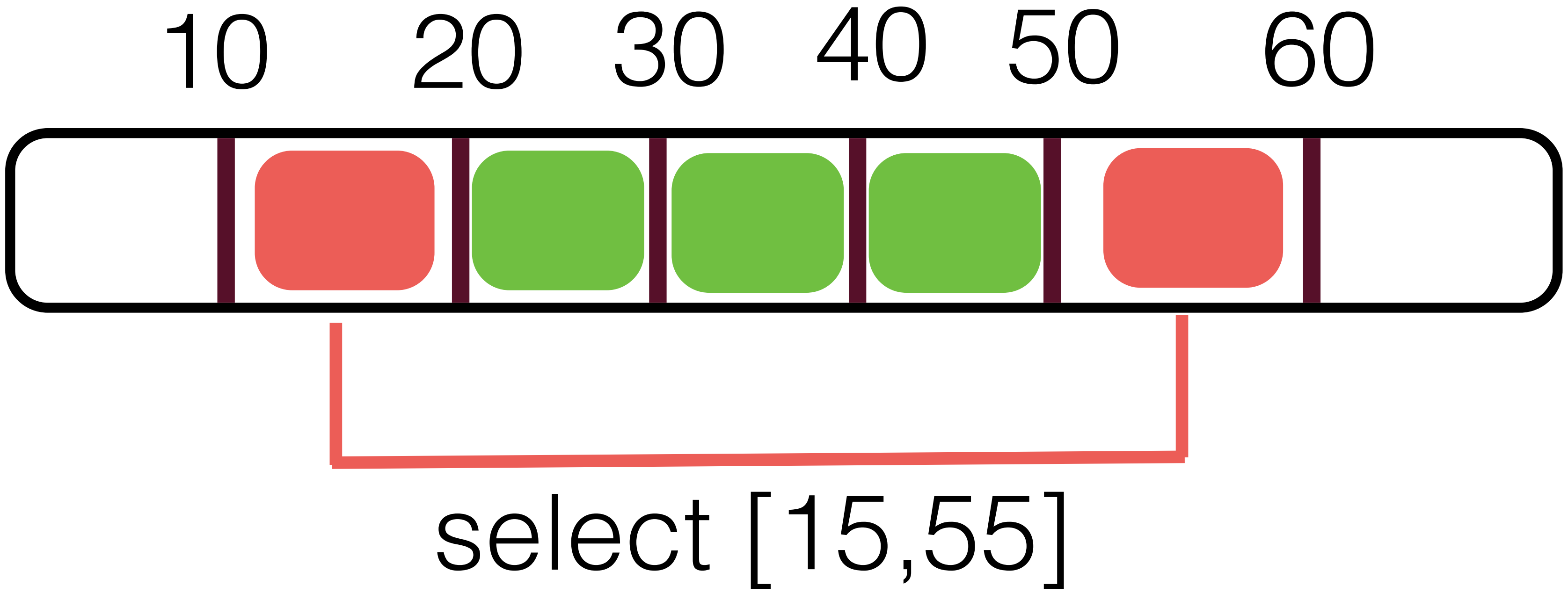
select [15,55]

select [15,55]

Database Cracking CIDR 2007

touch at most two pieces at a time

pieces become smaller and smaller

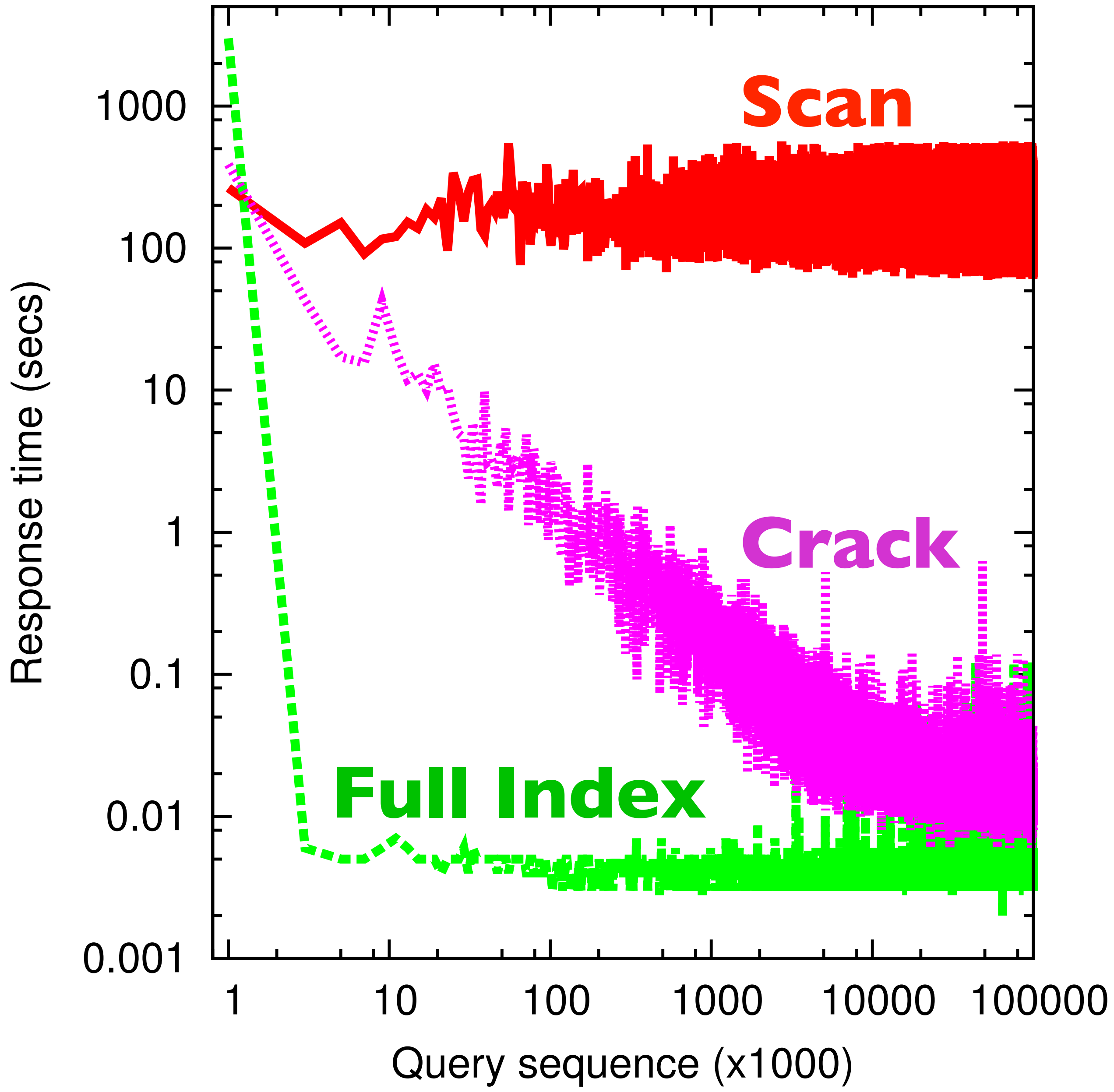


Database Cracking CIDR 2007

continuous adaptation

set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column



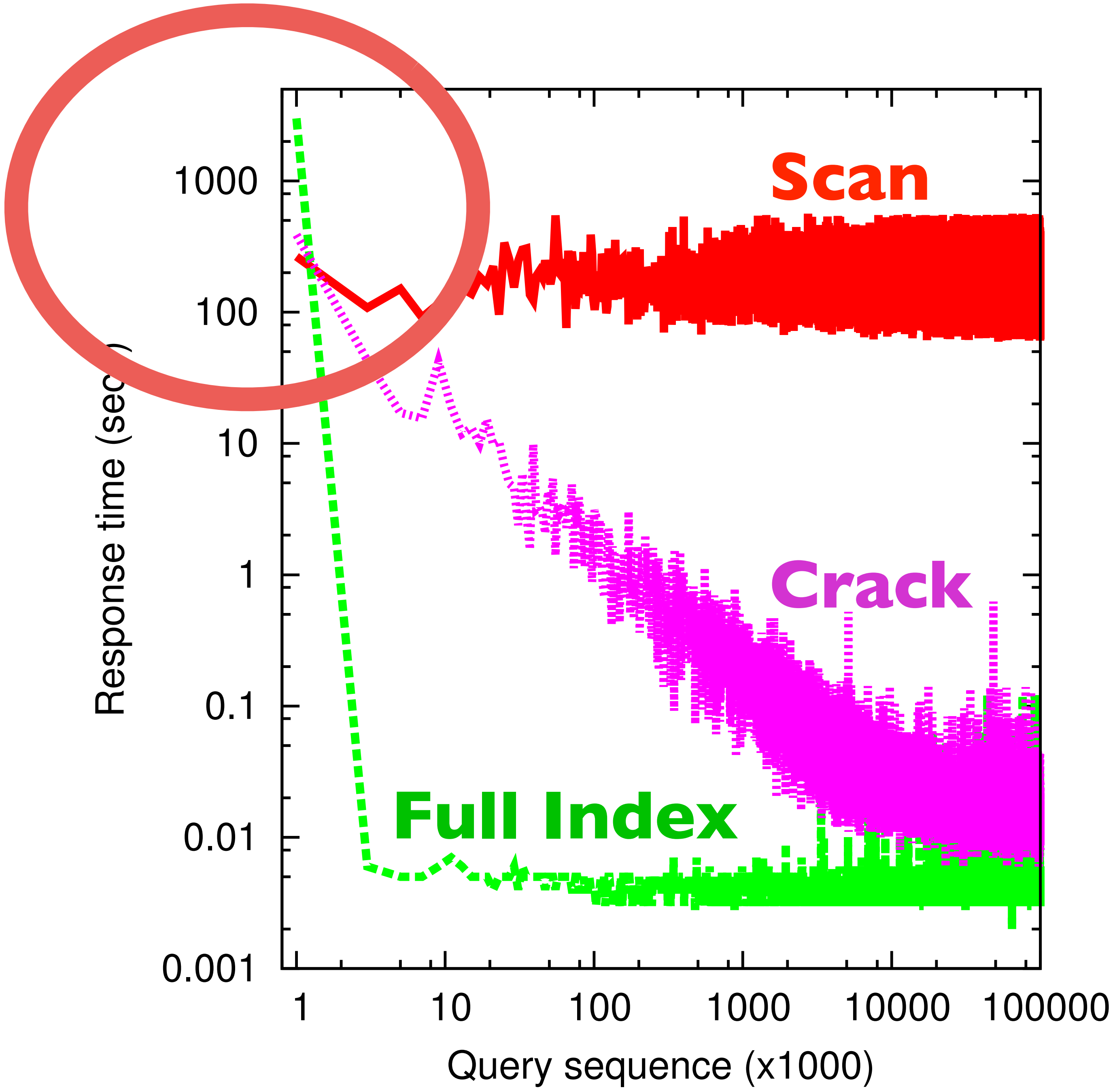
Database Cracking CIDR 2007

continuous adaptation

set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

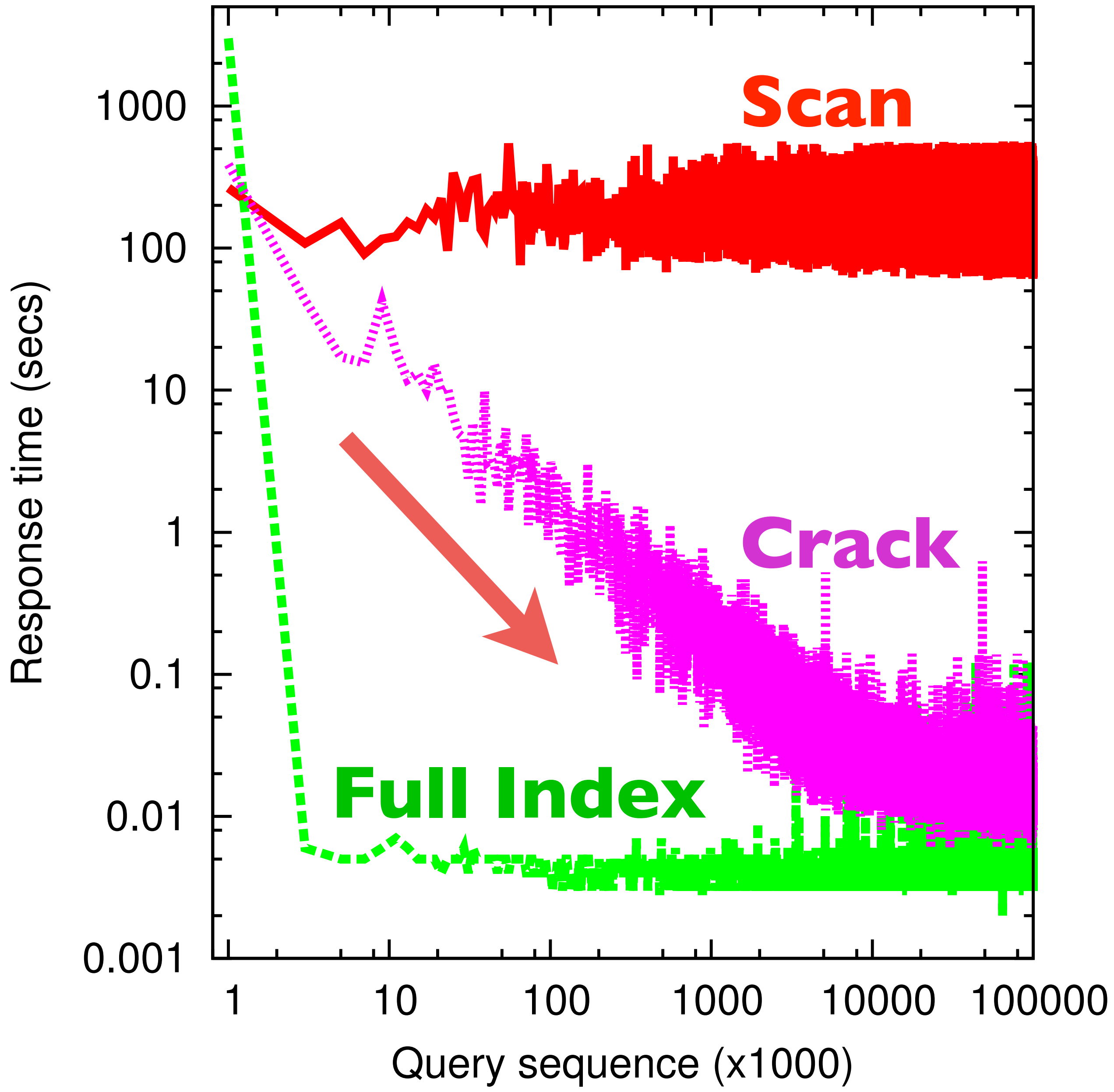


Database Cracking CIDR 2007

continuous adaptation

set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column



almost no
initialization overhead

continuous improvement

Database Cracking CIDR 2007

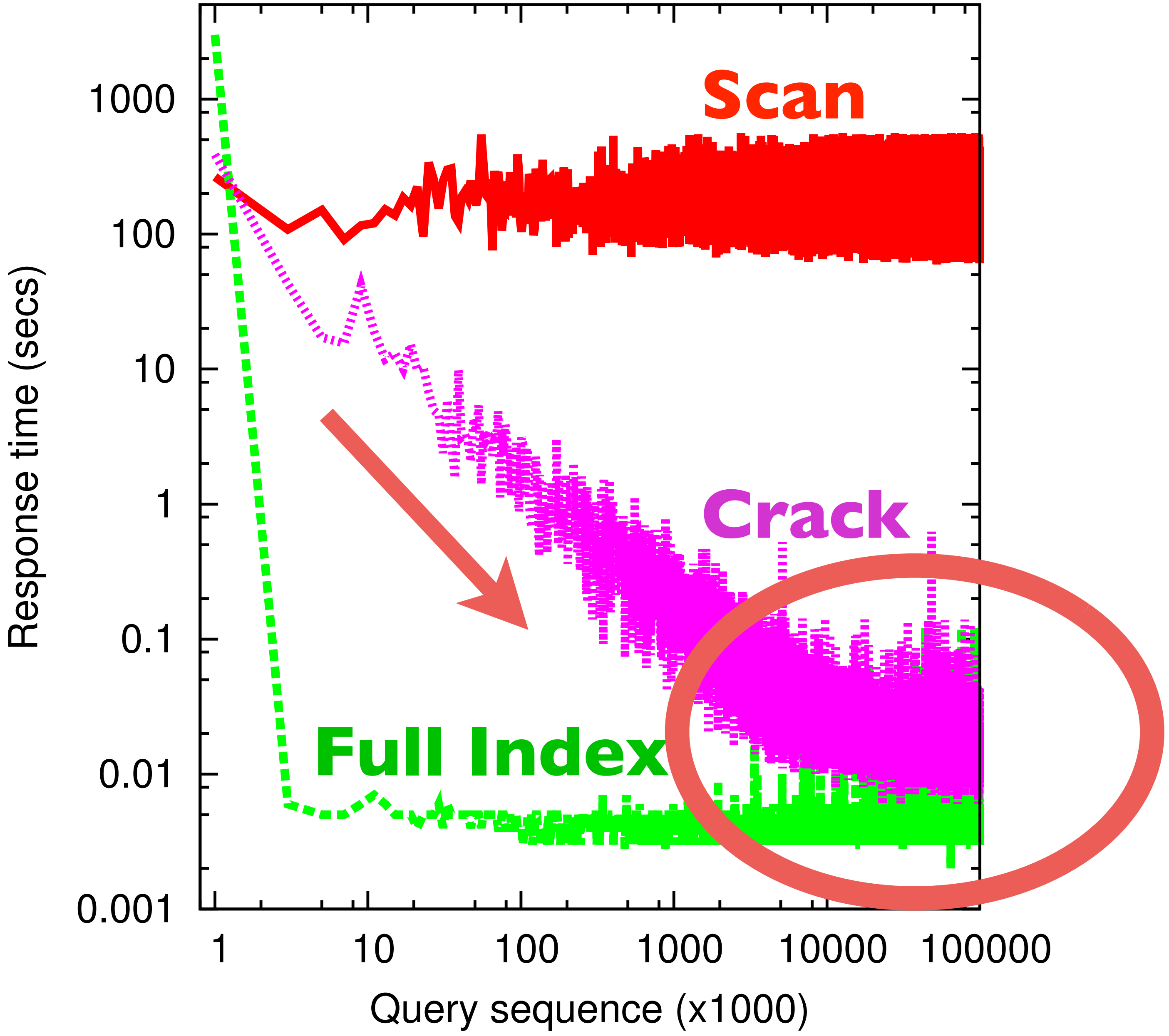
continuous adaptation

set-up

100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous improvement

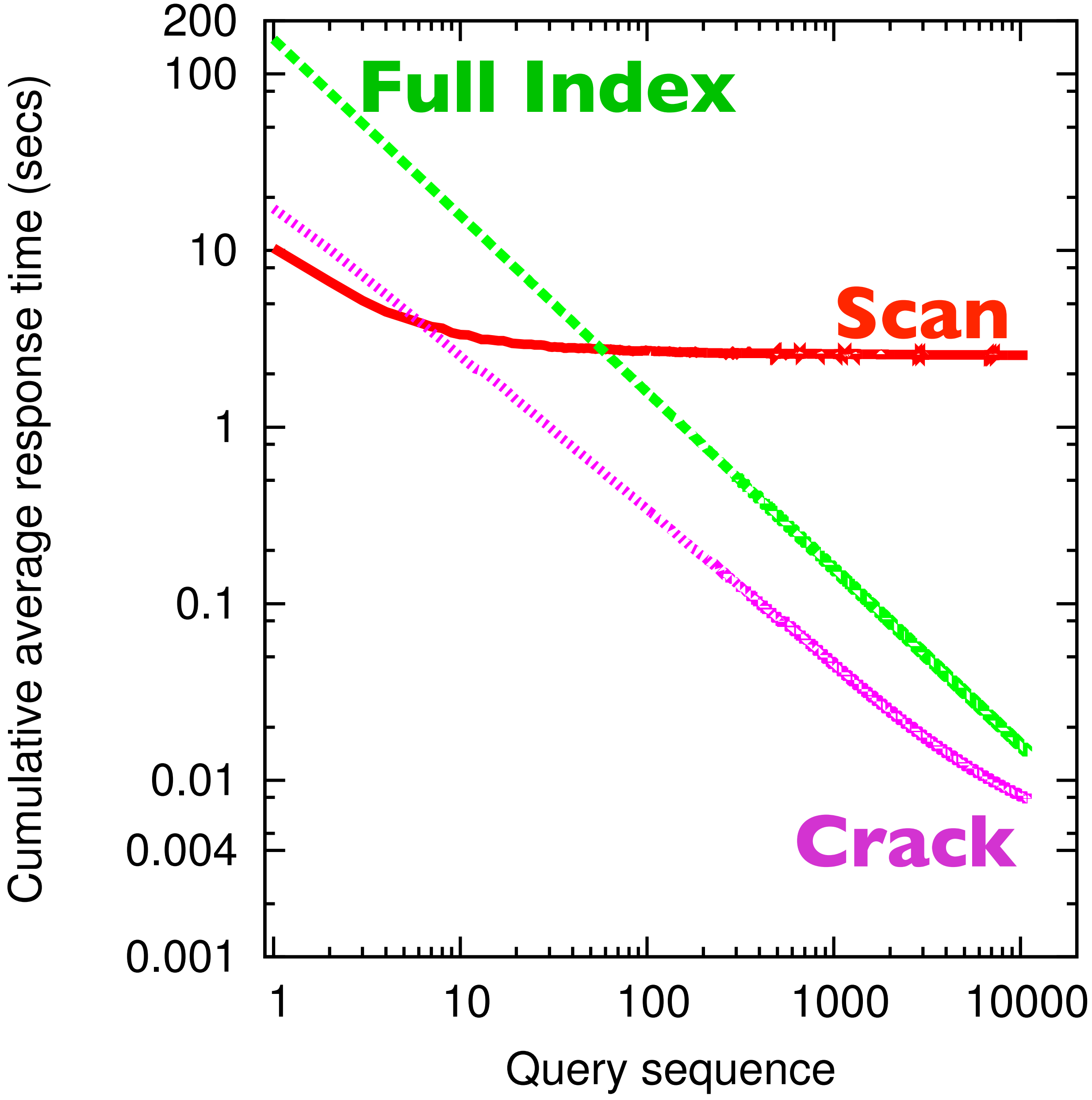


Database Cracking CIDR 2007

continuous adaptation

set-up

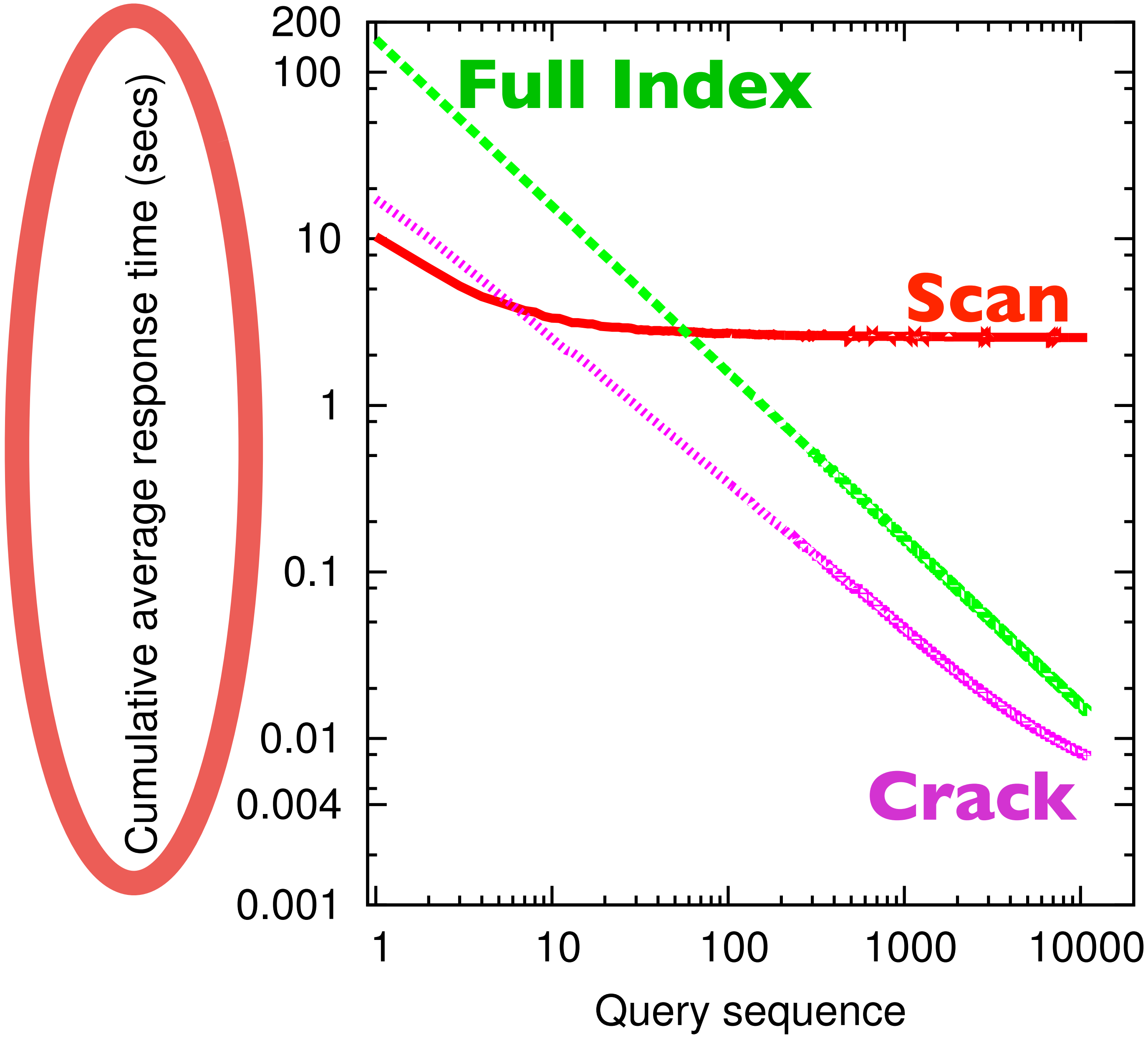
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



continuous adaptation

set-up

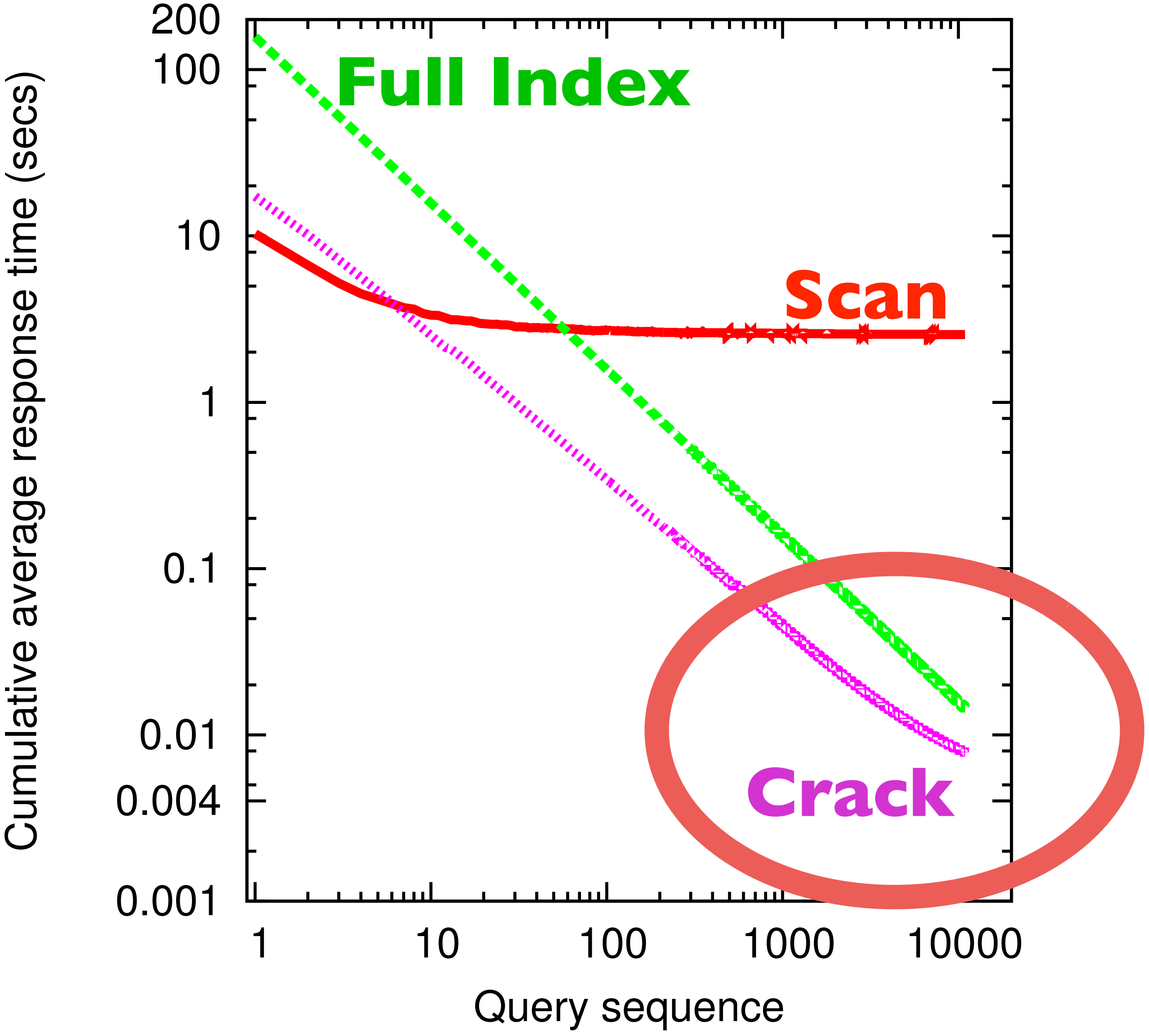
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



continuous adaptation

set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



10K queries later,
Full Index still has not
amortized the initialization costs

Database Cracking CIDR 2007

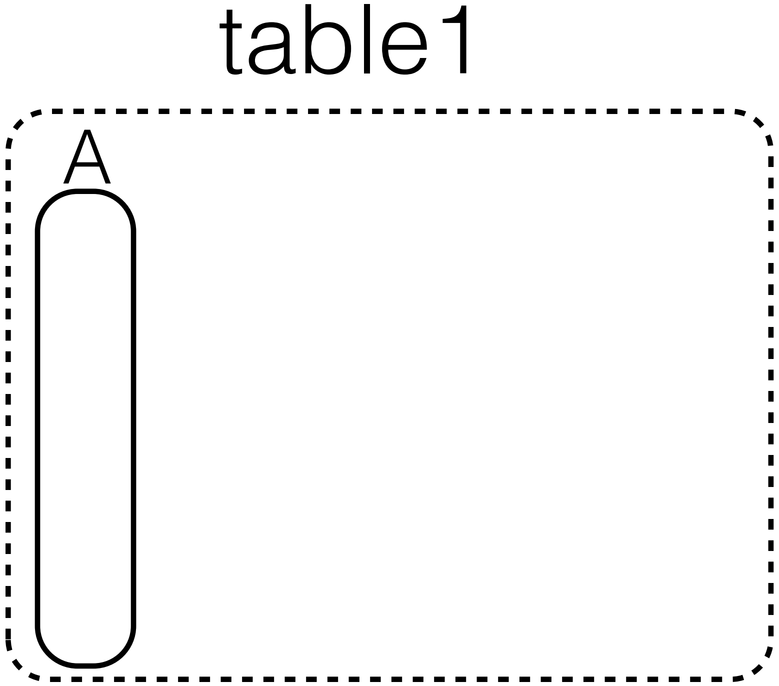


table1
A

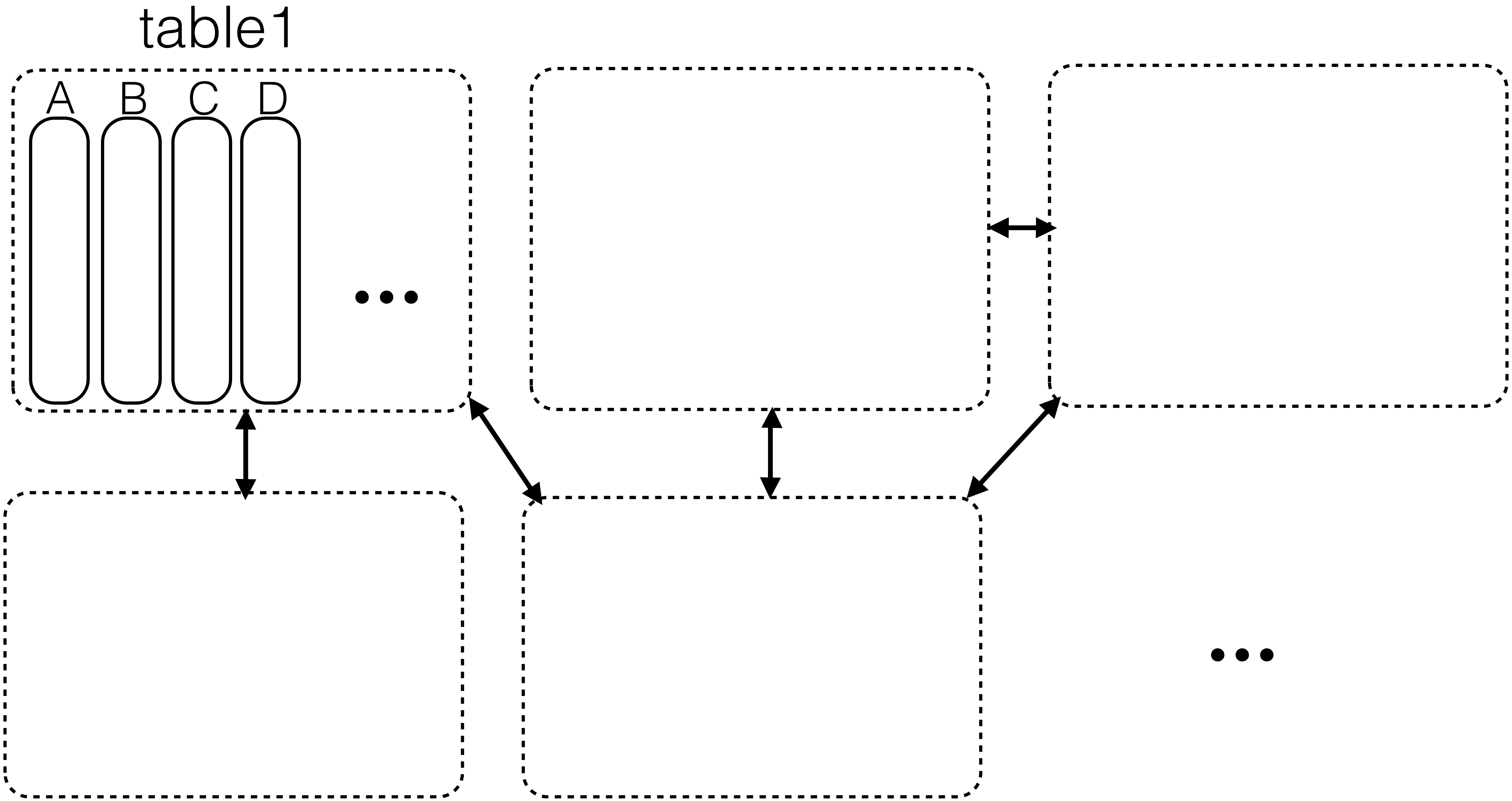
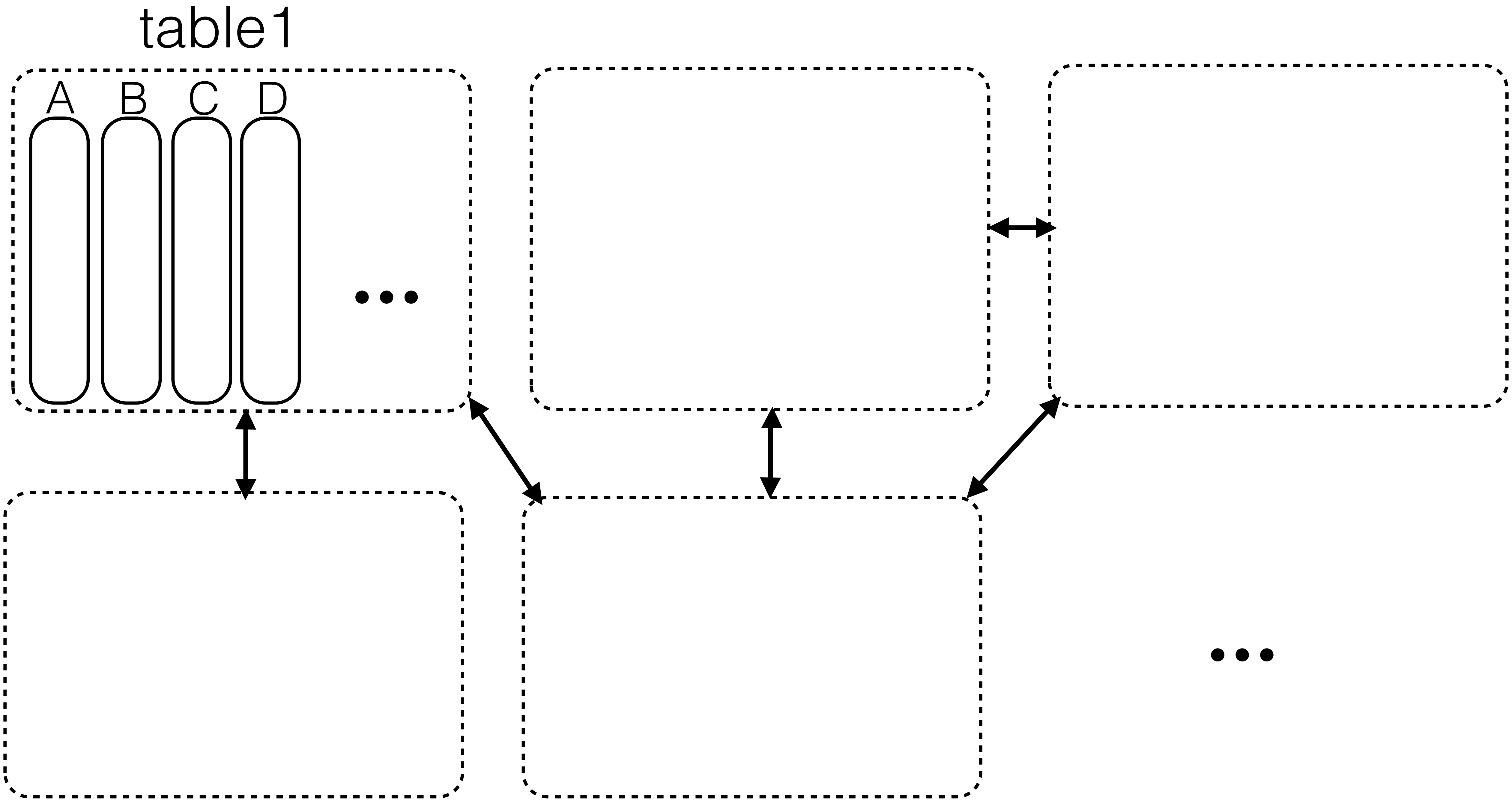


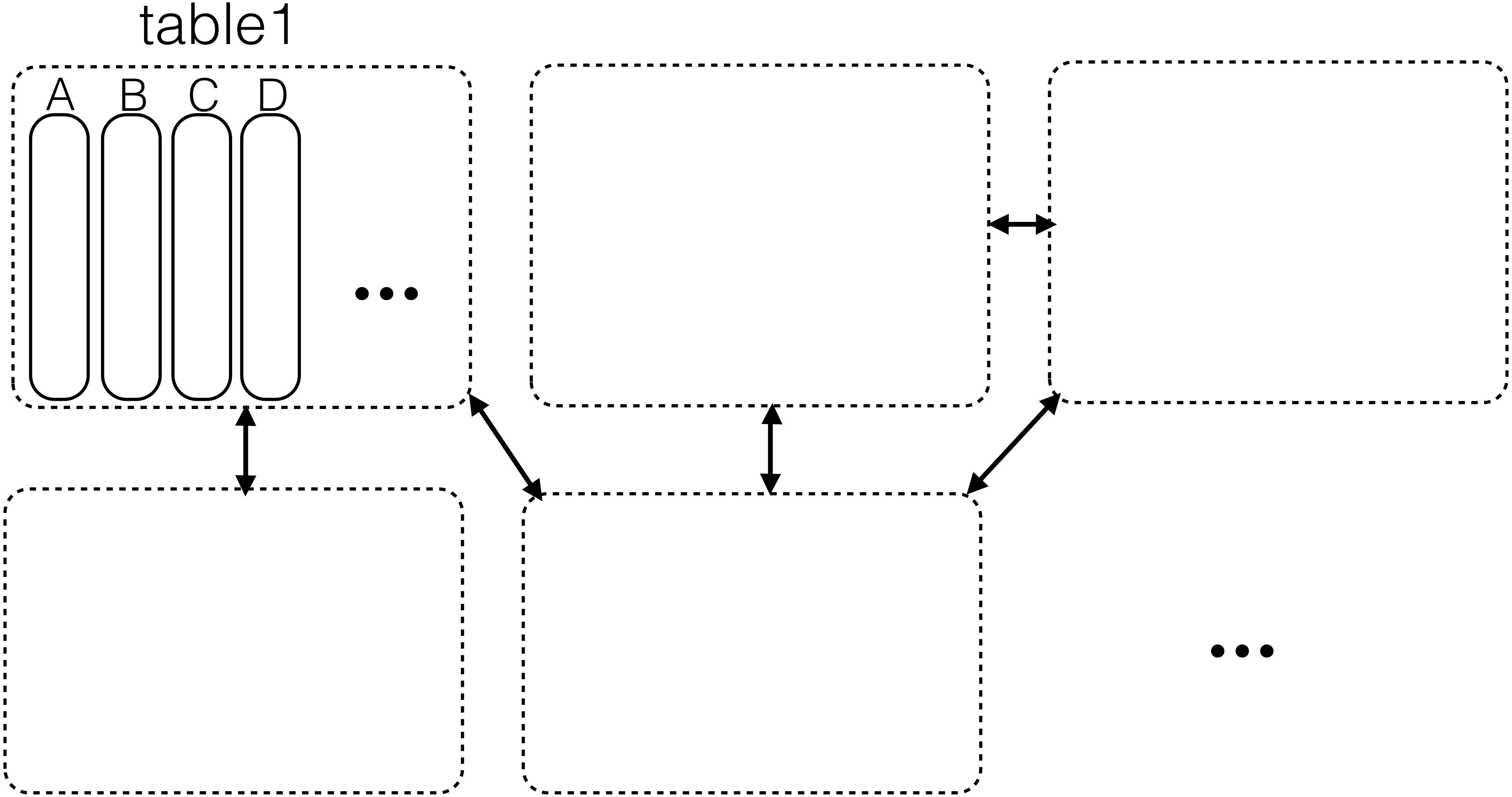
table1
A
B
C
D
...
...

select R.A from R where R.A>10 and R.A<14



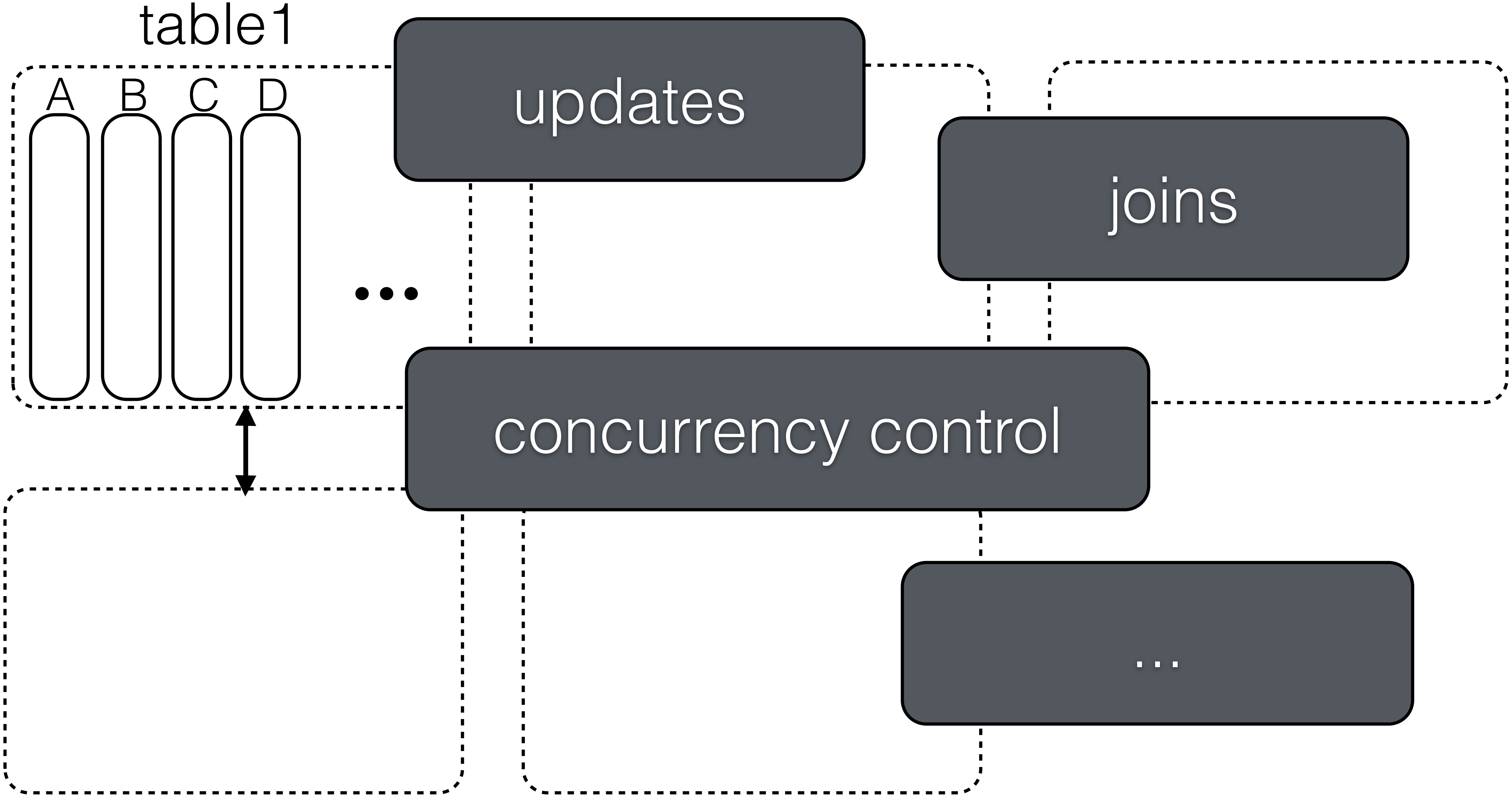
select R.A from R where R.A>10 and R.A<14

select max(R.A),max(R.B),max(S.A),max(S.B) from R,S
where v1 <R.C<v2 and v3 <R.D<v4
and v5 <R.E<v6 and k1 <S.C<k2 and k3 <S.D<k4 and k5 <S.E<k
and R.F = S.F

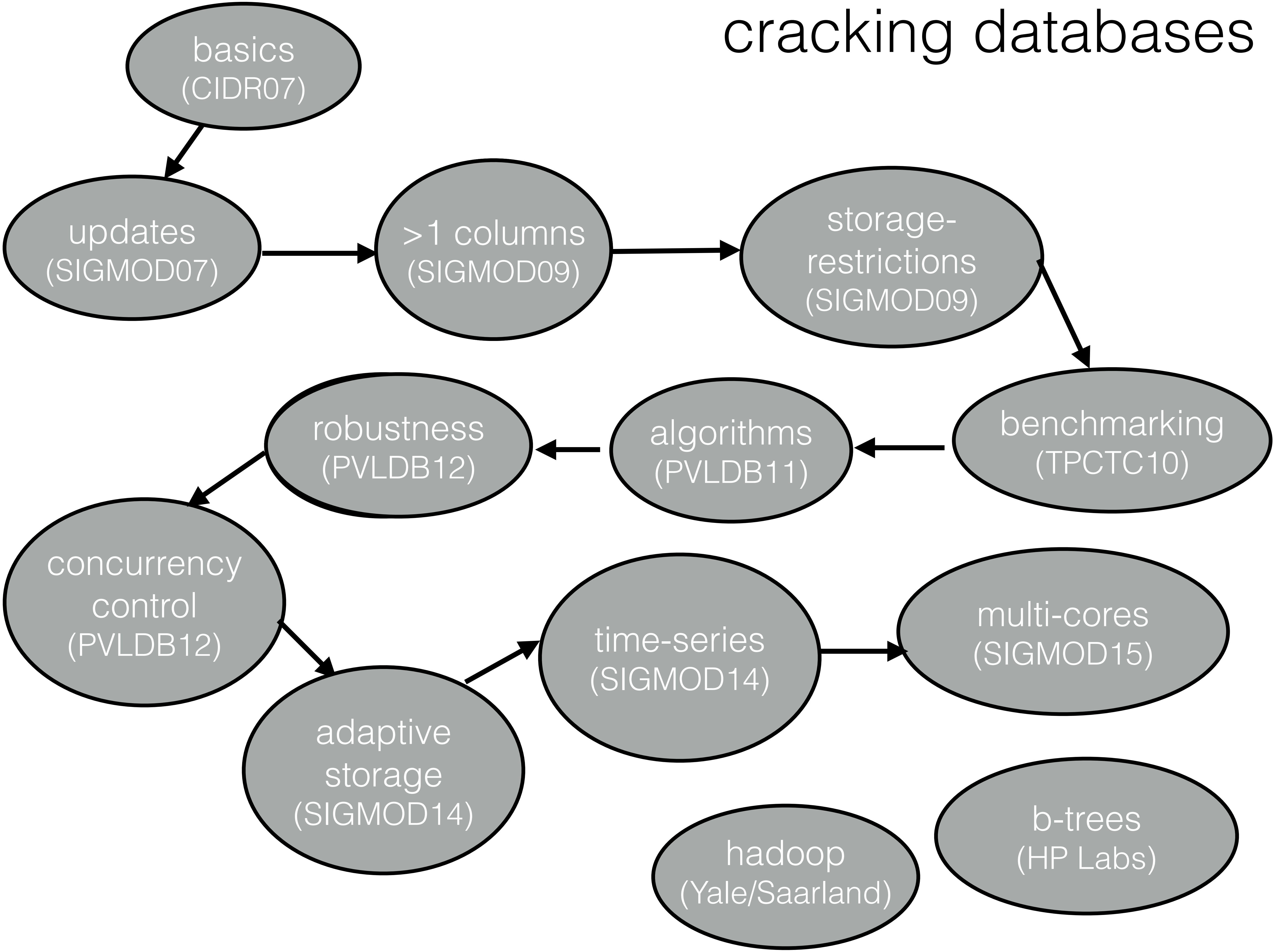


```
select R.A from R where R.A>10 and R.A<14
```

```
select max(R.A),max(R.B),max(S.A),max(S.B) from R,S
where v1 <R.C<v2 and v3 <R.D<v4
and v5 <R.E<v6 and k1 <S.C<k2 and k3 <S.D<k4 and k5 <S.E<k
and R.F = S.F
```



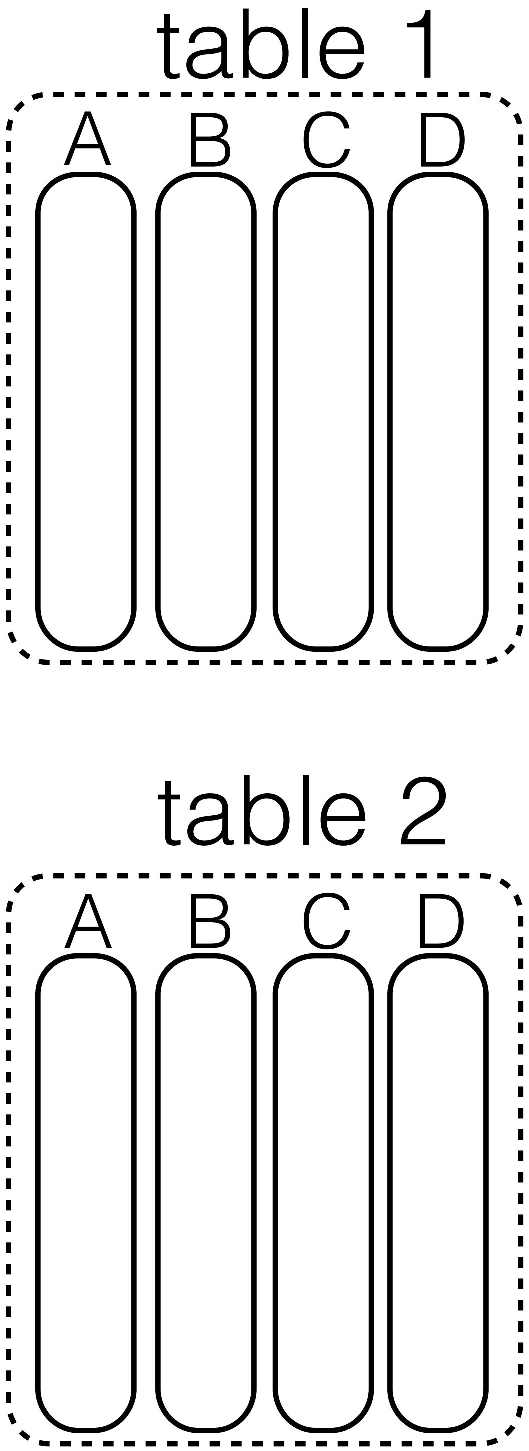
cracking databases



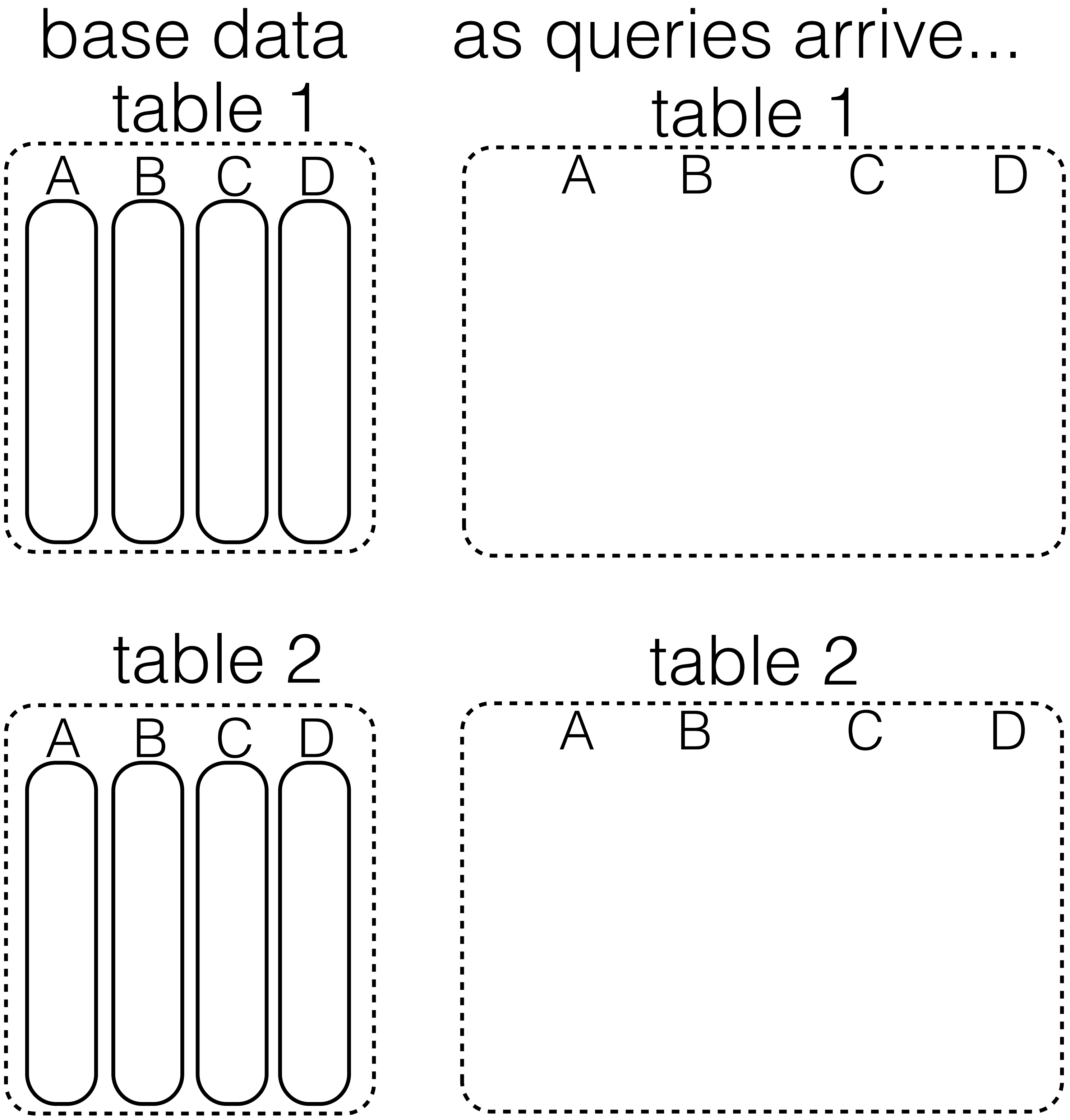
cracking tangram

base data

as queries arrive...



cracking tangram



base data

as queries arrive...

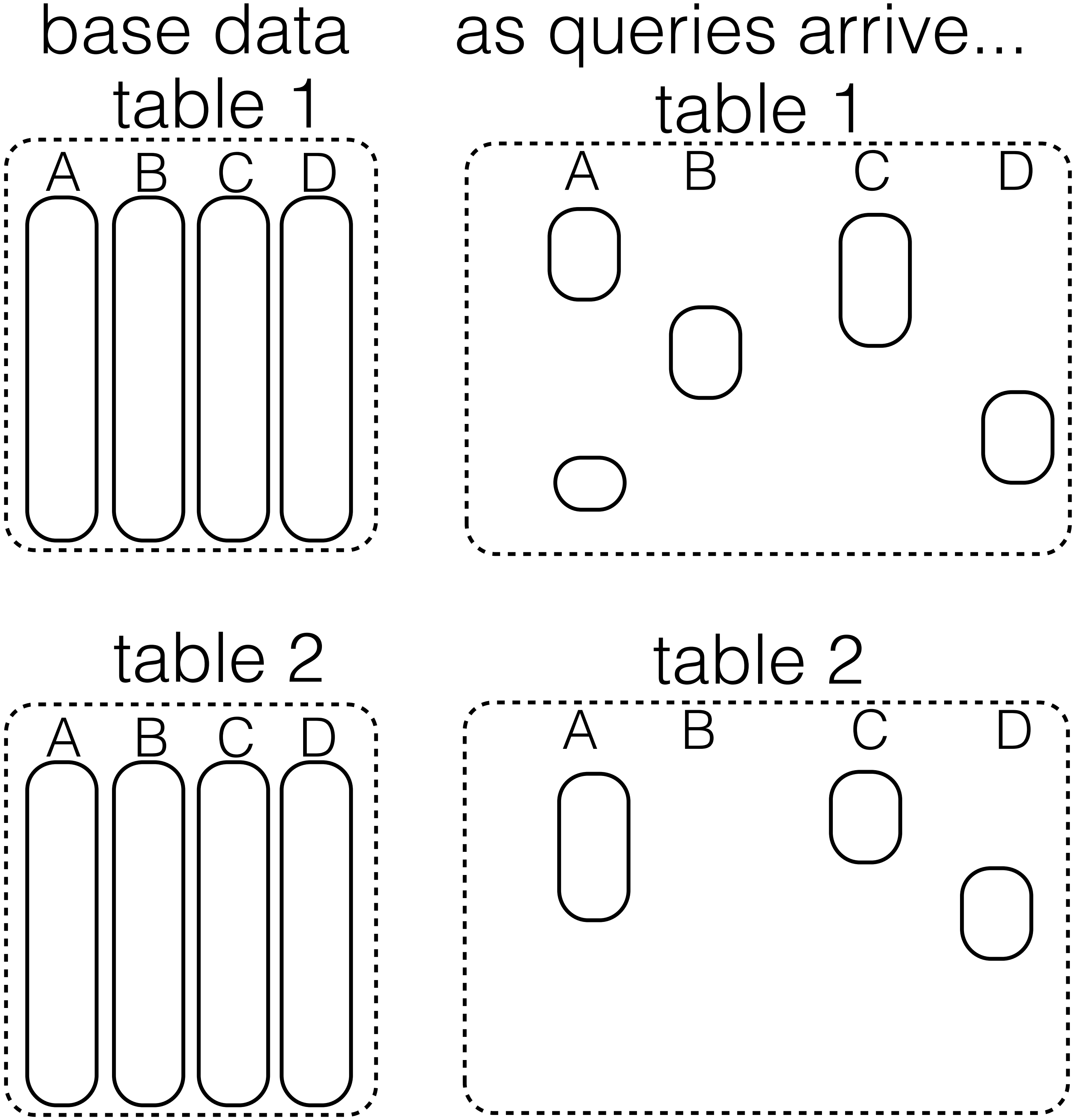
table 1

table 1

table 2

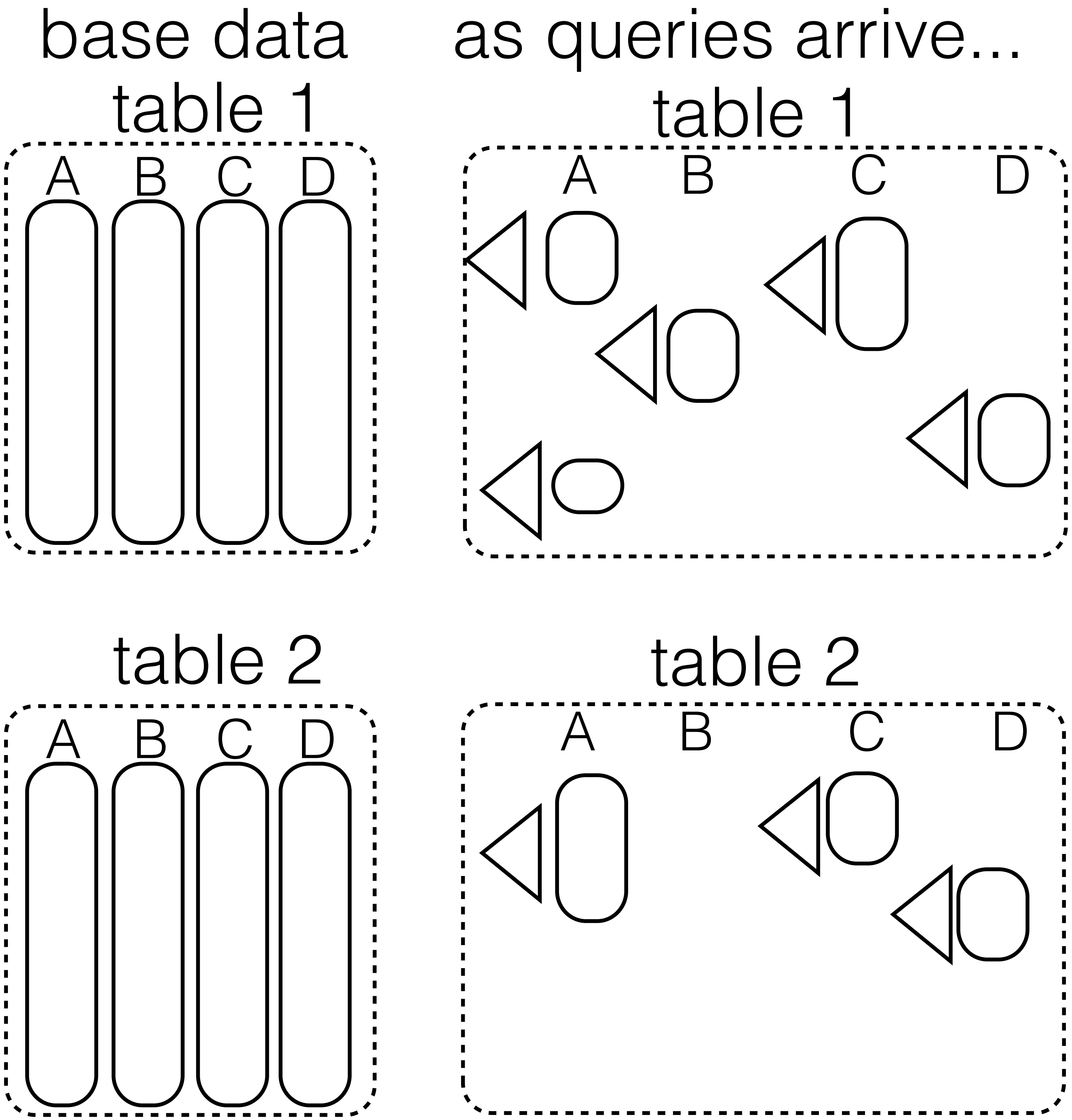
table 2

cracking tangram



partial materialization

cracking tangram



base data

as queries arrive...

table 1

table 1

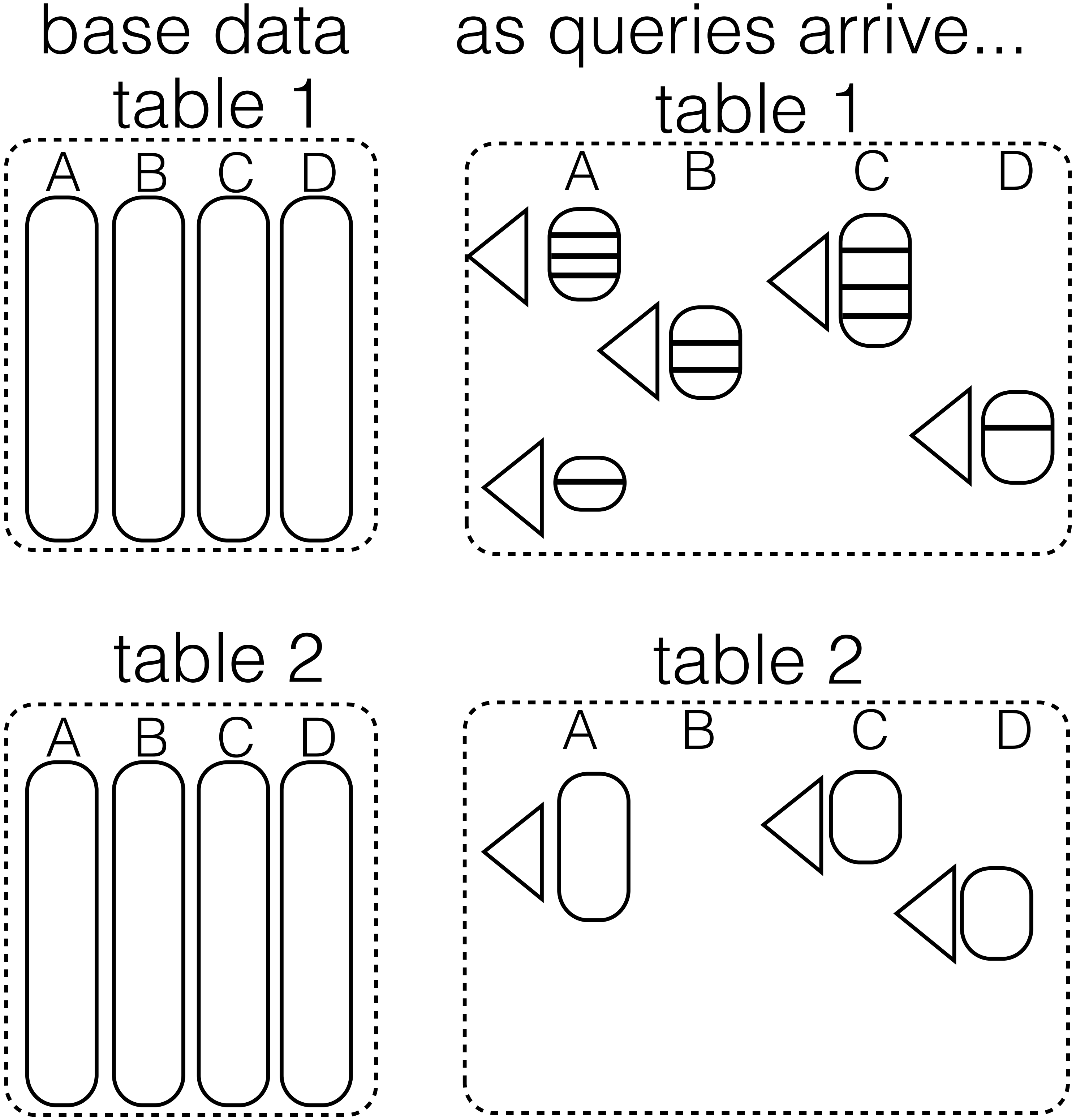
table 2

table 2

partial materialization

partial indexing

cracking tangram


base data

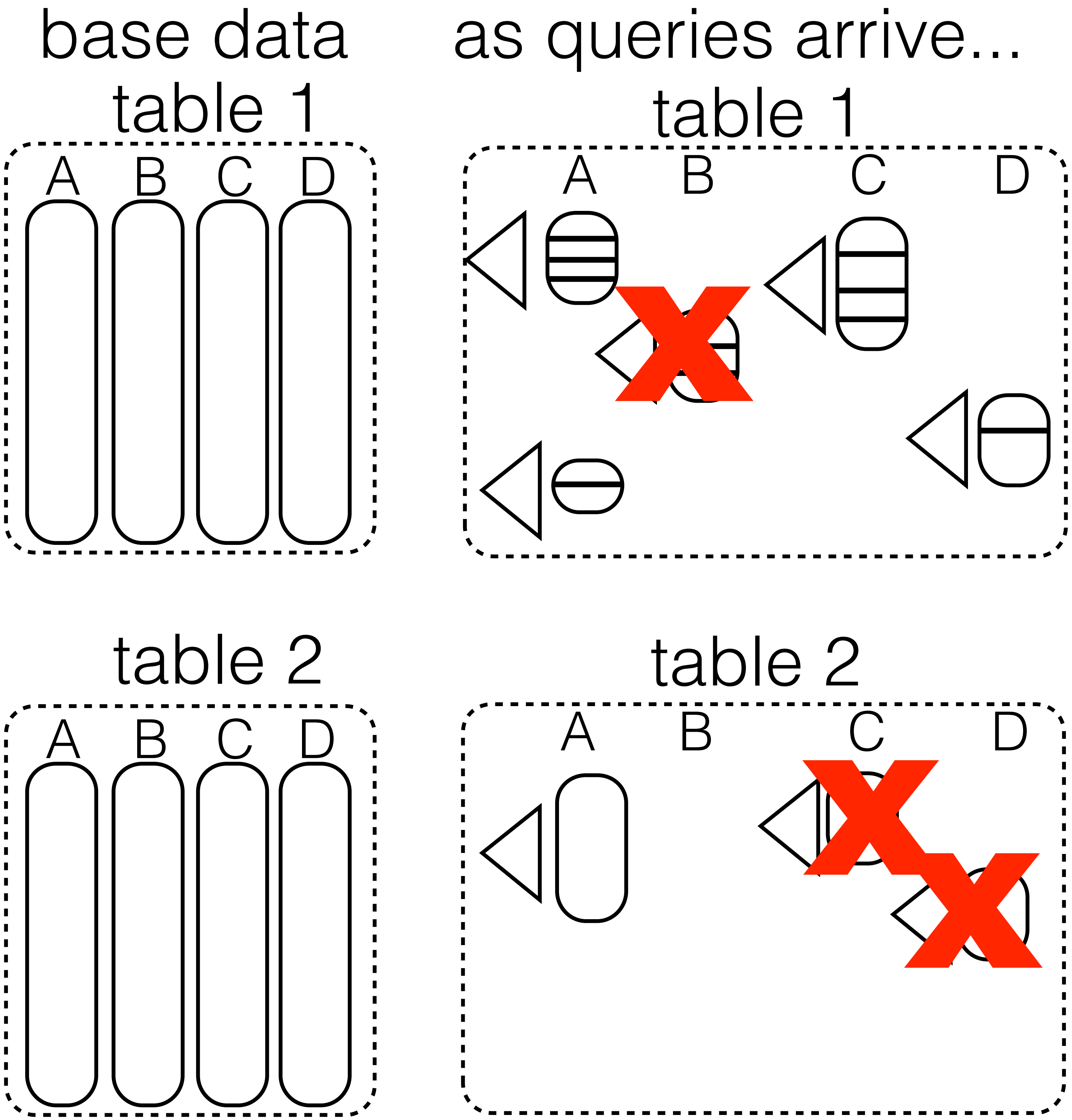
as queries arrive...

partial materialization

partial indexing

continuous adaptation

cracking tangram



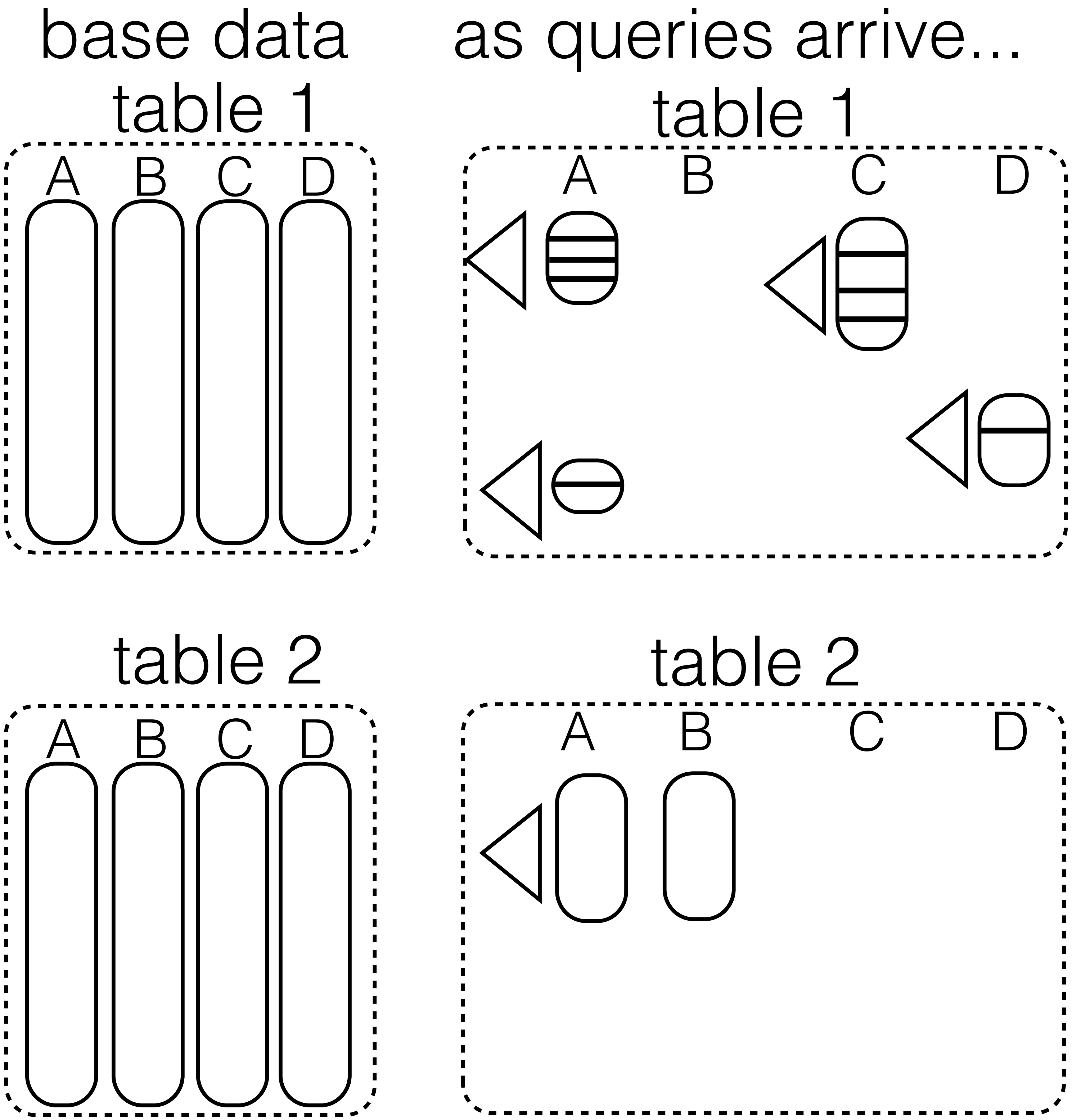
partial materialization

partial indexing

continuous adaptation

storage adaptation

cracking tangram



base data

table 1

A B C D

table 2

A B C D

as queries arrive...

table 1

A B C D

table 2

A B C D

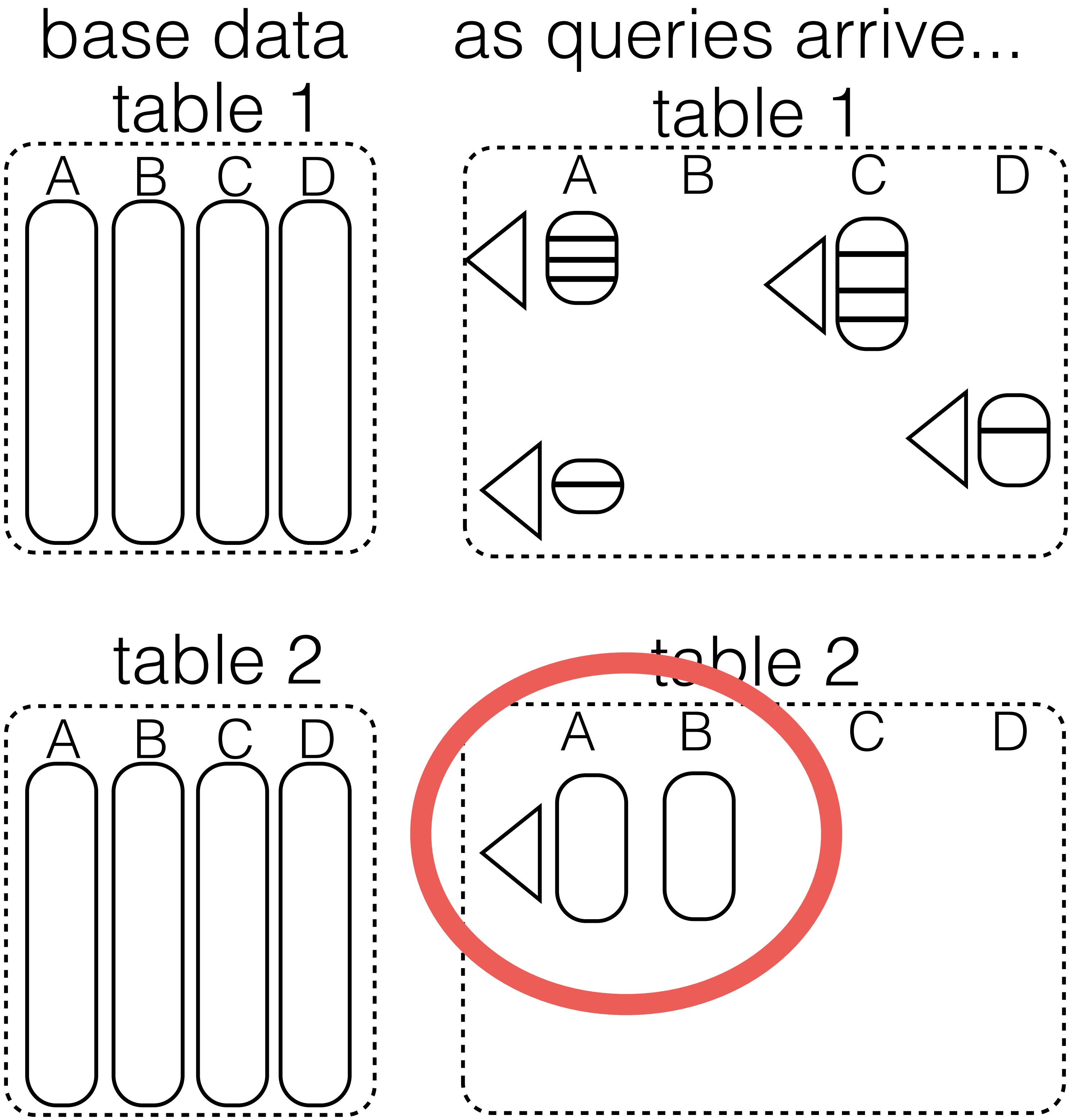
partial materialization

partial indexing

continuous adaptation

storage adaptation

cracking tangram



partial materialization

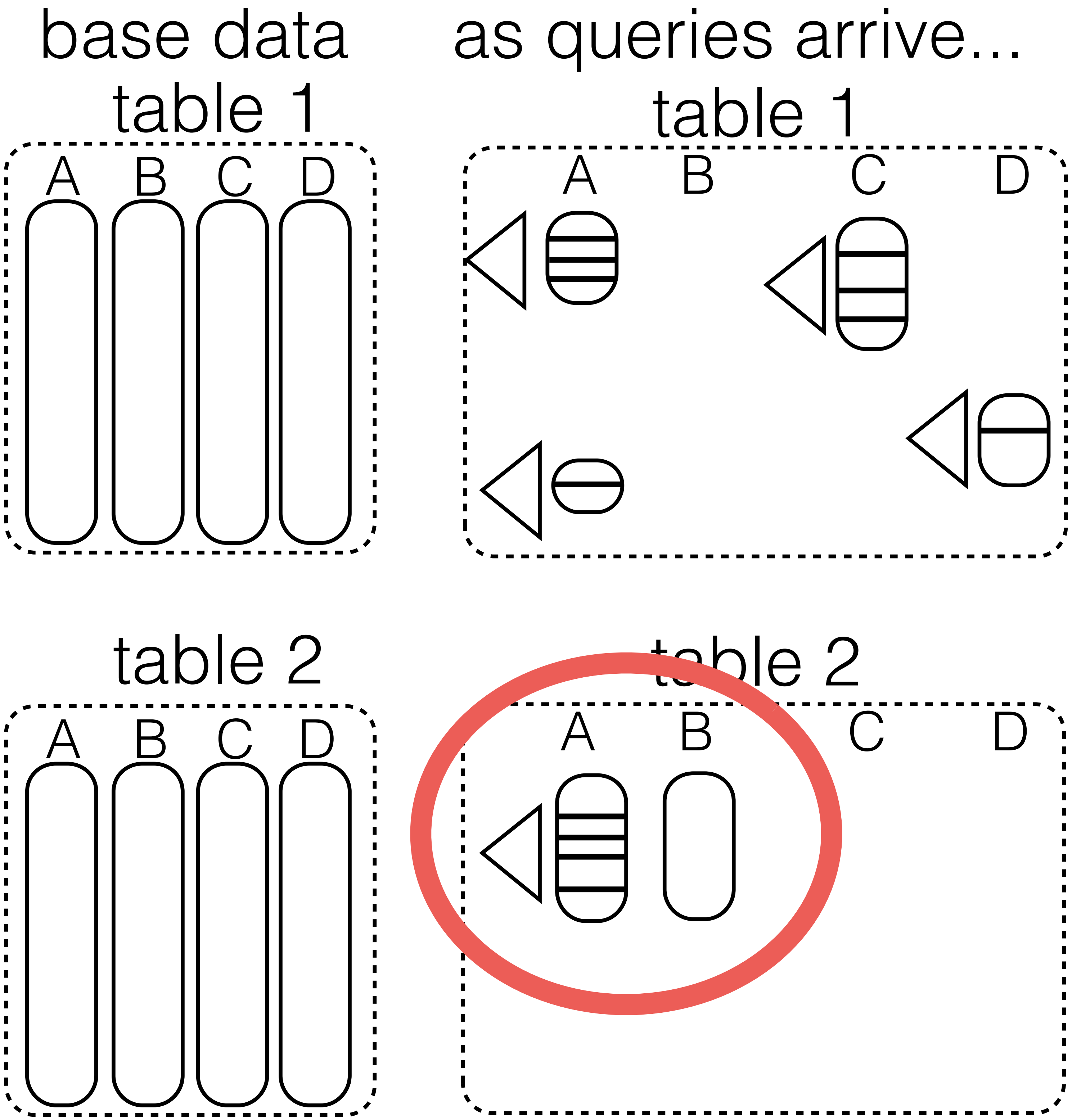
partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

cracking tangram



partial materialization

partial indexing

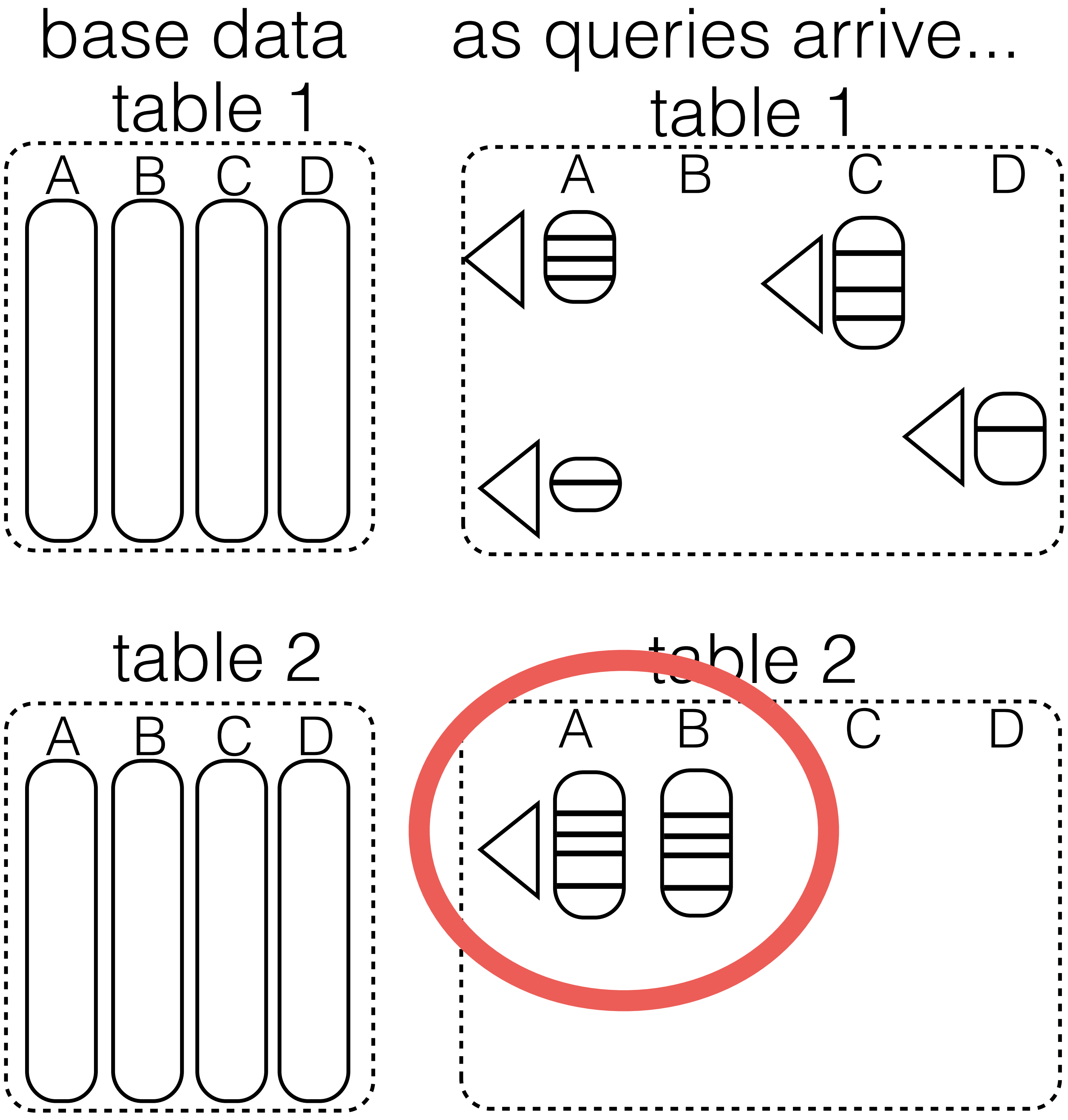
continuous adaptation

storage adaptation

no tuple reconstruction

adaptive alignment

cracking tangram



partial materialization

partial indexing

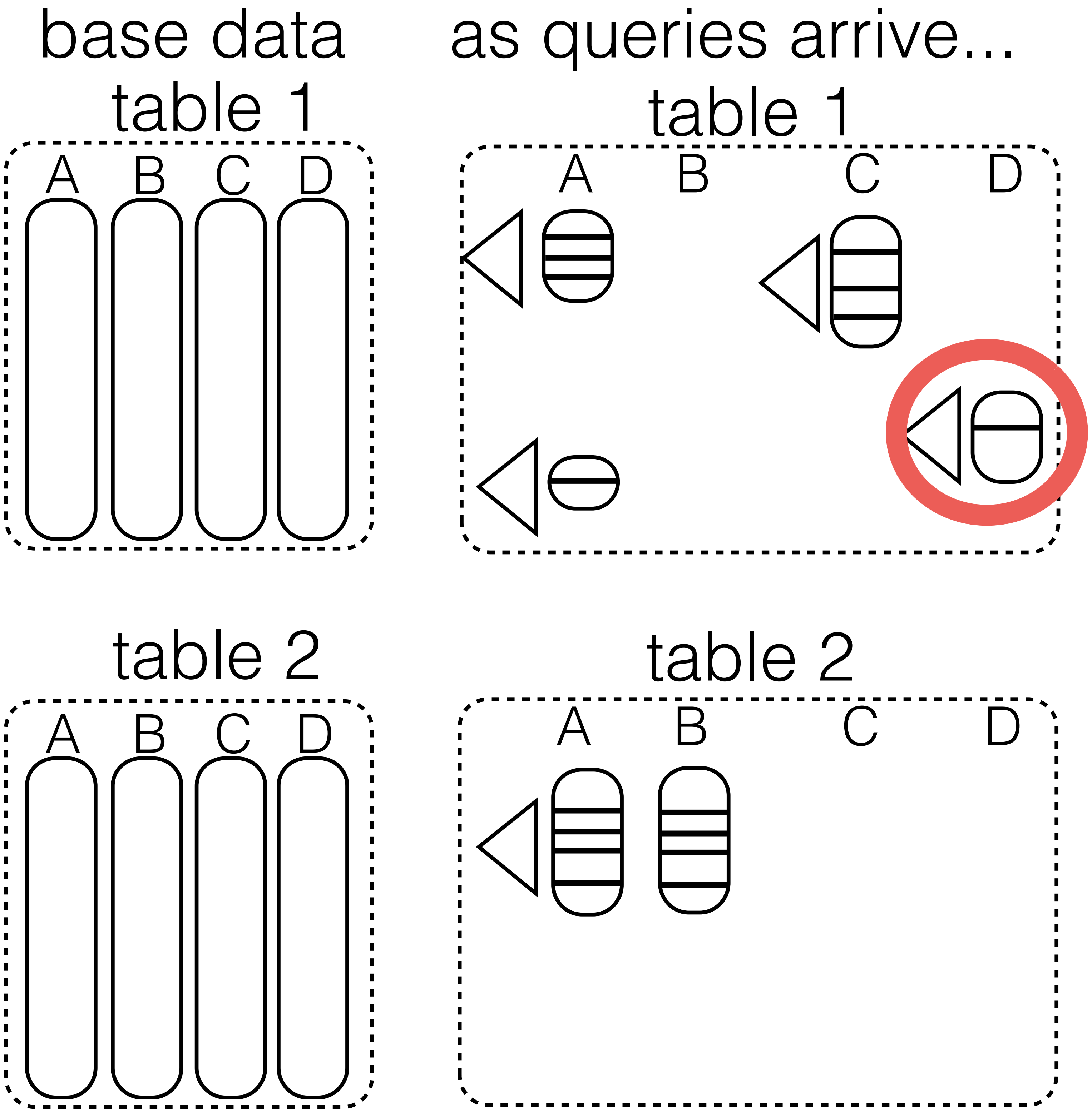
continuous adaptation

storage adaptation

no tuple reconstruction

adaptive alignment

cracking tangram



partial materialization

partial indexing

continuous adaptation

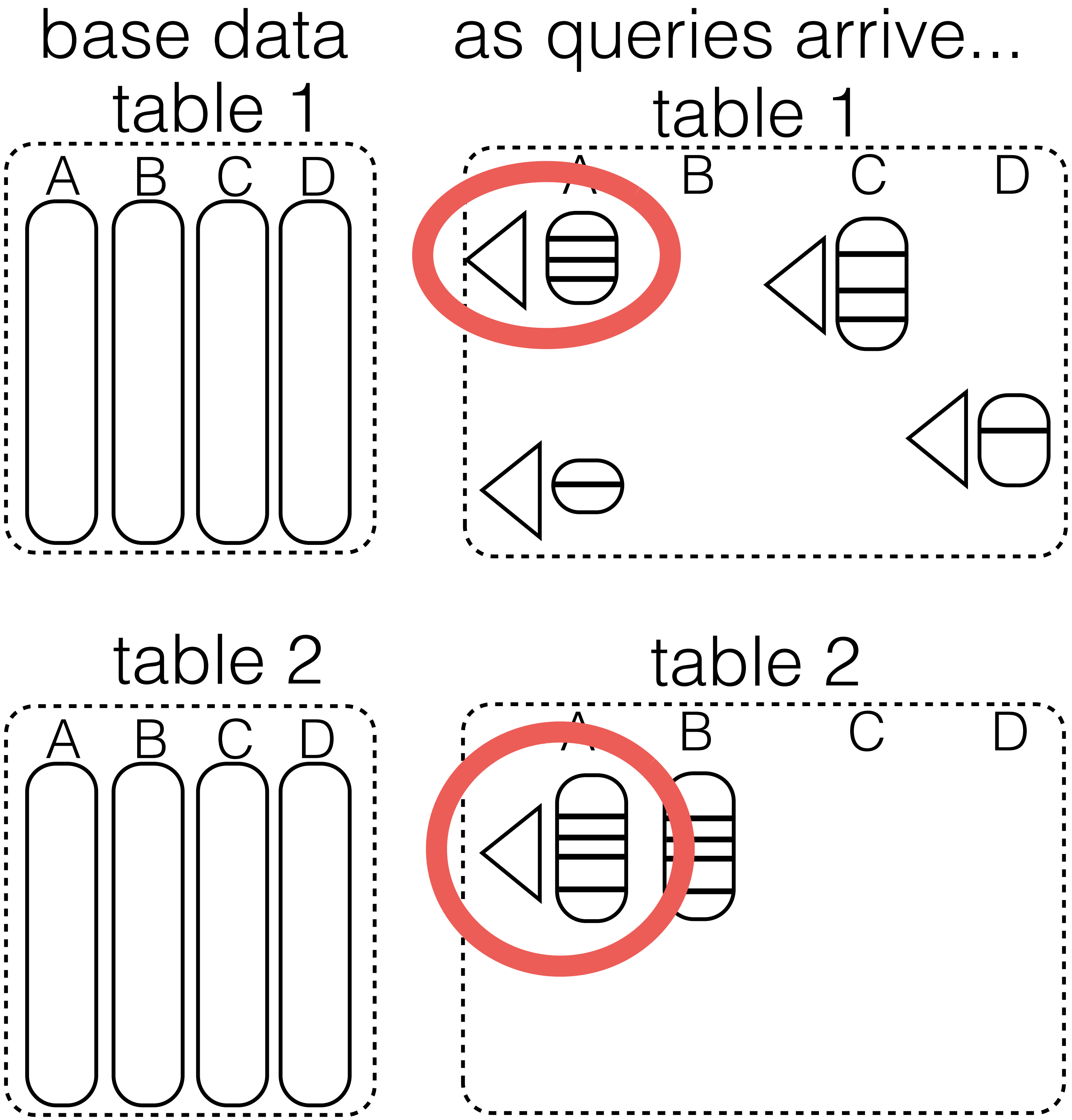
storage adaptation

no tuple reconstruction

adaptive alignment

sort in caches

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

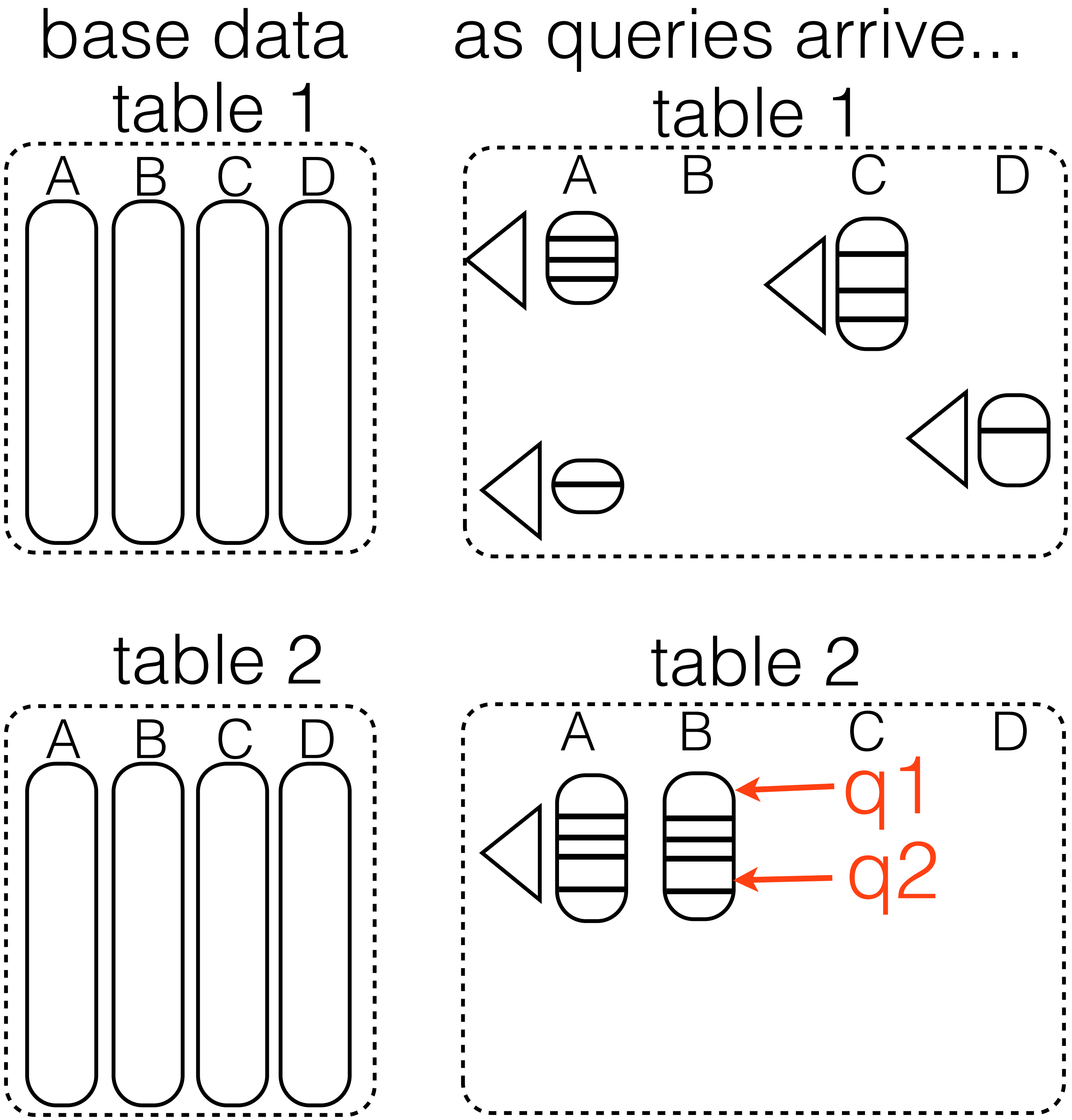
no tuple reconstruction

adaptive alignment

sort in caches

crack joins

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

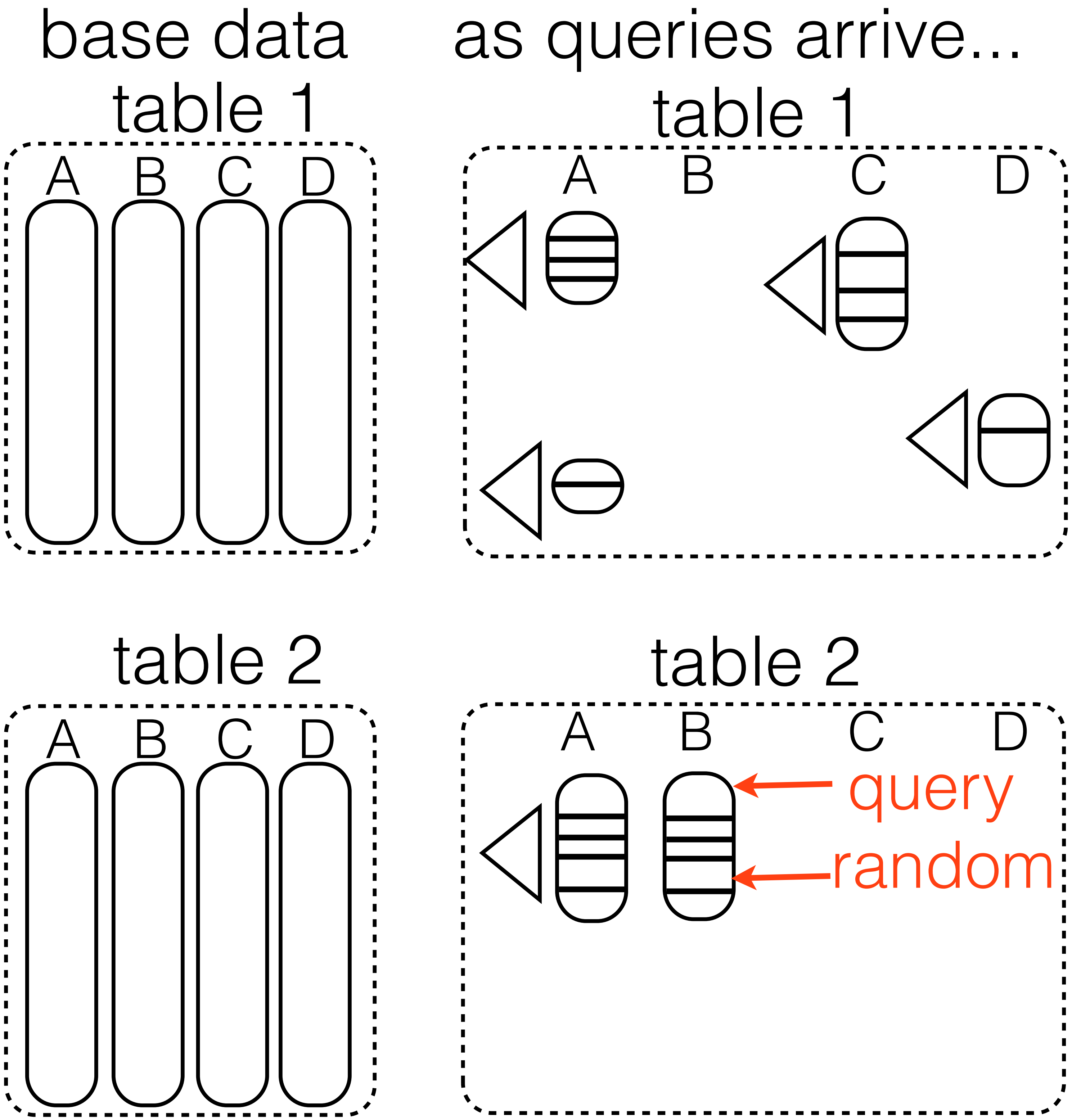
adaptive alignment

sort in caches

crack joins

lightweight locking

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

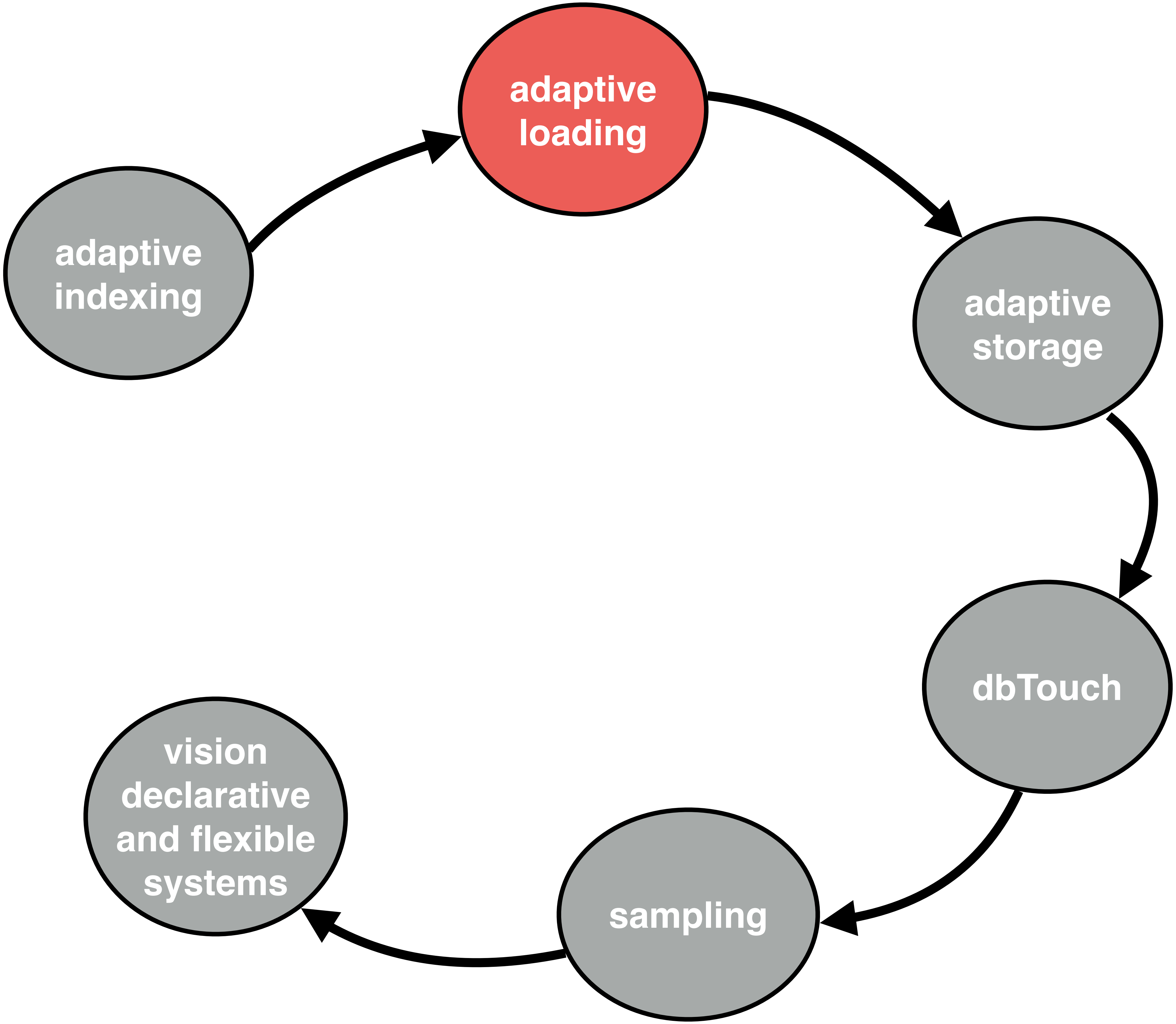
adaptive alignment

sort in caches

crack joins

lightweight locking

stochastic cracking



adaptive indexing

adaptive loading

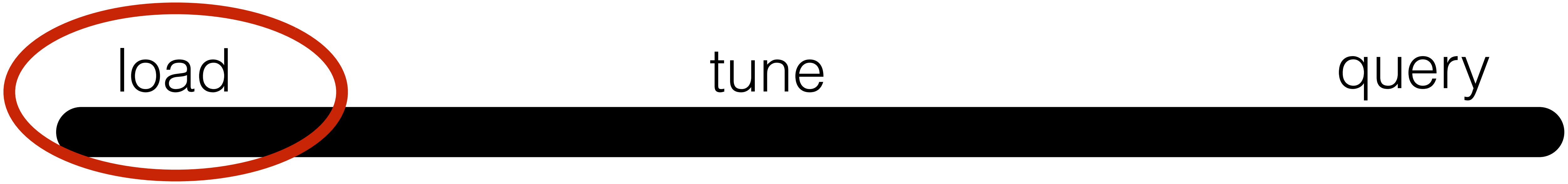
adaptive storage

dbTouch

sampling

vision declarative and flexible systems

loading



load

tune

query

copy data inside the database

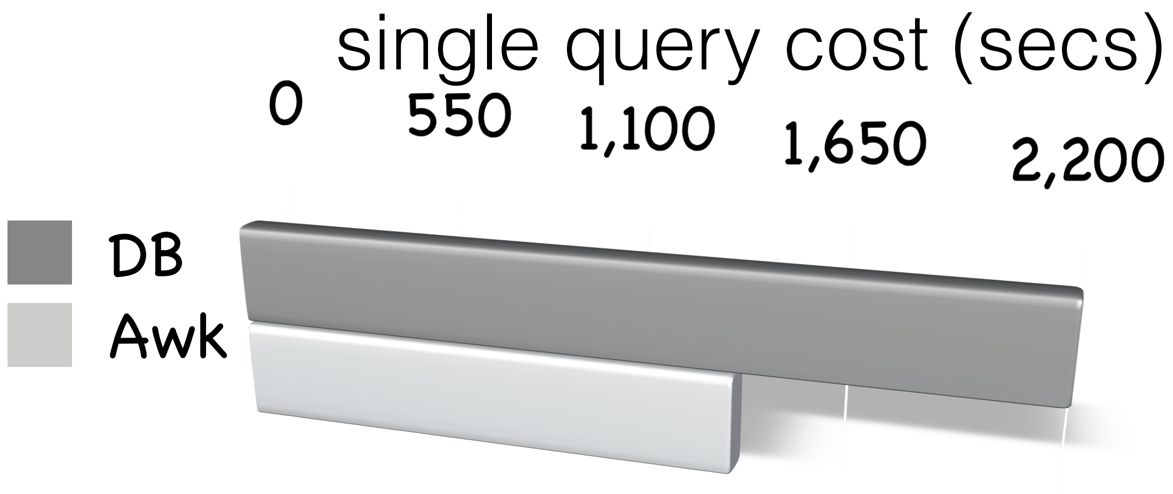
database now has full control

slow process…
not all data might be needed all the time

Adaptive Loading, CIDR 11

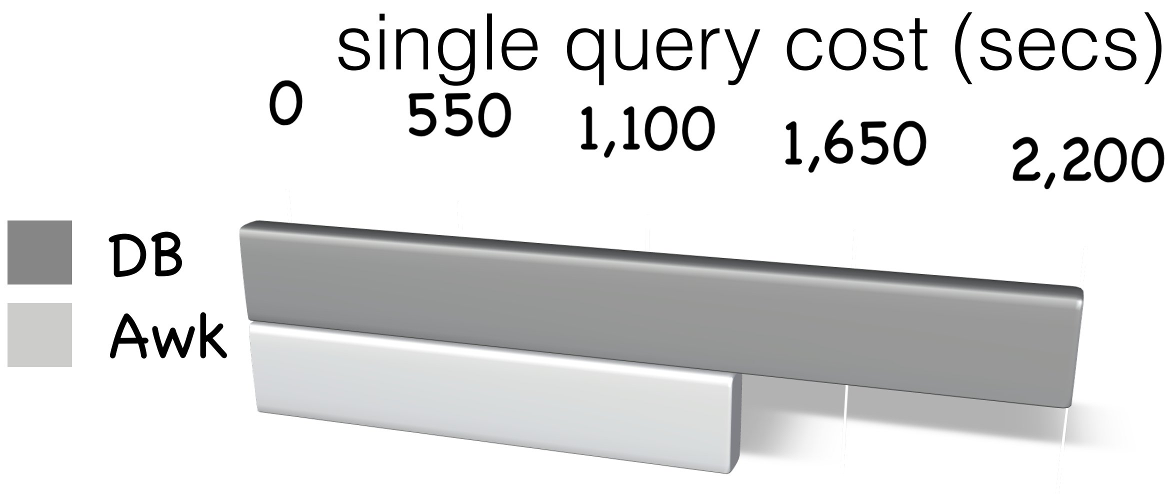
database vs. unix tools

1 file, 4 attributes,
1 billion tuples



database vs. unix tools

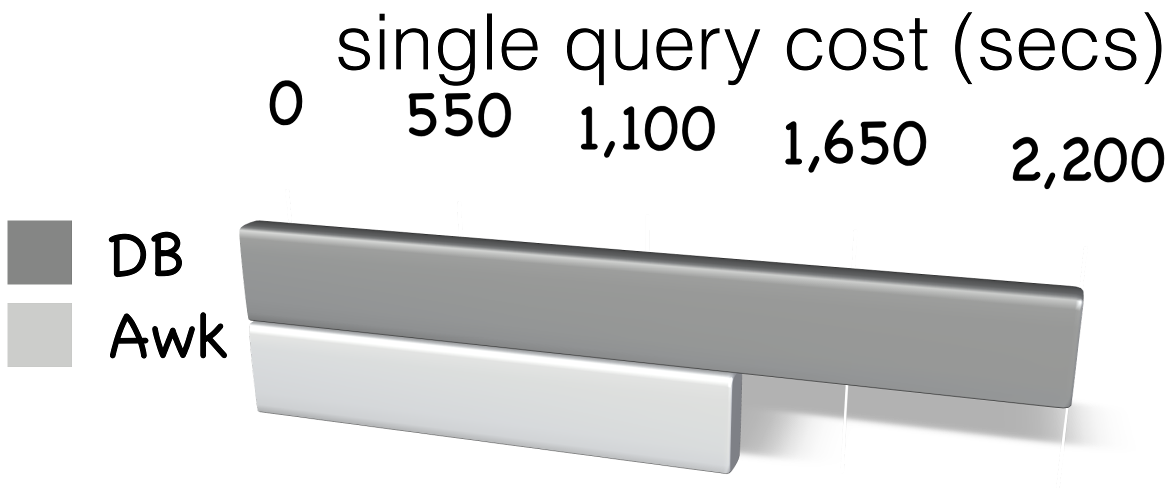
1 file, 4 attributes,
1 billion tuples



break down db cost

database vs. unix tools

1 file, 4 attributes,
1 billion tuples

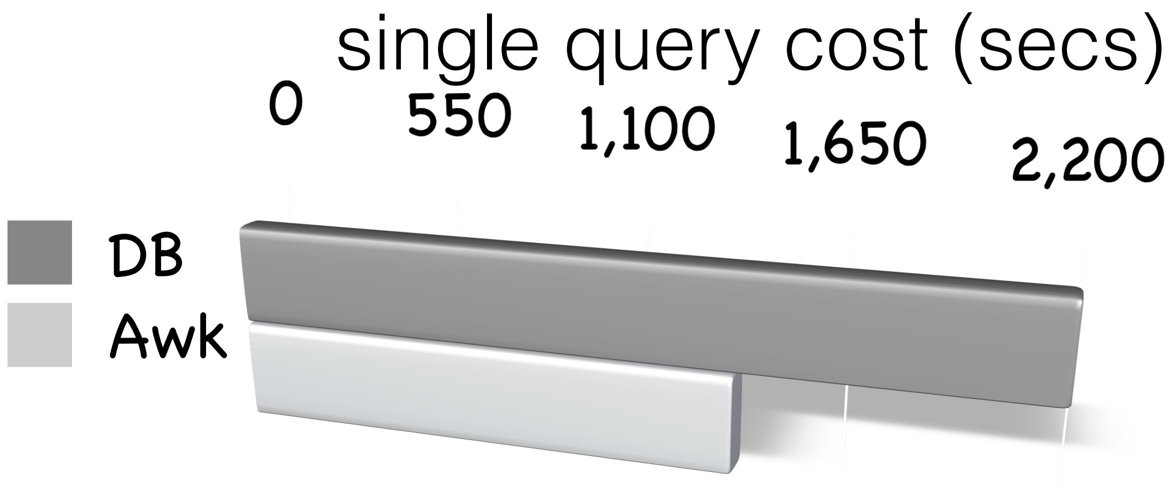


break down db cost

loading is a major bottleneck

database vs. unix tools

1 file, 4 attributes,
1 billion tuples



break down db cost

loading is a major bottleneck

but writing/maintaining scripts does not scale

adaptive loading

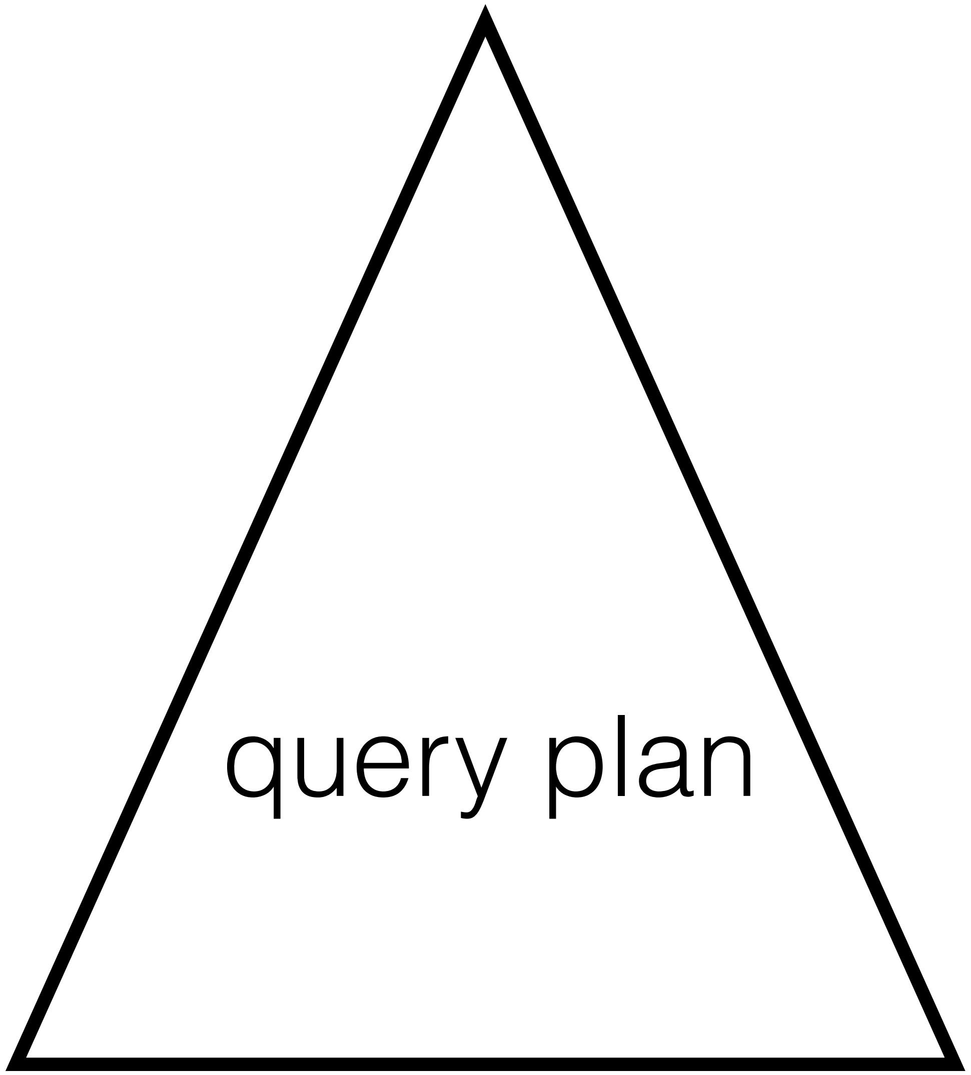
load/touch only what is needed and only when it is needed

Adaptive Loading, CIDR 11

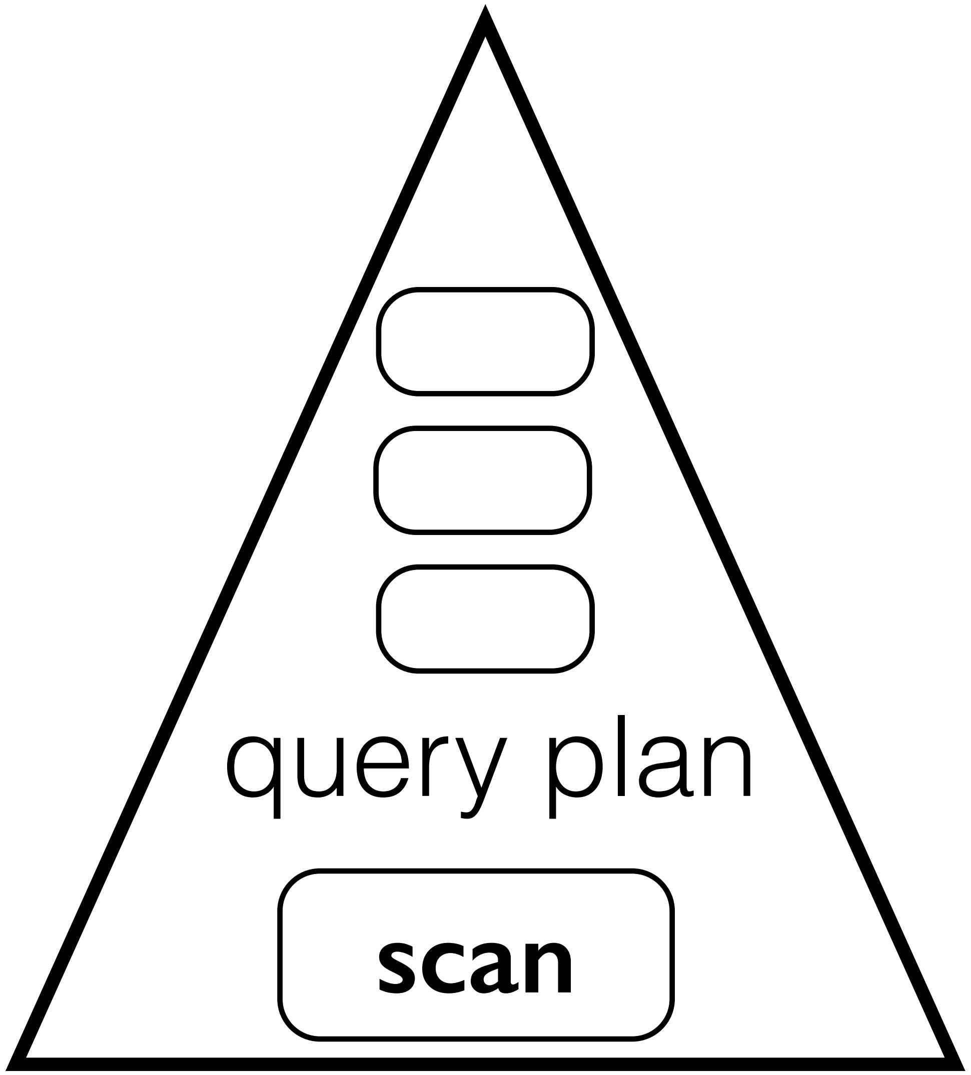
but raw data access is expensive

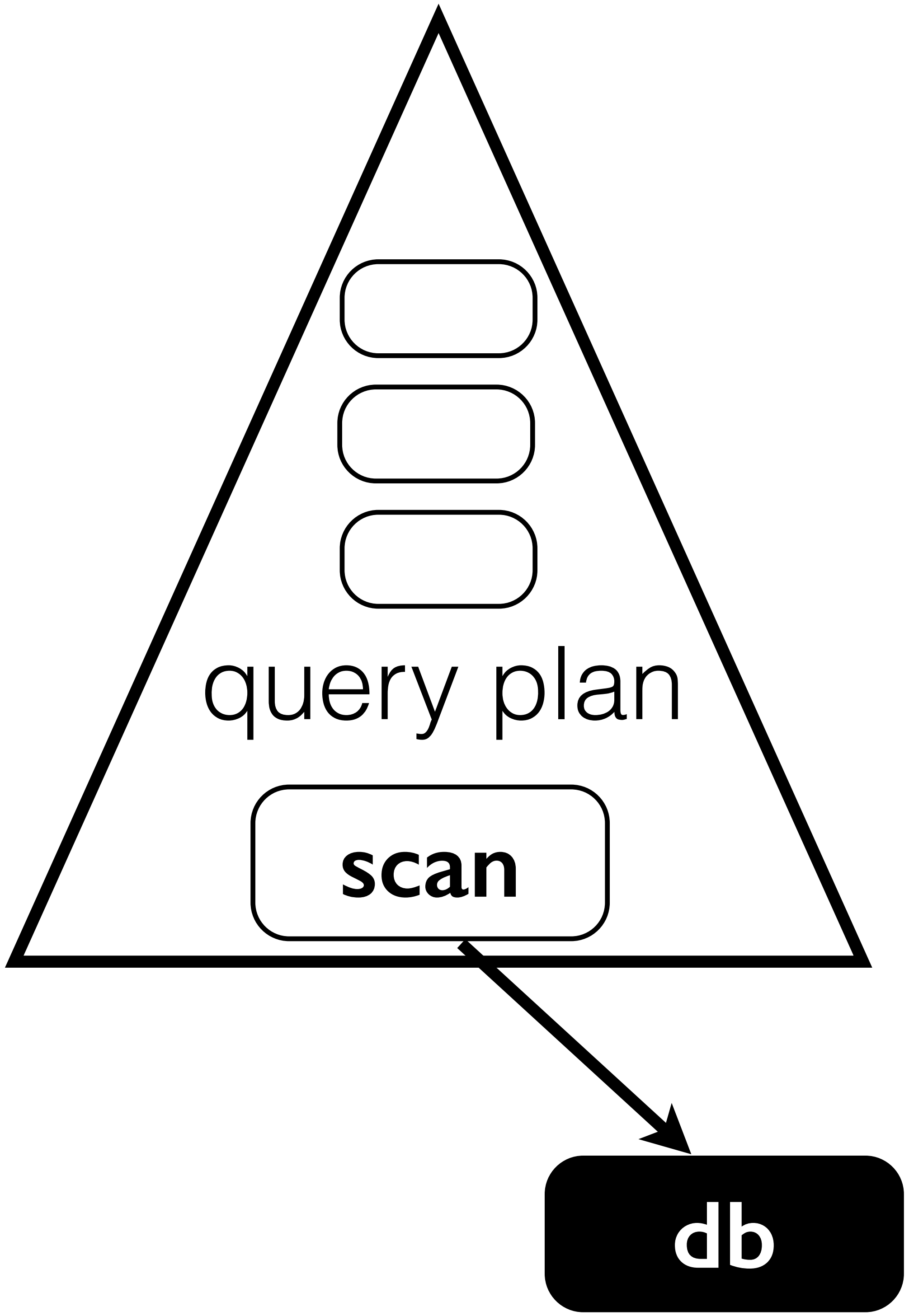
tokenizing - parsing - no indexing - no statistics

challenge: fast raw data access

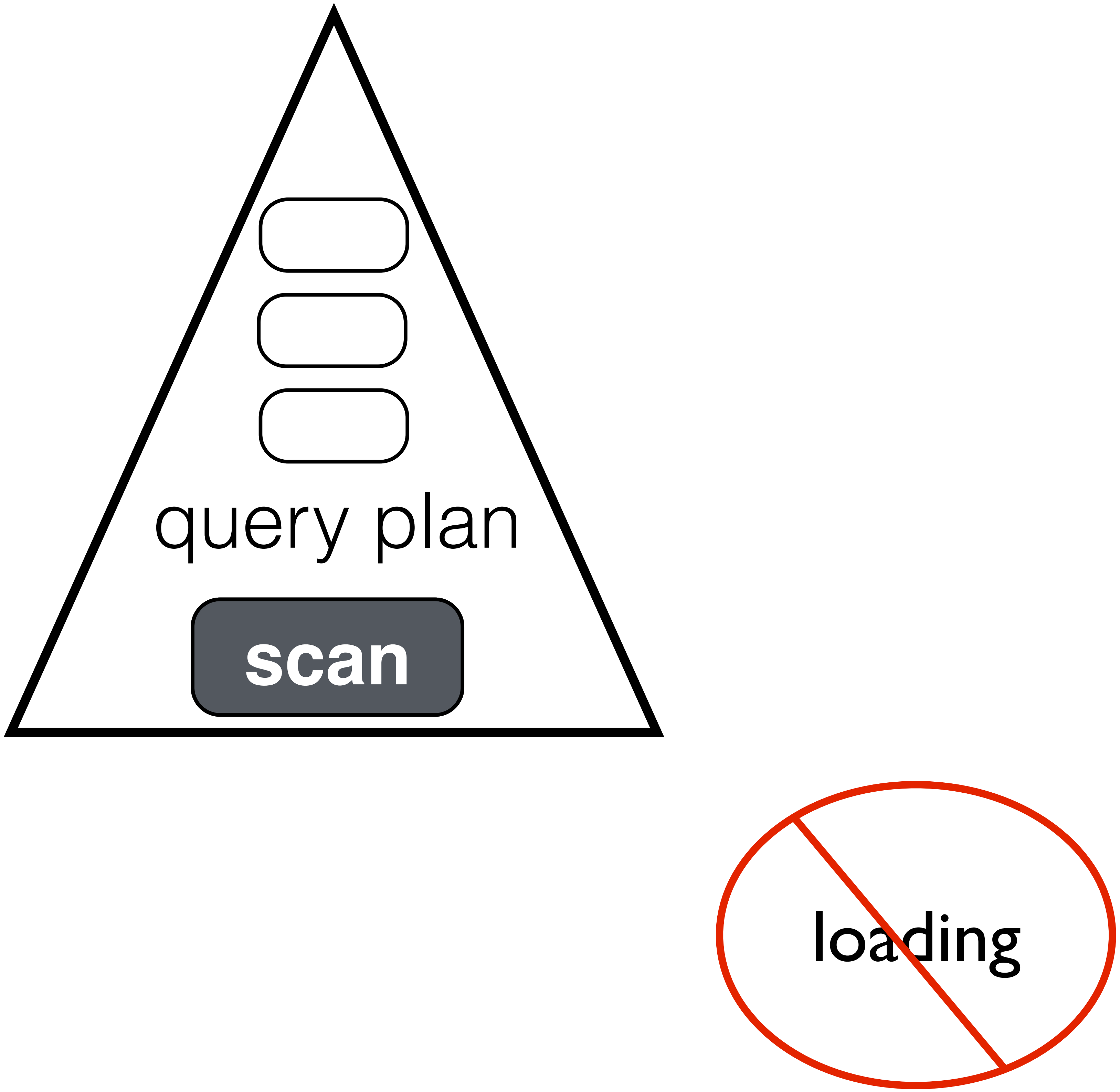


query plan


query plan
scan



Adaptive Loading, CIDR 11

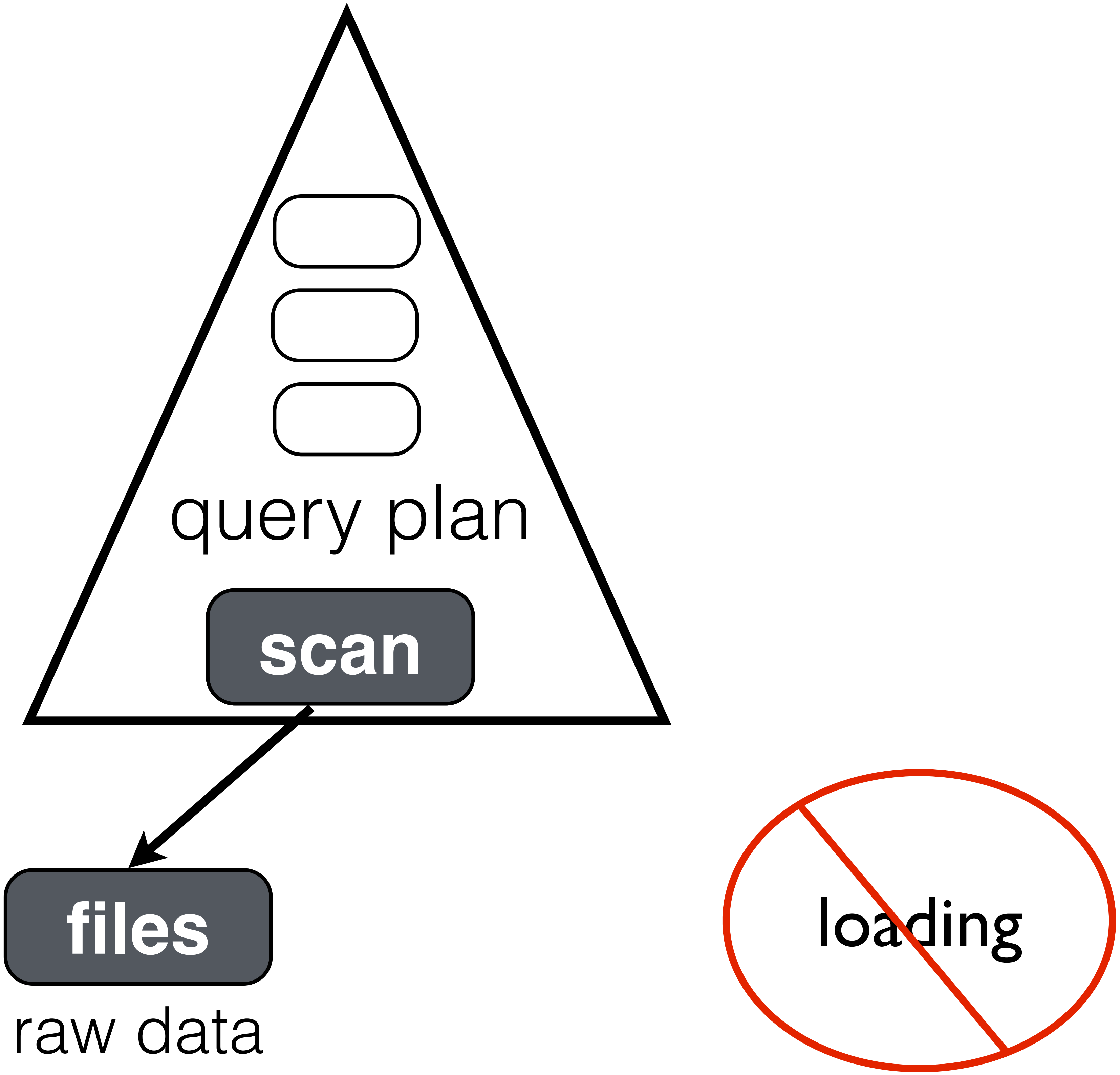


query plan

scan

loading

Adaptive Loading, CIDR 11



query plan

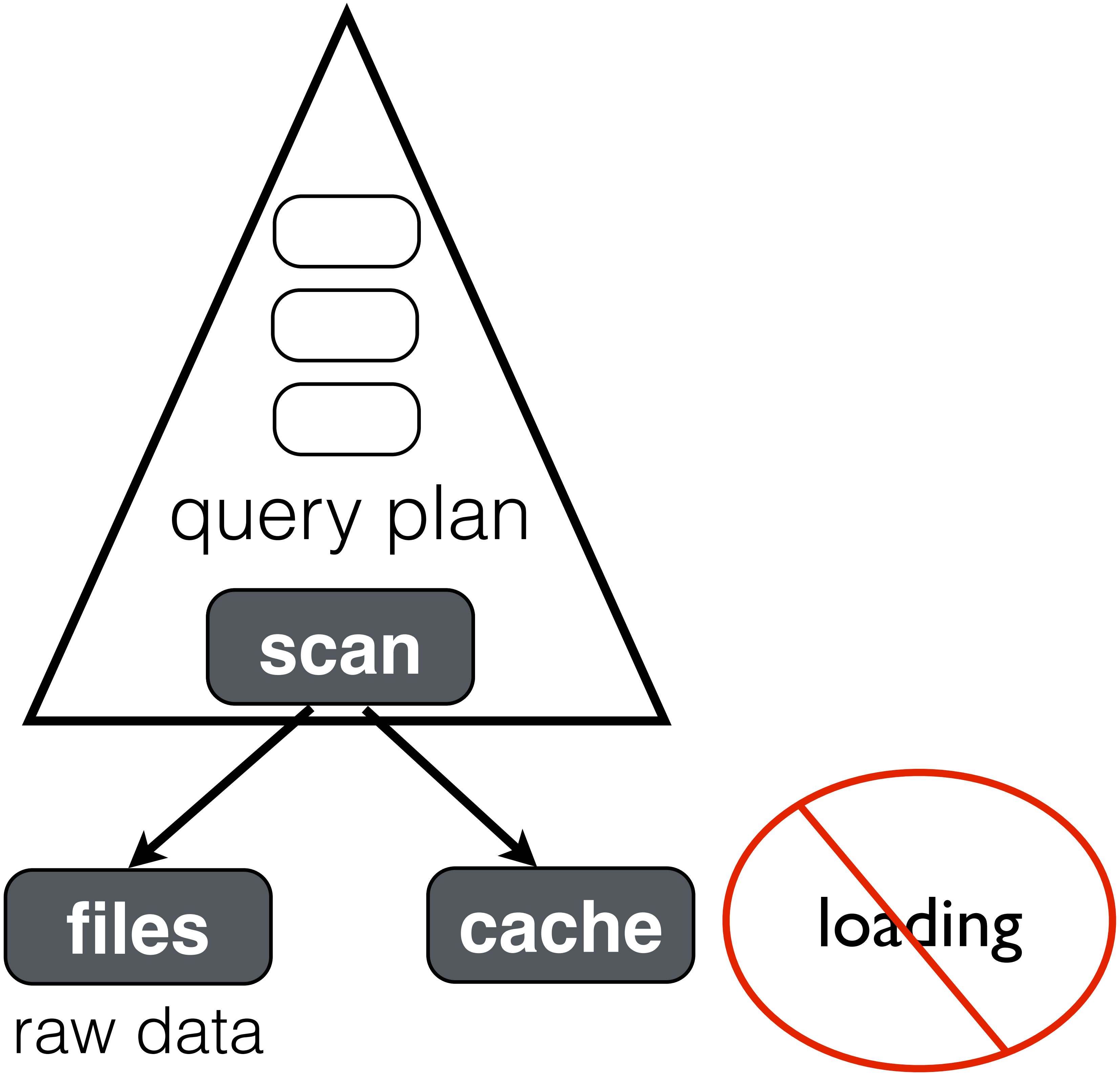
scan

files

~~loading~~

access raw data
adaptively on-the-fly

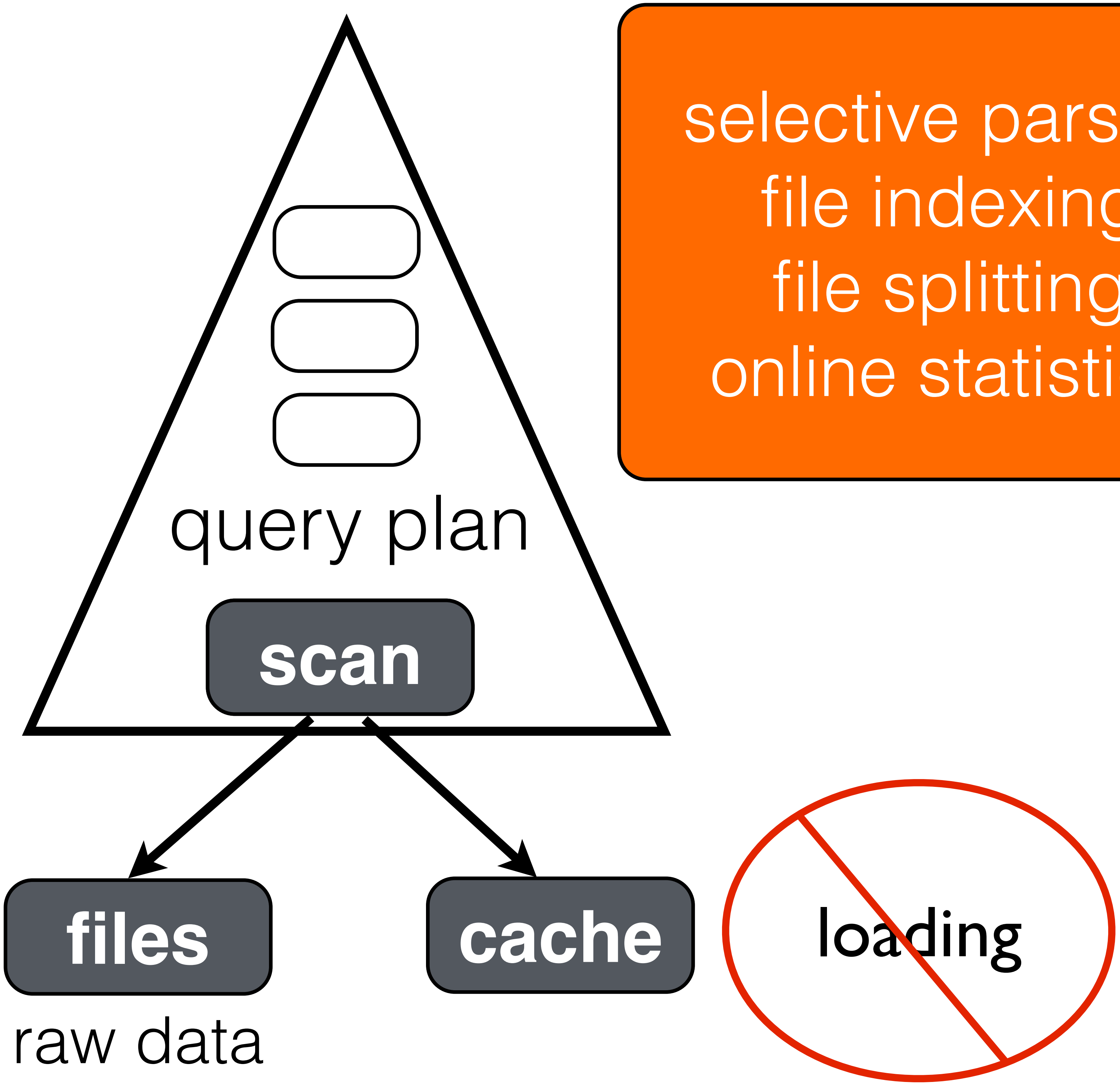
Adaptive Loading, CIDR 11



access raw data
adaptively on-the-fly

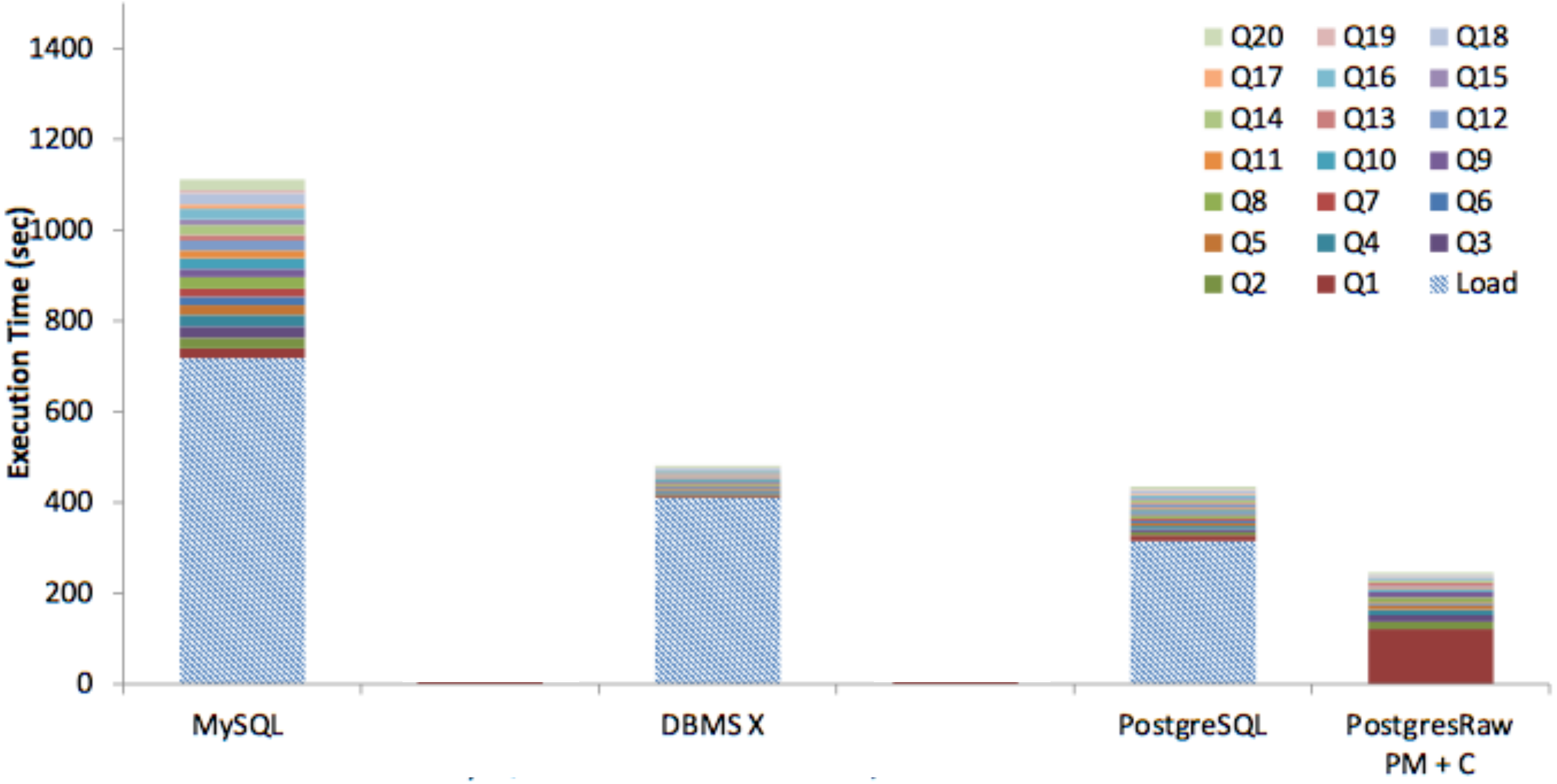
Adaptive Loading, CIDR 11

selective parsing
file indexing
file splitting
online statistics

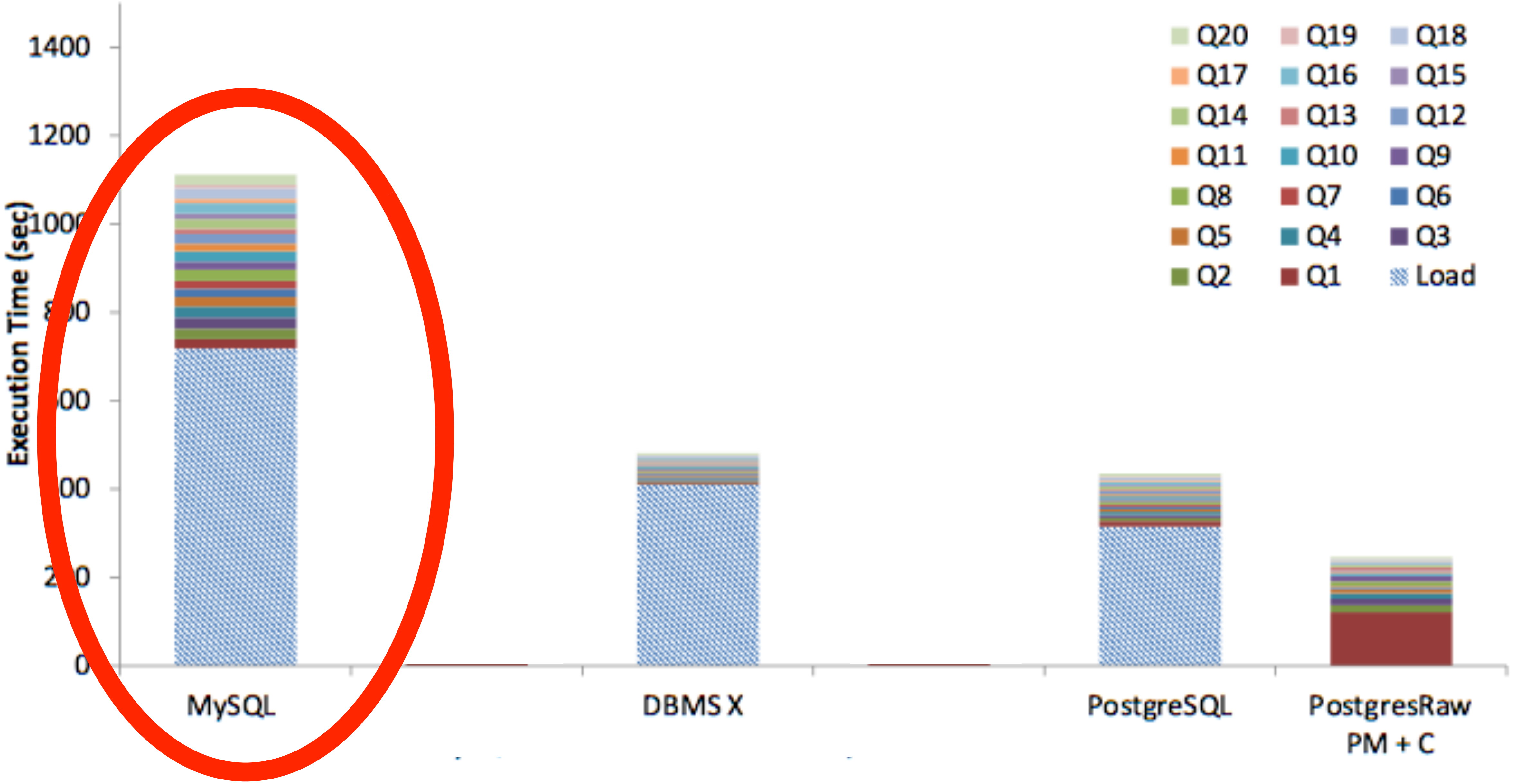


access raw data
adaptively on-the-fly

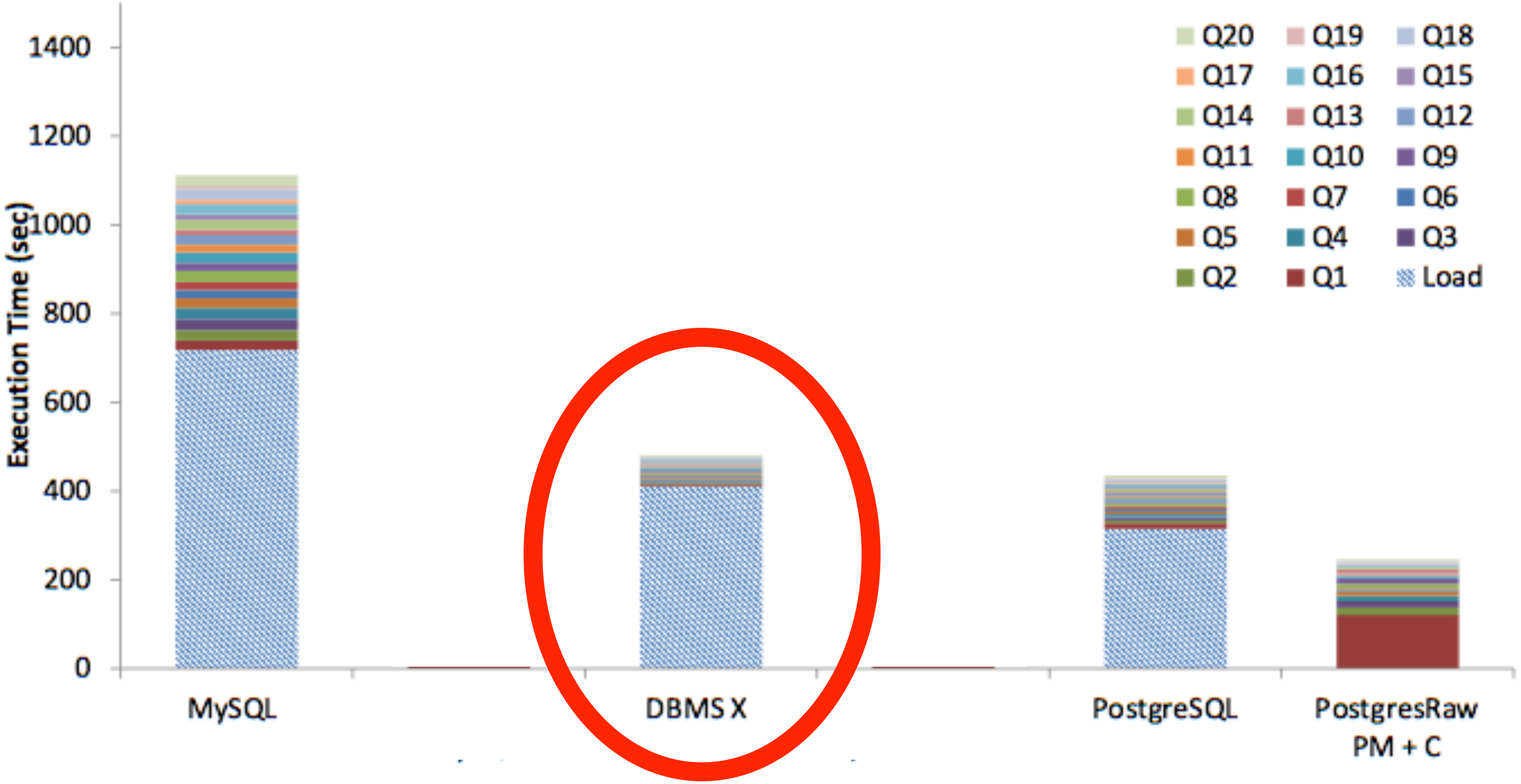
Adaptive Loading, CIDR 11



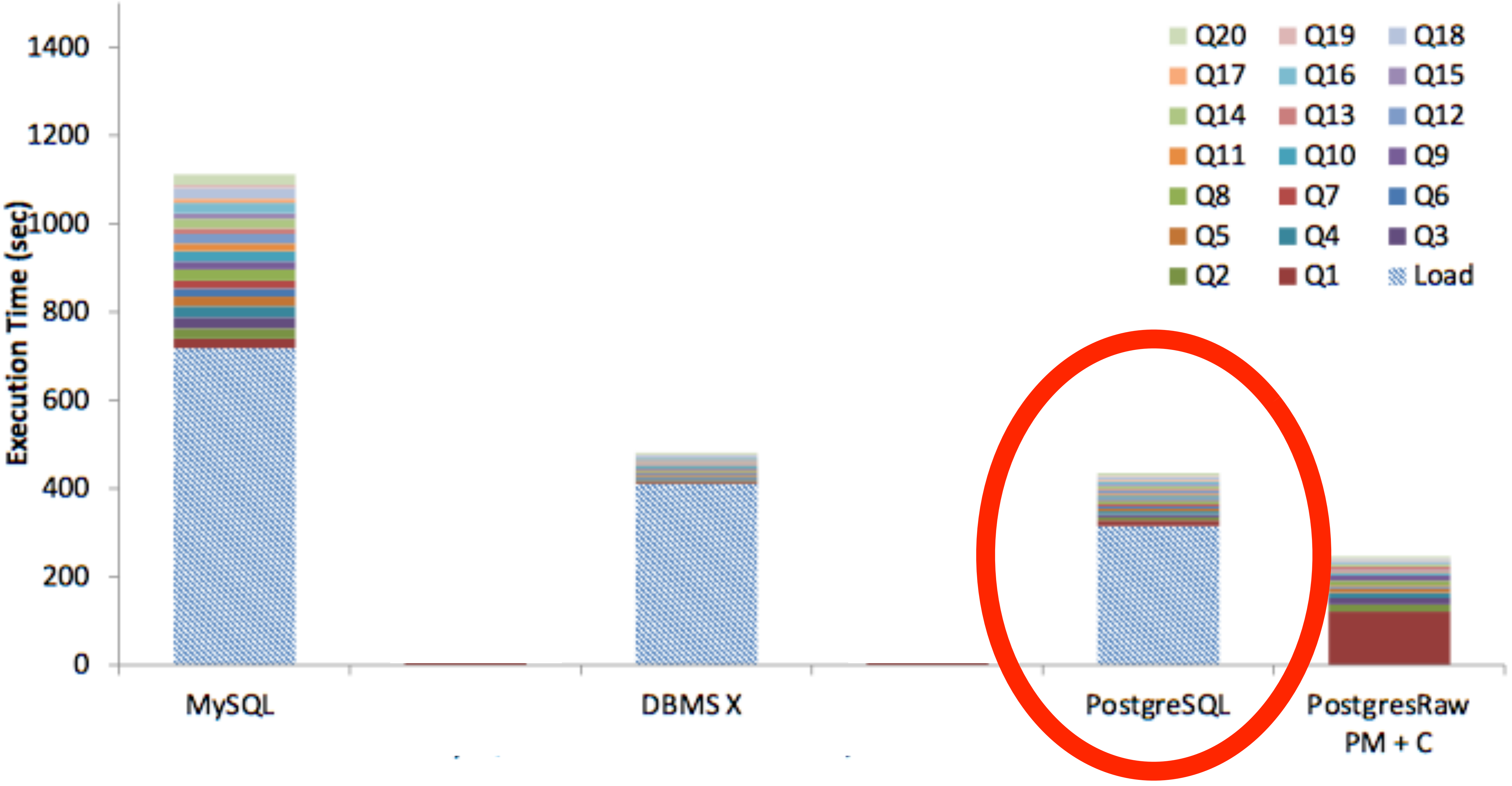
NoDB, SIGMOD 2012



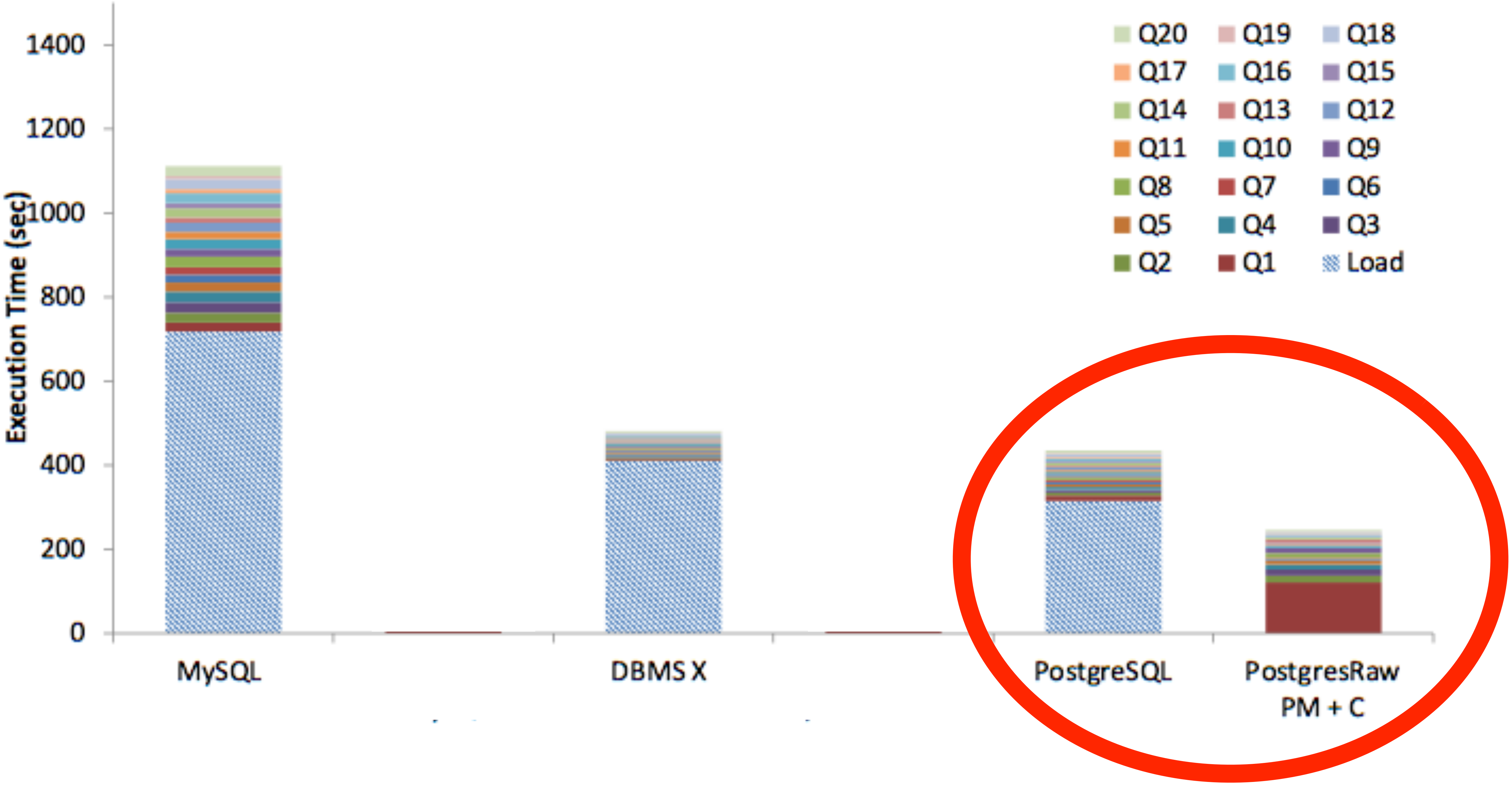
NoDB, SIGMOD 2012



NoDB, SIGMOD 2012

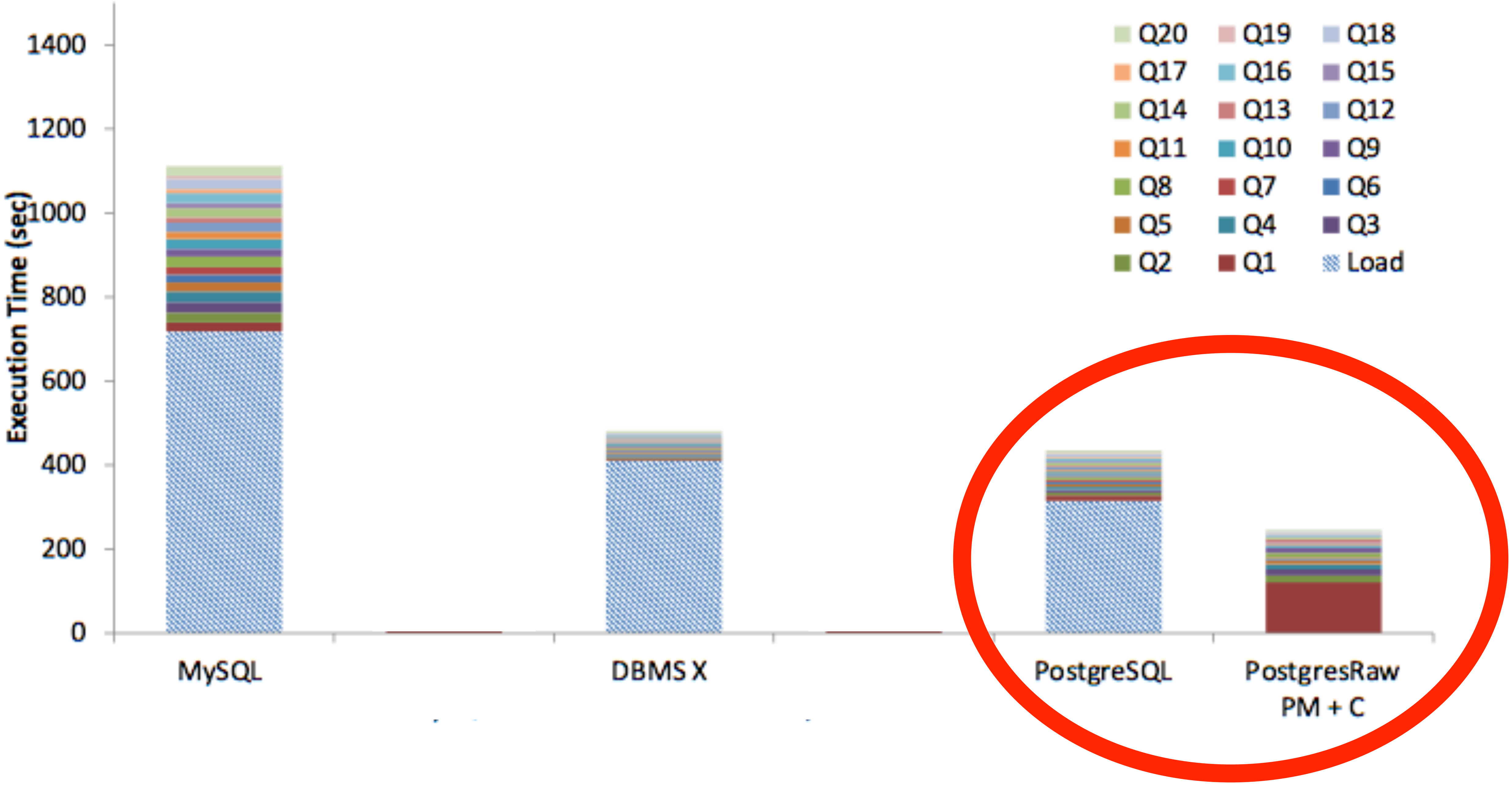


NoDB, SIGMOD 2012

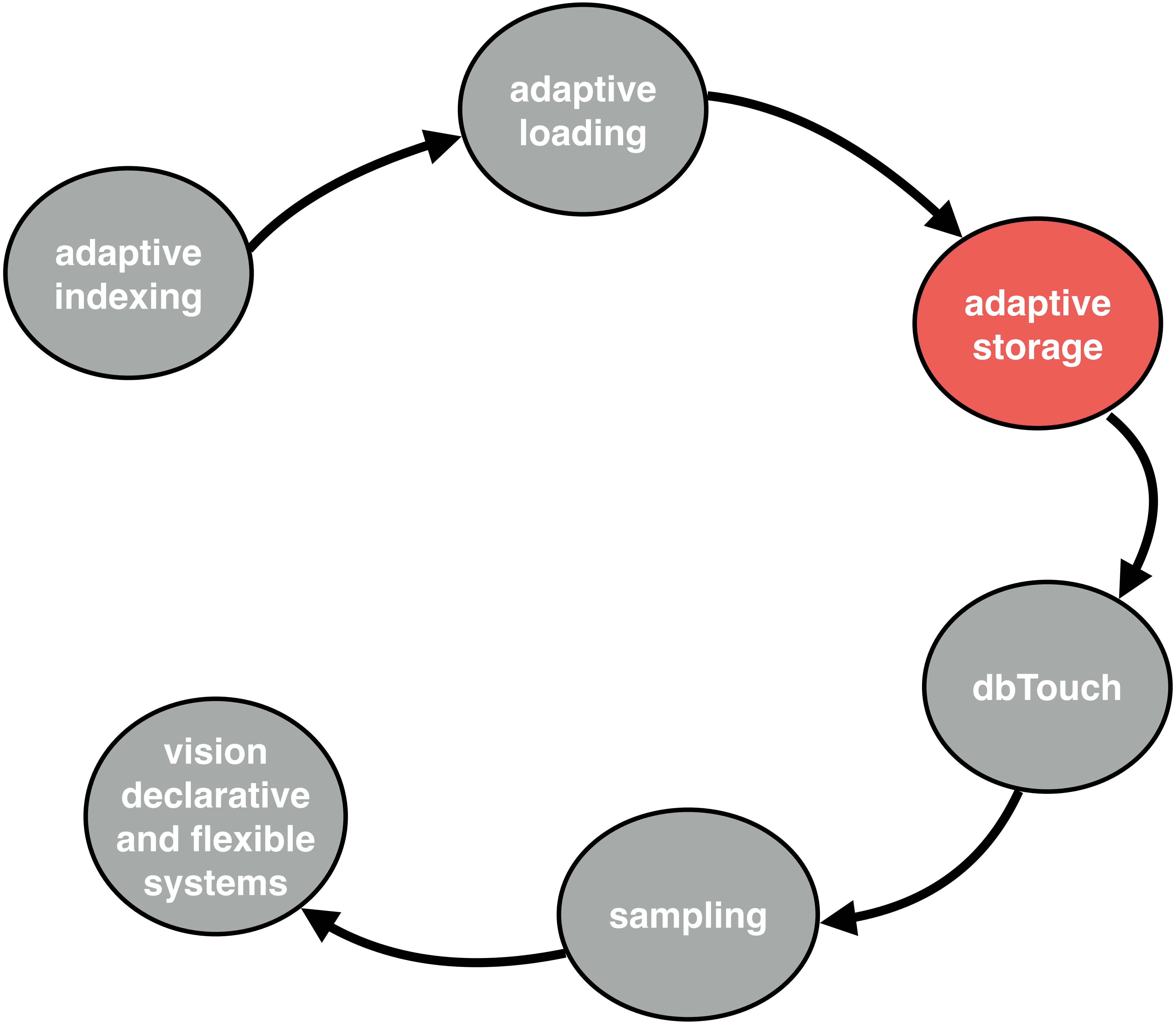


NoDB, SIGMOD 2012

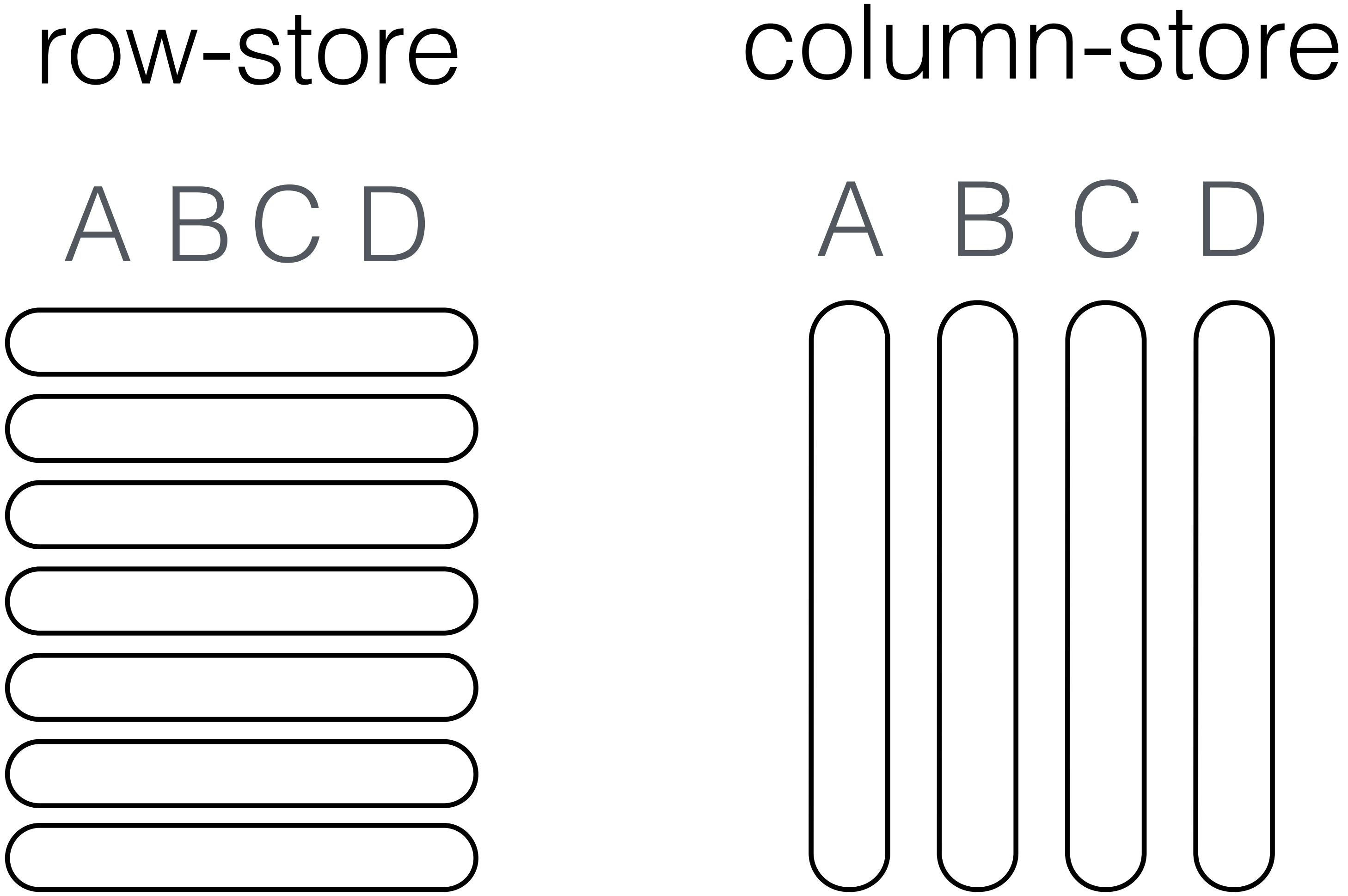
reducing data-to-query time



NoDB, SIGMOD 2012


adaptive indexing
adaptive loading
adaptive storage
dbTouch
sampling
vision declarative and flexible systems

rows & columns



row-store

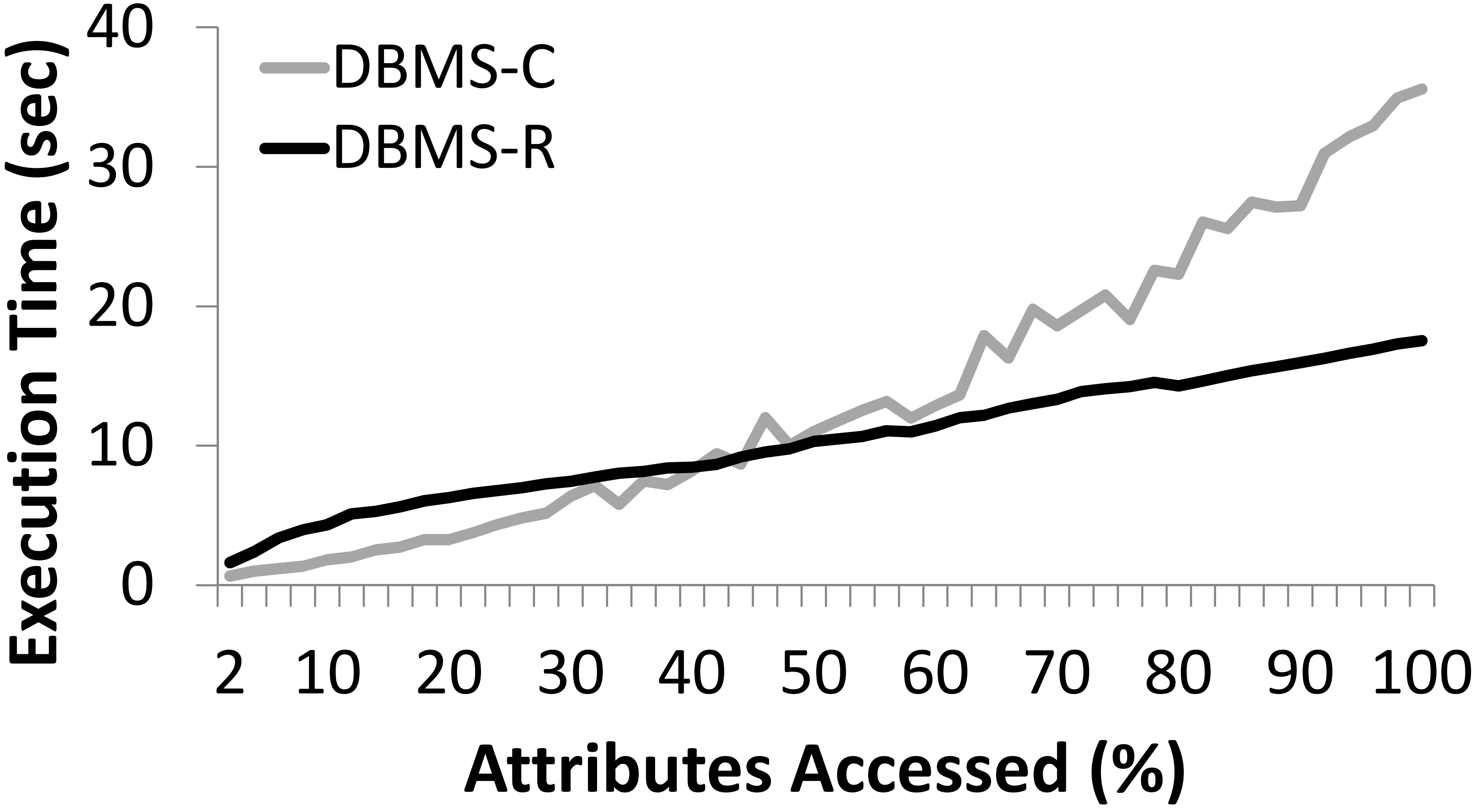
A B C D

column-store

A B C D

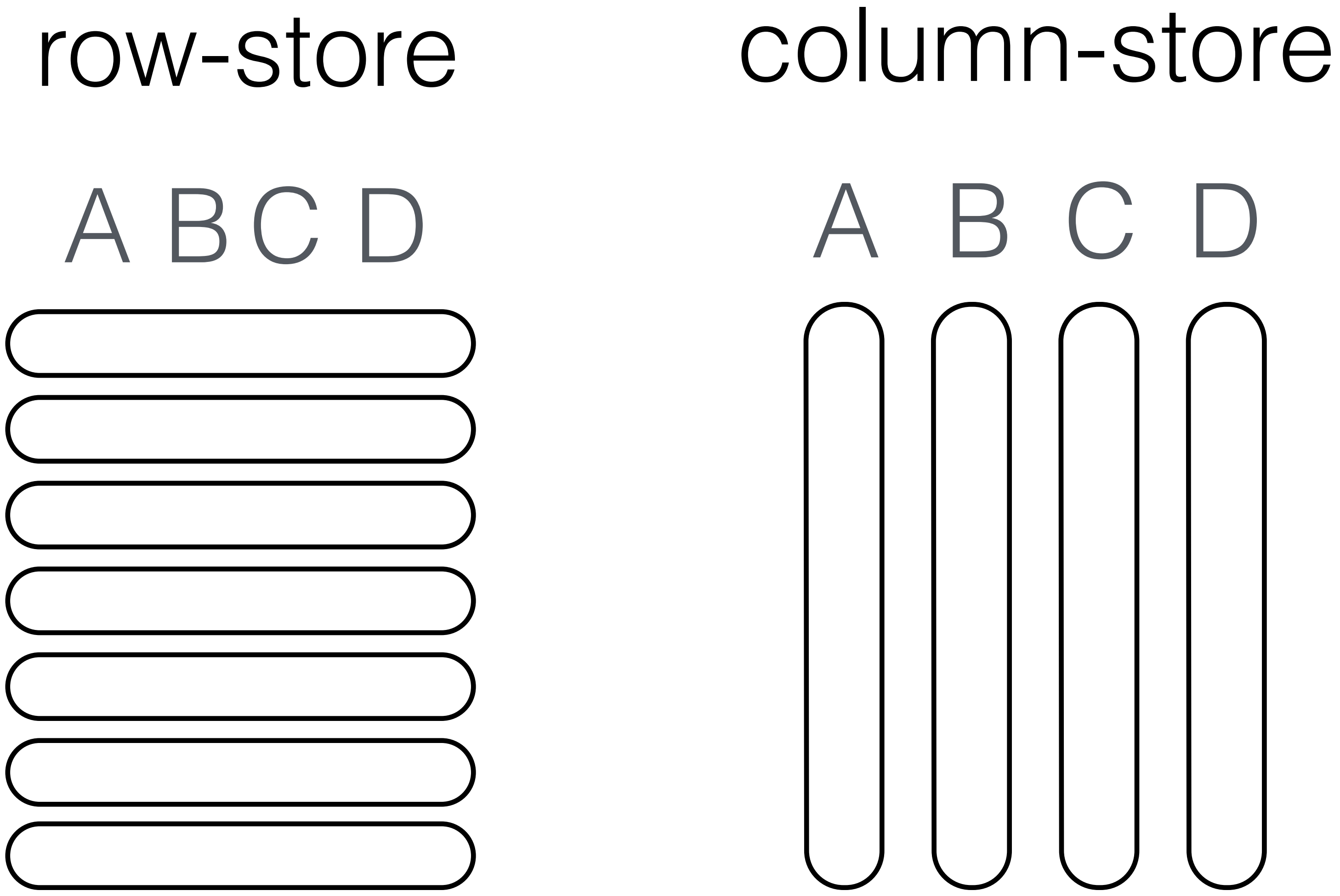
H20, SIGMOD 14

no fixed optimal solution



H20, SIGMOD 14

rows & columns



row-store

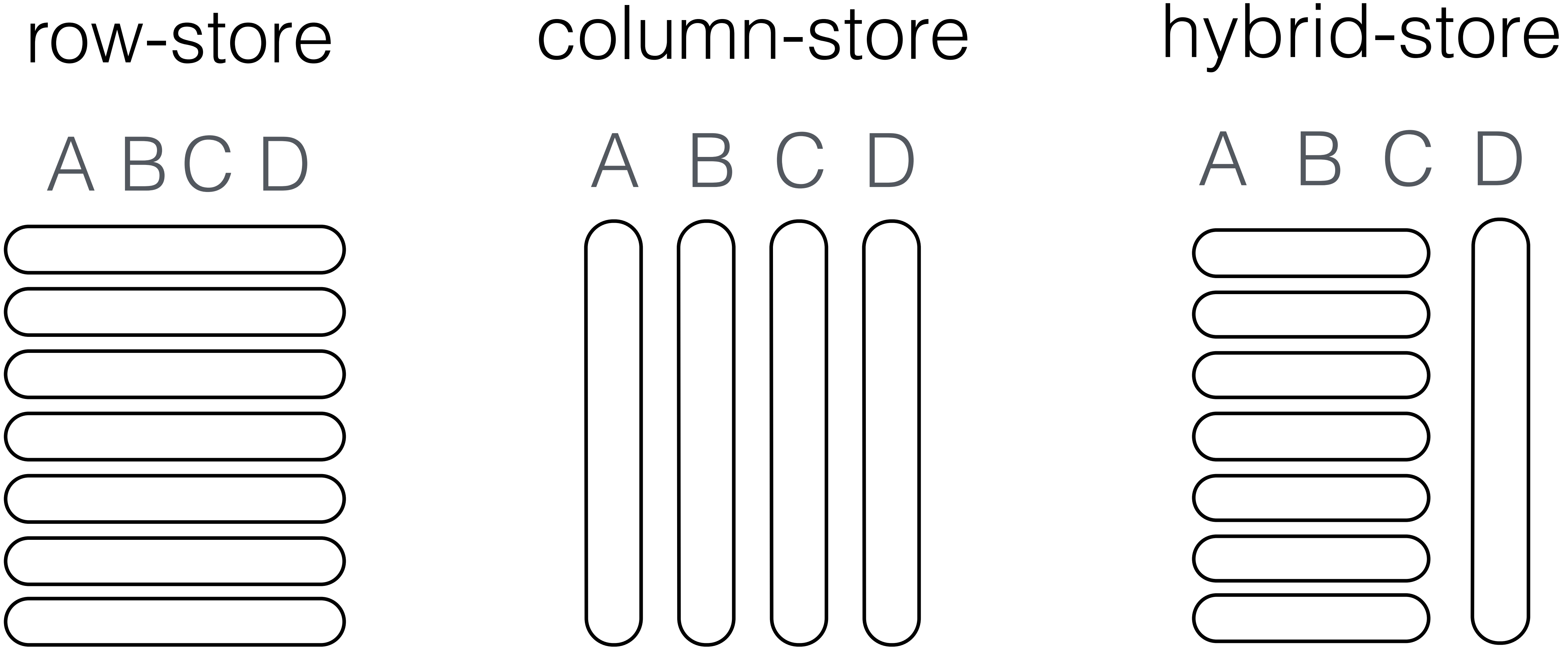
A B C D

column-store

A B C D

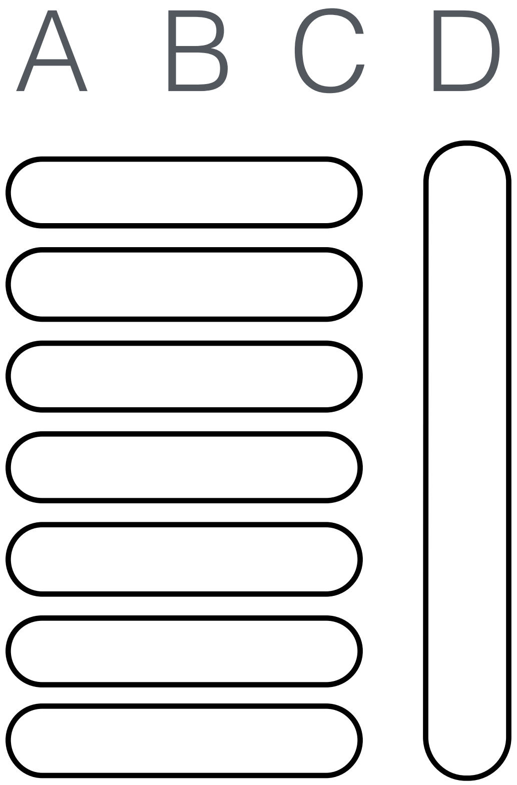
H20, SIGMOD 14

rows & columns



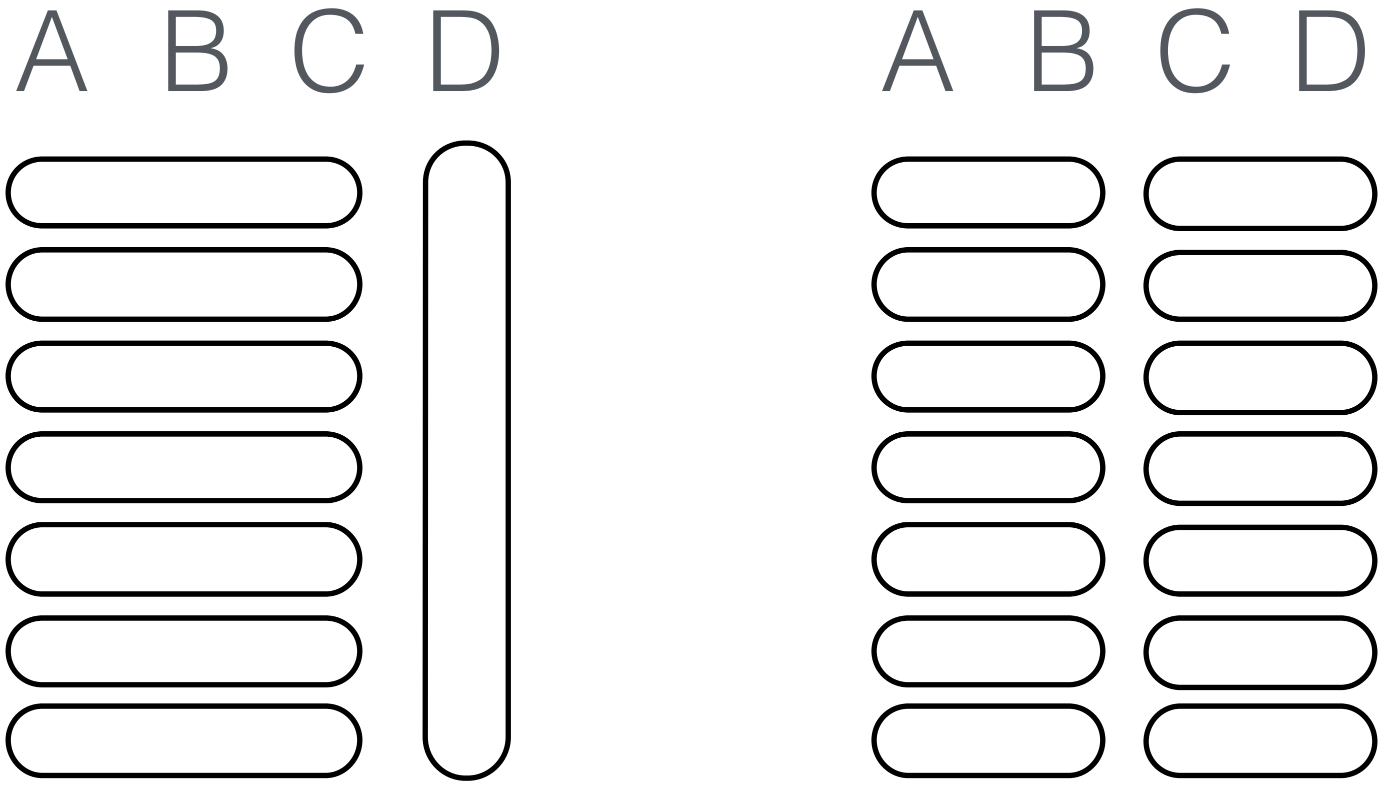
H20, SIGMOD 14

which layout is best?



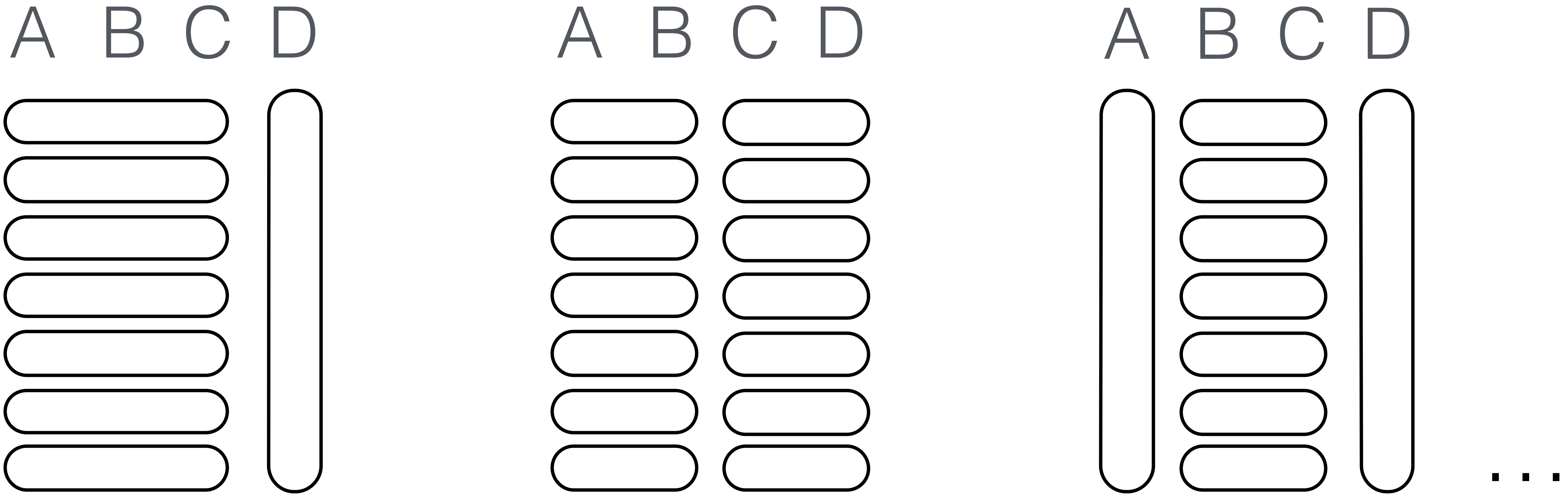
H20, SIGMOD 14

which layout is best?



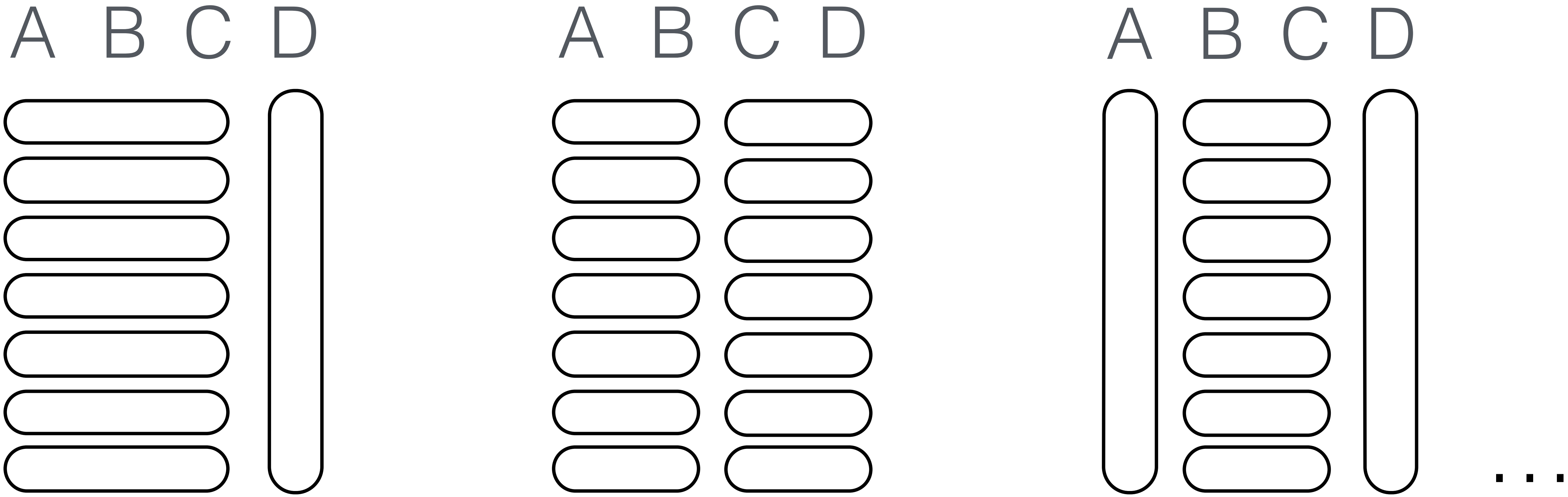
H20, SIGMOD 14

which layout is best?



H20, SIGMOD 14

which layout is best?



too many combinations to maintain in parallel

H20, SIGMOD 14

query cost

$$q(L) = \sum_{i=1}^{|L|} max(cost_i^{IO}, cost_i^{CPU})$$

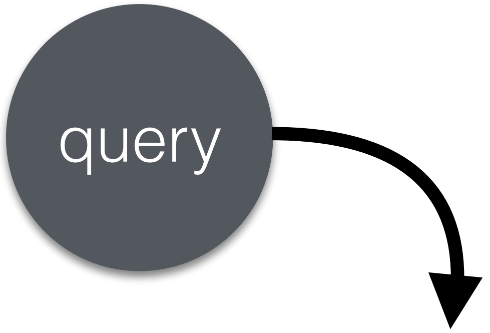
for a given query we can know which layout is best

the one that will cause the fewer cache misses if we know all queries up front we can choose the layouts

adaptive storage:
continuously adapt layouts based on incoming queries

H20, SIGMOD 14

but computing all possible combinations is expensive…

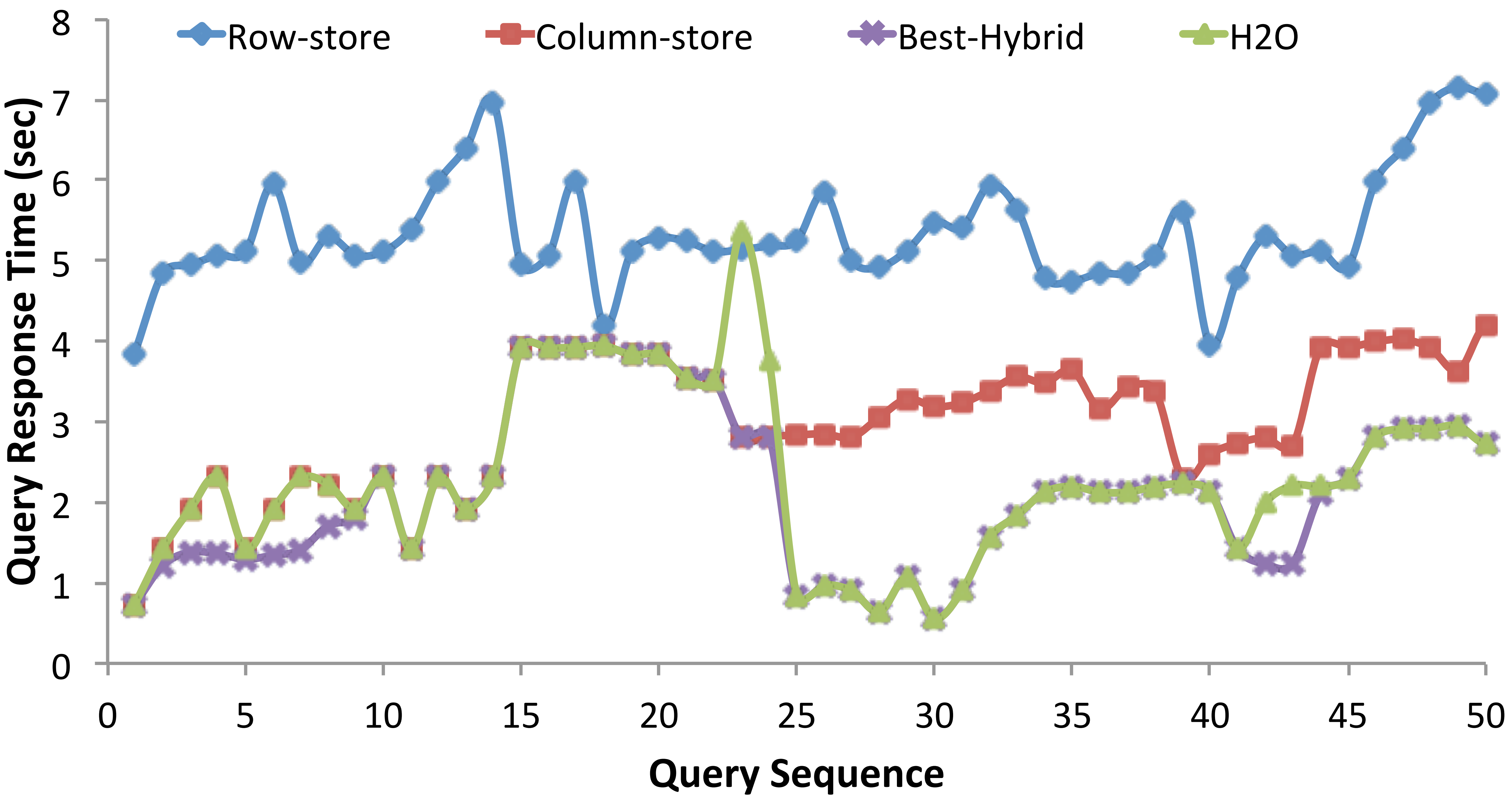


query

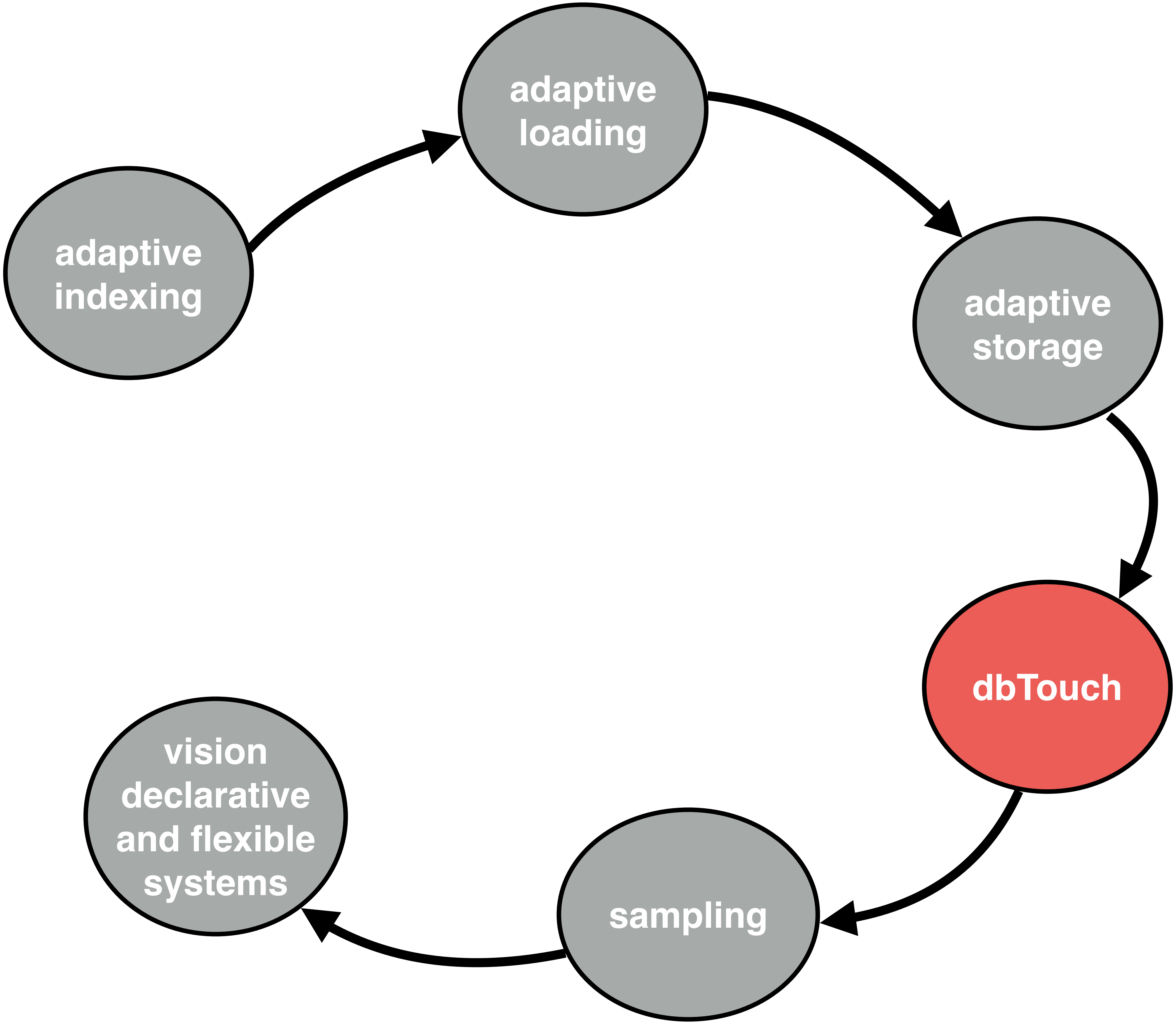
select A+B+C+D from R where A<10 and E>10

1. deal only with attributes referenced in queries
2. handle select clause separately from where clause
3. start from pure column-store and build up
4. stop when no improvement possible

H20, SIGMOD 14



H2O, SIGMOD 14



adaptive indexing

adaptive loading

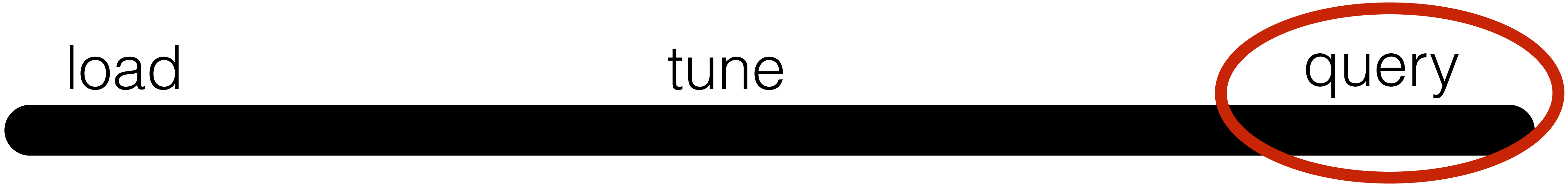
adaptive storage

dbTouch

sampling

vision declarative and flexible systems

querying



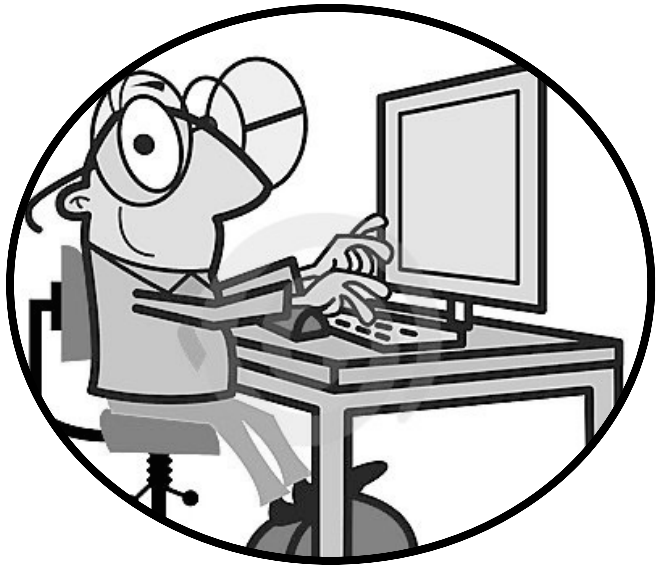
load

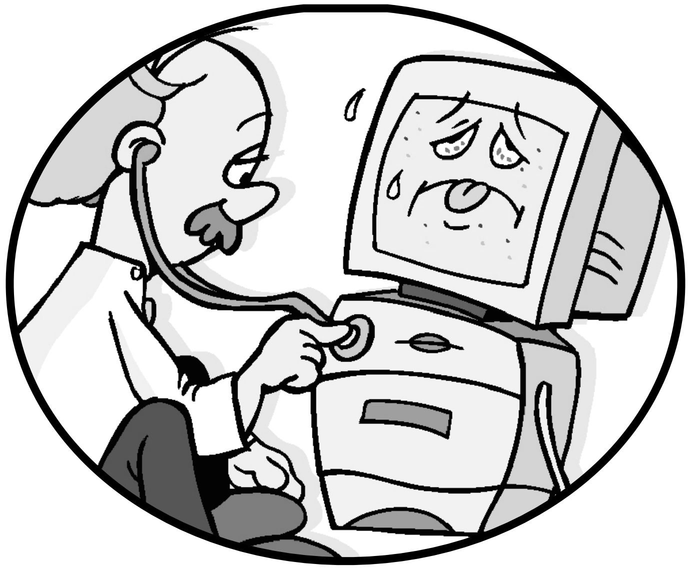
tune

query

SQL interface

correct and complete answers



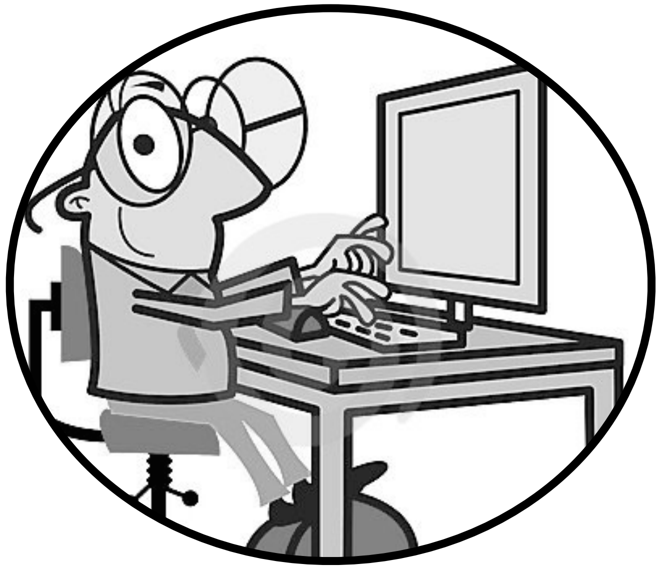


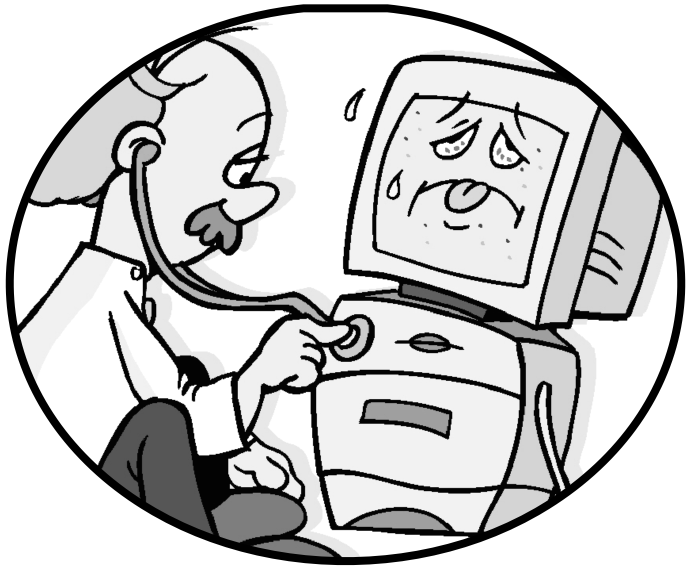
querying

complex and slow - not fit for exploration

SQL interface

correct and complete answers

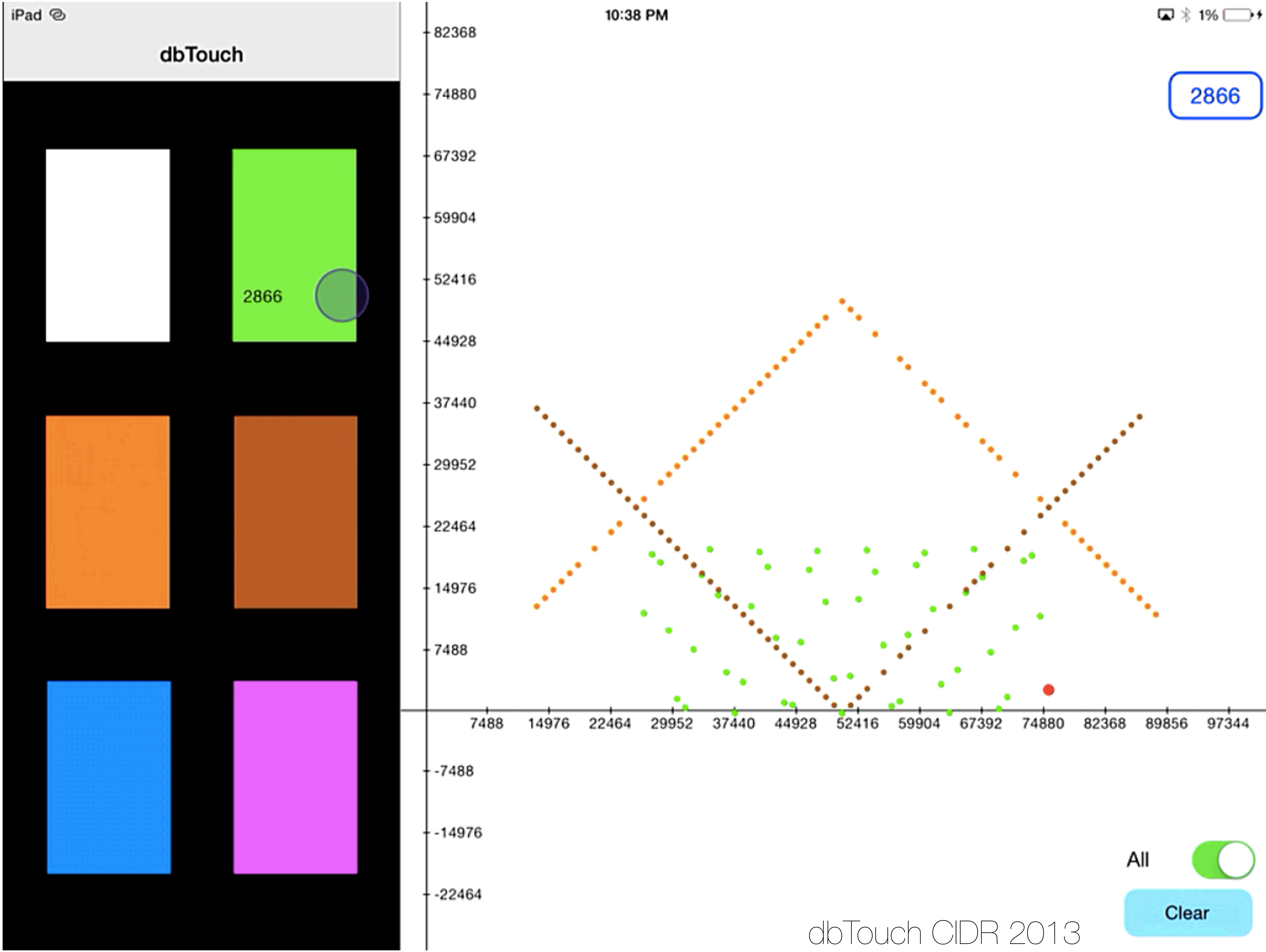




just touch the data you need

just touch the data you need

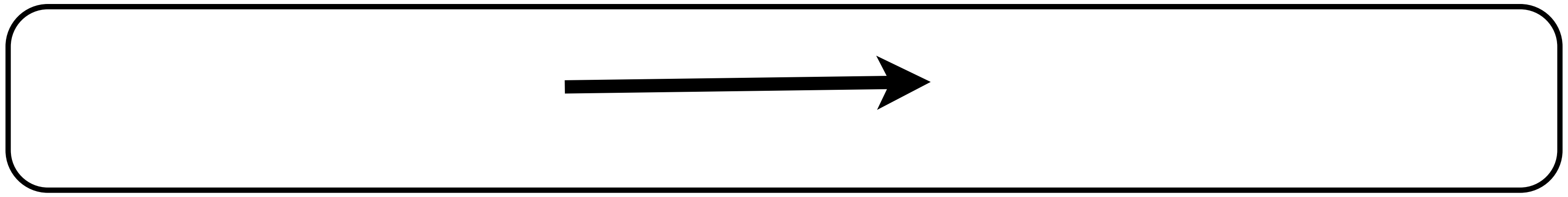
this is not about query building
it is about query processing



dbTouch CIDR 2013

what does this mean for db kernels?

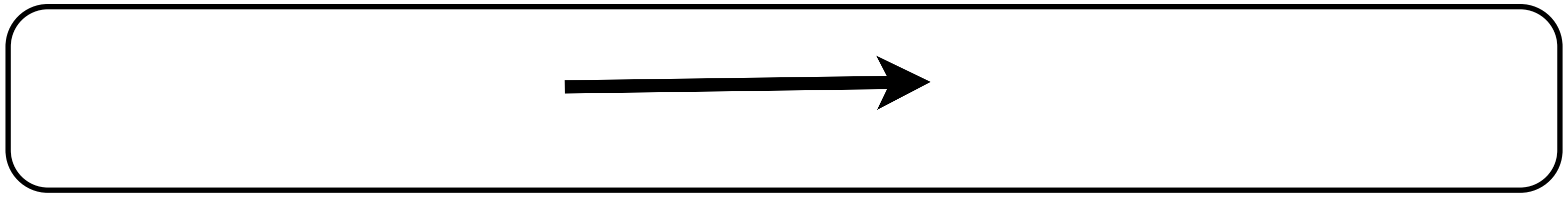
db



select R.a from R

what does this mean for db kernels?

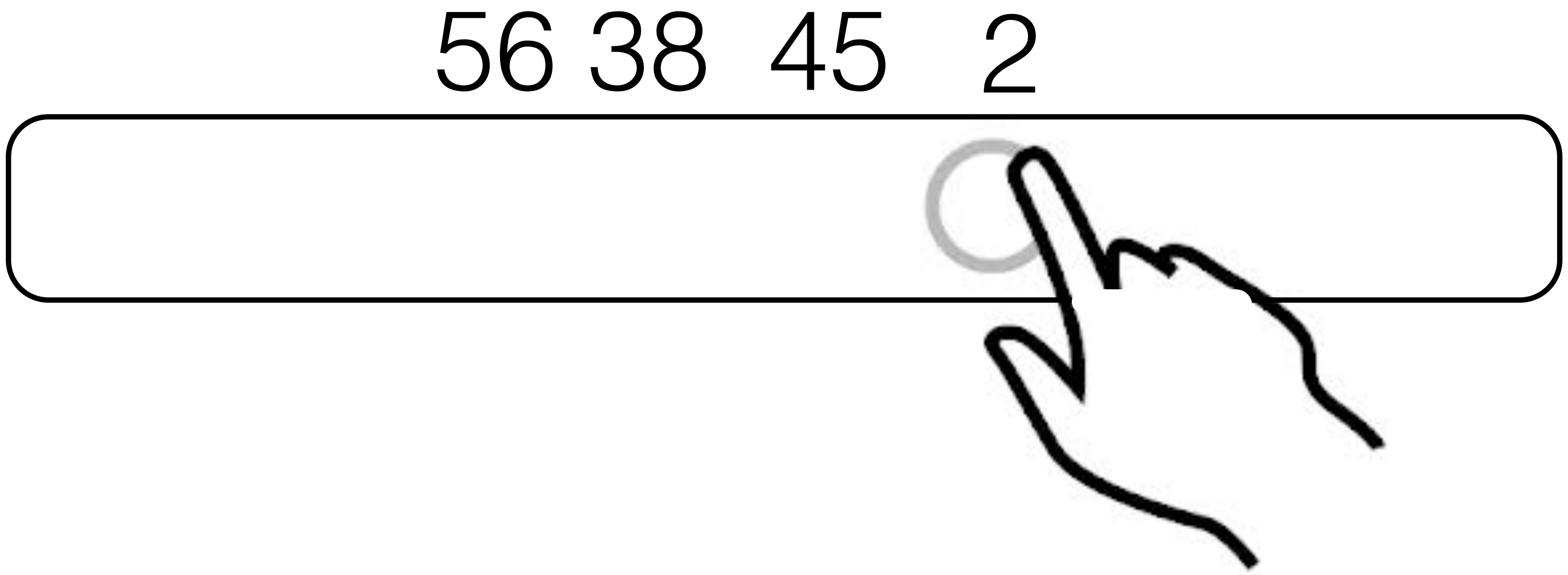
db



select R.a from R

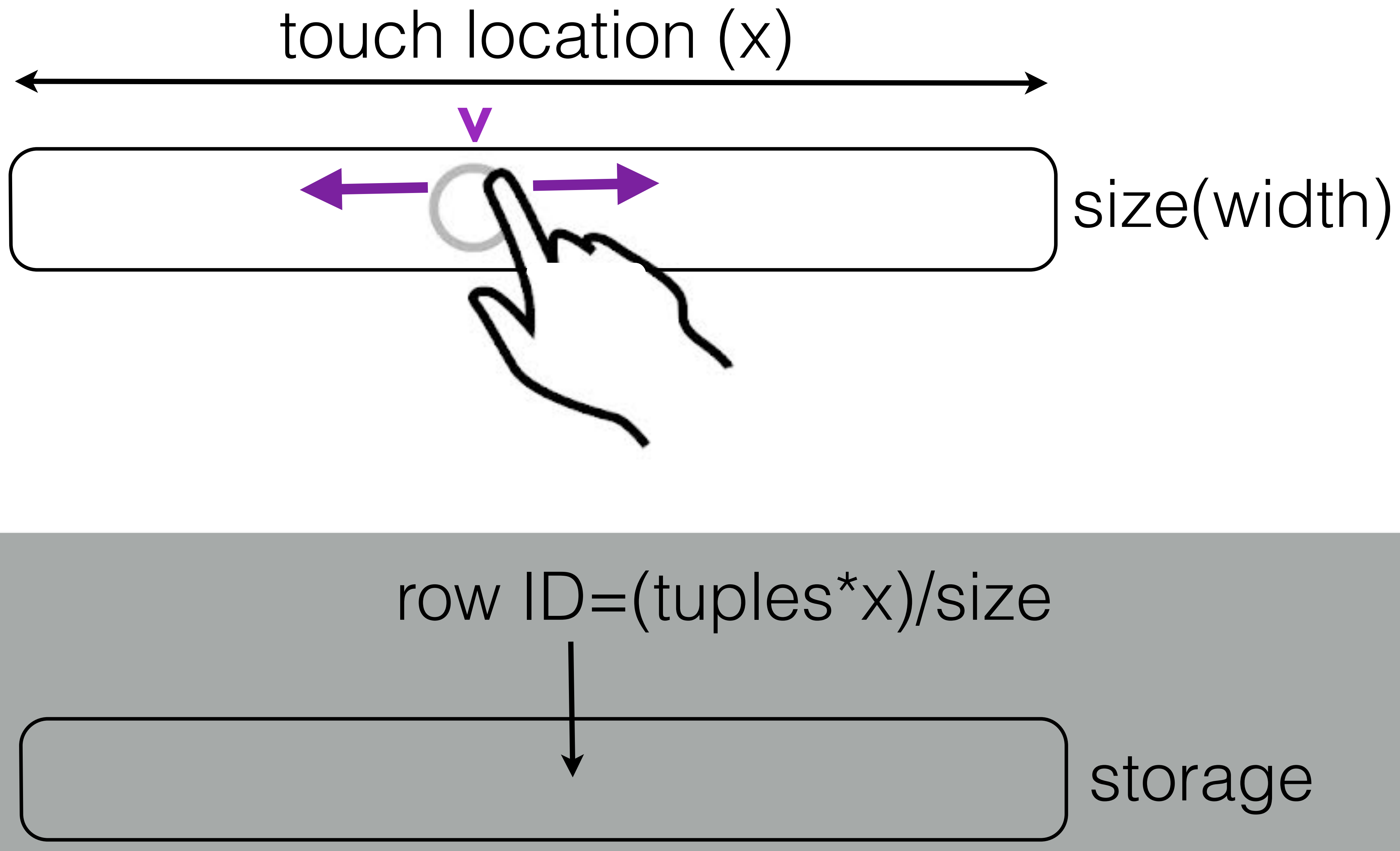
what does this mean for db kernels?

dbTouch



process only
what you touch

from touch to query processing



touch location (x)

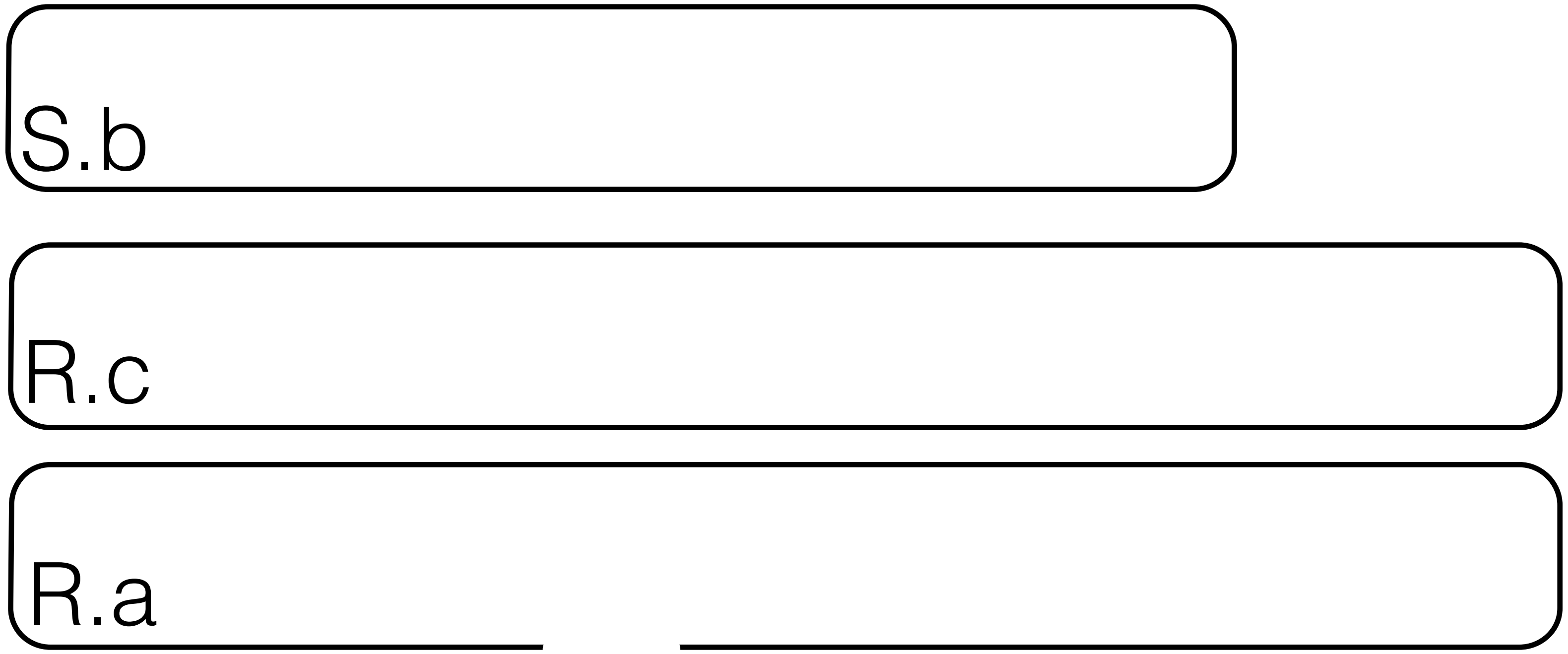
v

size(width)

row ID=(tuples*x)/size

storage

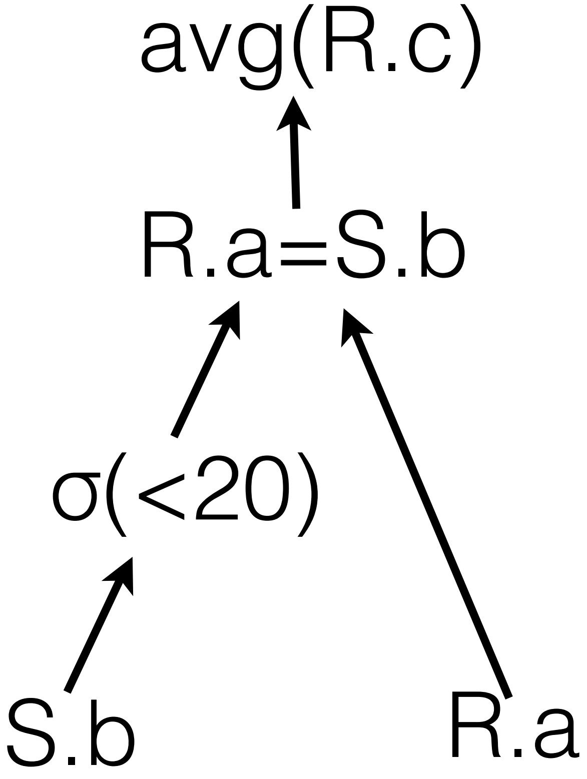
select avg(R.c)
where R.a=S.b and S.b<20



S.b

R.c

R.a



avg(R.c)

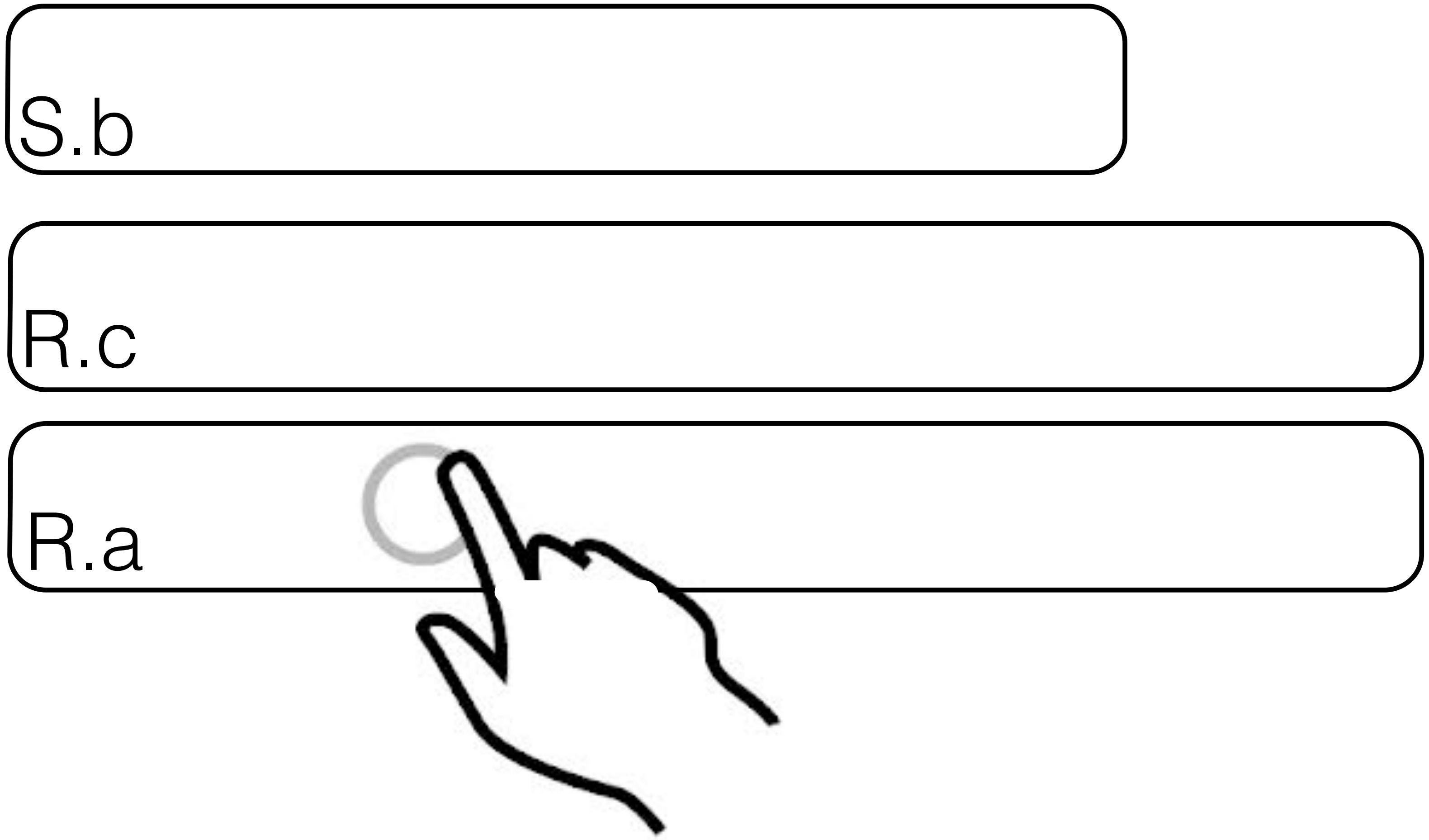
R.a=S.b

σ(<20)

S.b

R.a

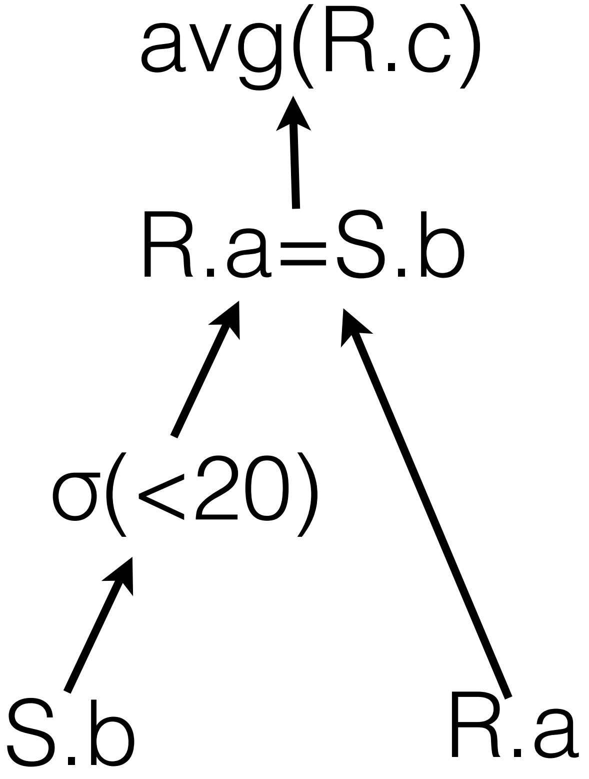
select avg(R.c)
where R.a=S.b and S.b<20



S.b

R.c

R.a



avg(R.c)

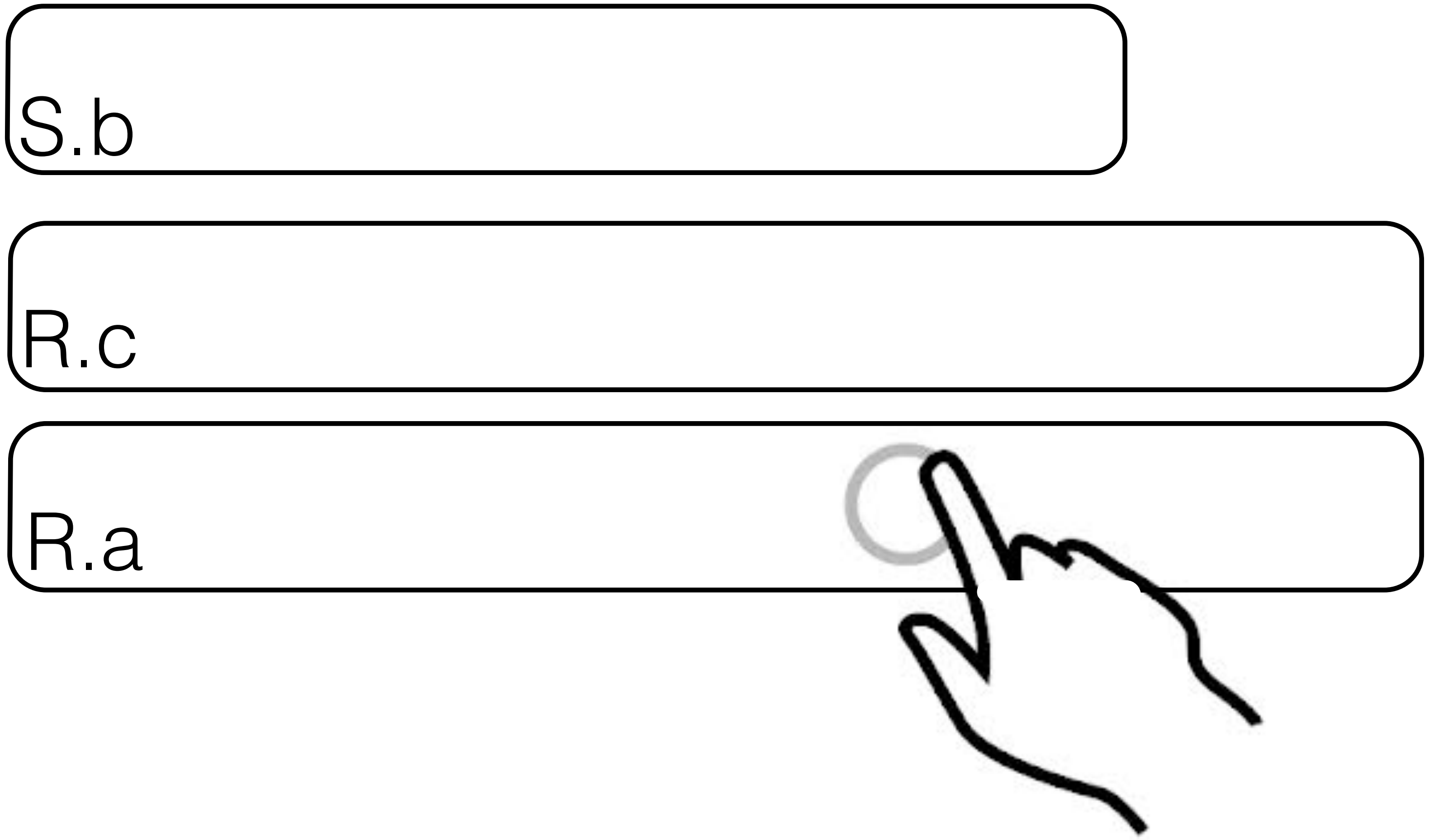
R.a=S.b

σ(<20)

S.b

R.a

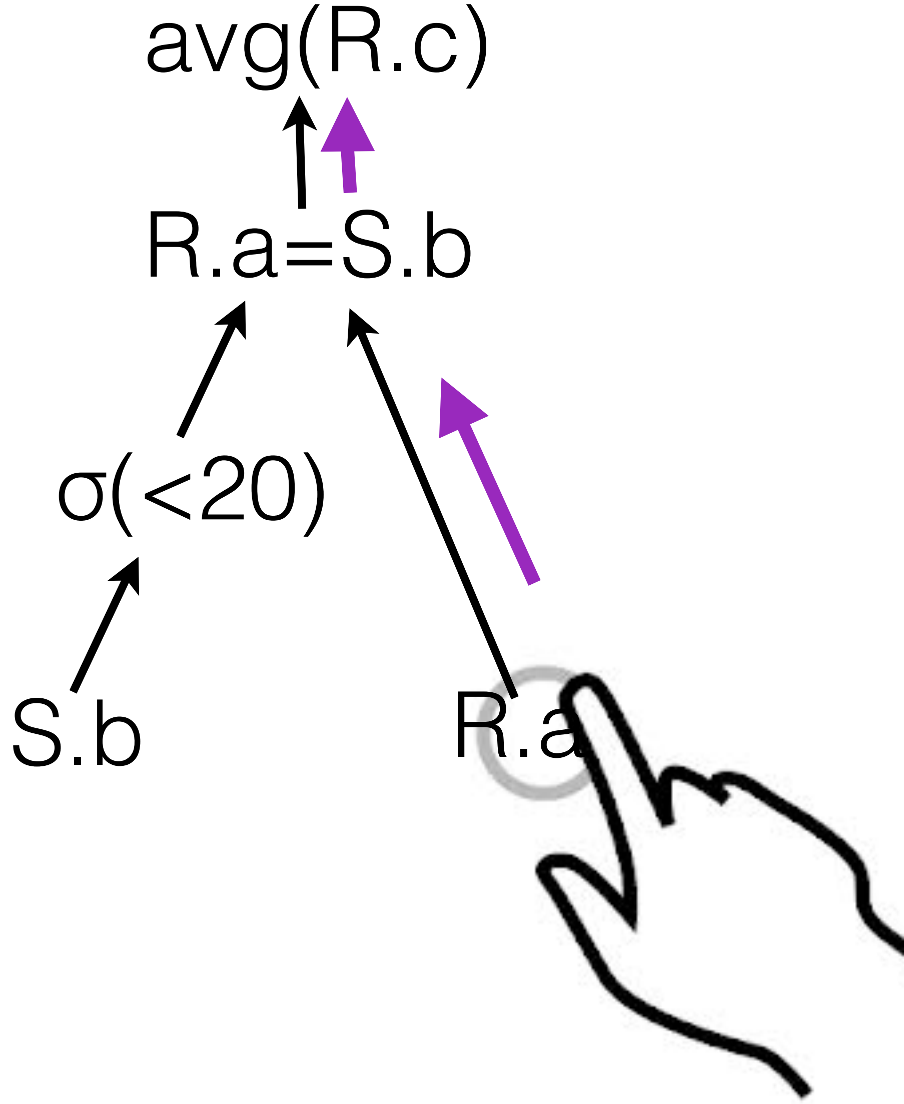
select avg(R.c)
where R.a=S.b and S.b<20



S.b

R.c

R.a



avg(R.c)

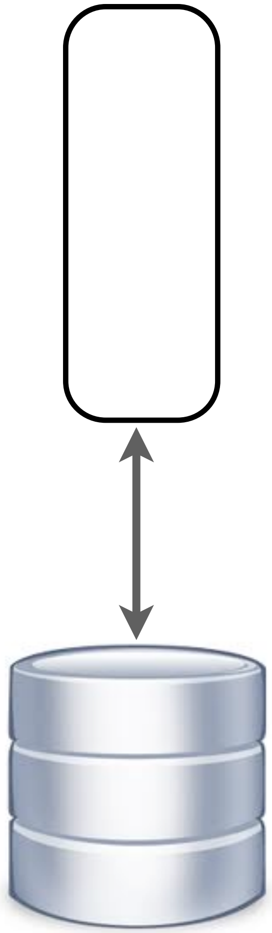
R.a=S.b

σ(<20)

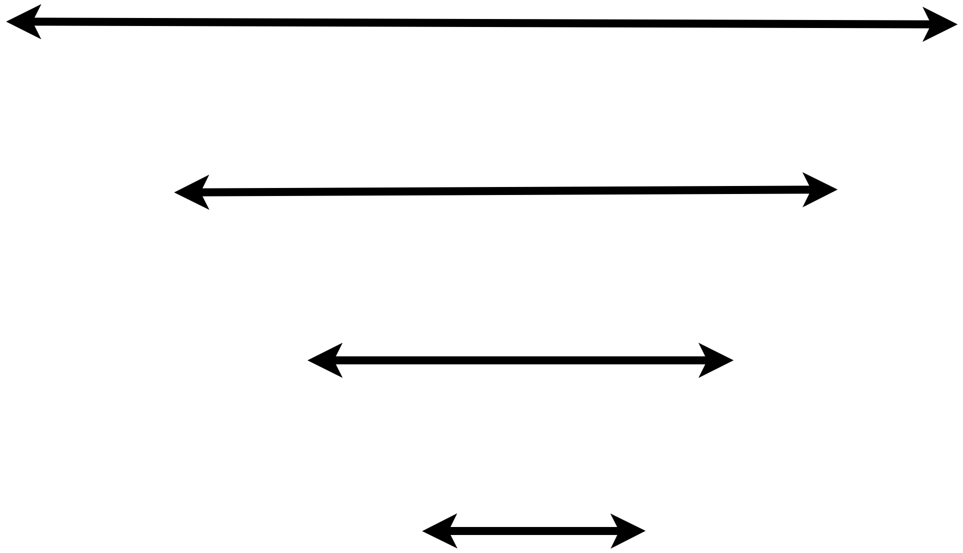
S.b

R.a

visual objects

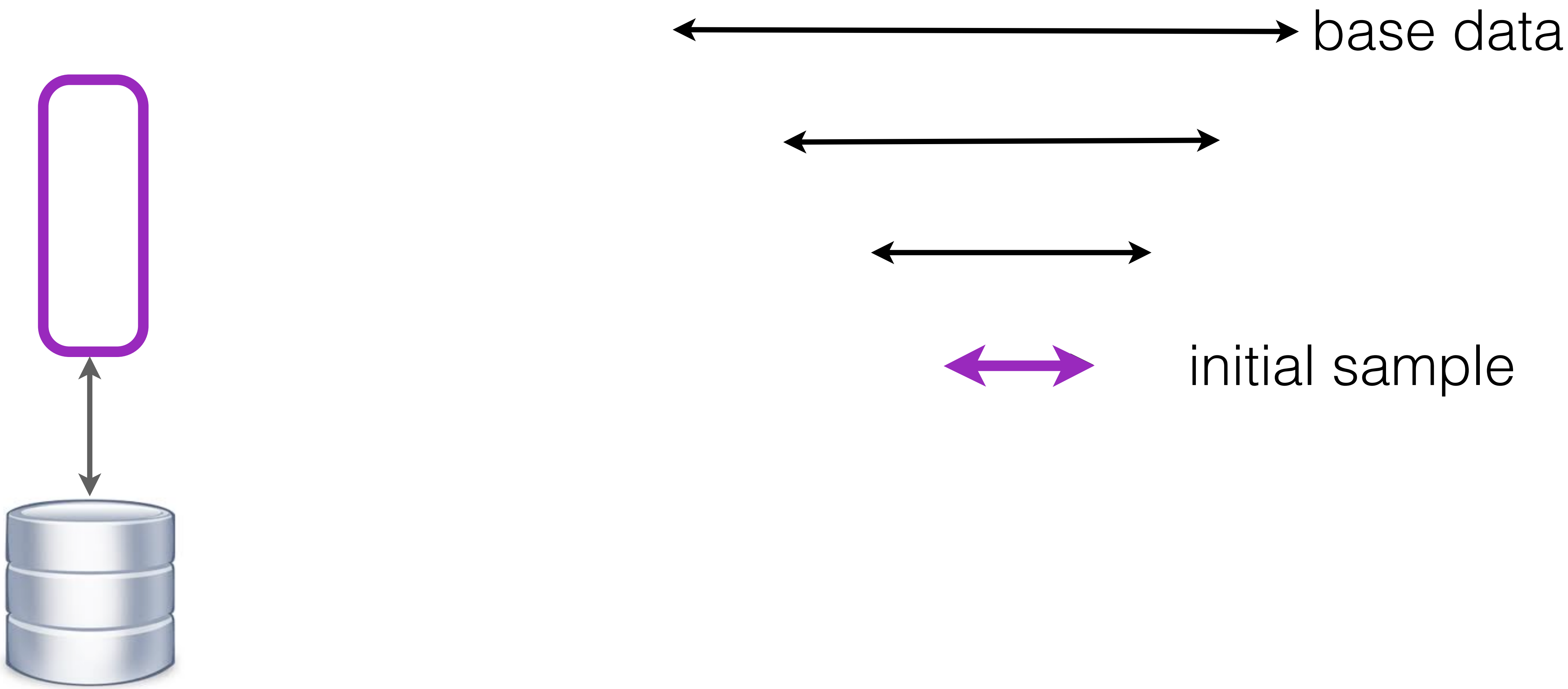


samples hierarchy



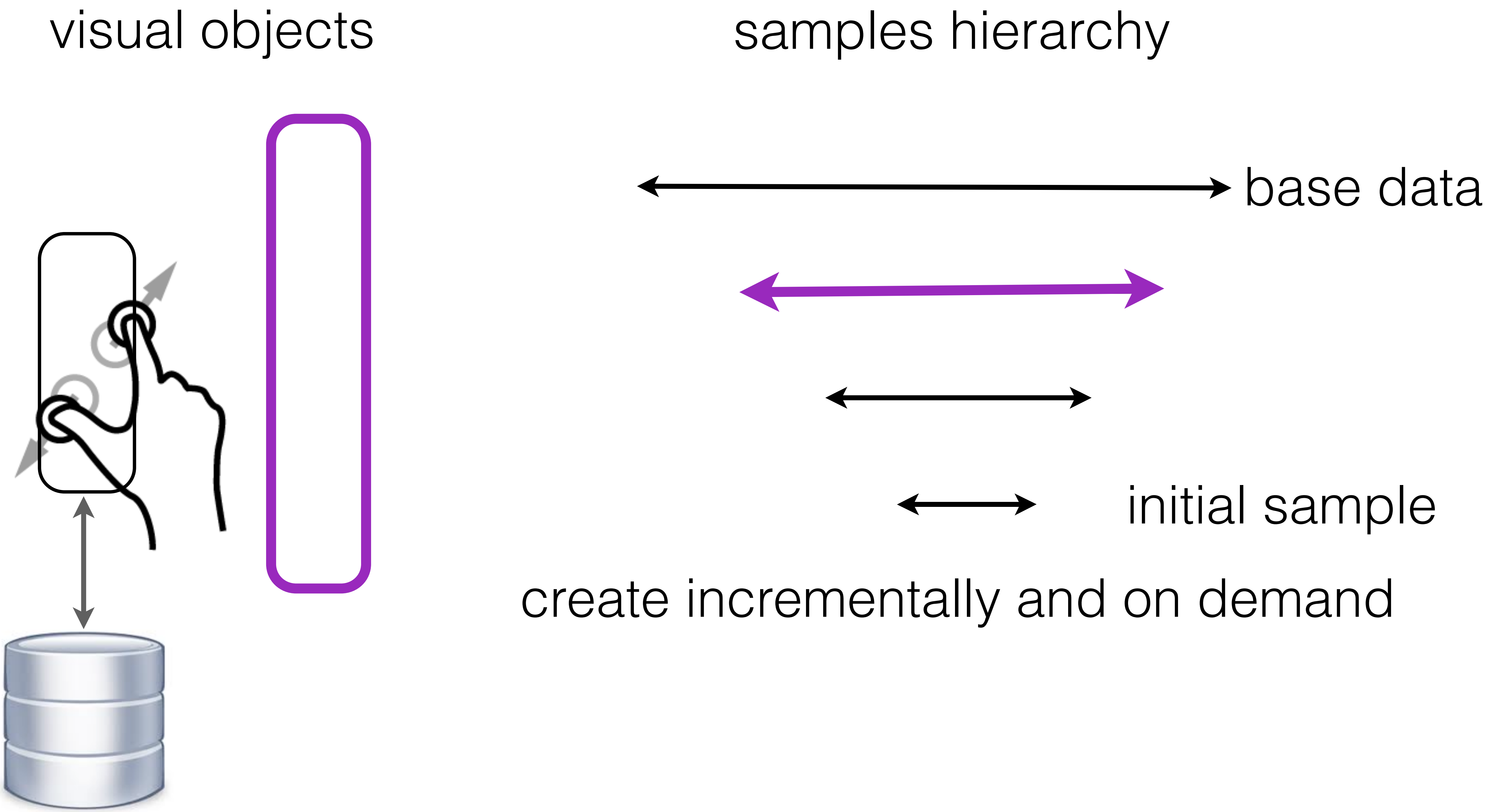
visual objects

samples hierarchy



base data

initial sample



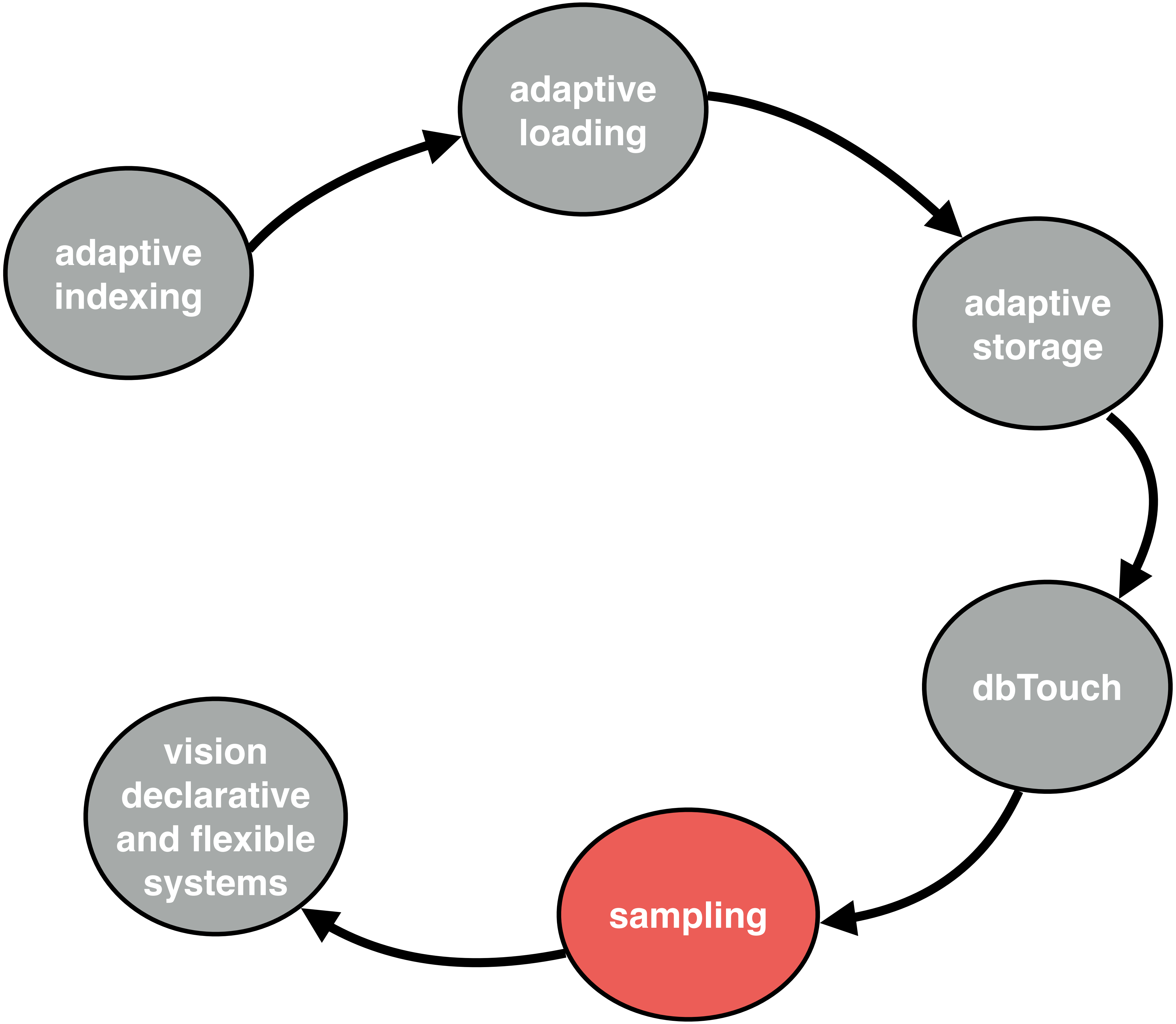
visual objects

samples hierarchy

base data

initial sample

create incrementally and on demand



adaptive indexing

adaptive loading

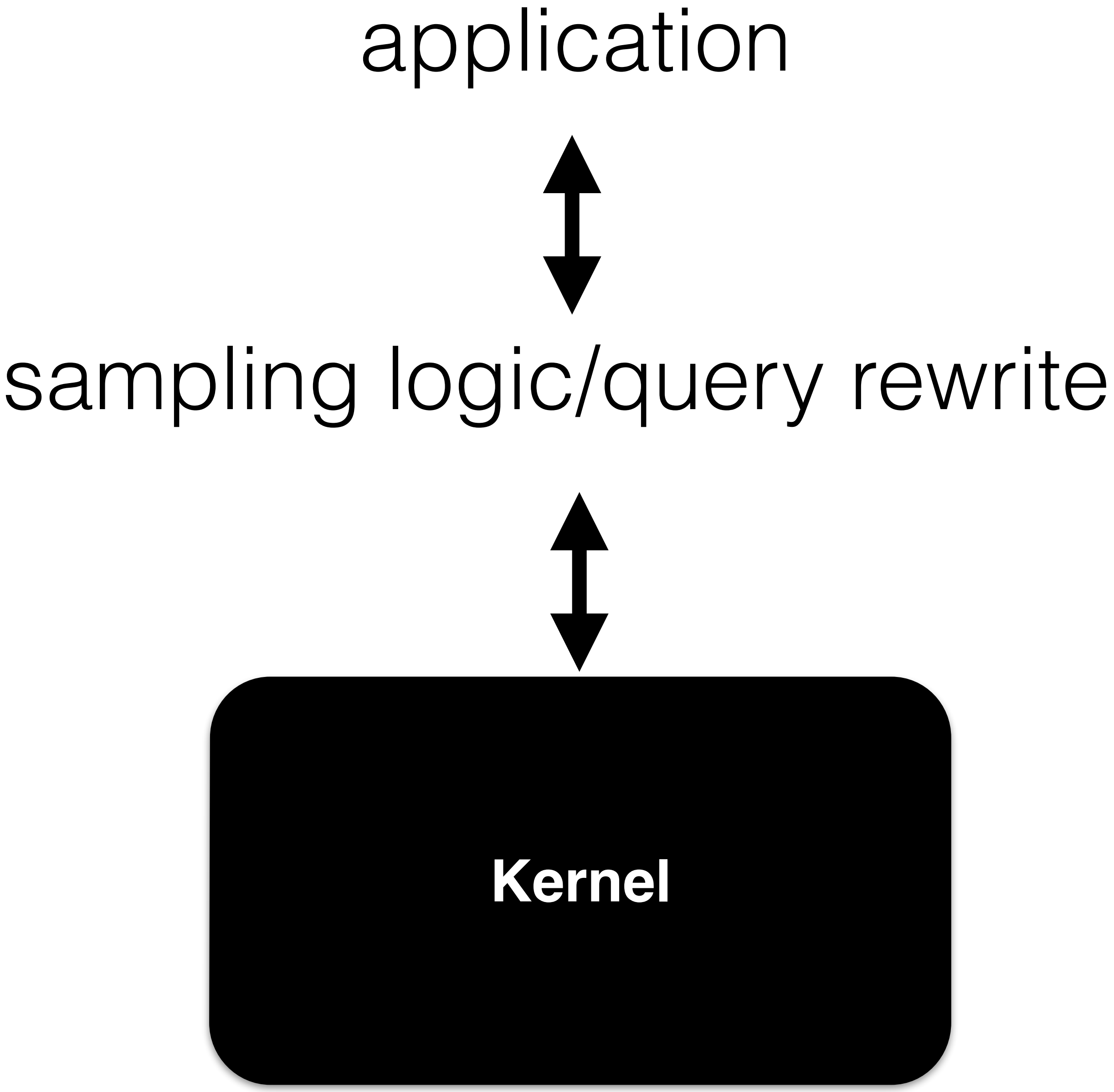
adaptive storage

dbTouch

sampling

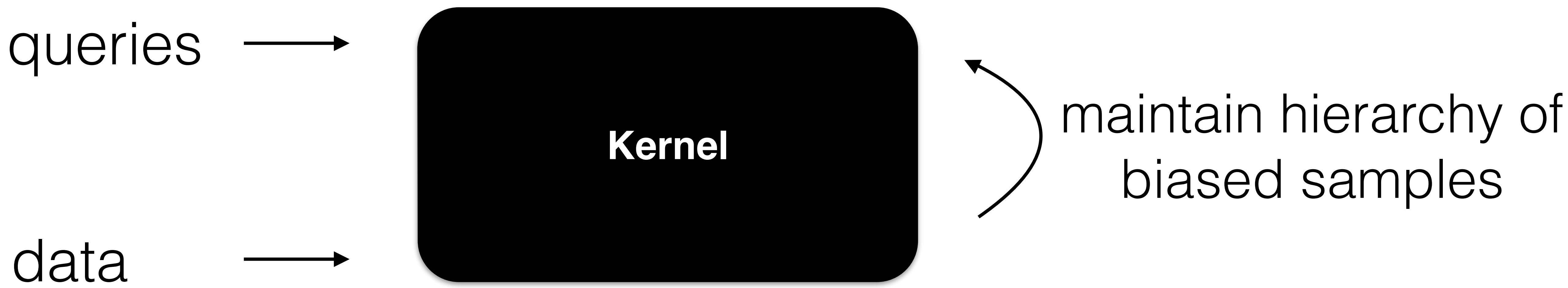
vision declarative and flexible systems

sampling outside the engine



Aqua, VLDB 1999
BlinkDB, Eurosys 2013

sampling with SciBORG



queries

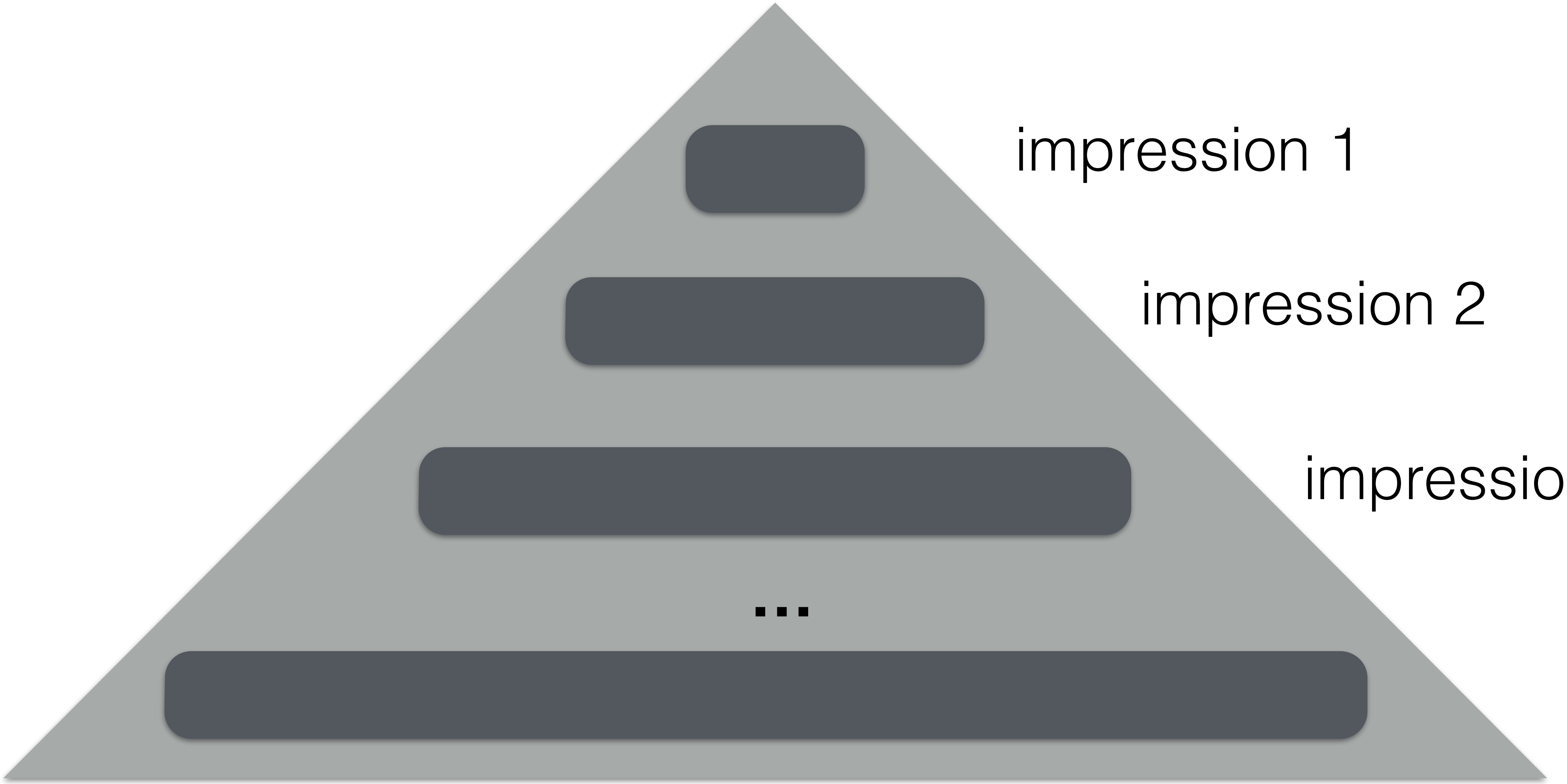
data

Kernel

maintain hierarchy of biased samples

SciBORG, CIDR 2011

sampling with SciBORG



impression 1

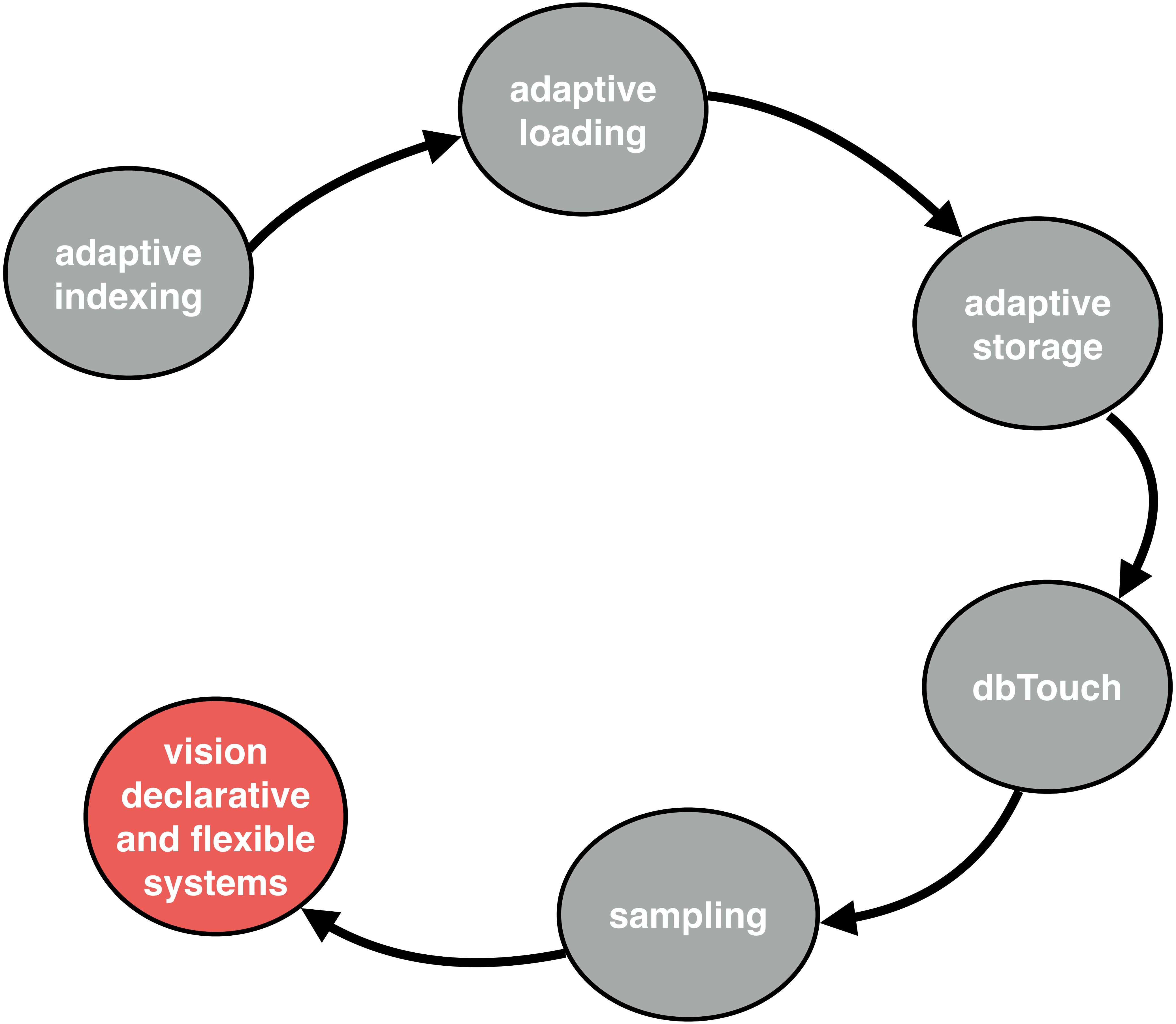
impression 2

impression 3

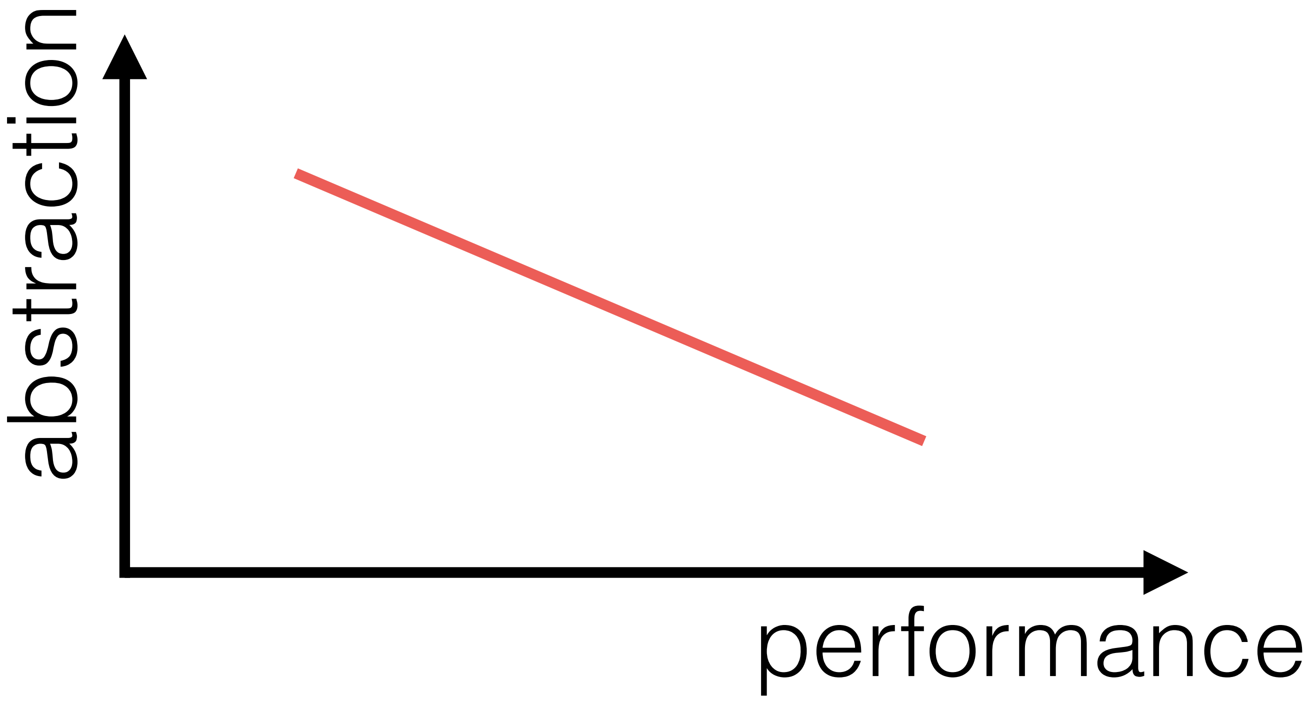
base column

continuously reorganized based on the workload

SciBORG, CIDR 2011


adaptive indexing
adaptive loading
adaptive storage
dbTouch
sampling
vision declarative and flexible systems

building systems declaratively

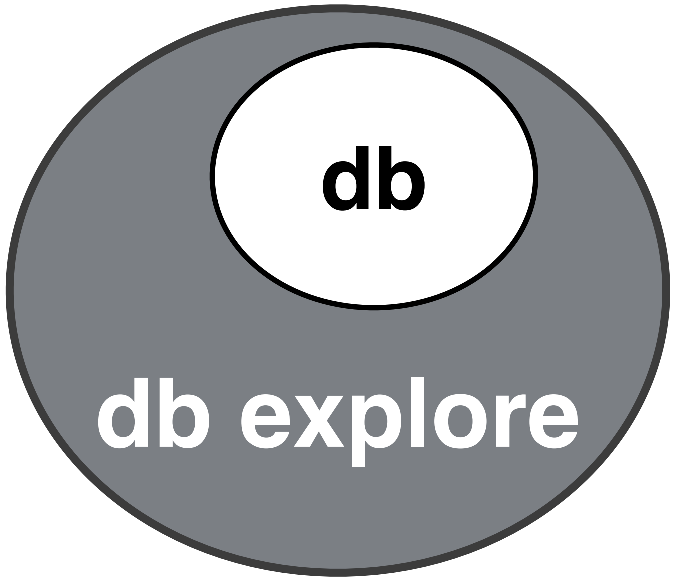


vision: being able to define system components in a higher level language without significant performance penalty

RodentStore, CIDR 2009
Abstraction without regrets, IEEE Data Engin. Bulletin/PVLDB 2014

data systems today

allow us to answer queries fast



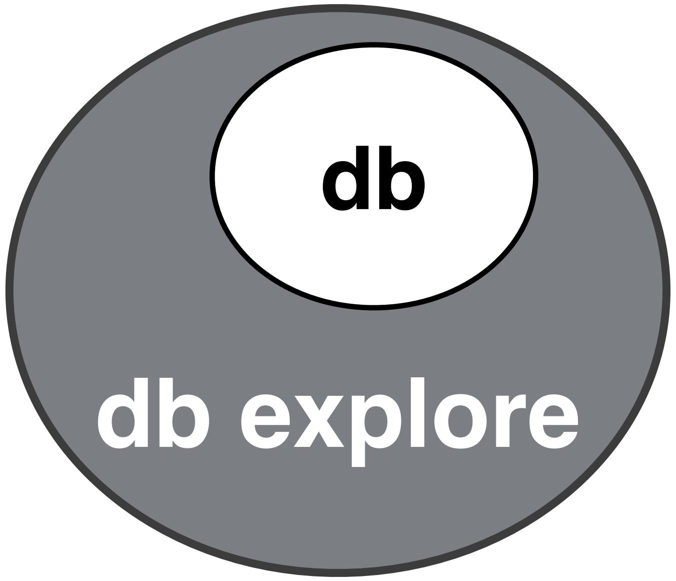
data systems for exploration

should allow us to find fast which queries to ask

+ approximate processing techniques

data systems today

allow us to answer queries fast



data systems for exploration

should allow us to find fast which queries to ask

+ approximate processing techniques

thank you!