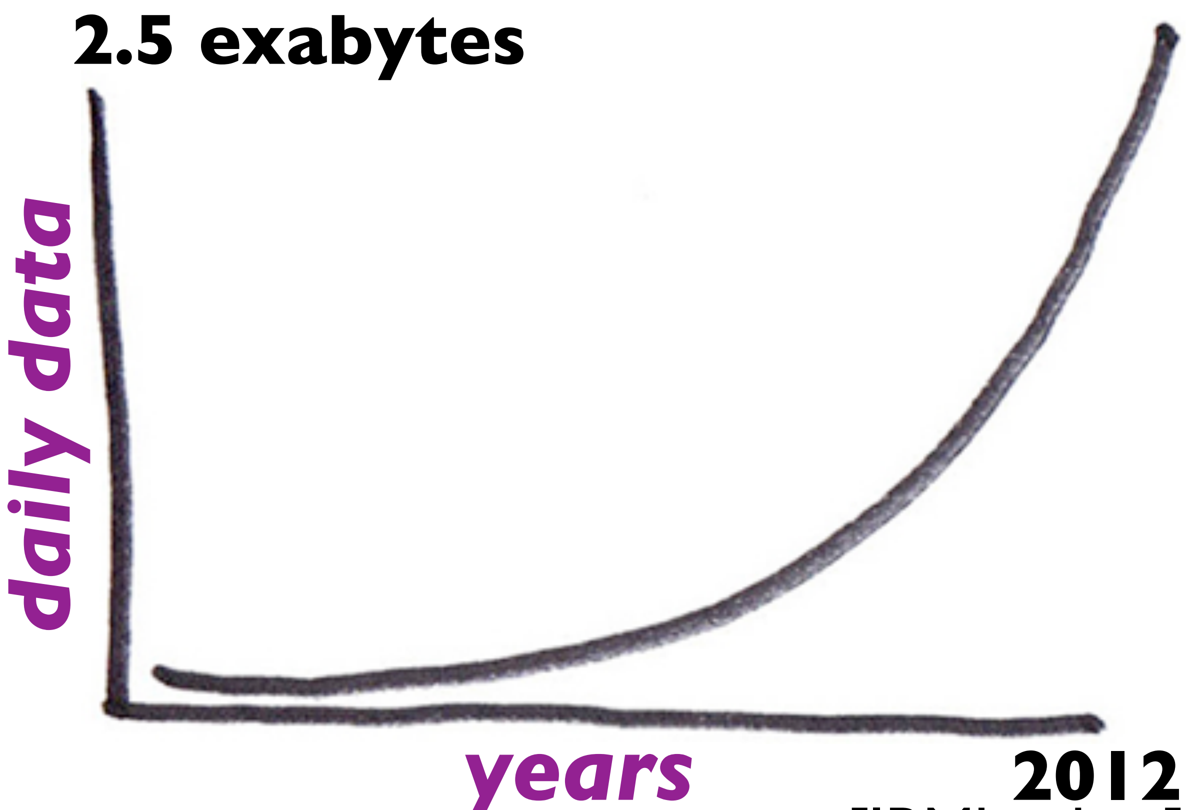
Database Architectures for Big Data Exploration

Stratos Idreos

CWI

HARVARD School of Engineering and Applied Sciences

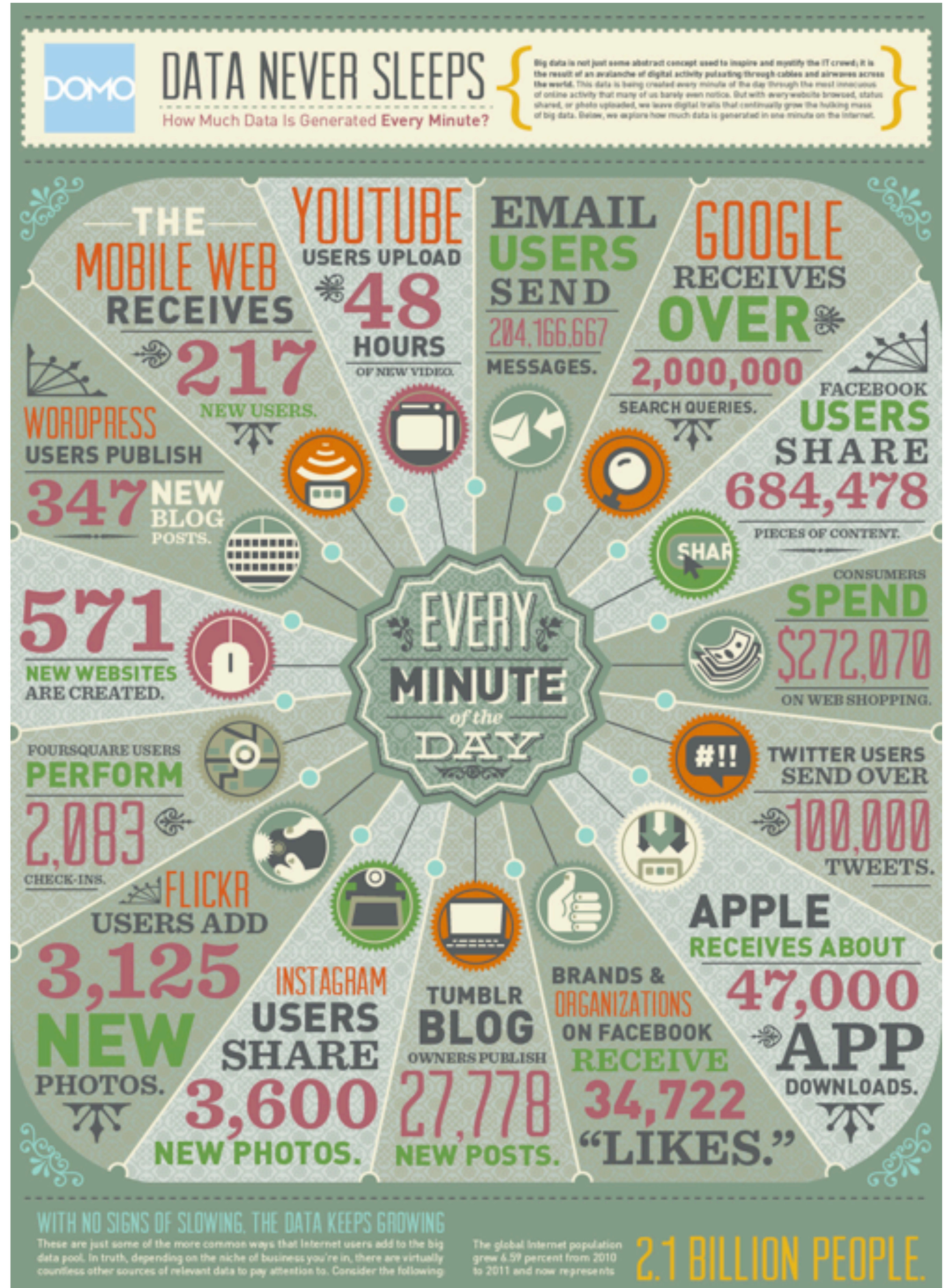
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE



[IBMbigdata]

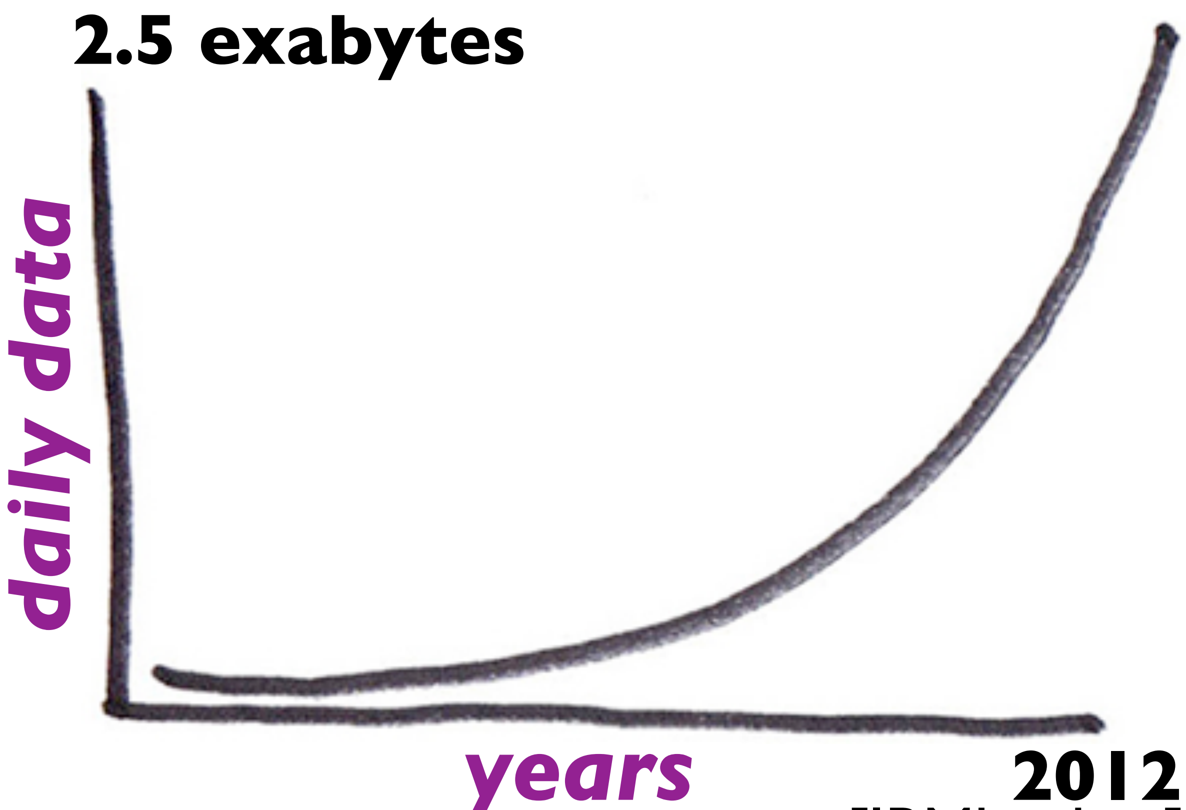
Every two days we create as much data as much we did from dawn of humanity to 2003

[Eric Schmidt]



[Domo]

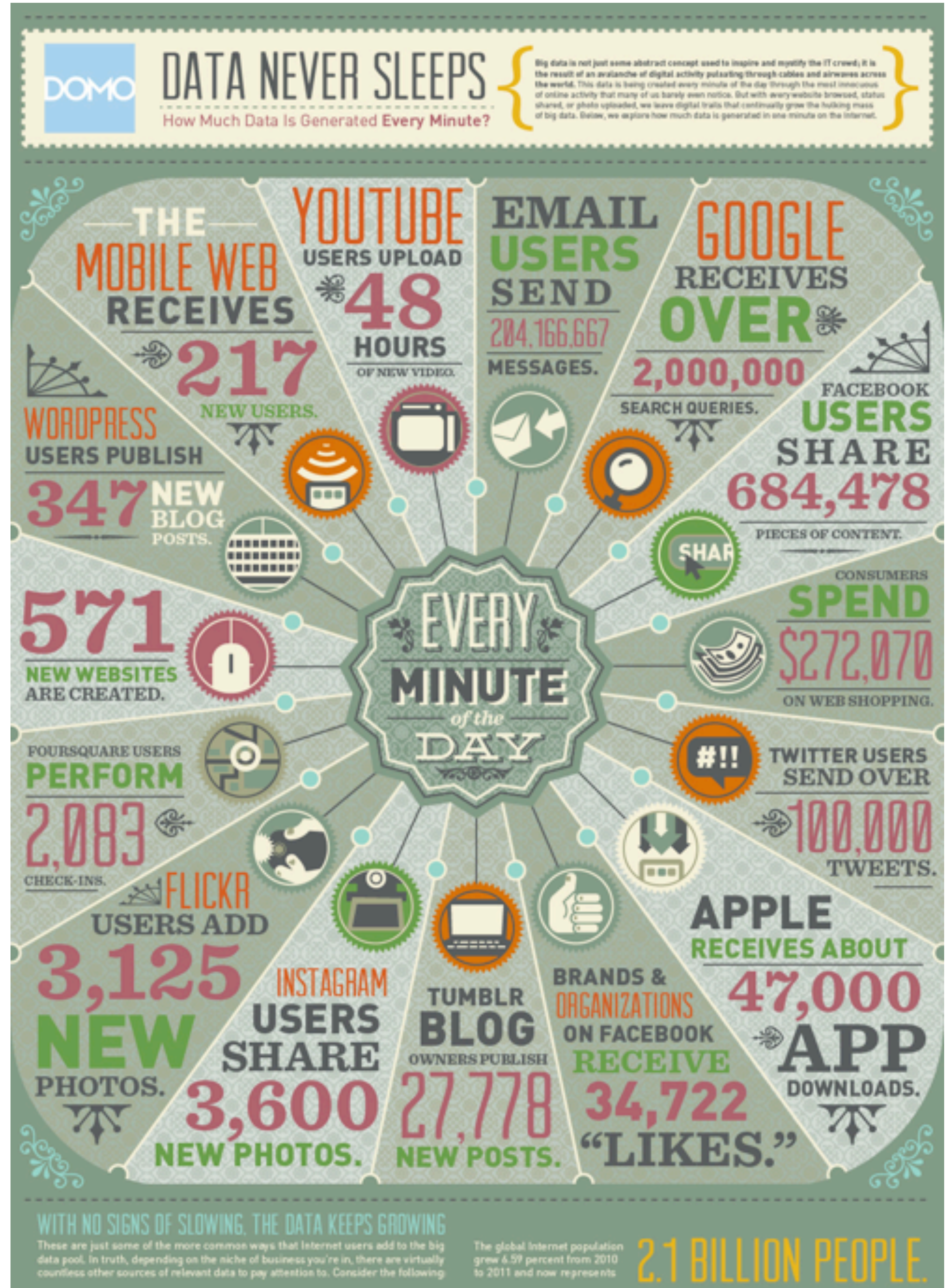
make it easy to turn data into knowledge



[IBMbigdata]

Every two days we create as much data as much we did from dawn of humanity to 2003

[Eric Schmidt]



[Domo]

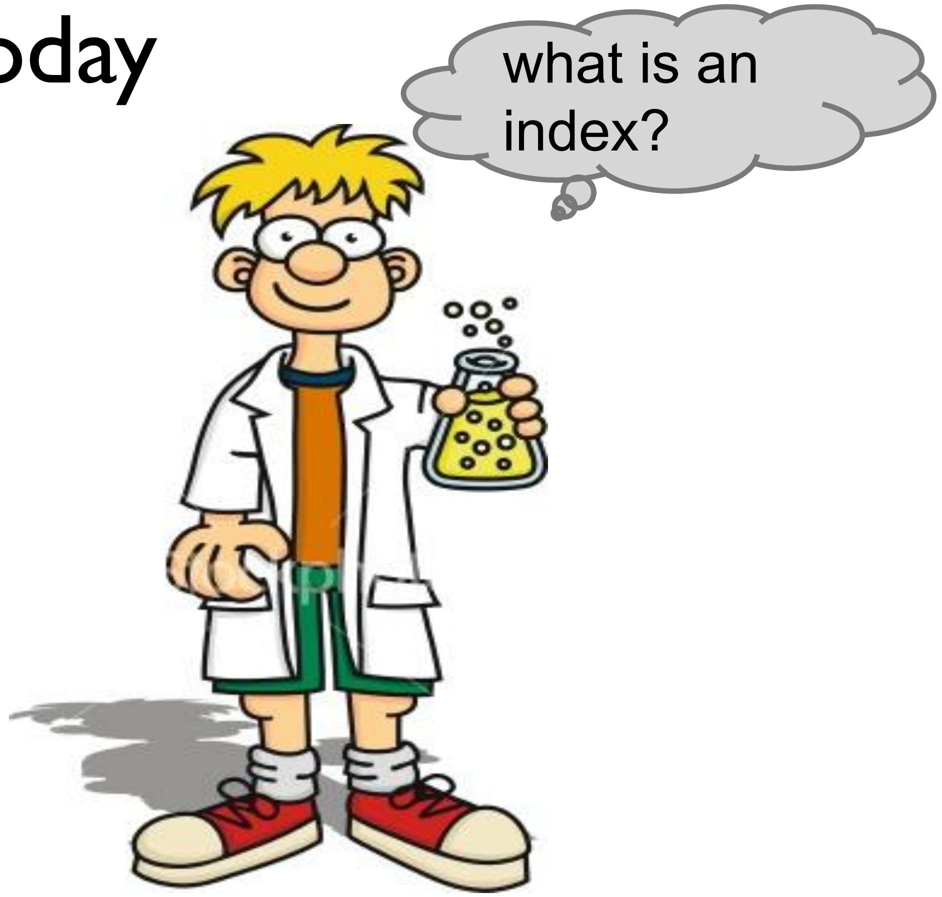
data exploration

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

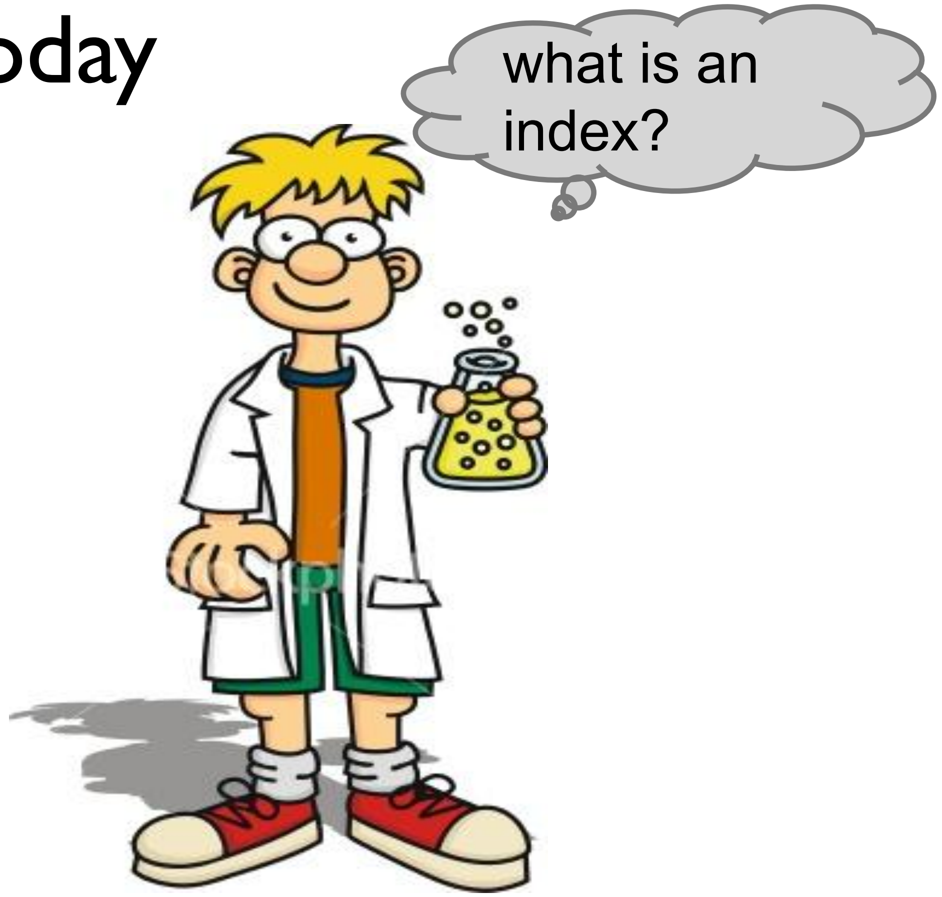
Big Data V's

- volume
- velocity
- variety
- veracity

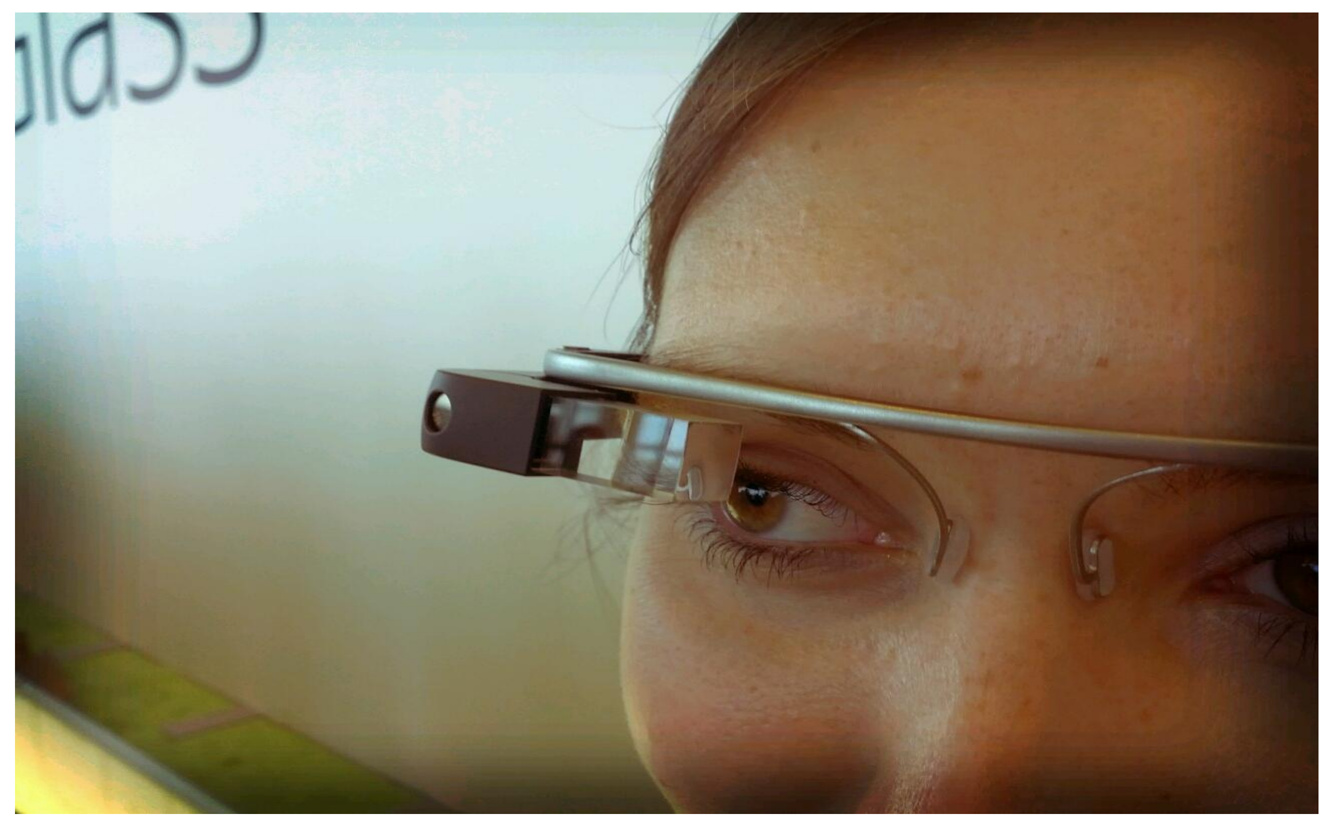
today



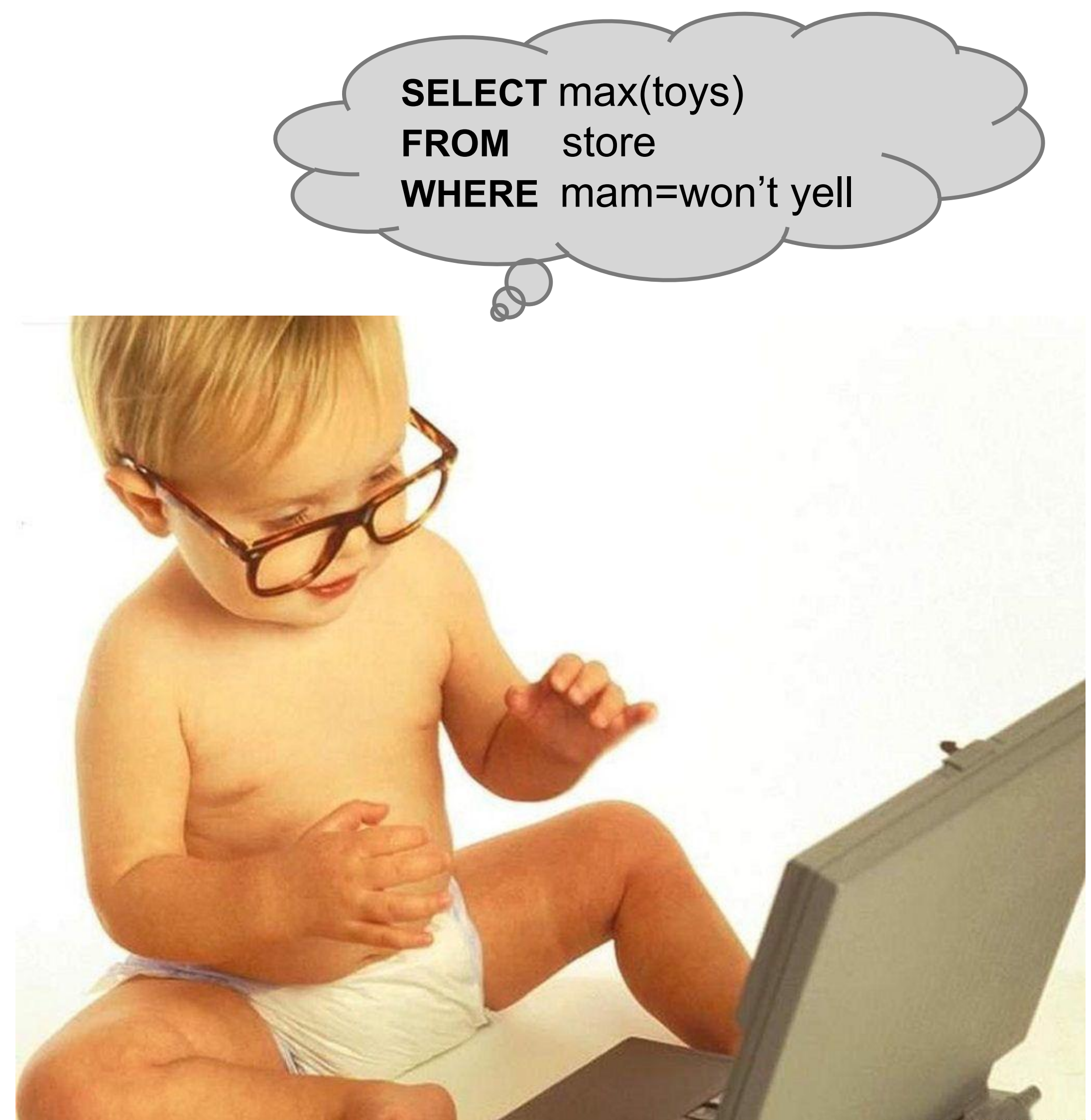
today



tomorrow



everybody will need to be a "data scientist"



WIRED MAGAZINE: 16.07

SCIENCE : DISCOVERIES

The End of Theory: The Data Deluge Makes the Scientific Method Obsolete

By Chris Anderson 06.23.08

Illustration: Marian Bantjes

THE PETABYTE AGE:

"**All models are wrong,** but some are useful."

it is time for a paradigm shift in how we design databases systems

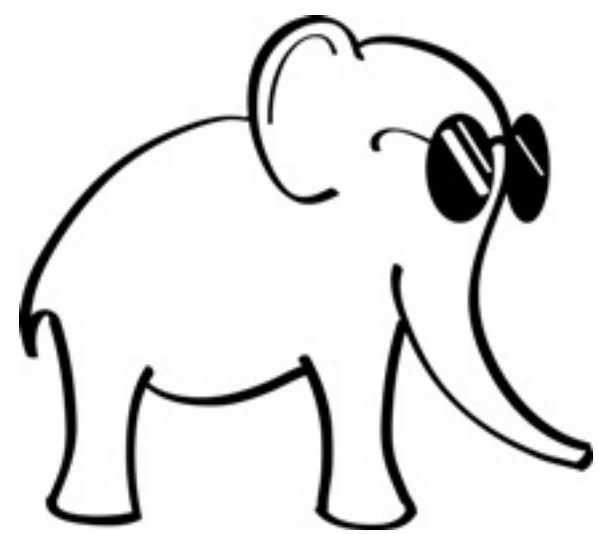
database systems great...

declarative processing, back-end to numerous apps

database systems great...

declarative processing, back-end to numerous apps

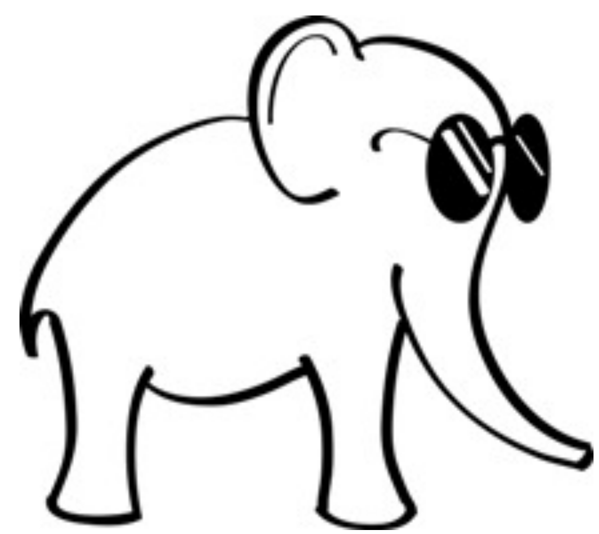
but databases have become too heavy and blind!



database systems great...

declarative processing, back-end to numerous apps

but databases have become too heavy and blind!

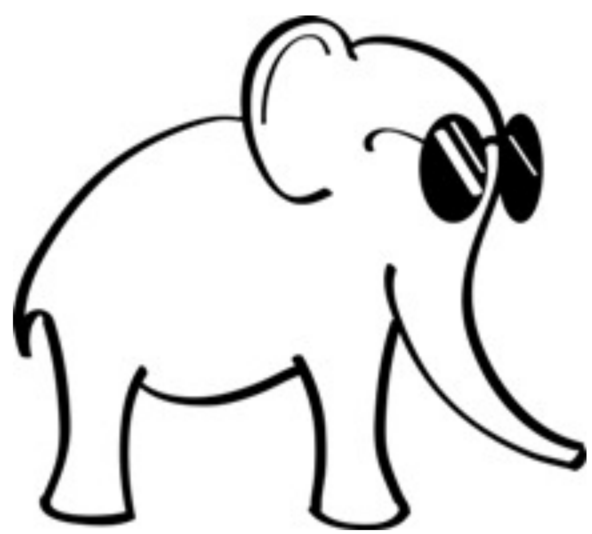


timeline

database systems great...

declarative processing, back-end to numerous apps

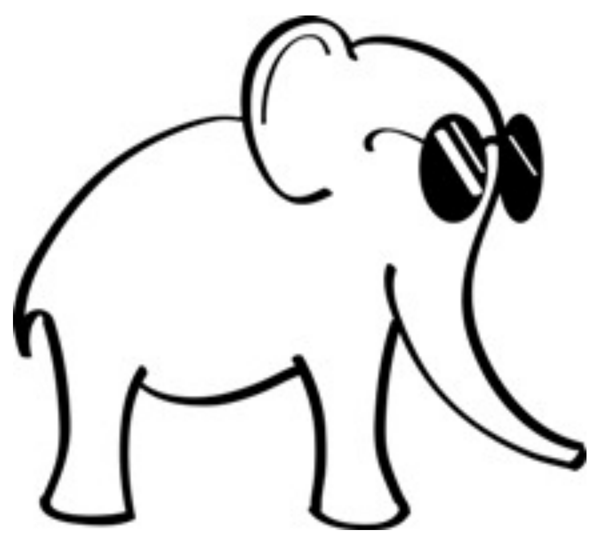
but databases have become too heavy and blind!

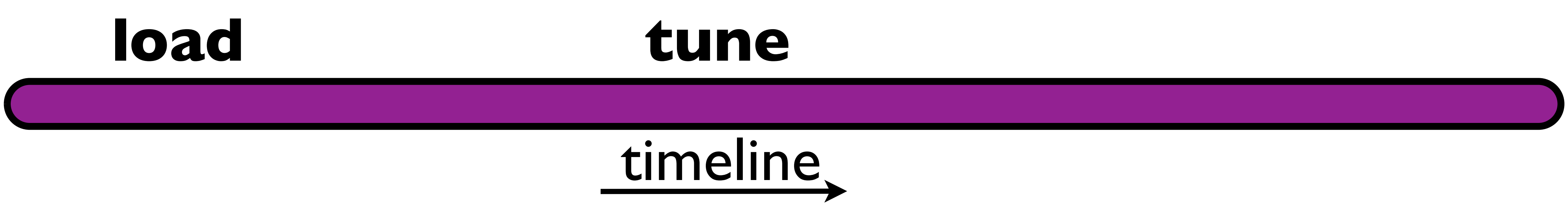


database systems great...

declarative processing, back-end to numerous apps

but databases have become too heavy and blind!

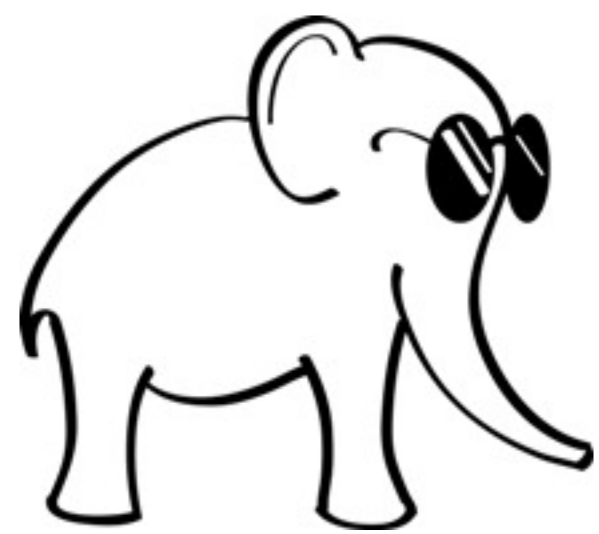


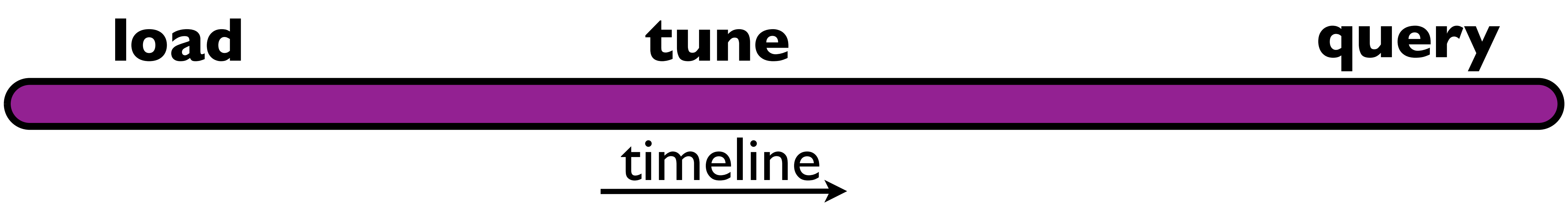


database systems great...

declarative processing, back-end to numerous apps

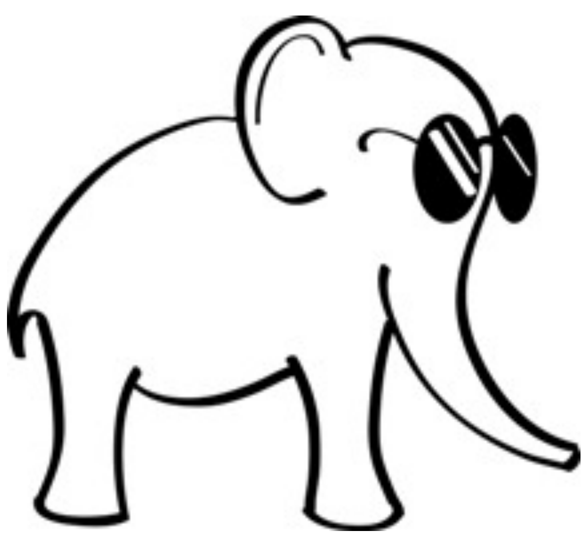
but databases have become too heavy and blind!





expert users - idle time - workload knowledge

but databases have become too heavy and blind!



load **tune** **query**

timeline

data systems tailored for data exploration

no workload knowledge

no installation steps

data systems tailored for data exploration no workload knowledge

no installation steps

minimize data-to-query time

data systems tailored for data exploration

adaptive indexing

adaptive loading

dbTouch

Martin Kersten, Stefan Manegold, Felix Halim, Panagiotis Karras, Roland Yap, Goetz Graefe, Harumi Kuno, Eleni Petraki, Themis Palpanas, Kostas Zoumpatianos, Anastasia Ailamaki, Ioannis Alagiannis, Renata Borovica, Miguel Branco, Ryan Johnson, Erietta Liarou

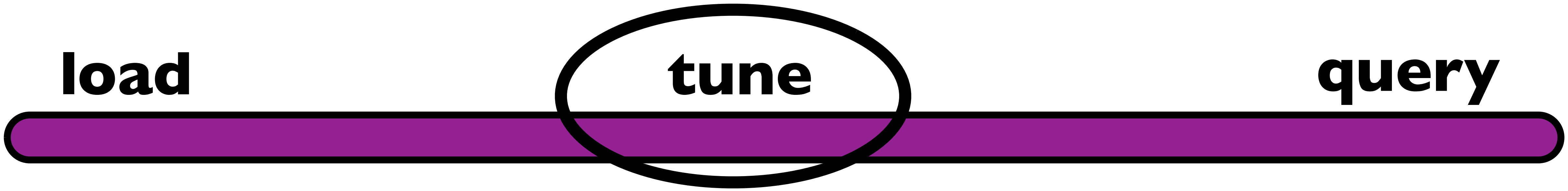
3 ideas

load — tune — query

adaptive loading — **adaptive indexing** — **dbTouch**

Martin Kersten, Stefan Manegold, Felix Halim, Panagiotis Karras, Roland Yap, Goetz Graefe, Harumi Kuno, Eleni Petraki, Themis Palpanas, Kostas Zoumpatianos, Anastasia Ailamaki, Ioannis Alagiannis, Renata Borovica, Miguel Branco, Ryan Johnson, Erietta Liarou

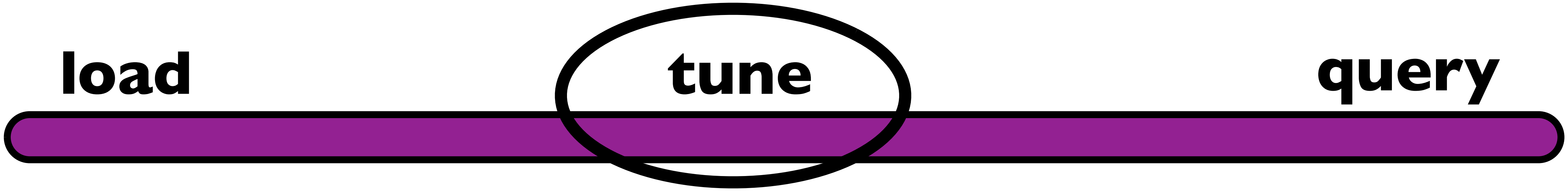
indexing



tune= create proper indices offline

performance 10-100X

indexing

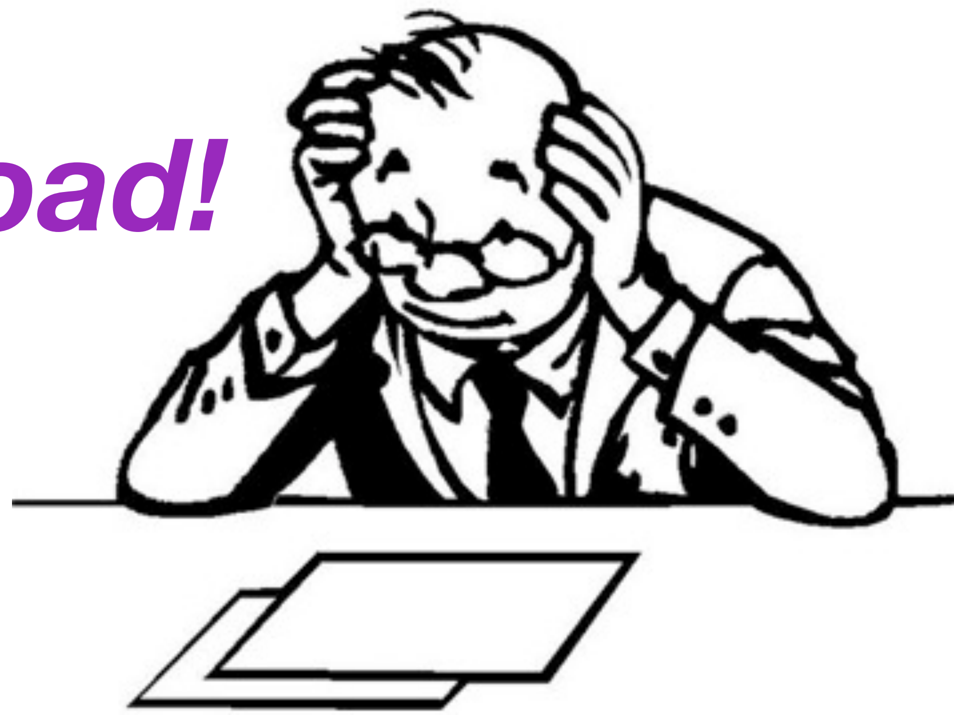


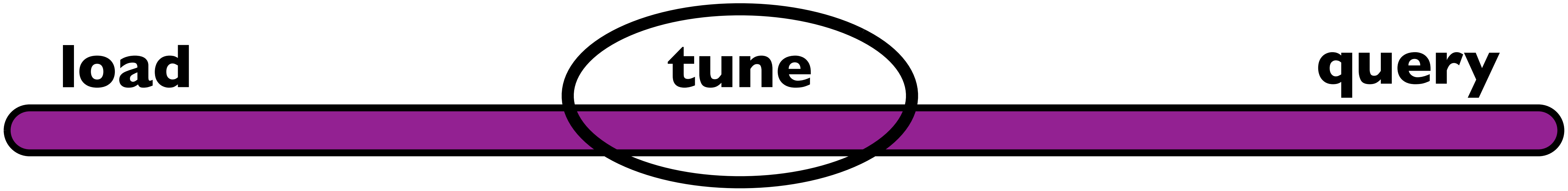
tune= create proper indices offline

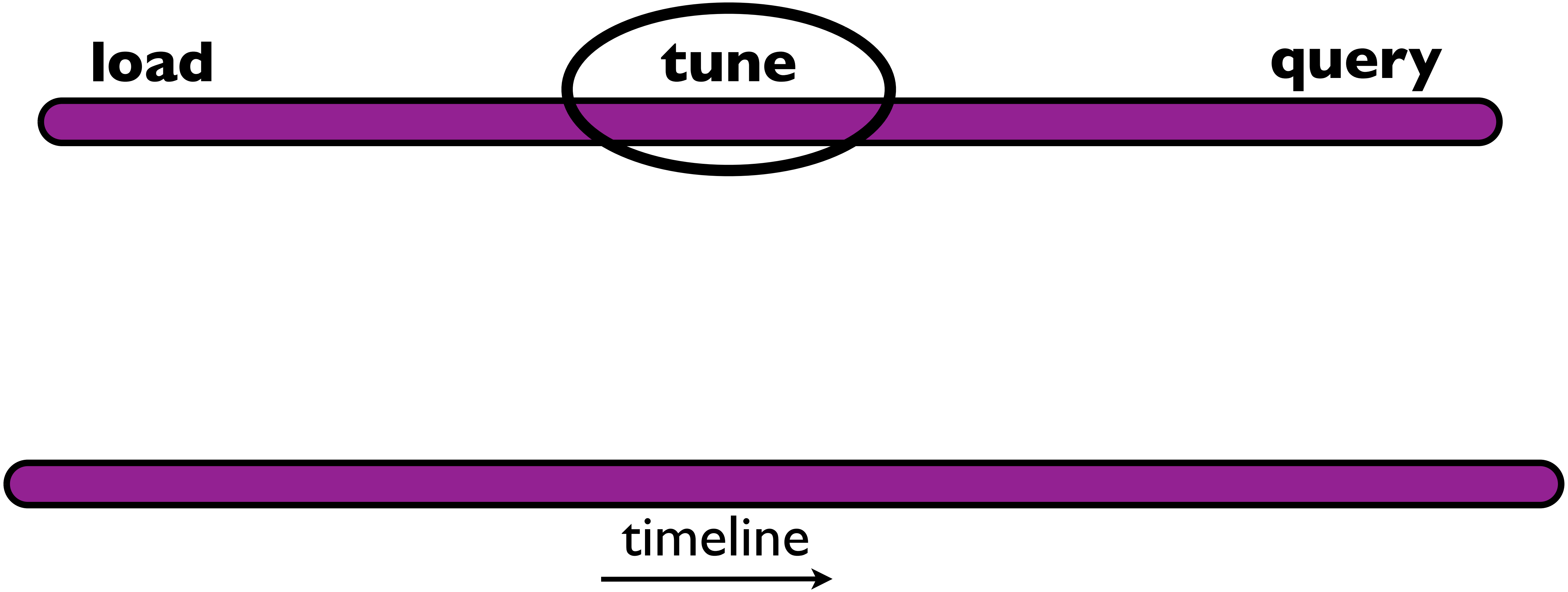
performance 10-100X

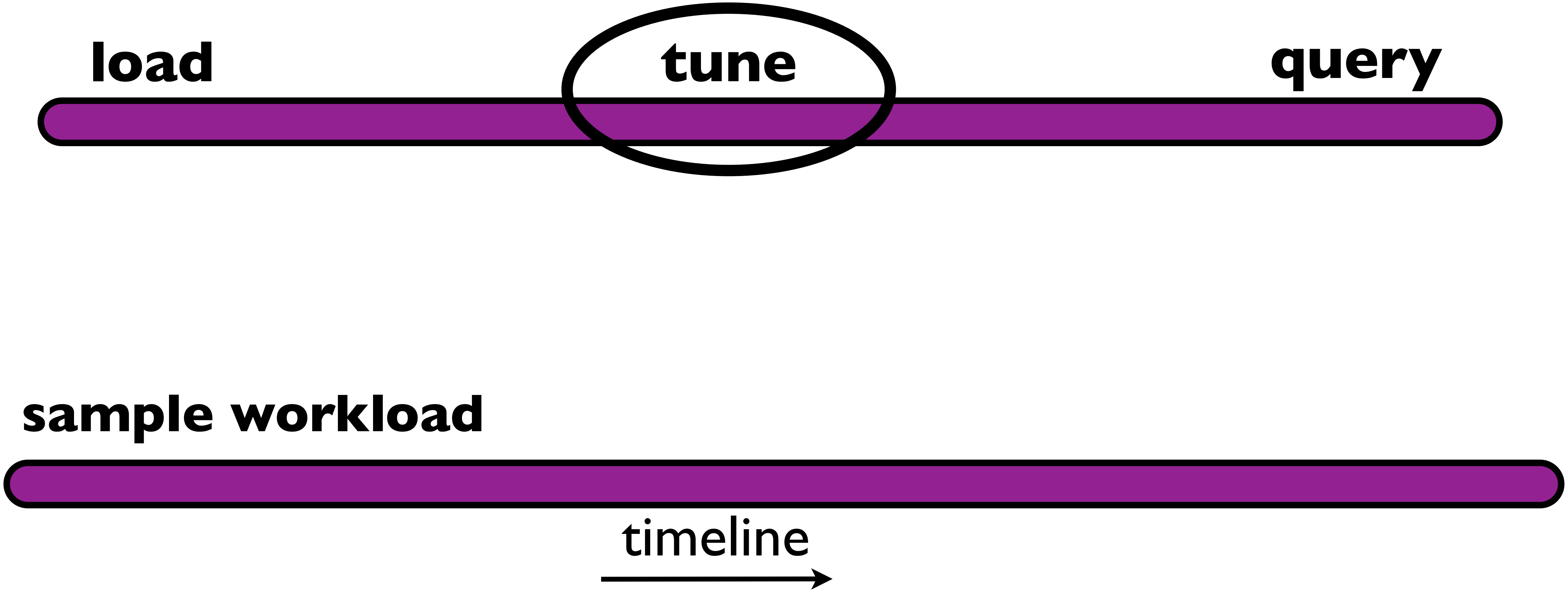
but it depends on workload!

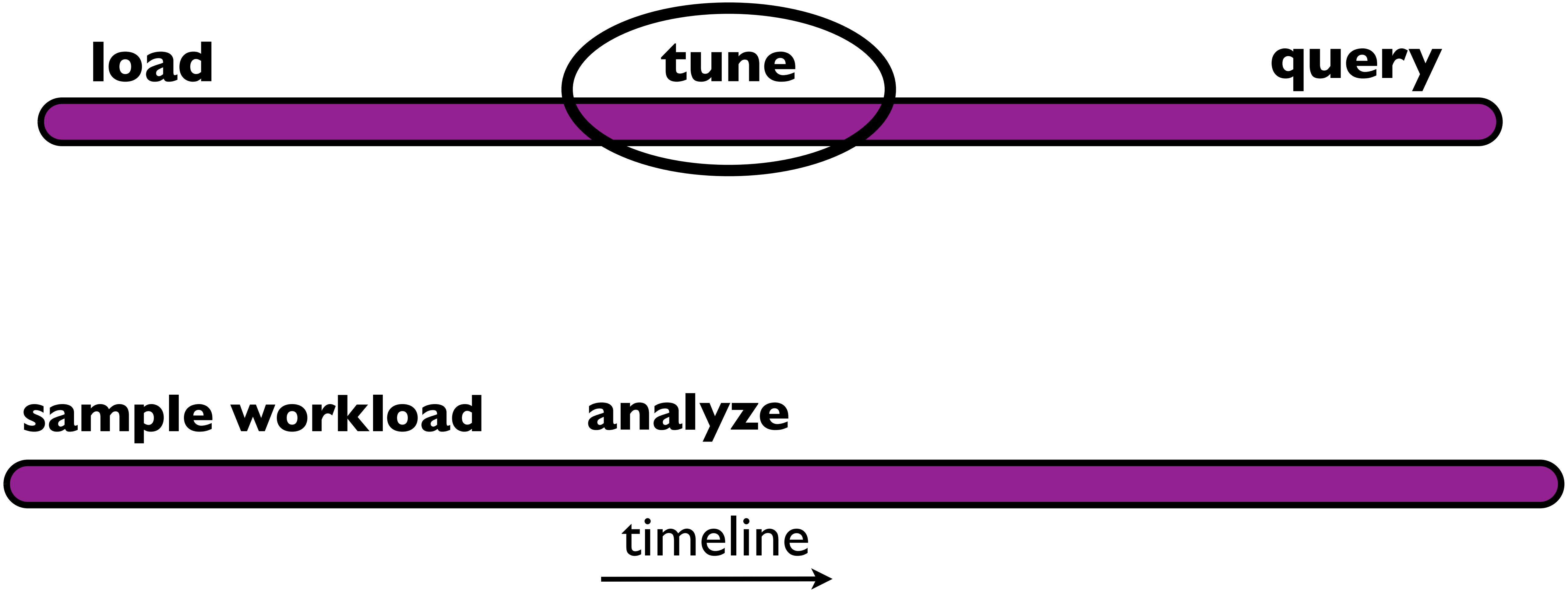
which indices to build?
on which data parts?
and when to build them?

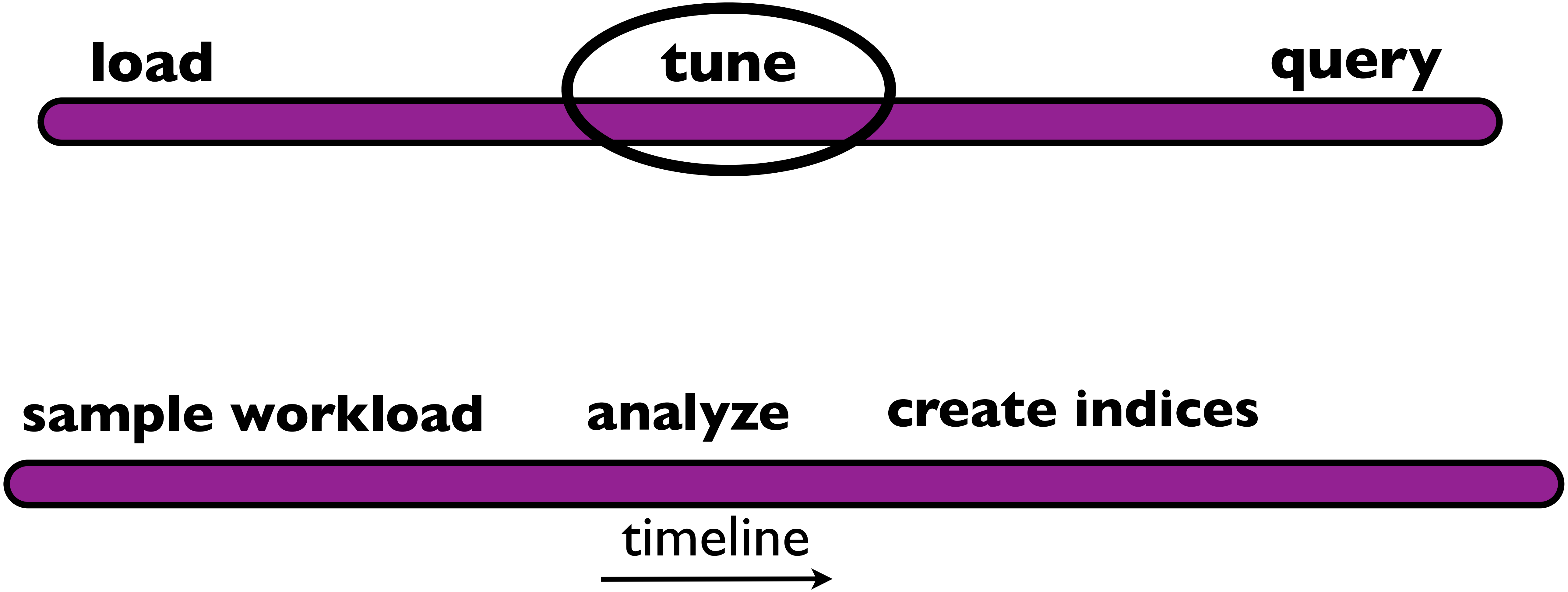


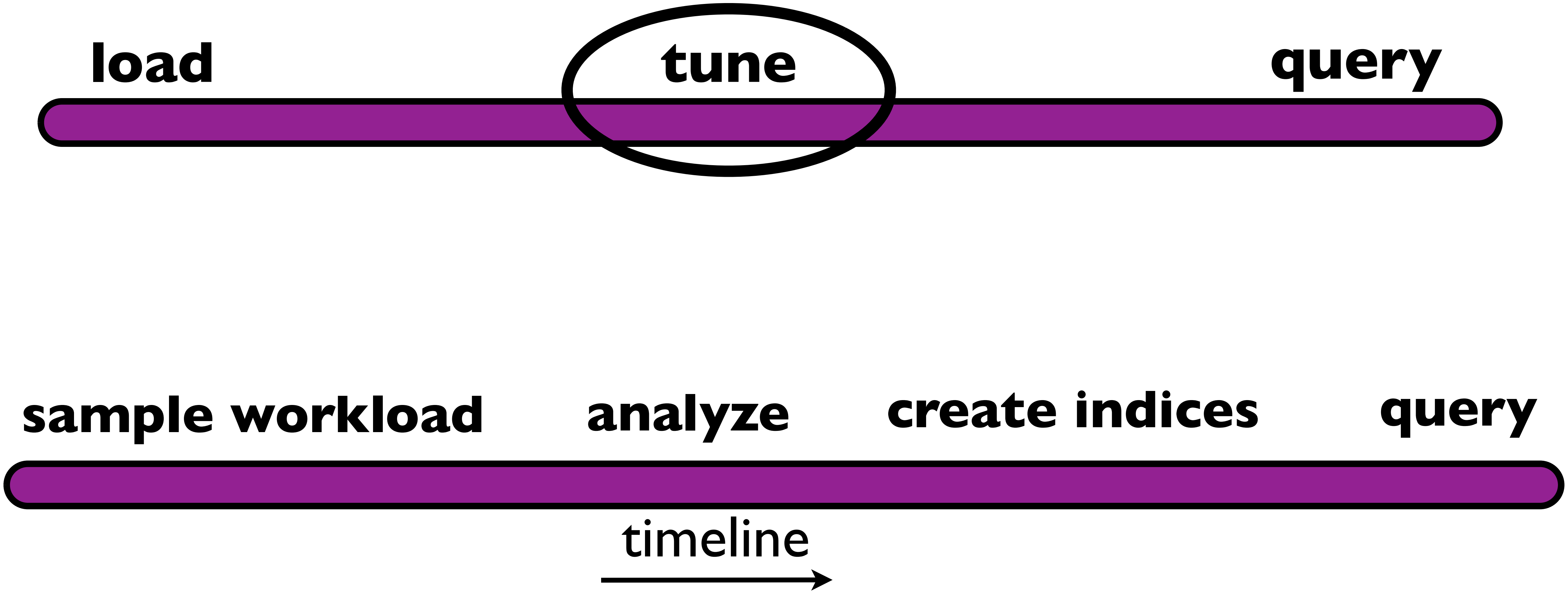

load
tune
query

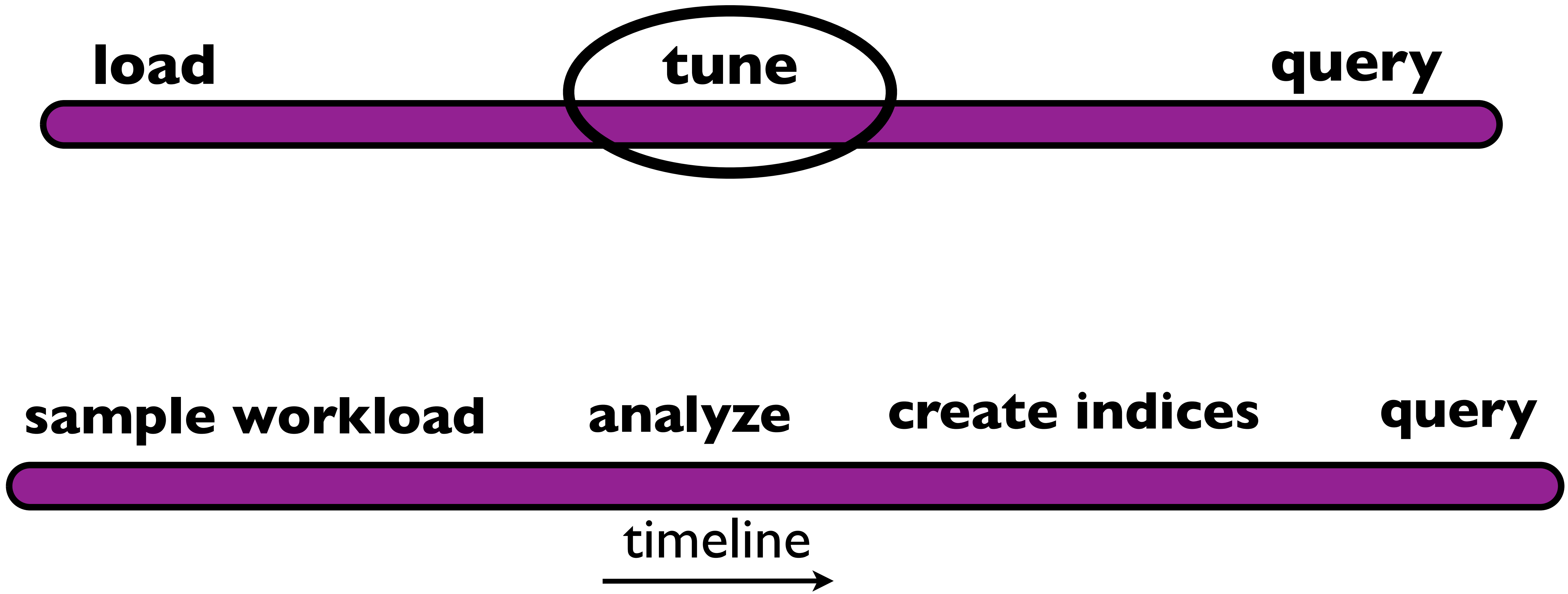

load
tune
query
timeline


load
tune
query
sample workload
timeline


load
tune
query
sample workload
analyze
timeline


load
tune
query
sample workload
analyze
create indices
timeline


load
tune
query
sample workload
analyze
create indices
query
timeline



complex and time consuming process

human administrators + auto-tuning tools

sample workload **analyze** **create indices** **query**

timeline →

complex and time consuming process

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

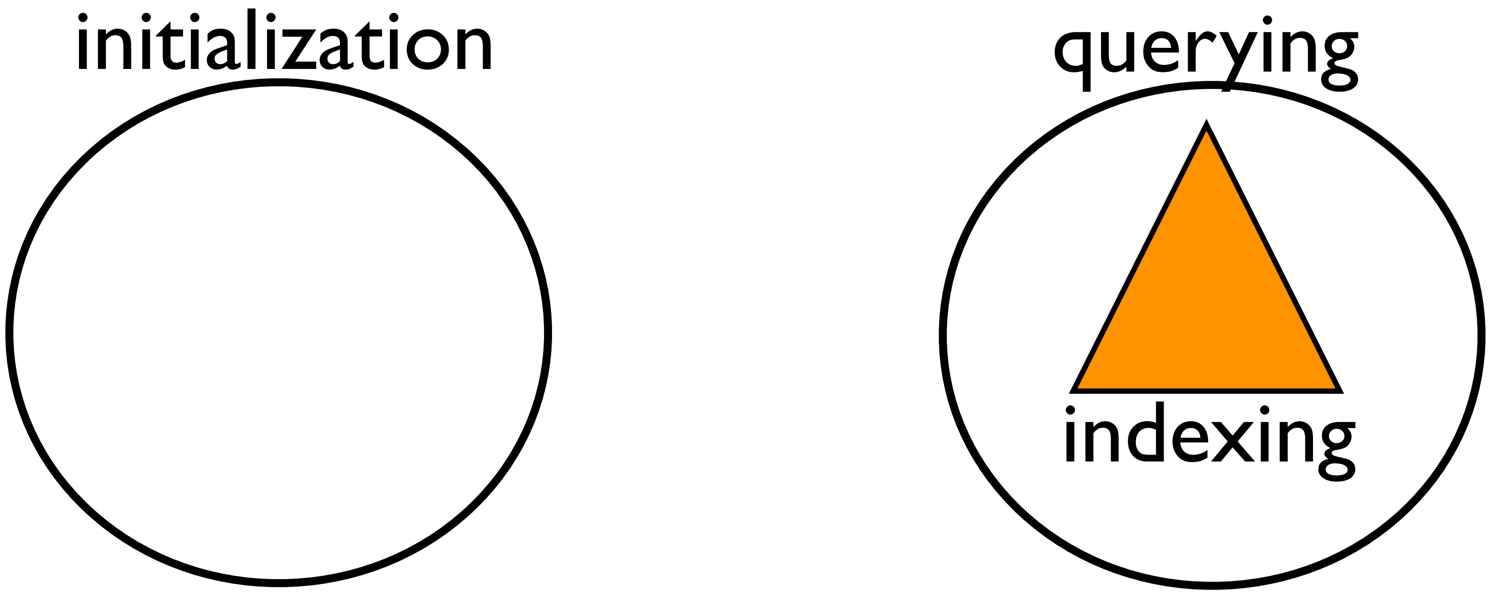
incremental, adaptive, partial indexing

~~idle time~~

~~workload knowledge~~

~~external tools~~

~~human control~~



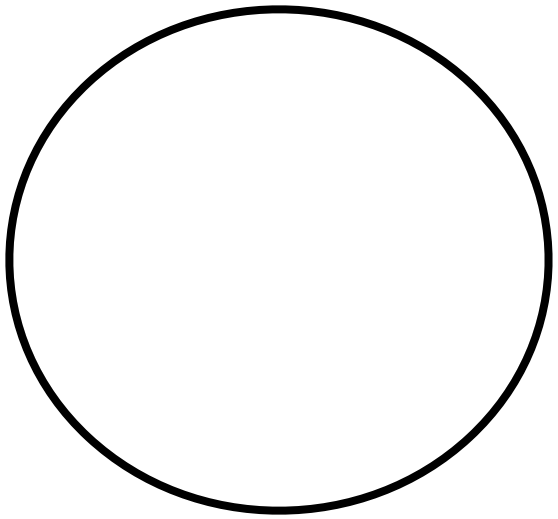
database cracking

auto-tuning database kernels

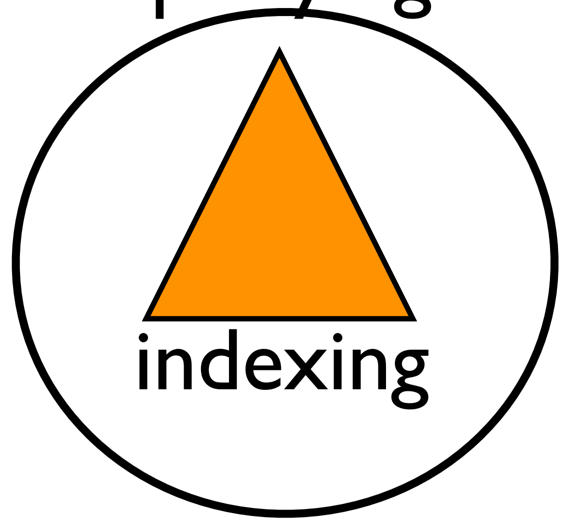
incremental, adaptive, partial indexing

~~idle time~~ ~~workload knowledge~~ ~~external tools~~ ~~human control~~

initialization



querying



database cracking

auto-tuning database kernels

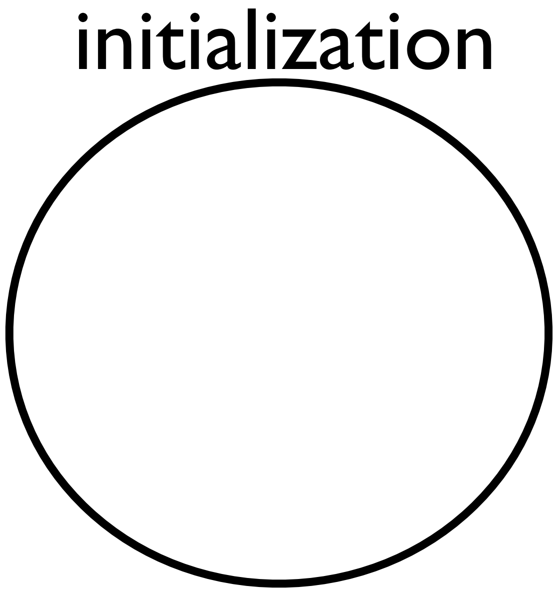
incremental, adaptive, partial indexing

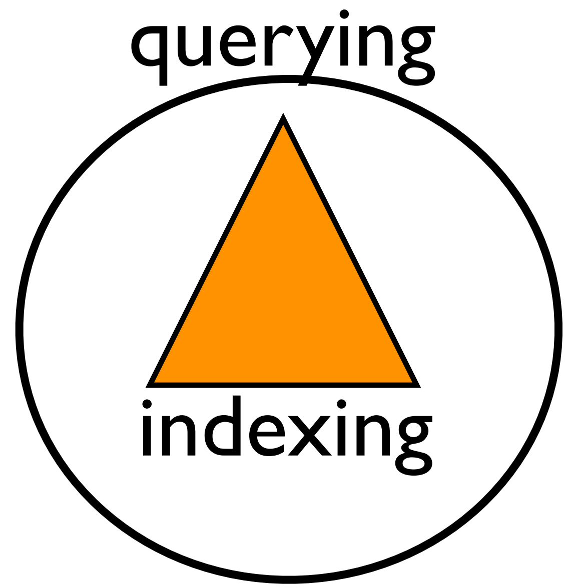
~~idle time~~

~~workload knowledge~~

~~external tools~~

~~human control~~





database cracking

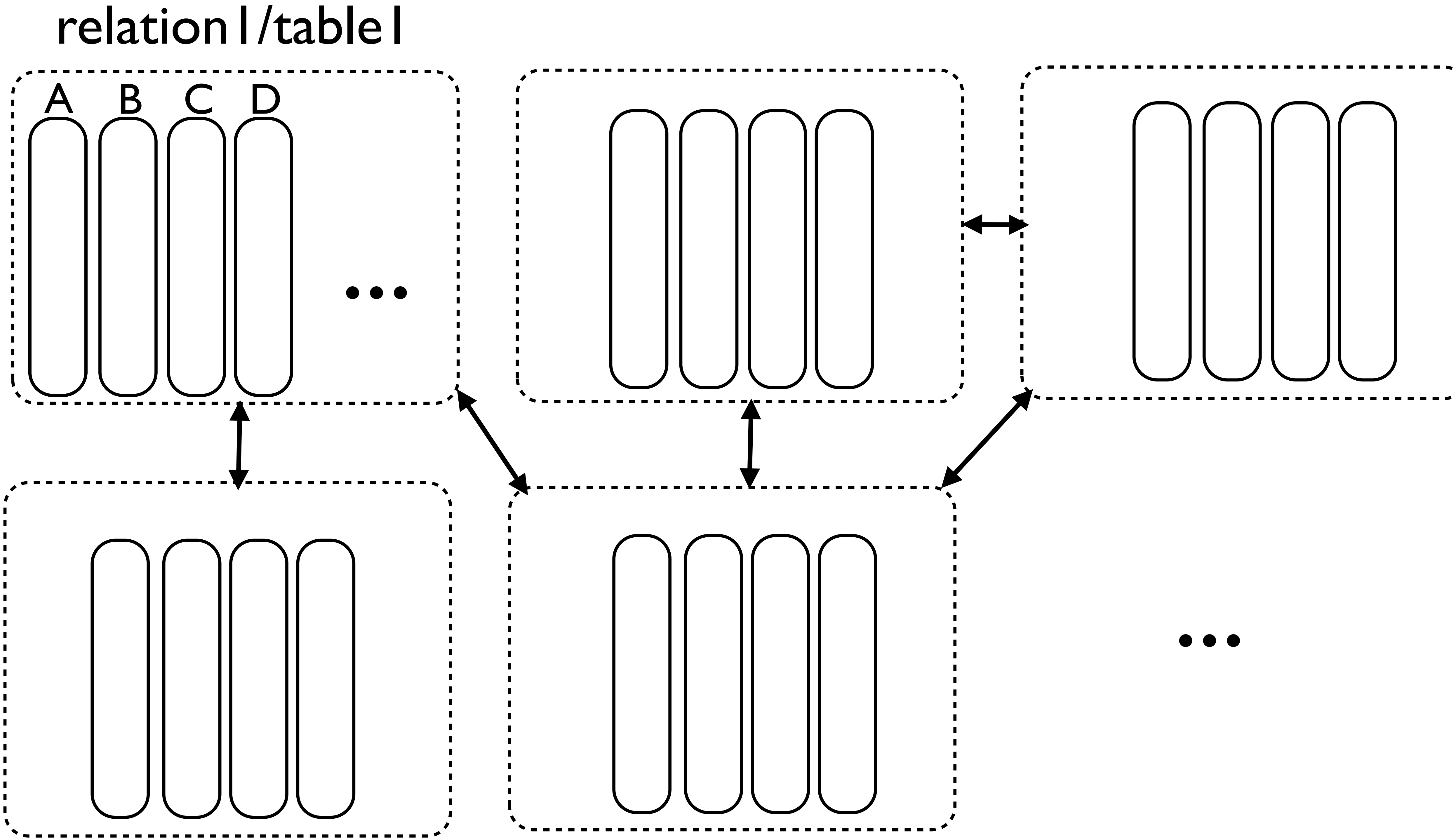
auto-tuning database kernels

incremental, adaptive, partial indexing

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

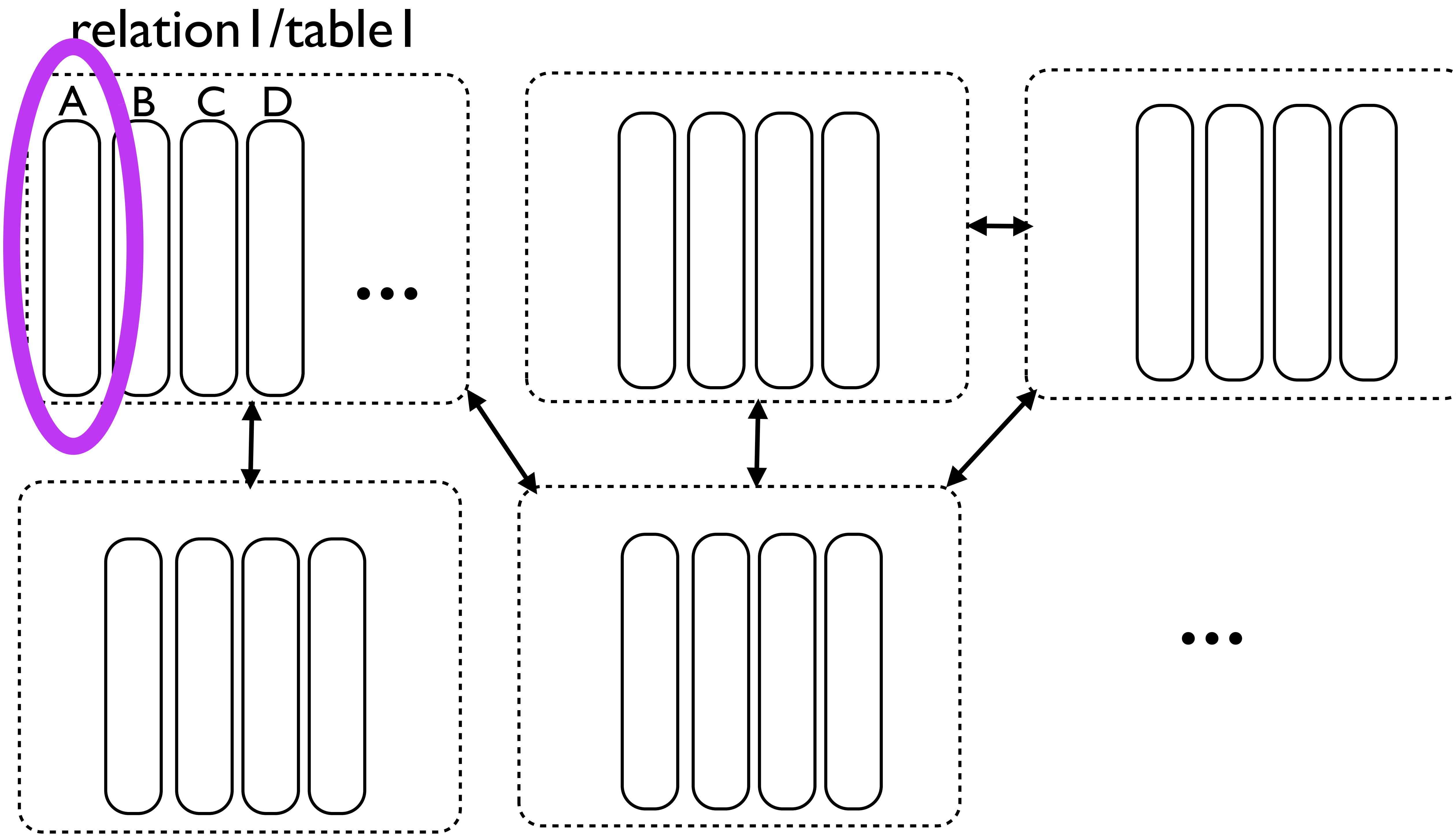
column-store database

a fixed-width and dense array per attribute



column-store database

a fixed-width and dense array per attribute



full indexing example

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

full indexing example

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

full indexing example

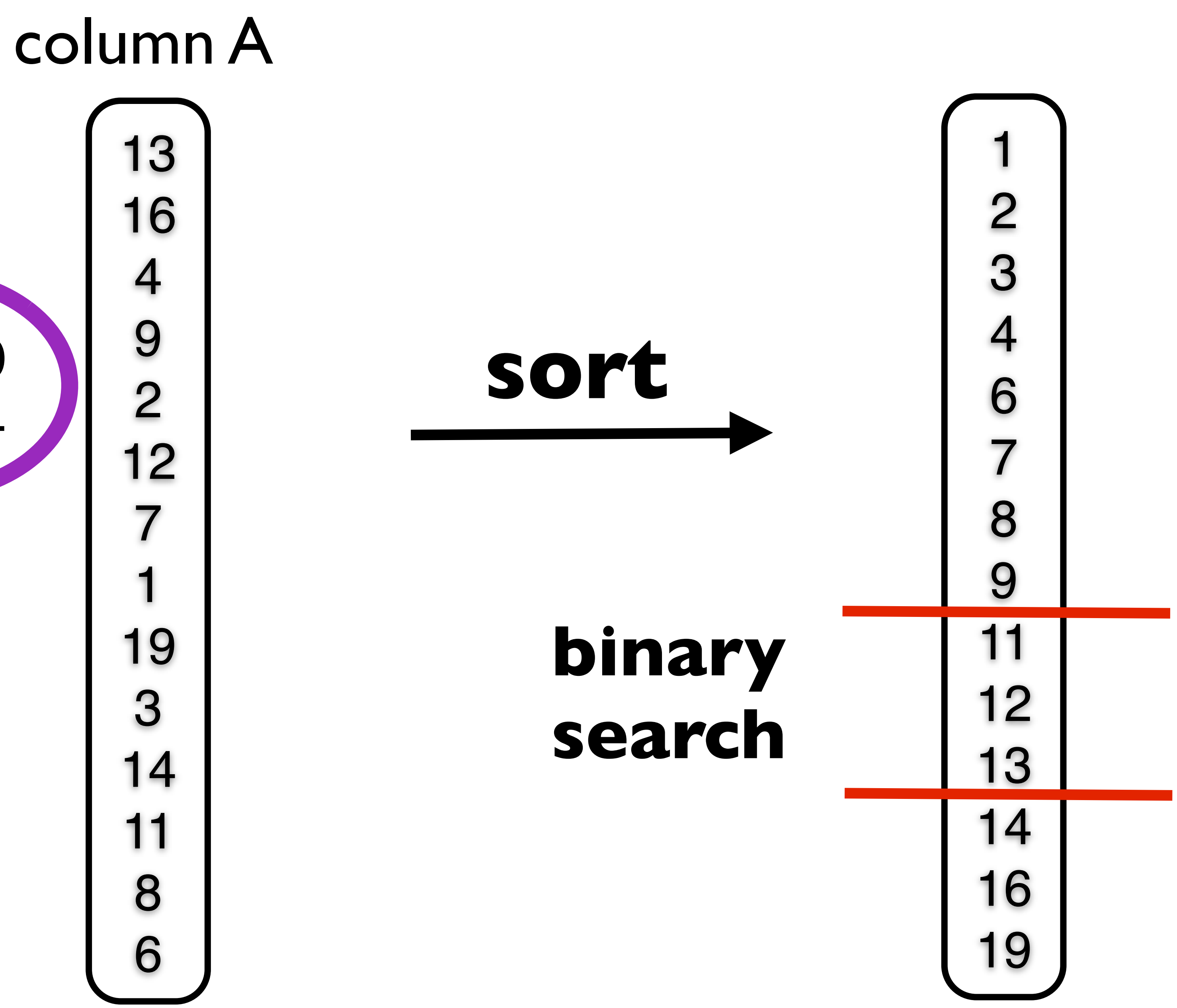
Q1:
select R.A
from R
where R.A>10
and R.A<14

column A

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

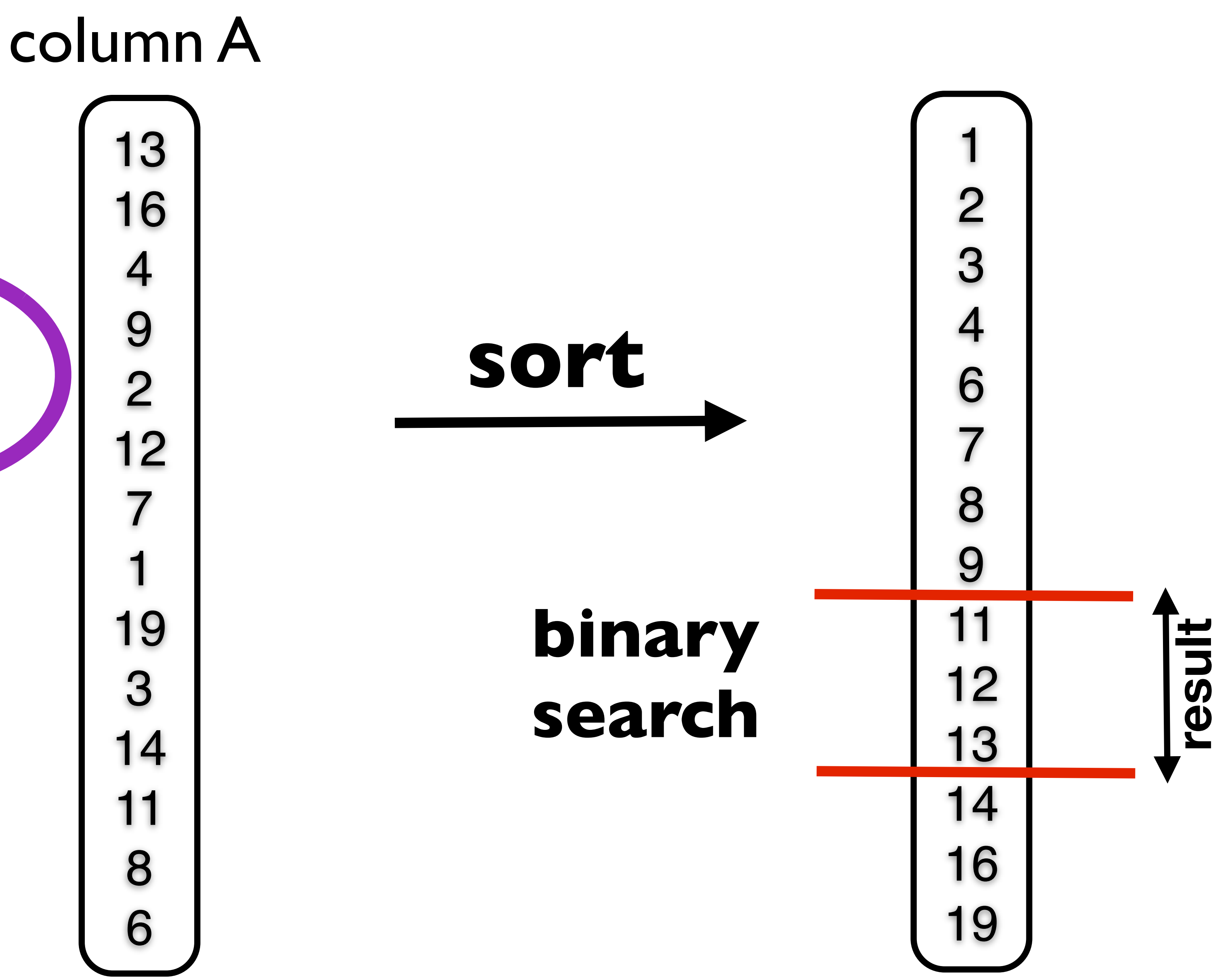
full indexing example

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

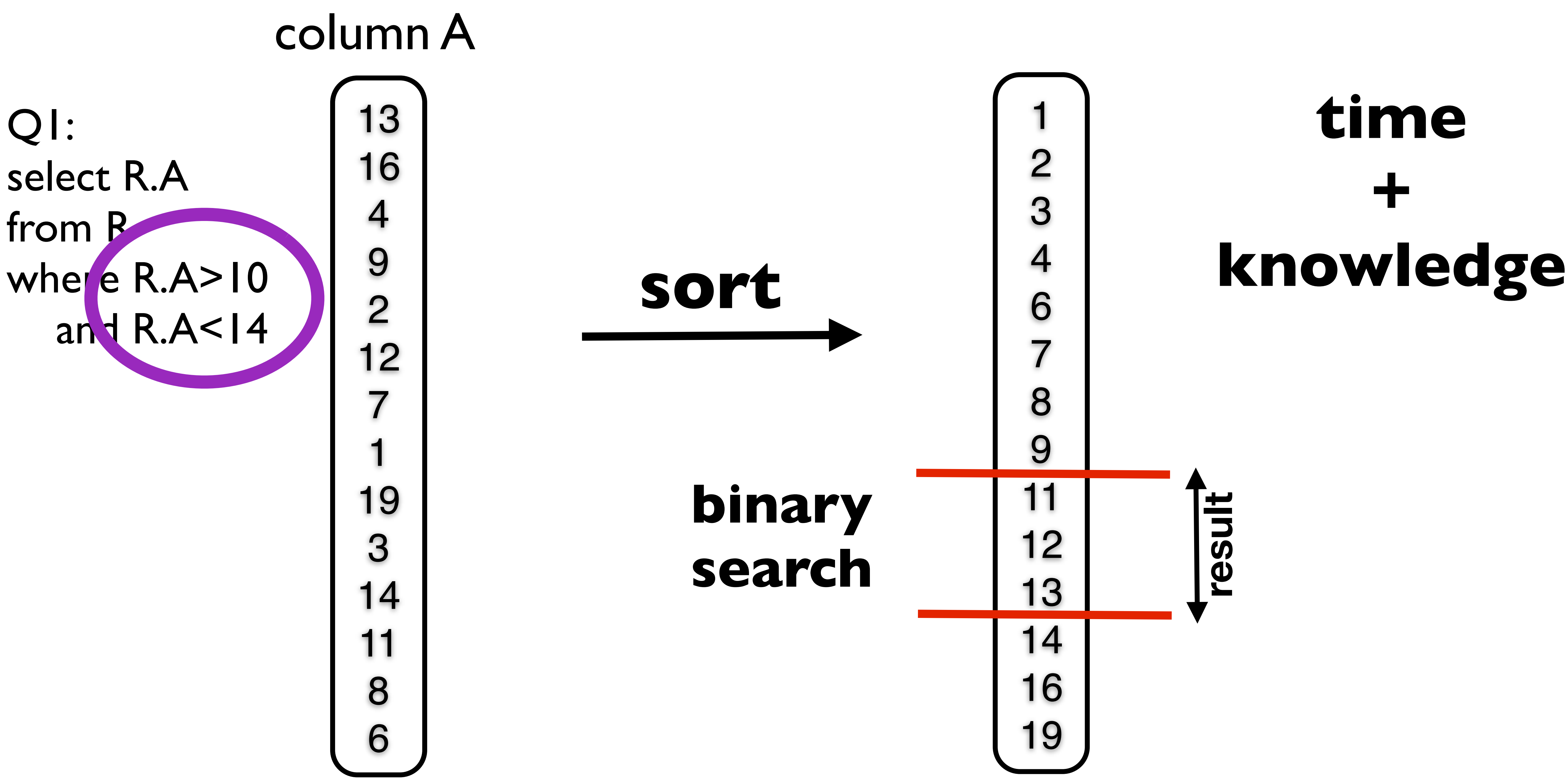


full indexing example

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



full indexing example



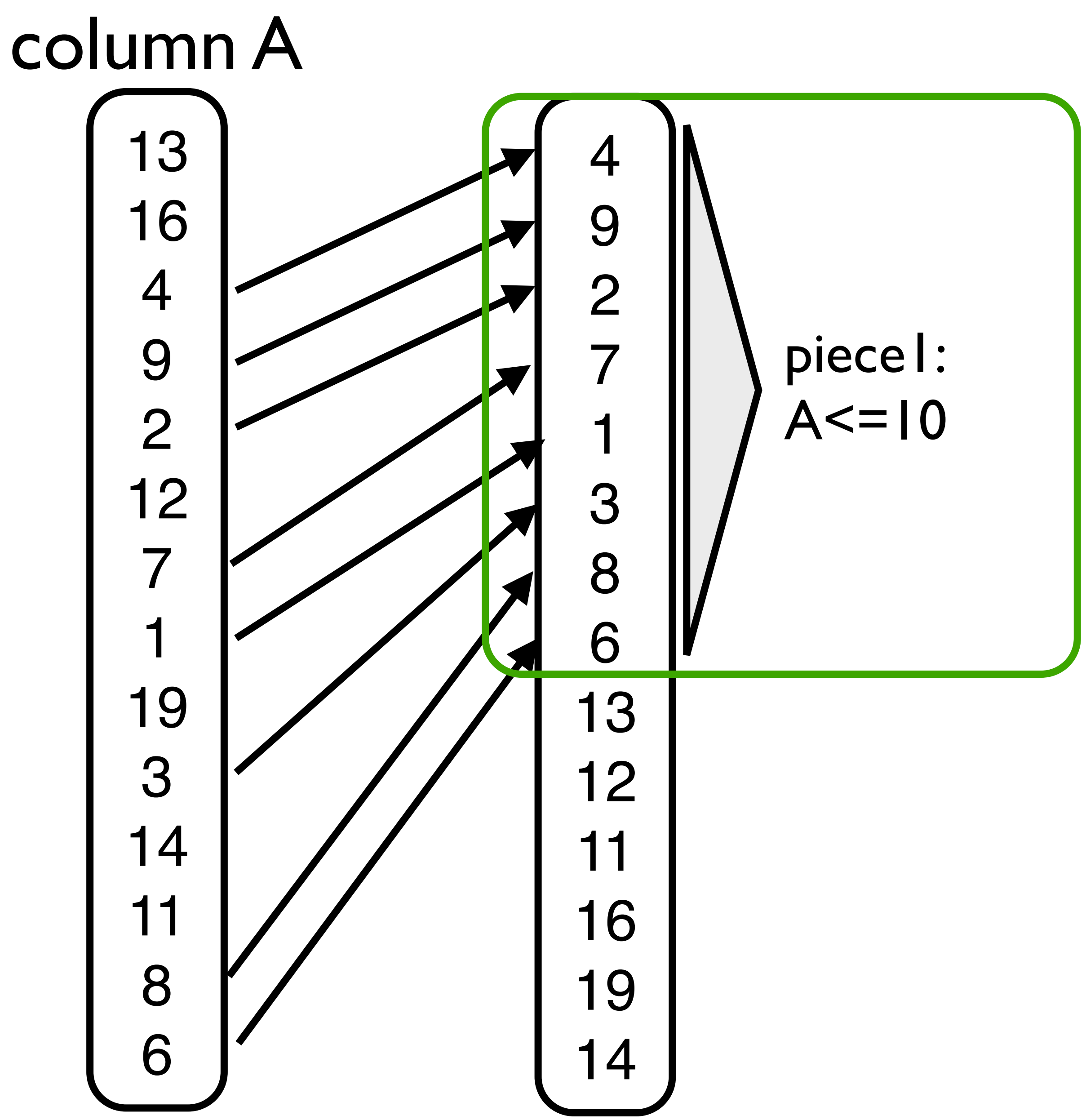
cracking example

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

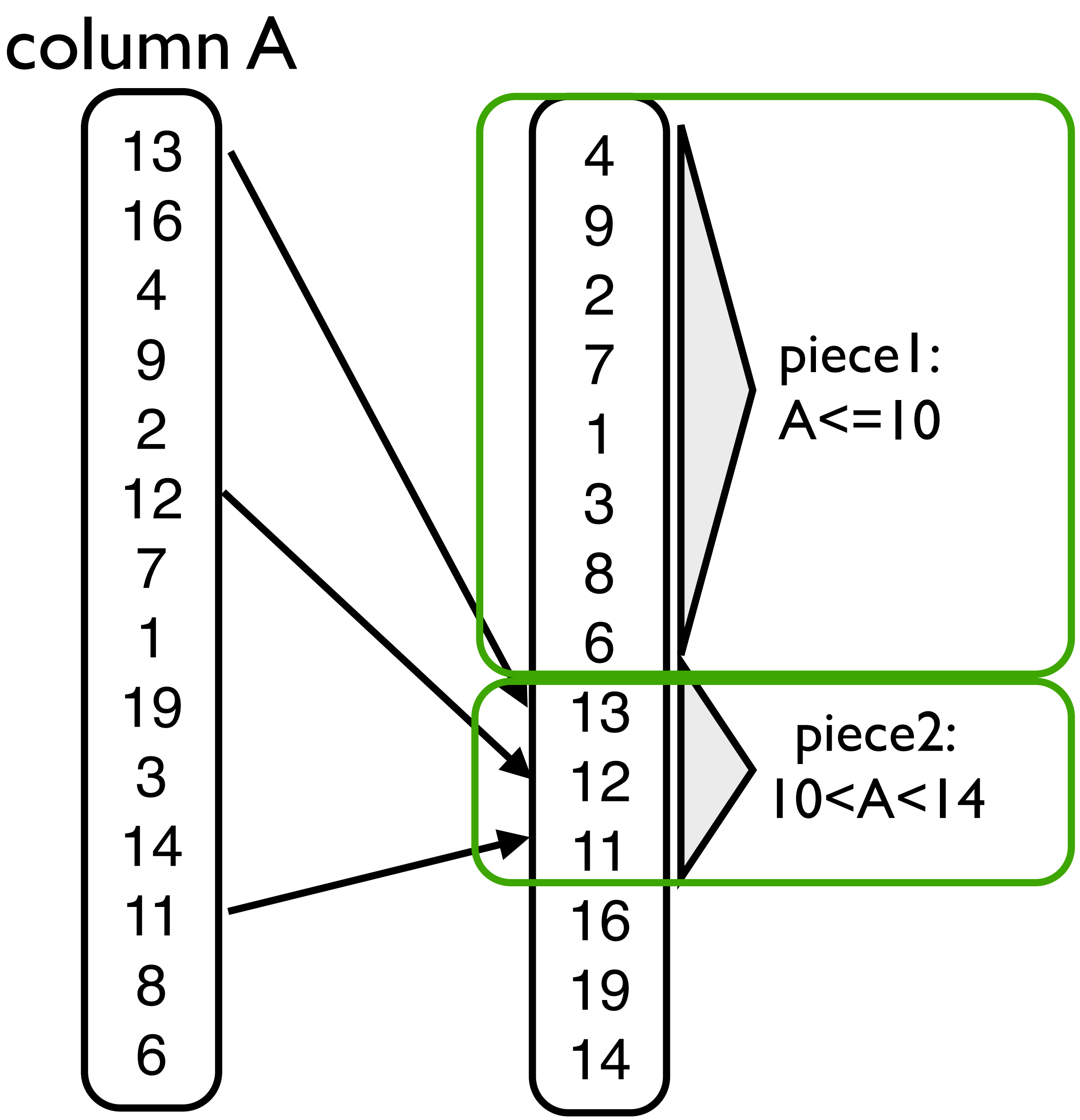
cracking example

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



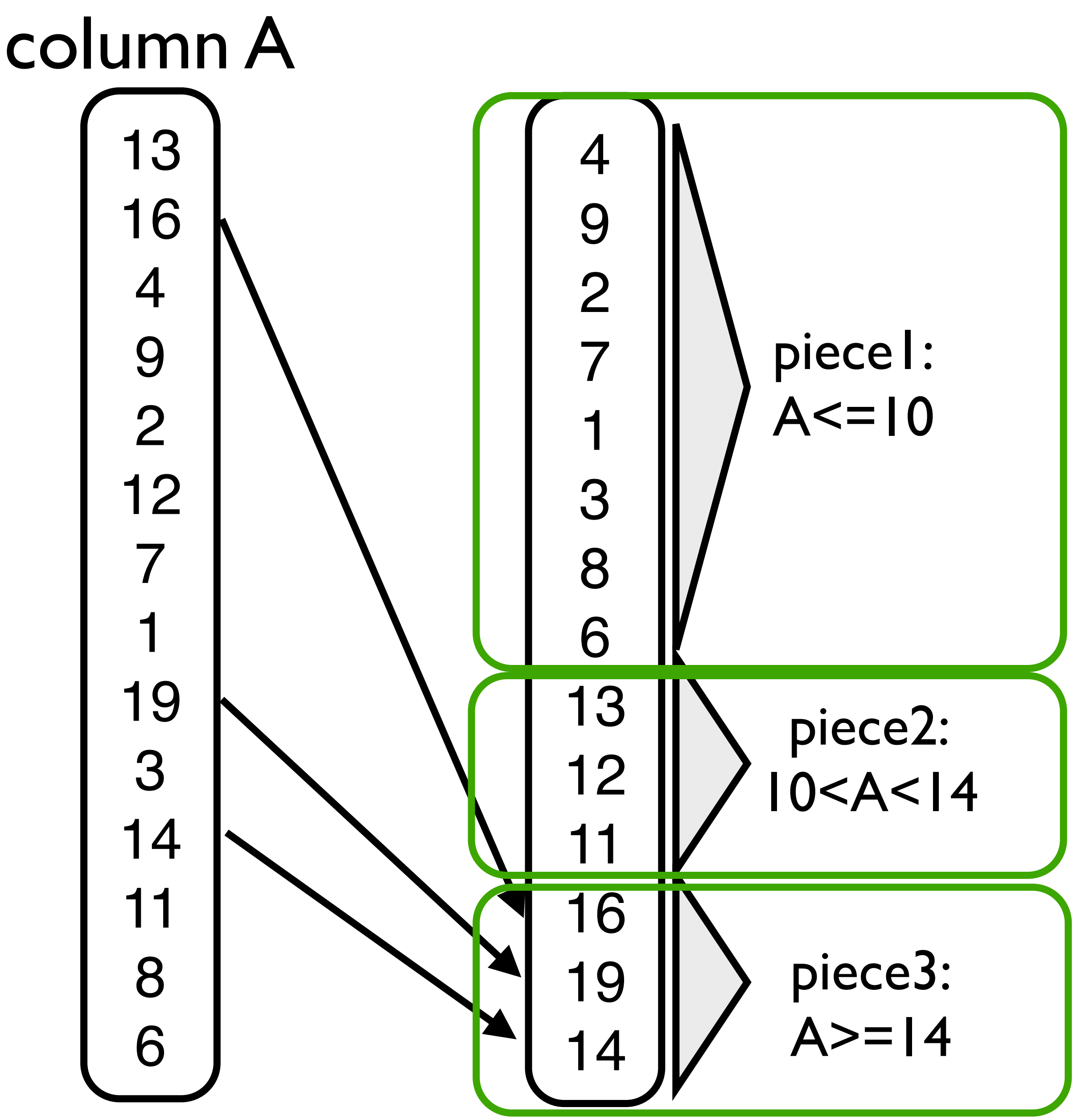
cracking example

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



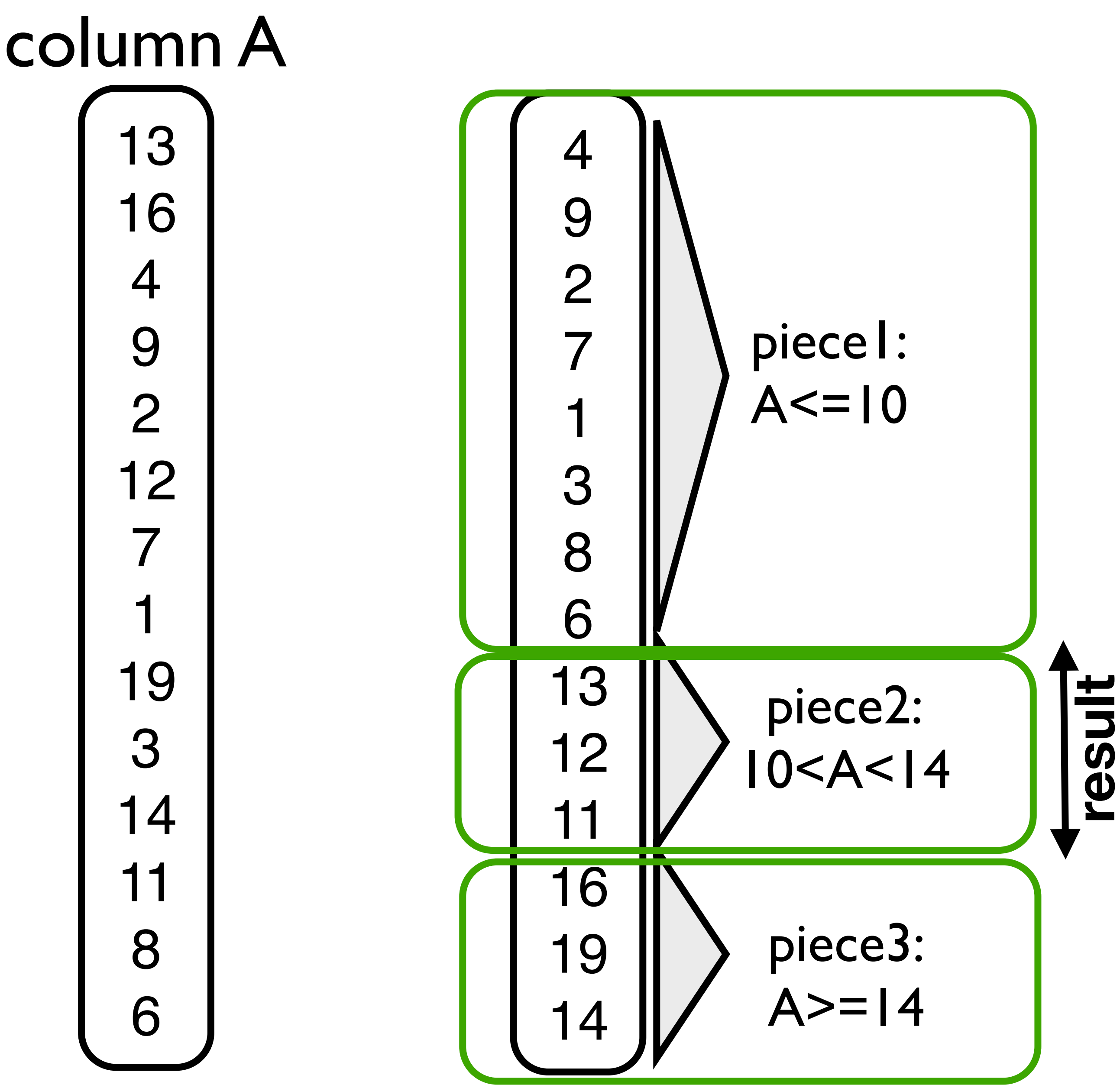
cracking example

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



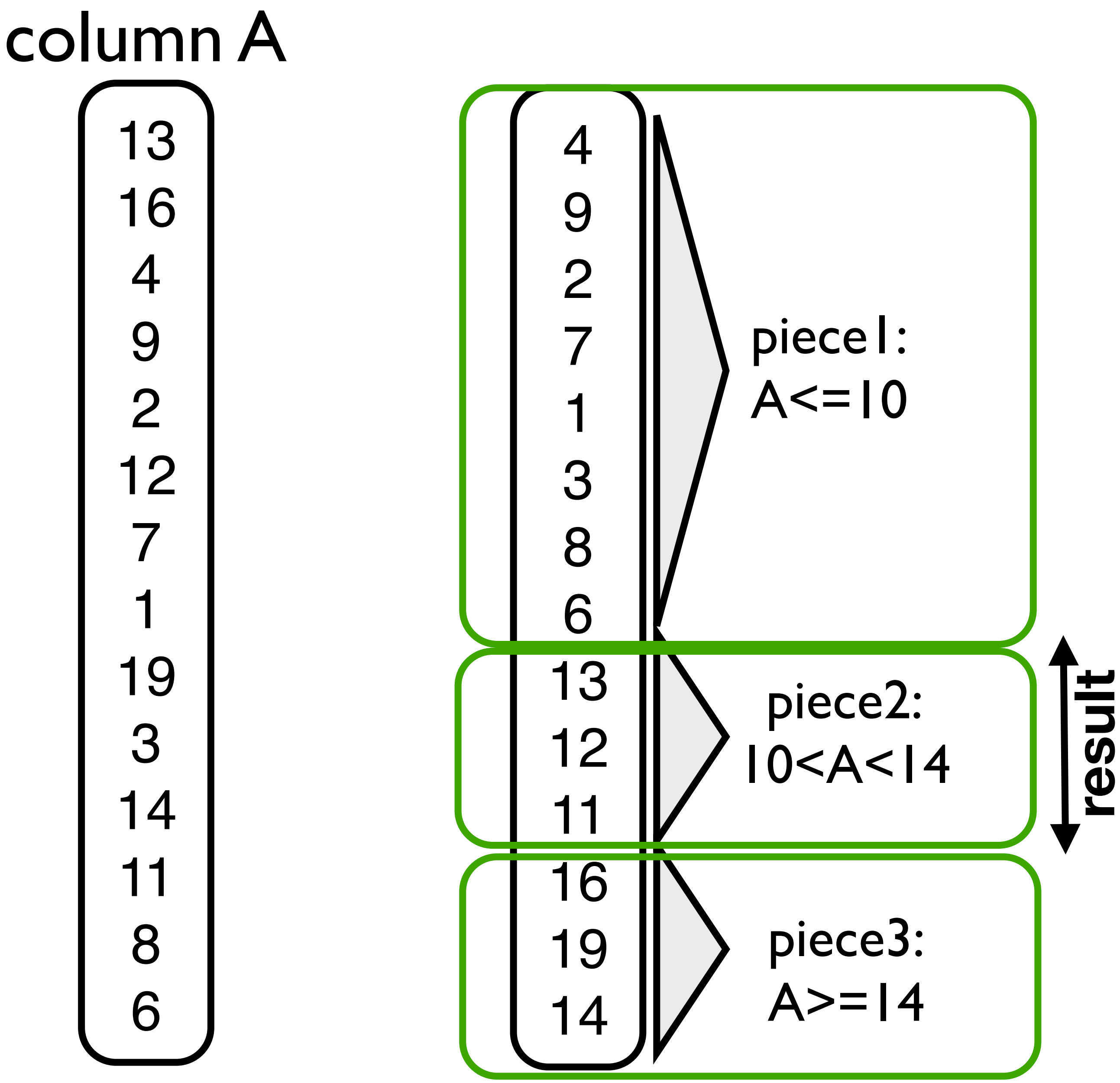
cracking example

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



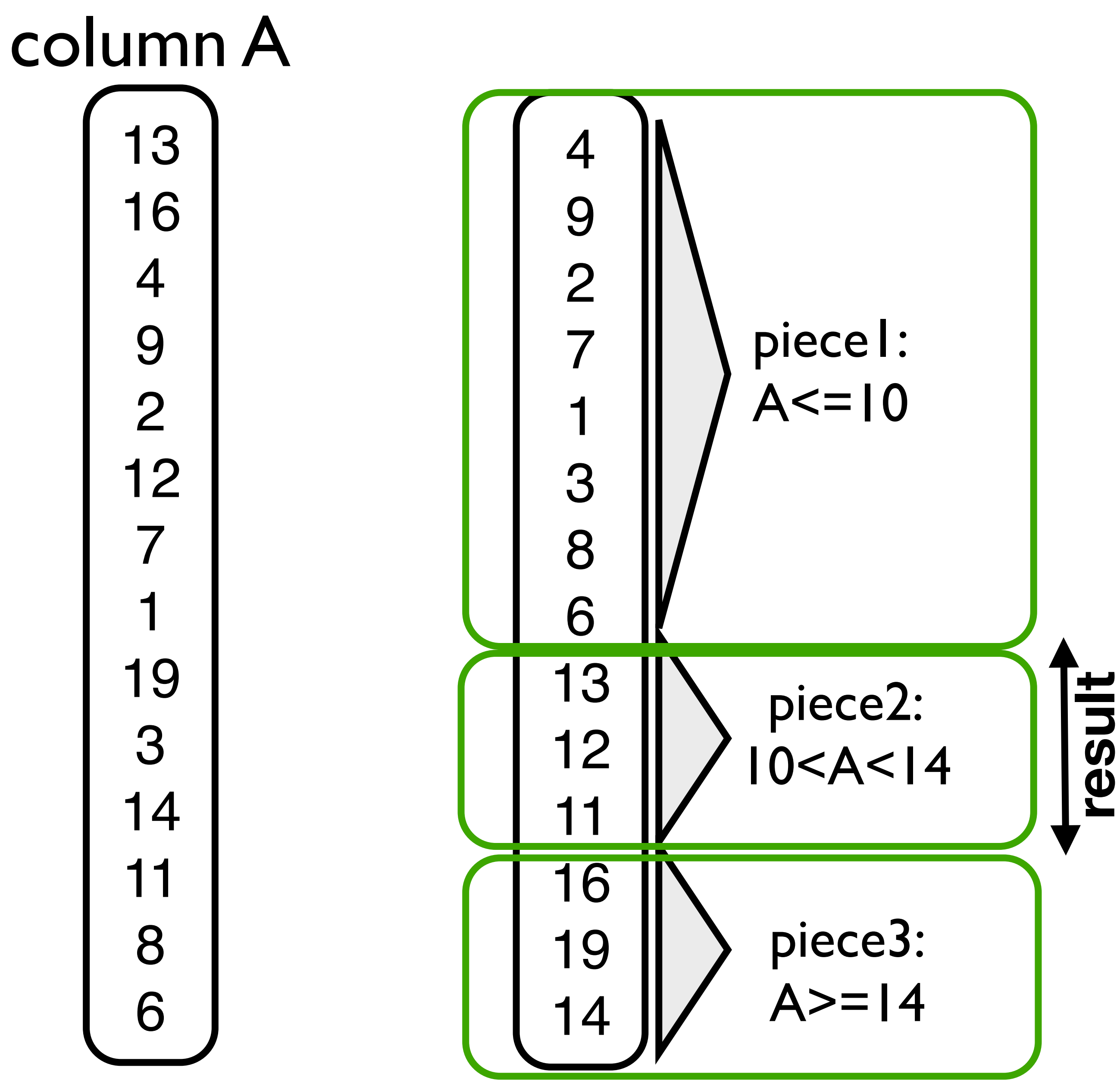
gain knowledge on how data is organized

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



gain knowledge on how data is organized

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

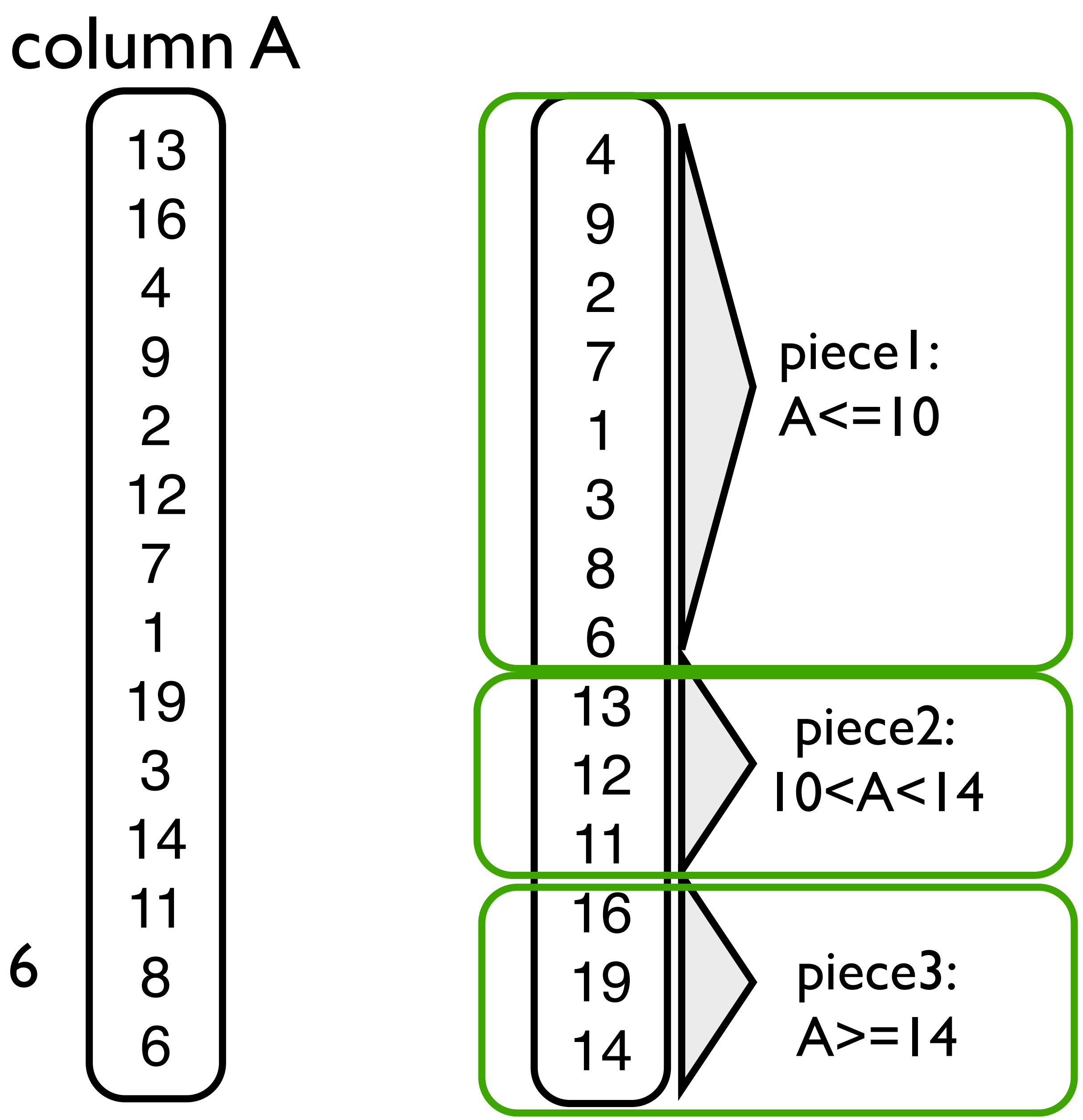


dynamically/on-the-fly within the select-operator

cracking example

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

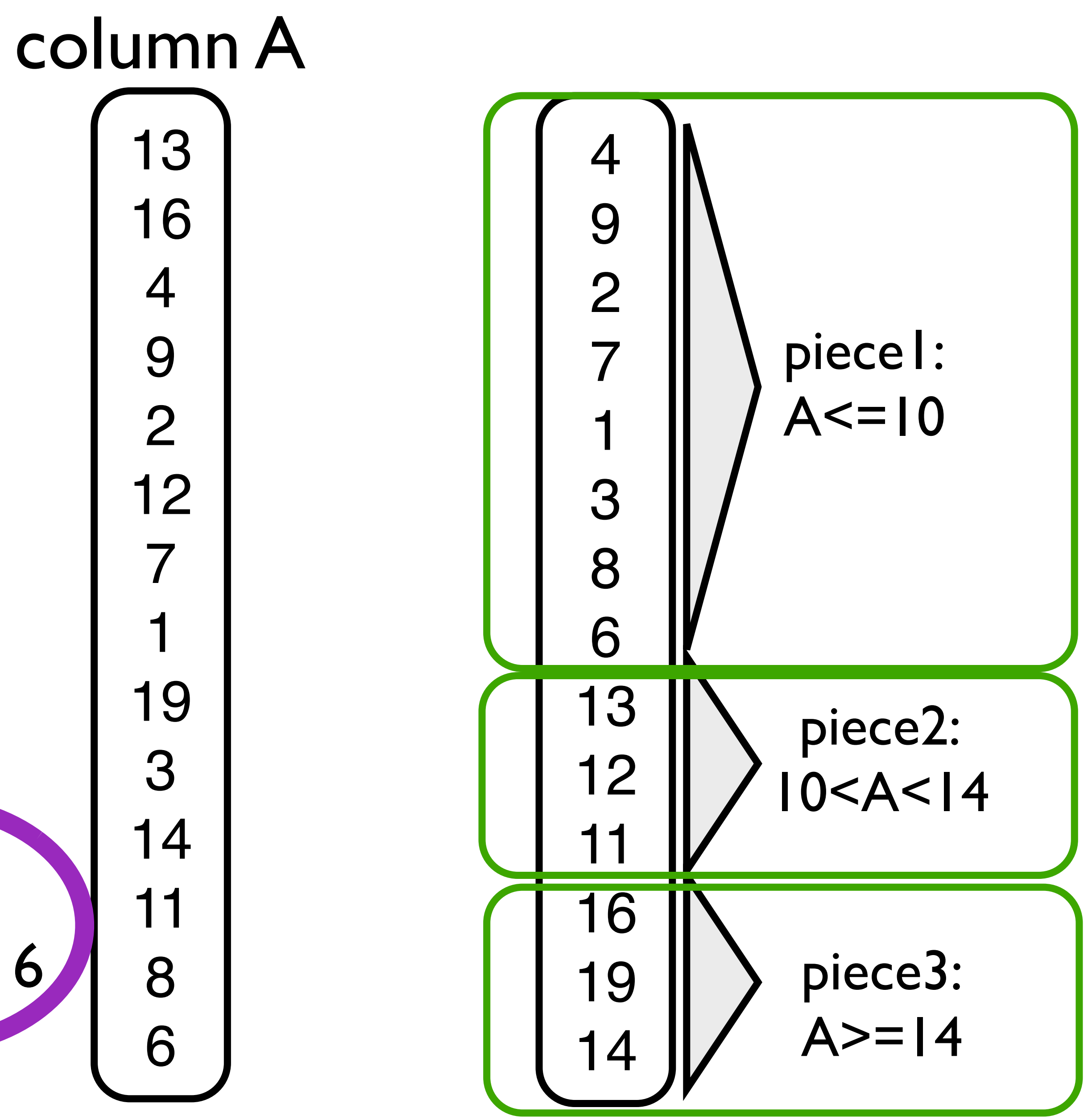


dynamically/on-the-fly within the select-operator

cracking example

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



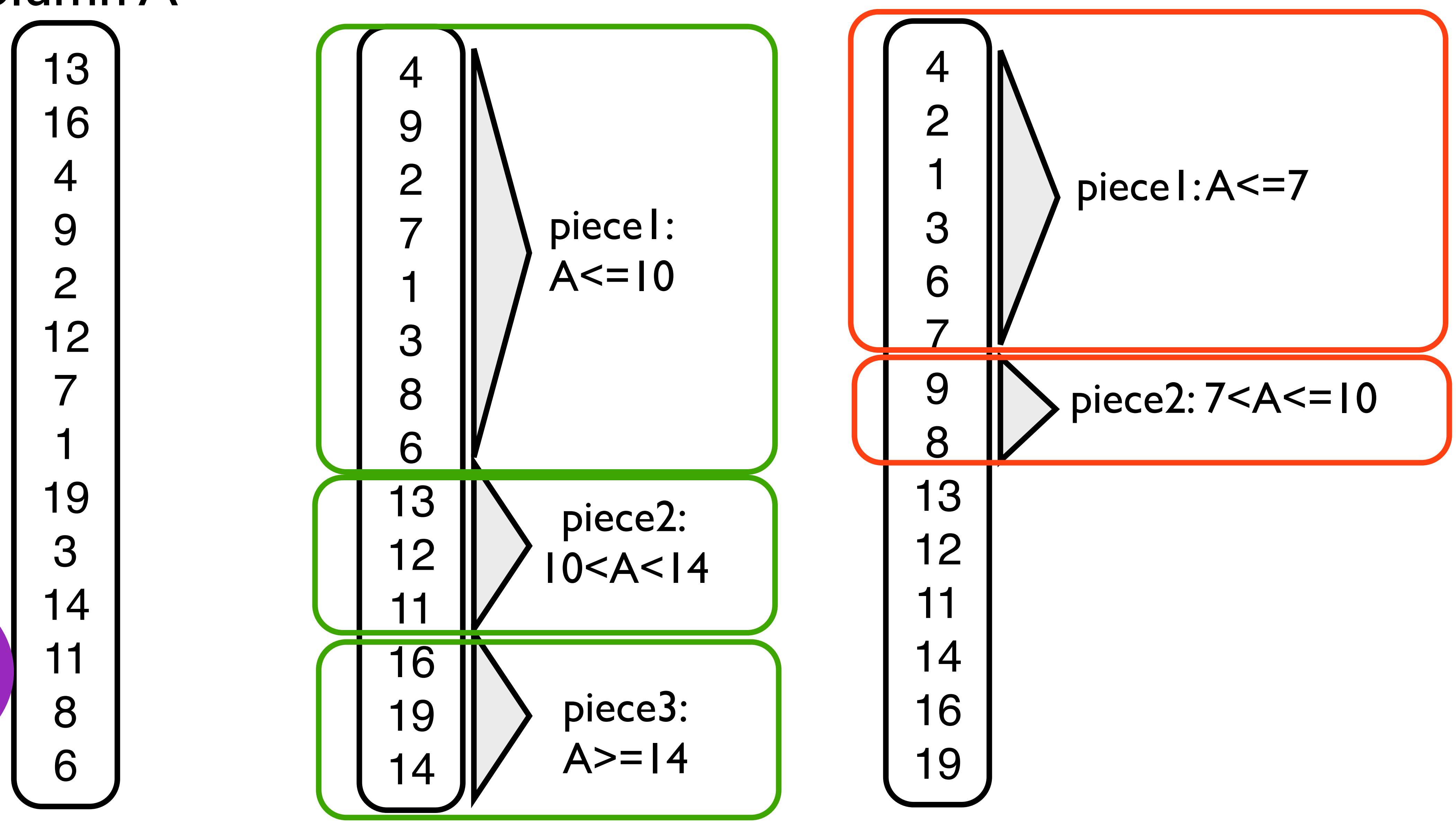
dynamically/on-the-fly within the select-operator

cracking example

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

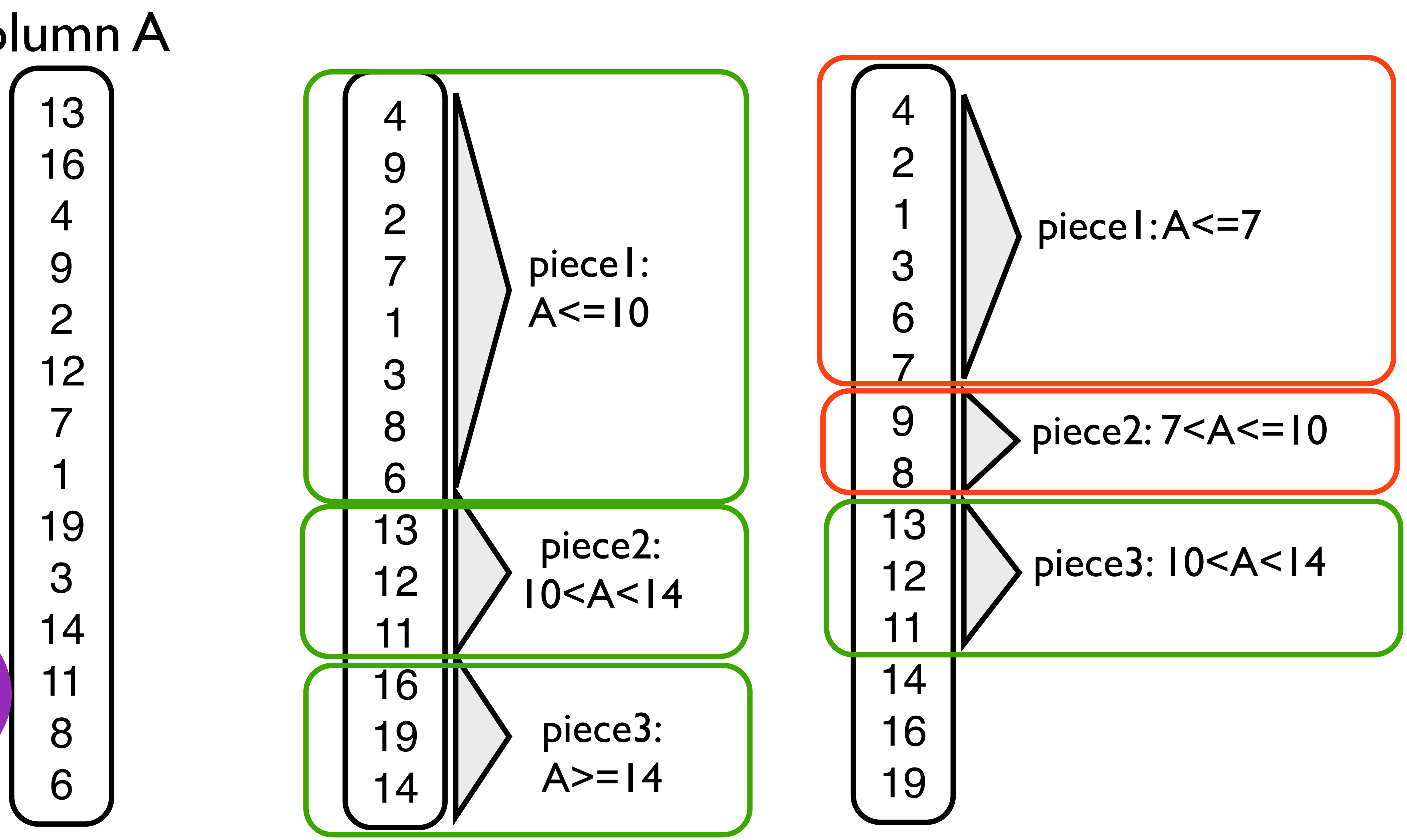


dynamically/on-the-fly within the select-operator

cracking example

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

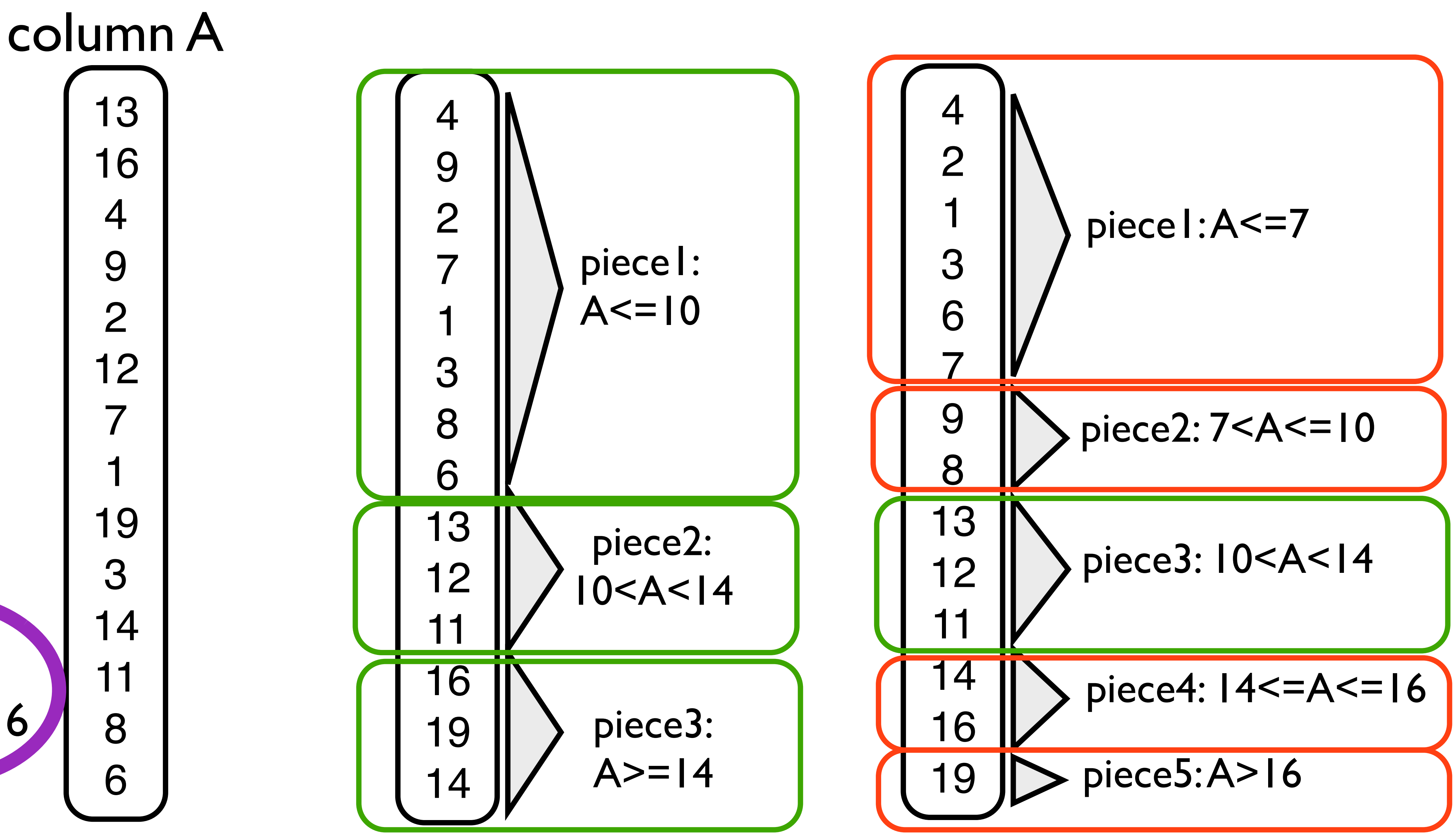


dynamically/on-the-fly within the select-operator

cracking example

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

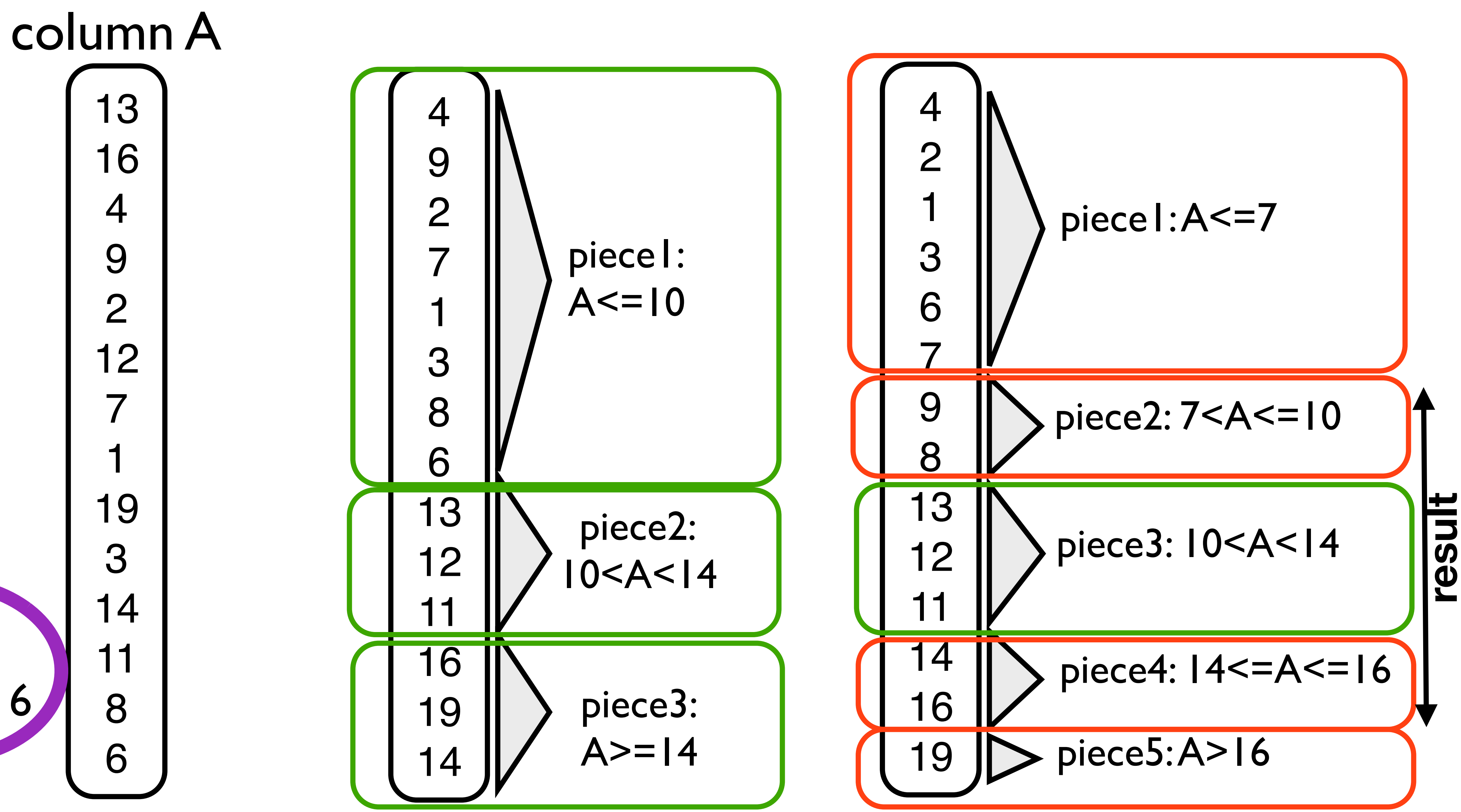


dynamically/on-the-fly within the select-operator

cracking example

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

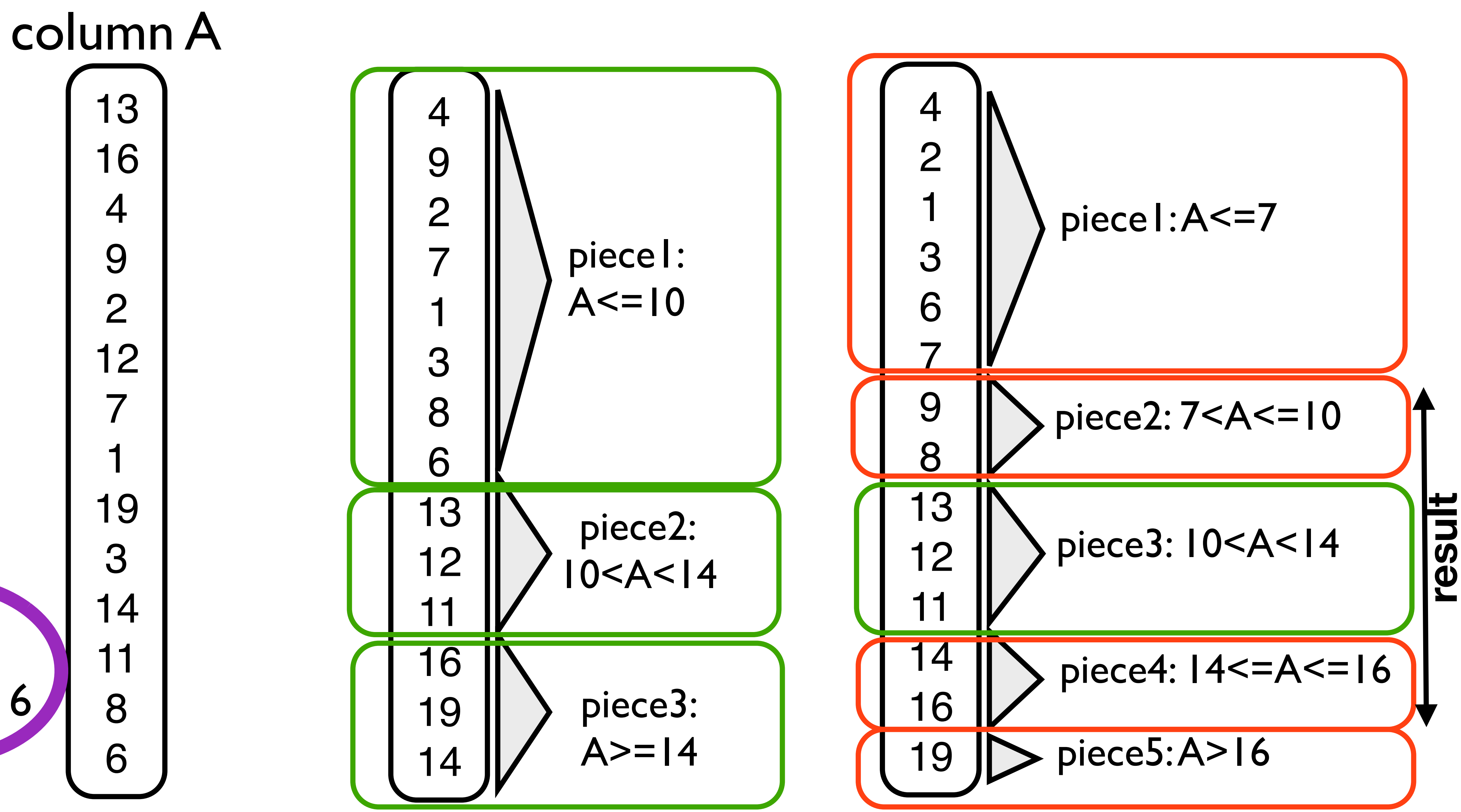


dynamically/on-the-fly within the select-operator

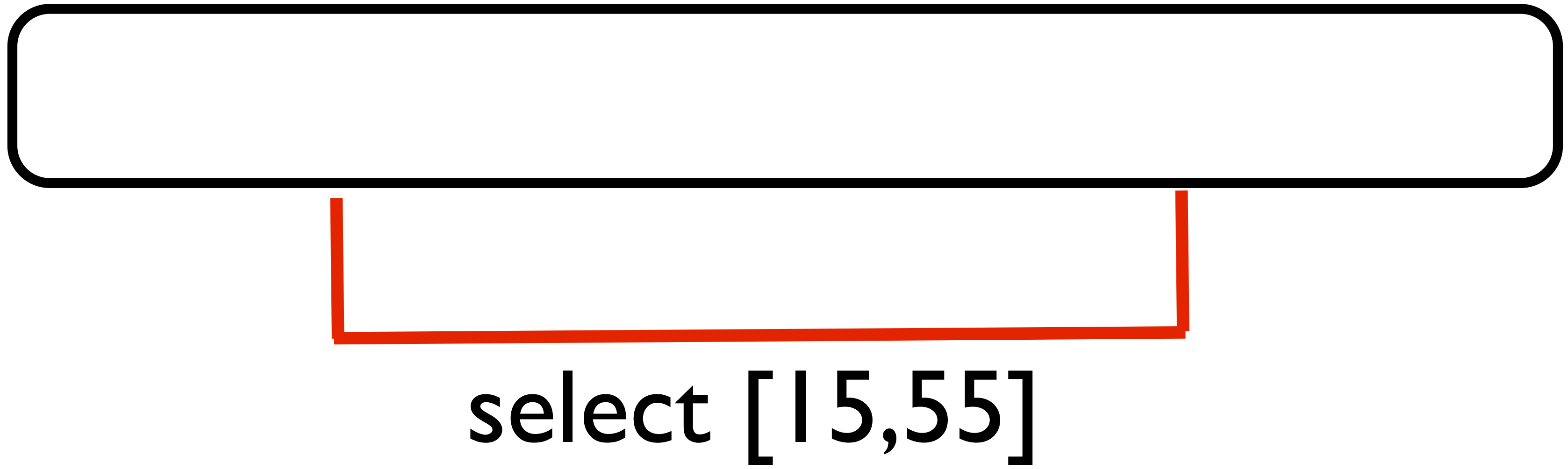
the more we crack, the more we learn

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

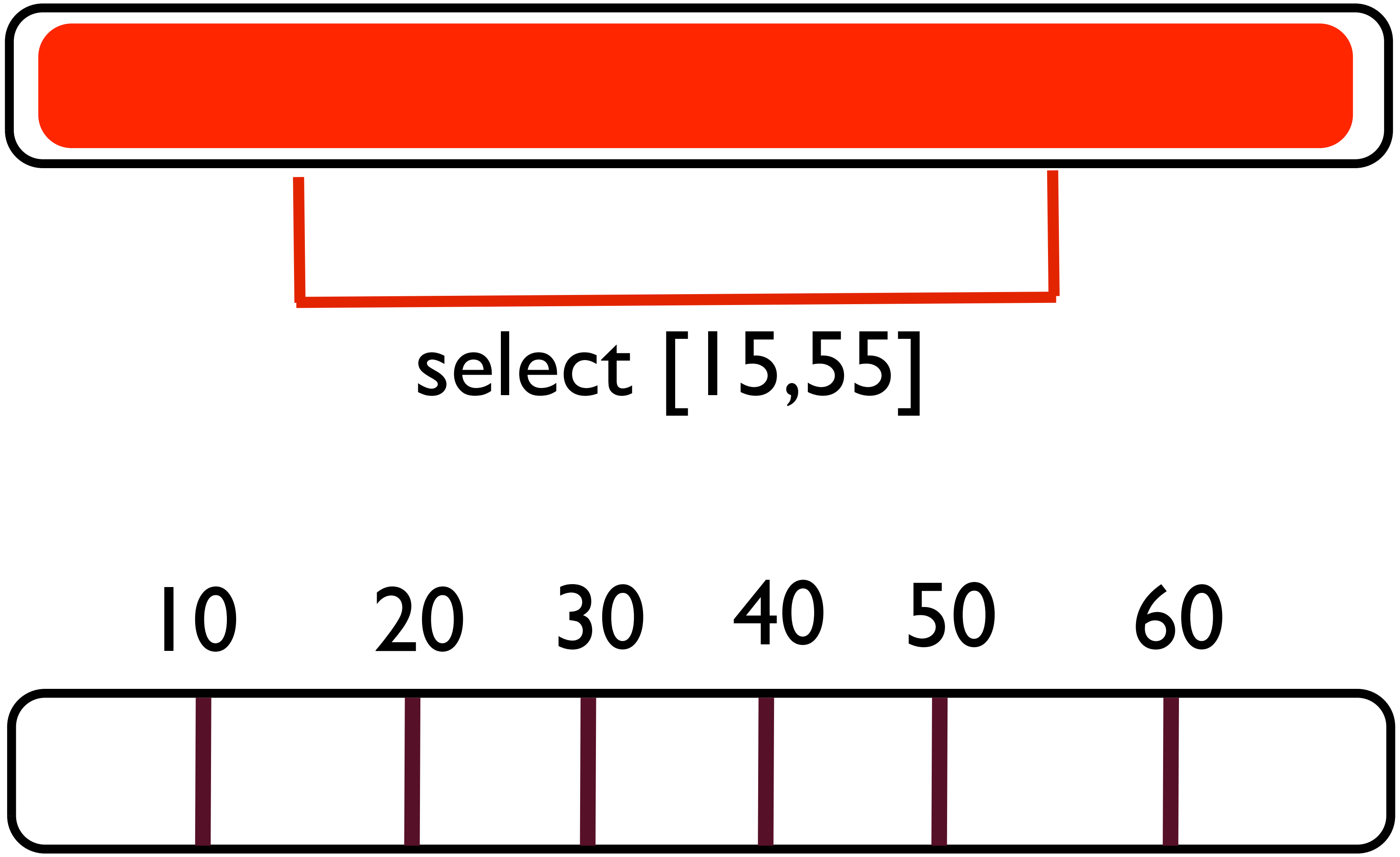
Q2:
select R.A
from R
where R.A>7
and R.A<=16

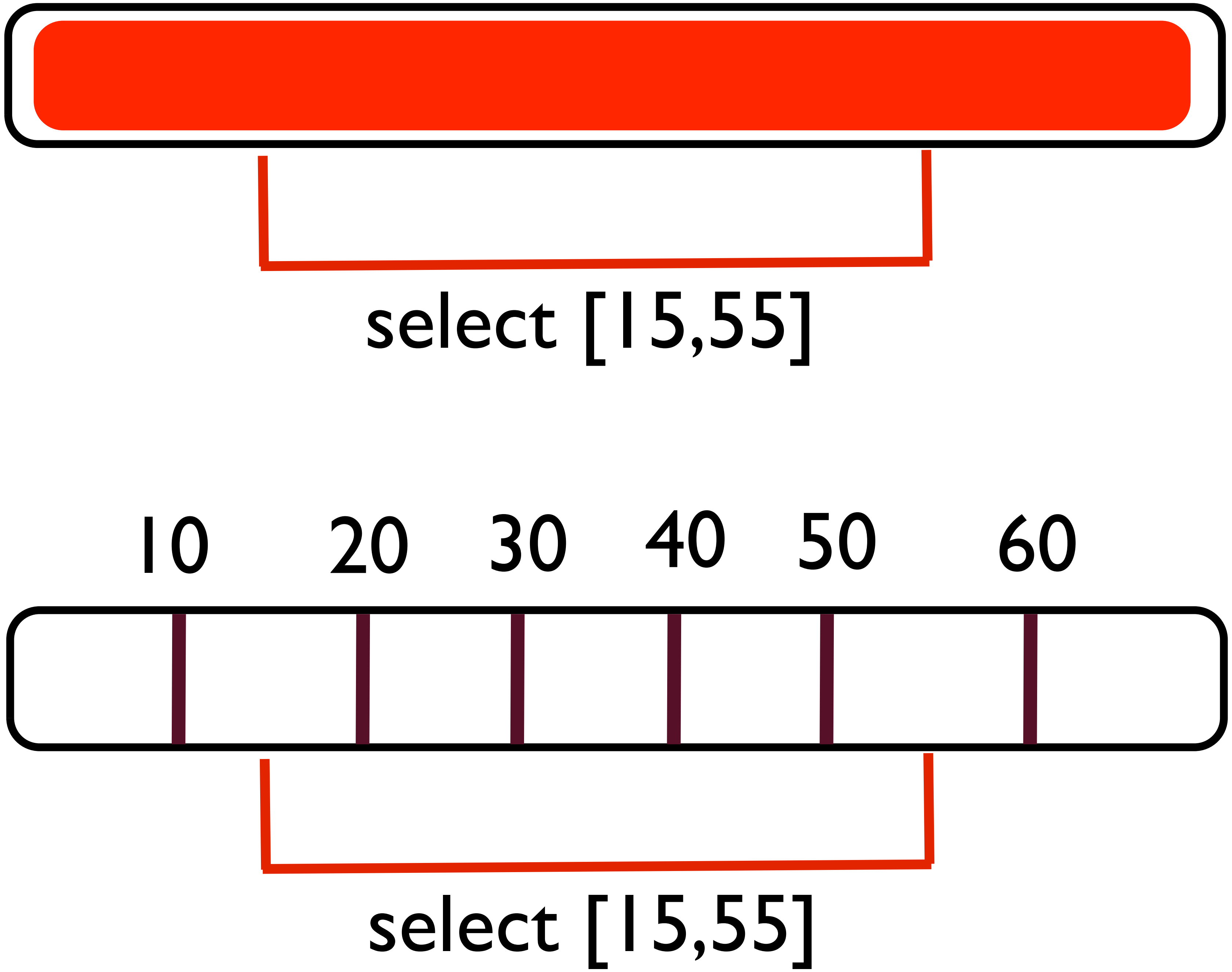


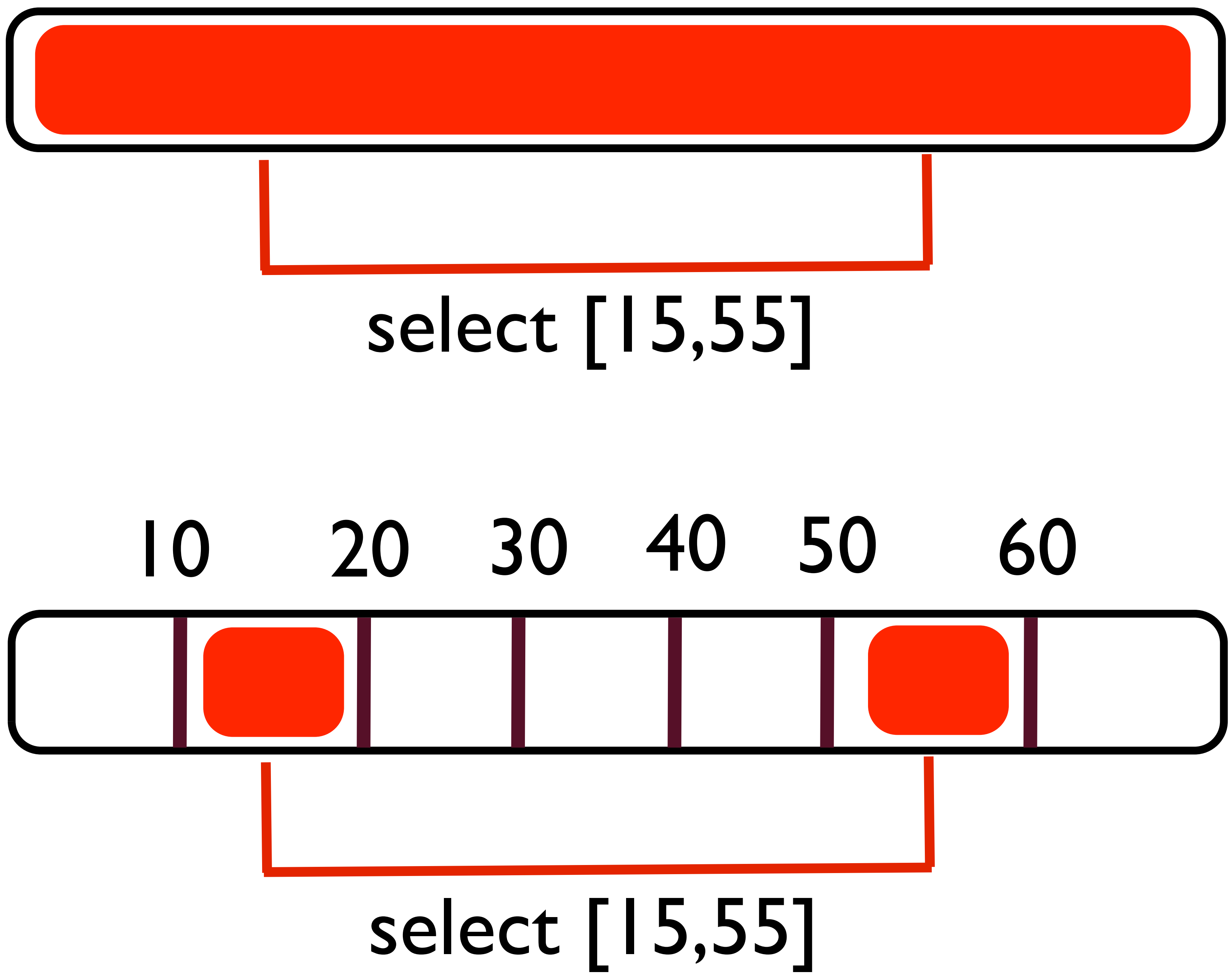
dynamically/on-the-fly within the select-operator


select [15,55]

select [15,55]

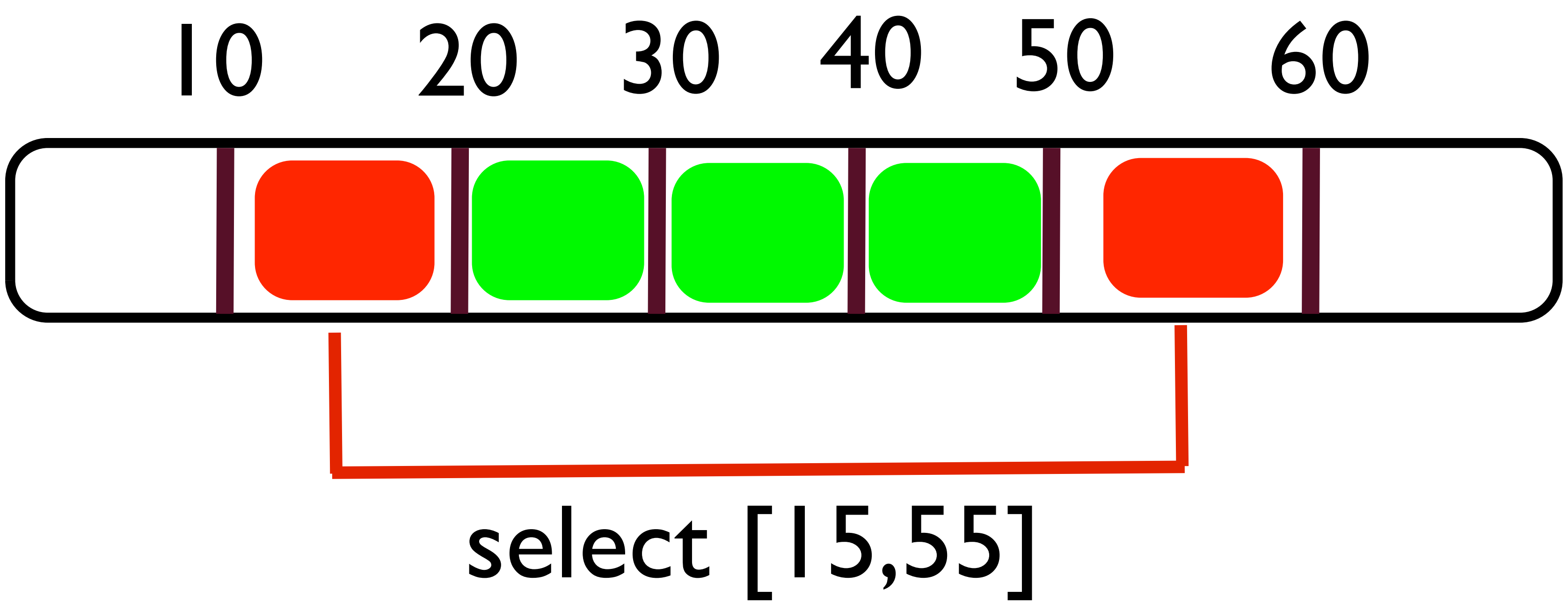

select [15,55]
10
20
30
40
50
60


select [15,55]
10
20
30
40
50
60
select [15,55]


select [15,55]
10
20
30
40
50
60
select [15,55]

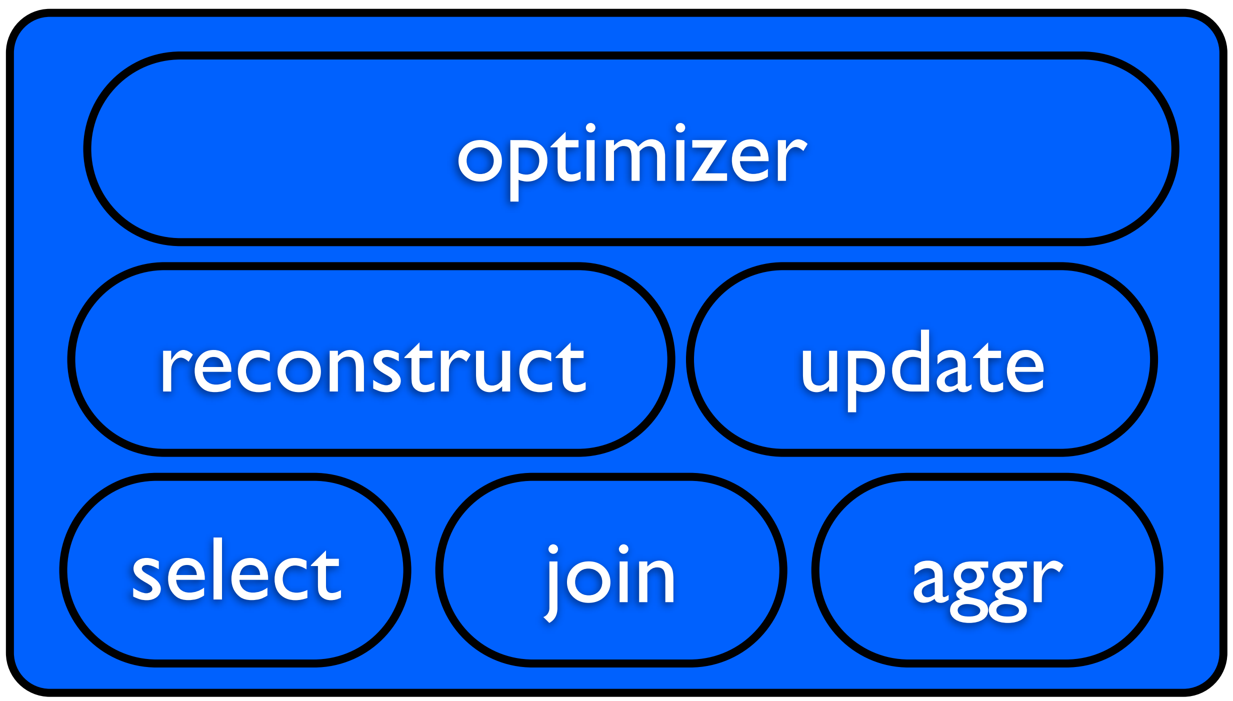
touch at most two pieces at a time

pieces become smaller and smaller



implemented in monetdb

open-source column-store



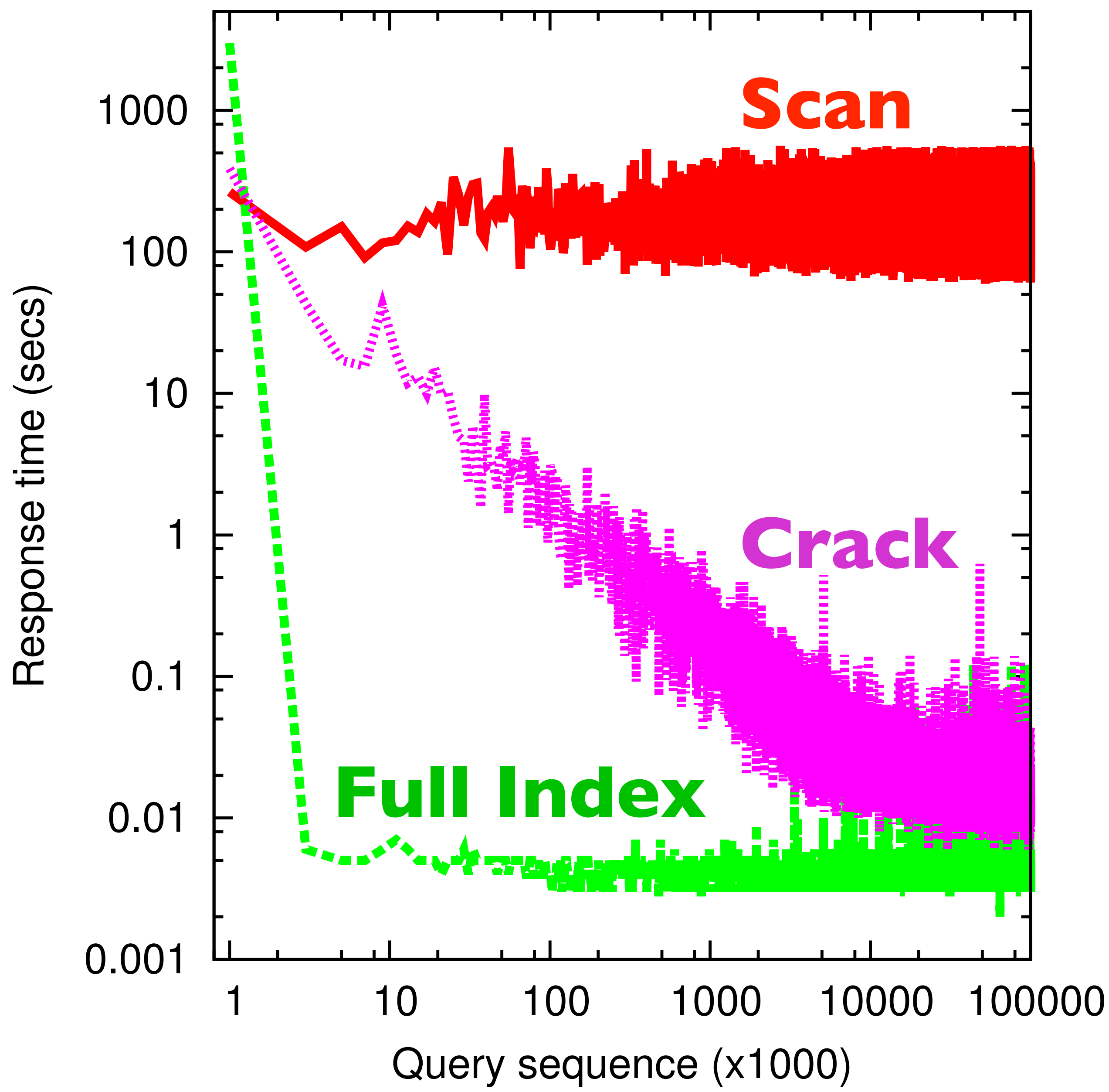
database kernel

code footprint
monetdb 2M

continuous adaptation

set-up

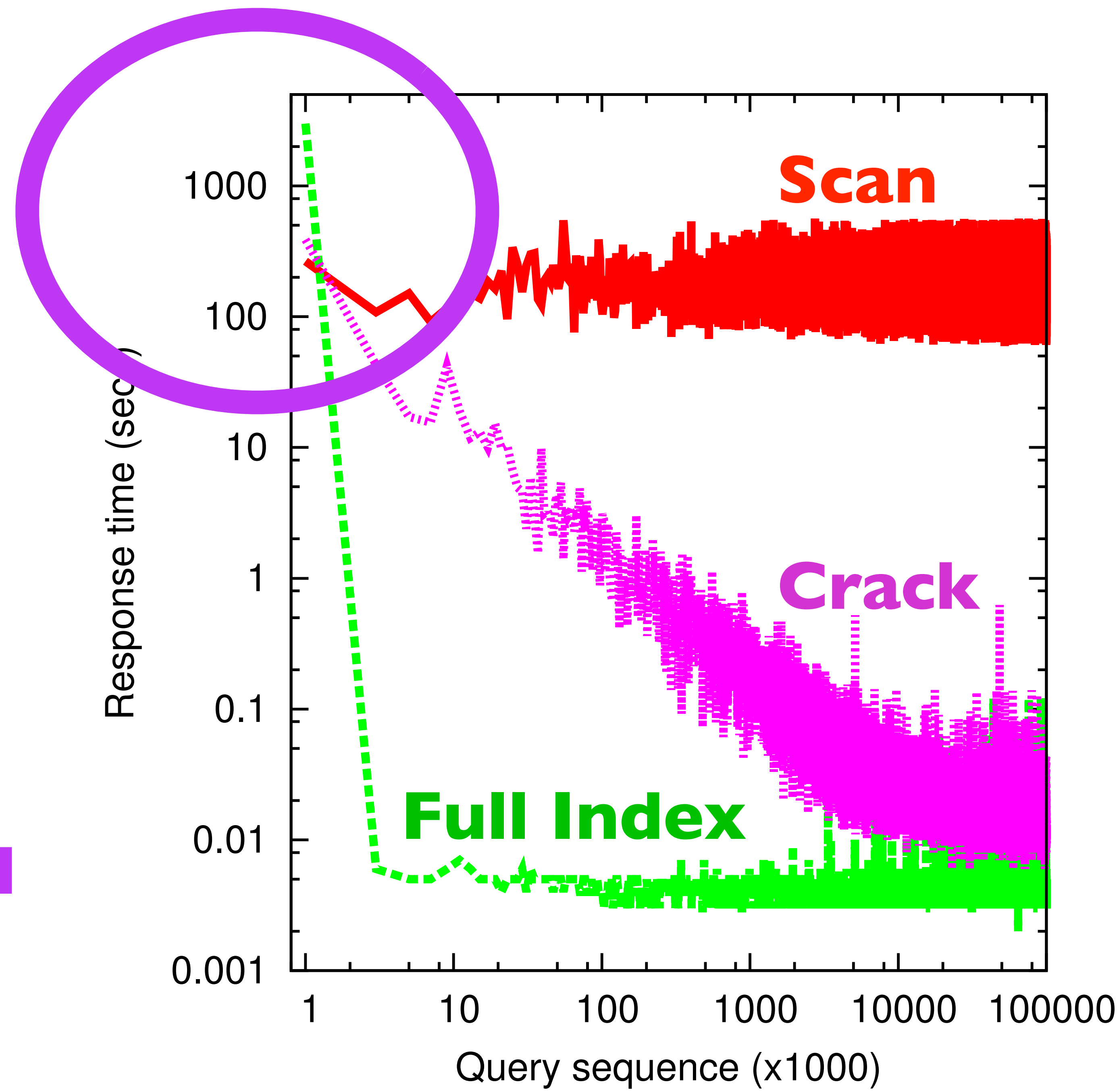
100K random selections
random selectivity
random value ranges
in a 10 million integer column



continuous adaptation

set-up

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

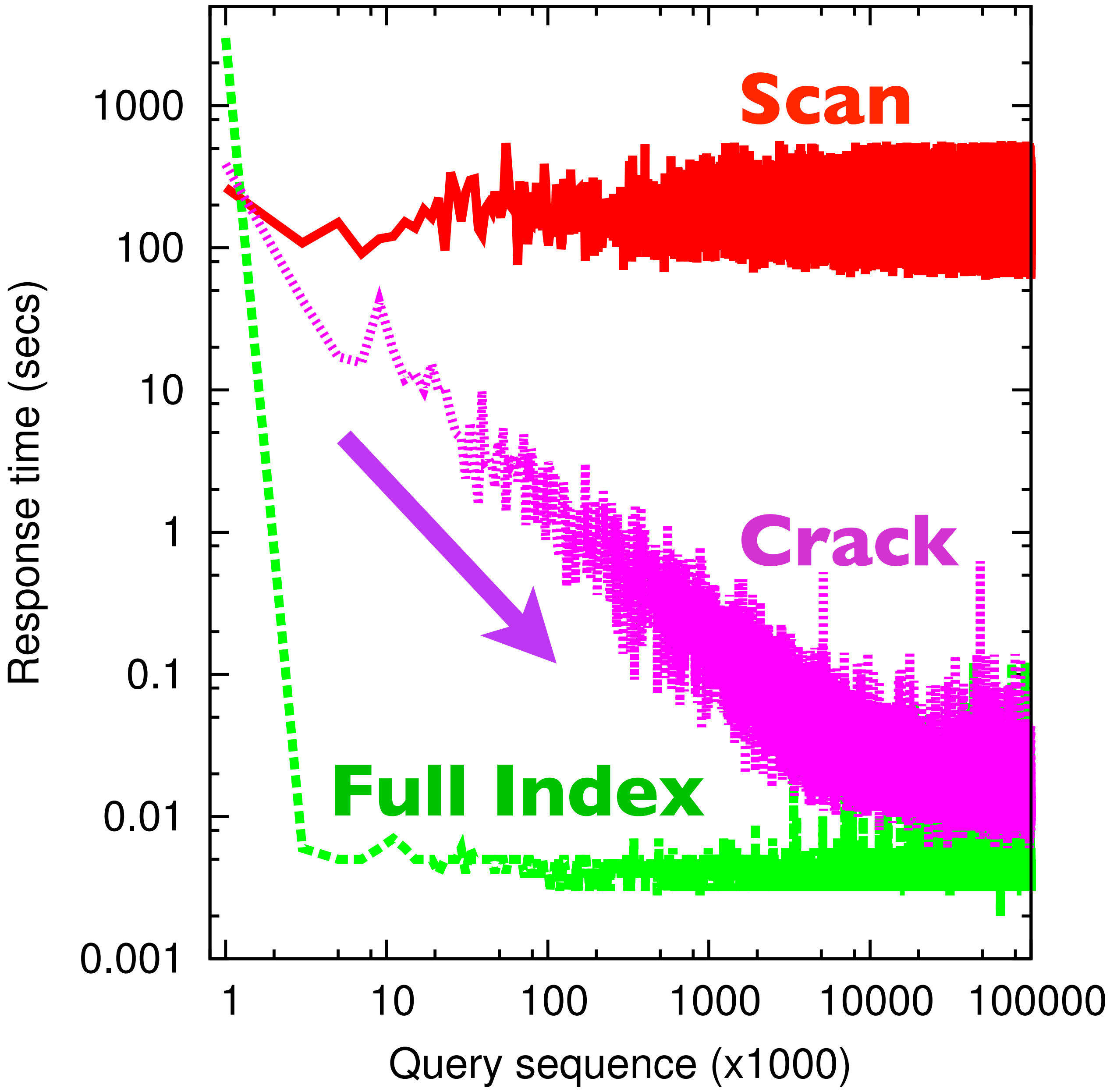


almost no initialization overhead

continuous adaptation

set-up

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



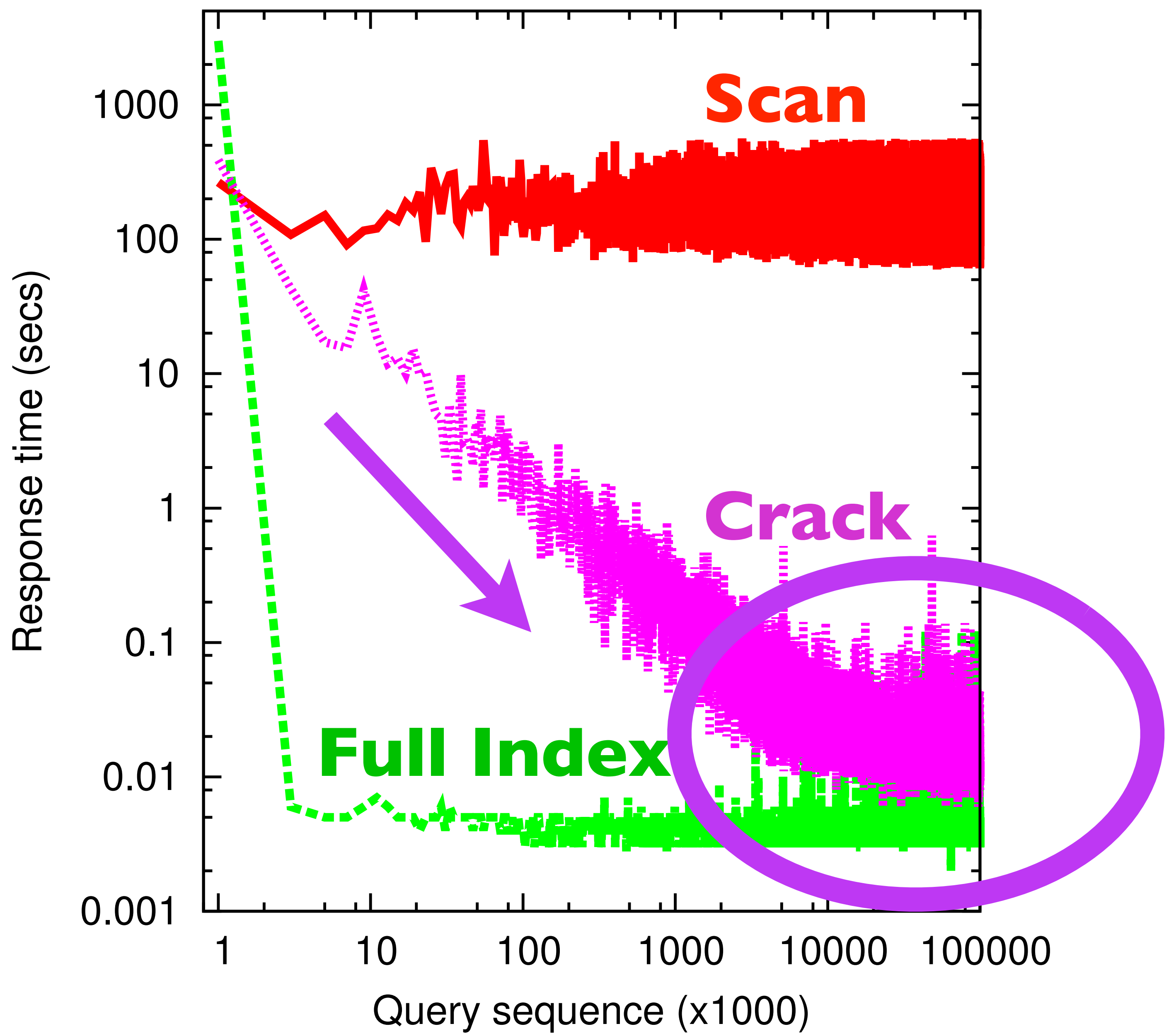
almost no initialization overhead

continuous improvement

continuous adaptation

set-up

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



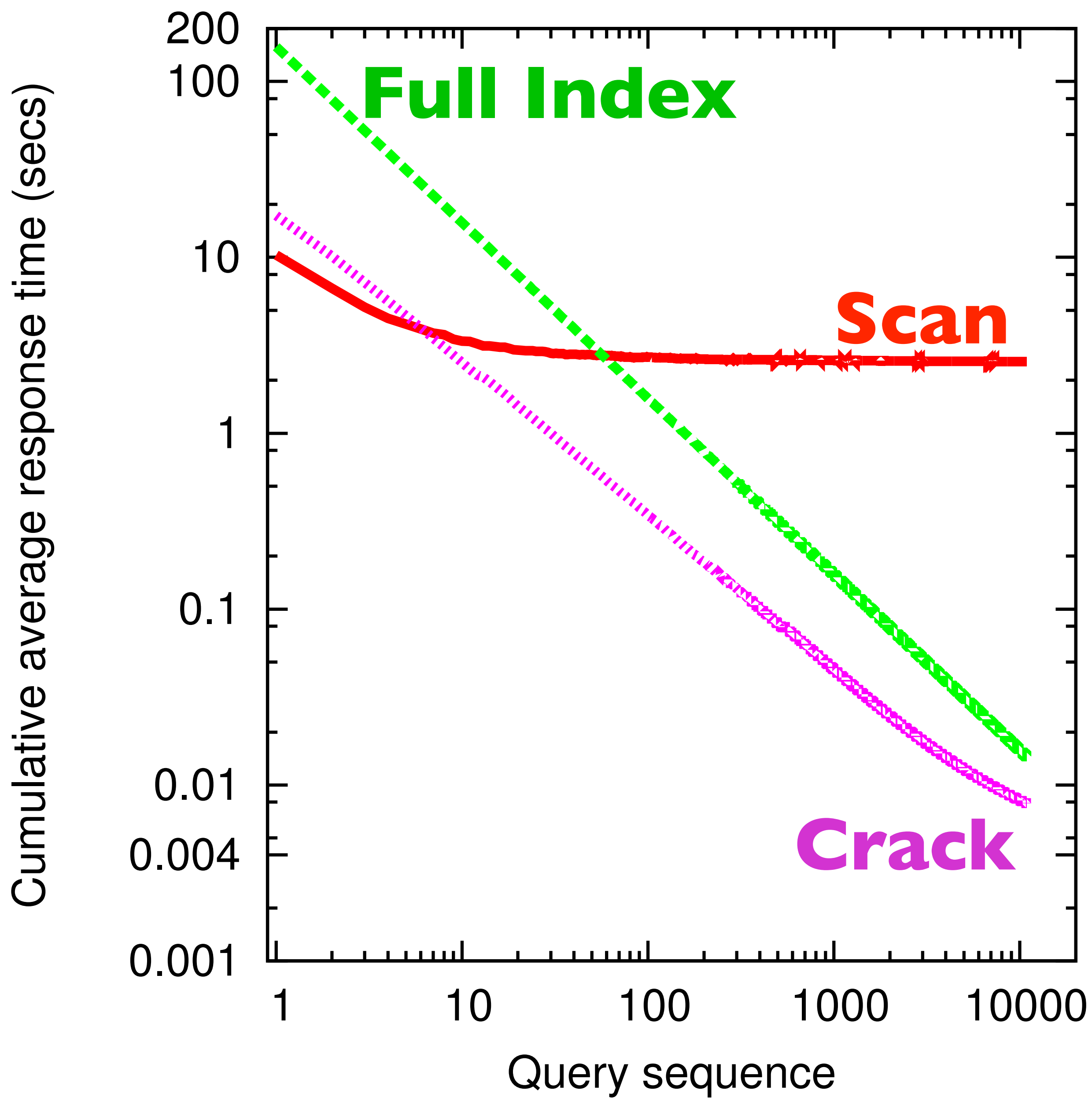
almost no initialization overhead

continuous improvement

continuous adaptation

set-up

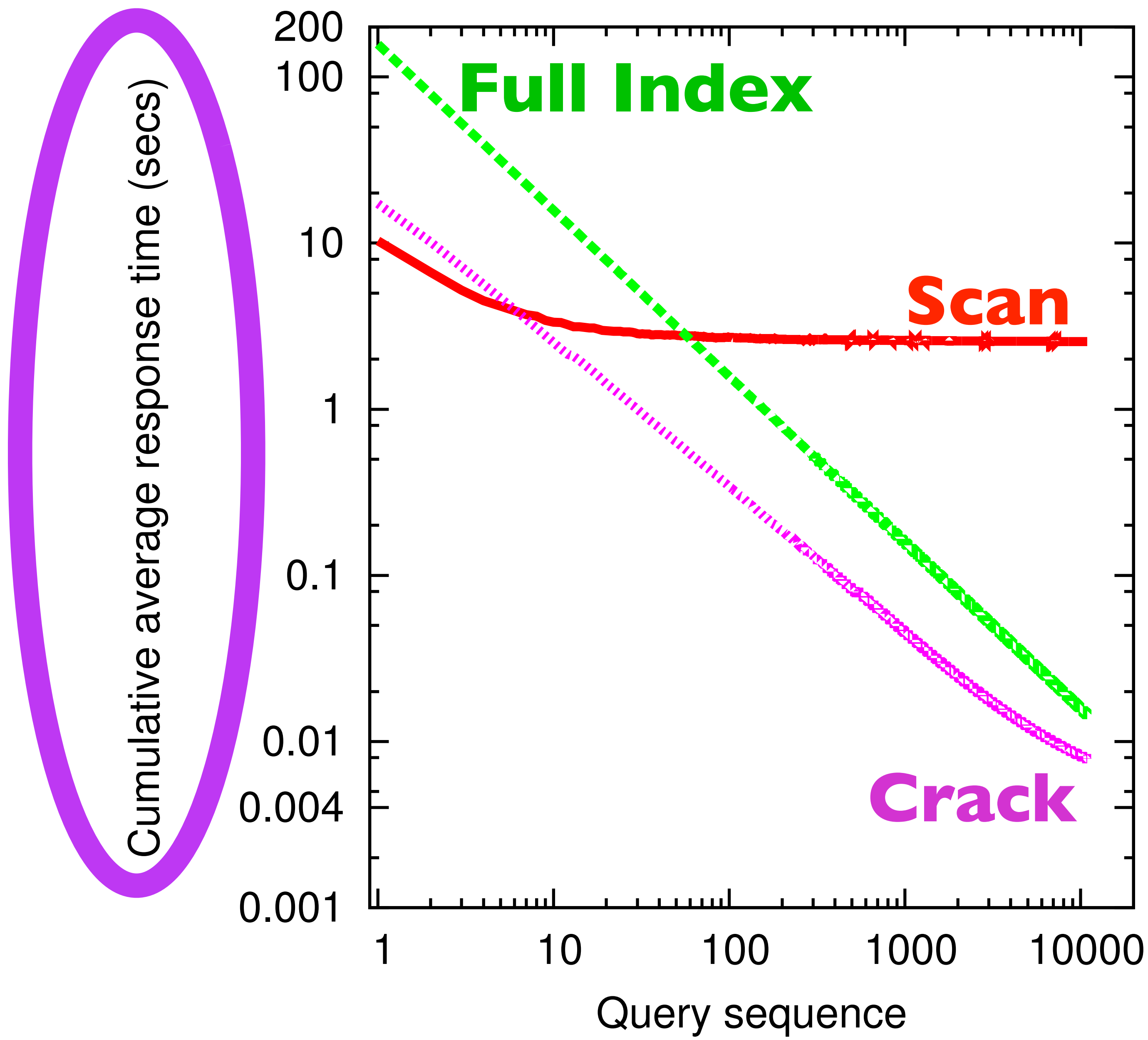
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



continuous adaptation

set-up

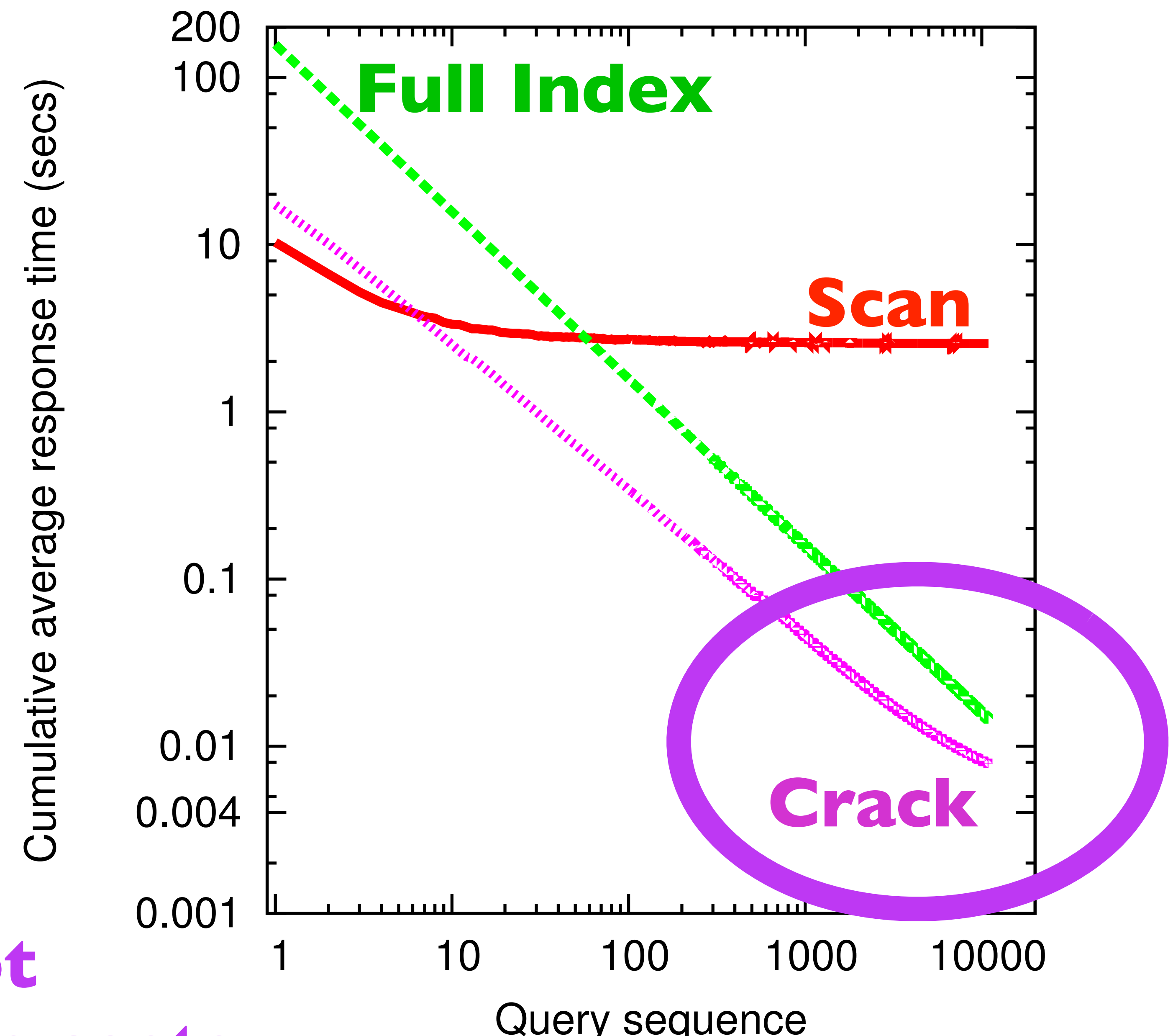
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



continuous adaptation

set-up

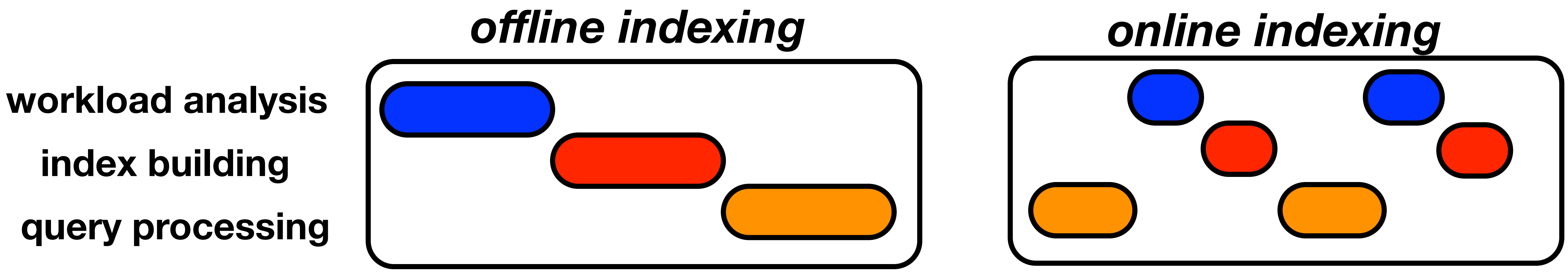
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



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

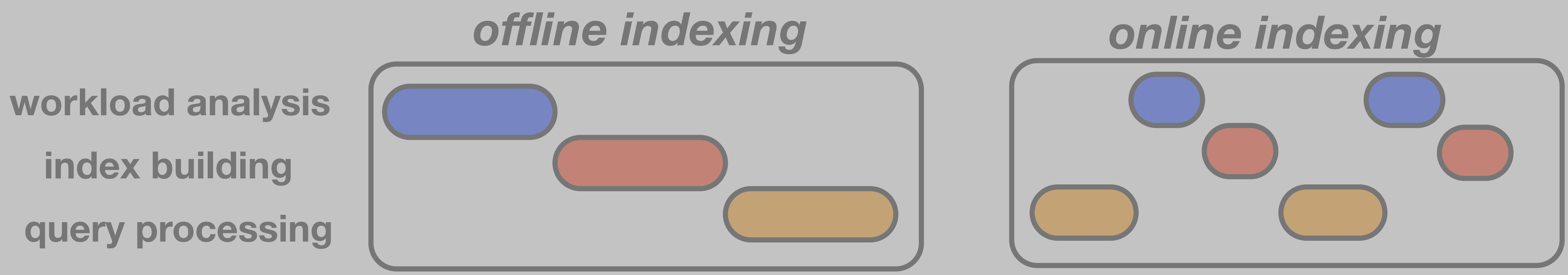
traditional databases

monolithic/full indexing



traditional databases

monolithic/full indexing



database cracking

partial/adaptive/continuous indexing

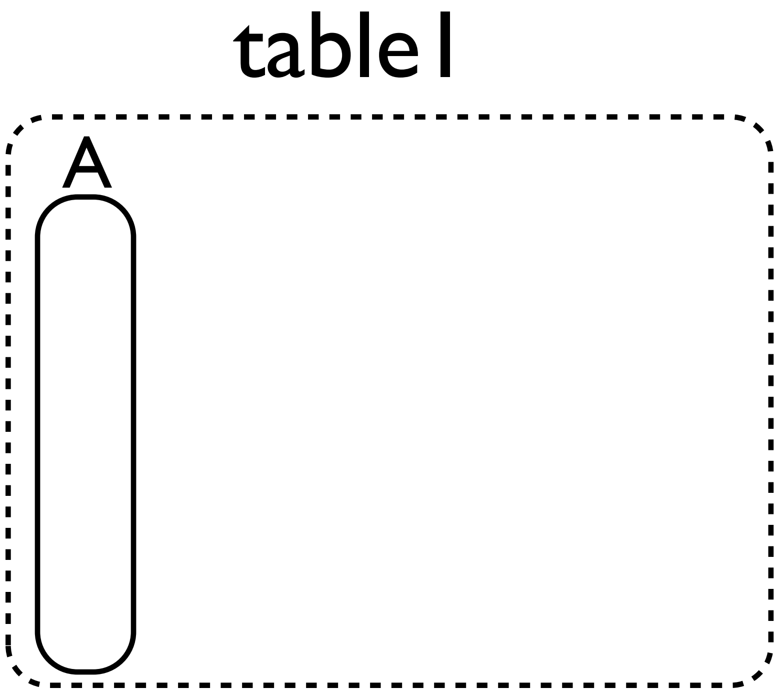


table1
A

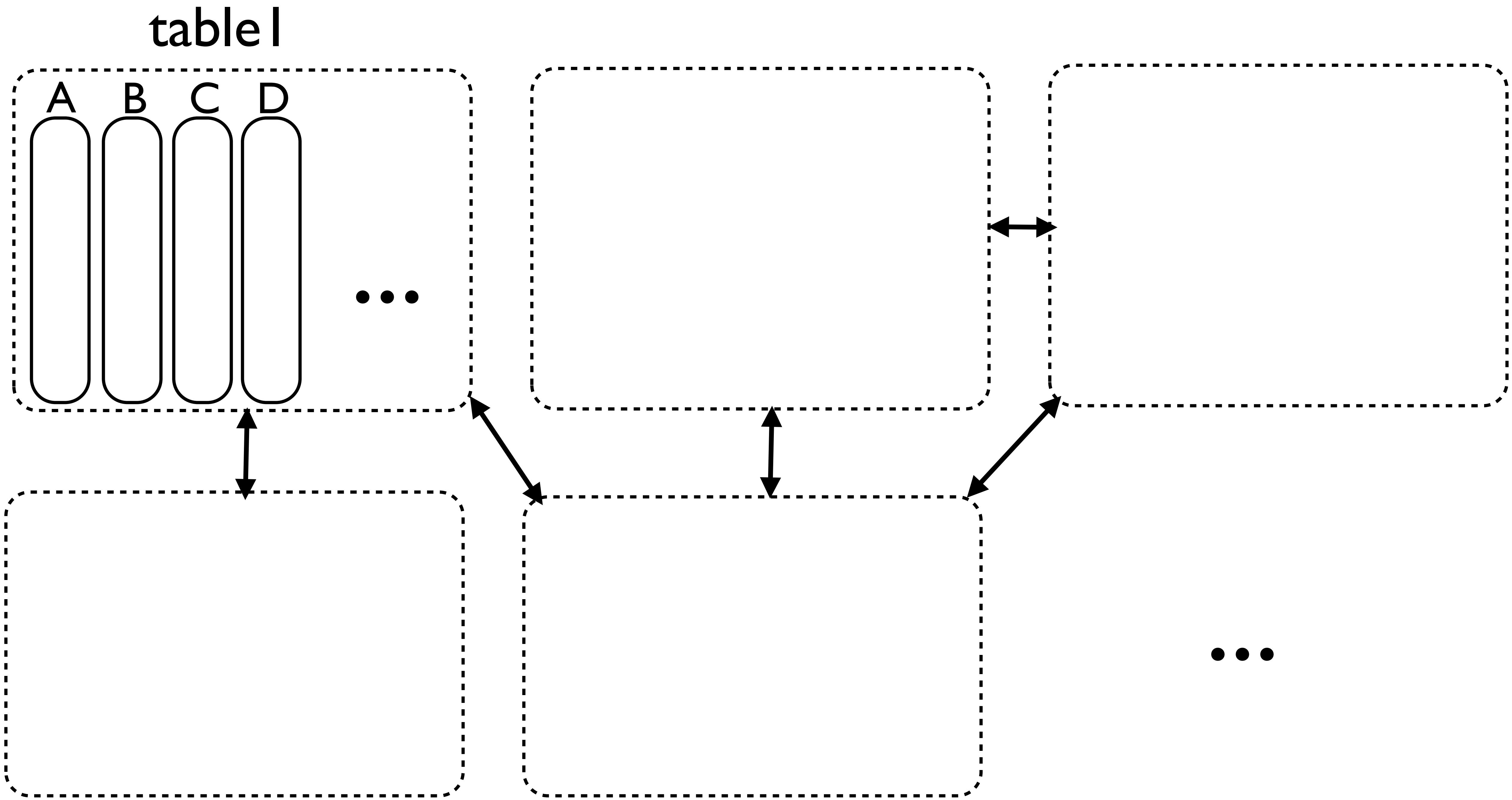
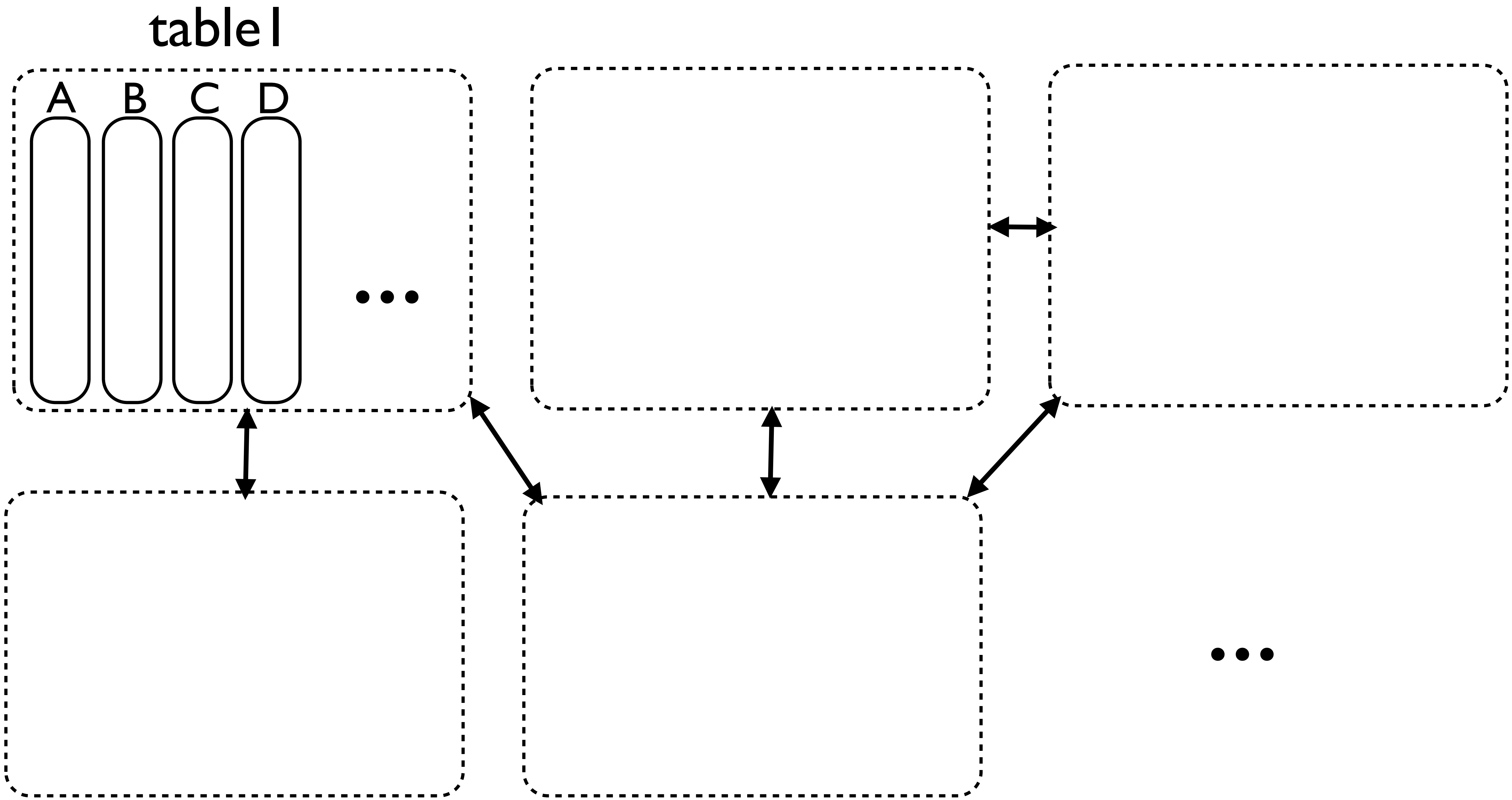


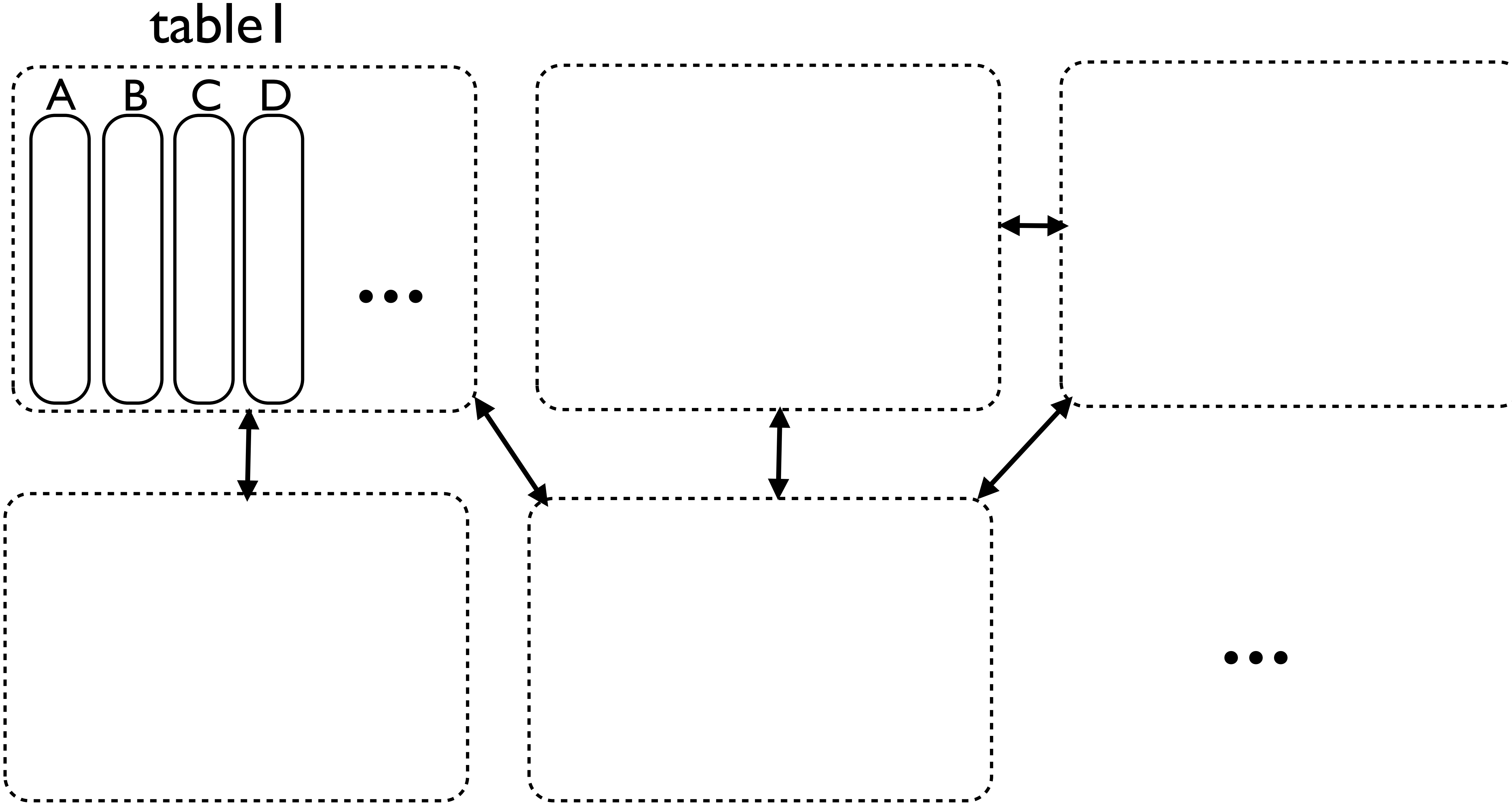
table1
A
B
C
D
...
...

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



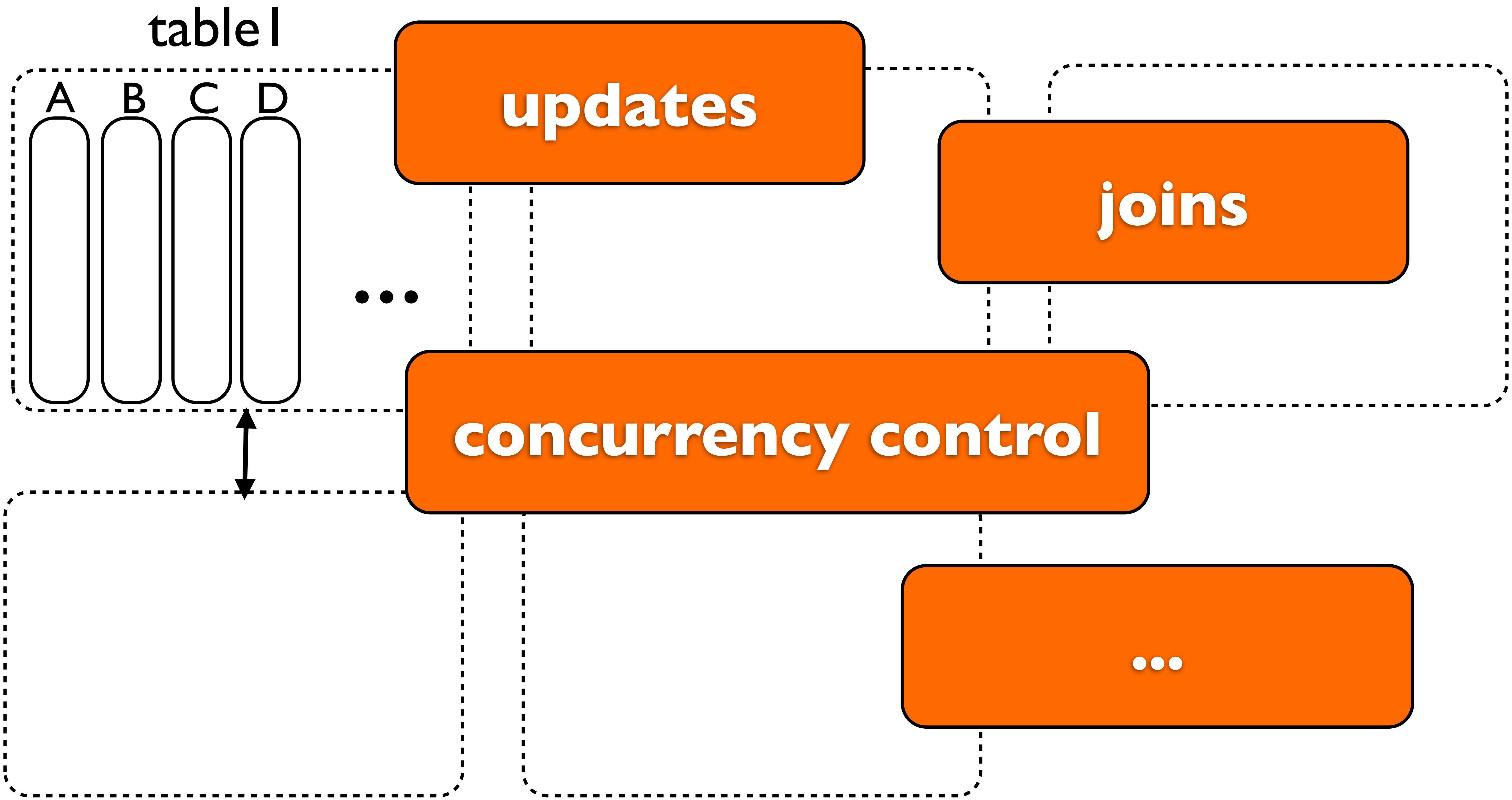
select R.A from R where R.A>10 and R.A<14

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<k6
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<k6
and R.F = S.F



cracking tangram

base data ***as queries arrive...***

table 1

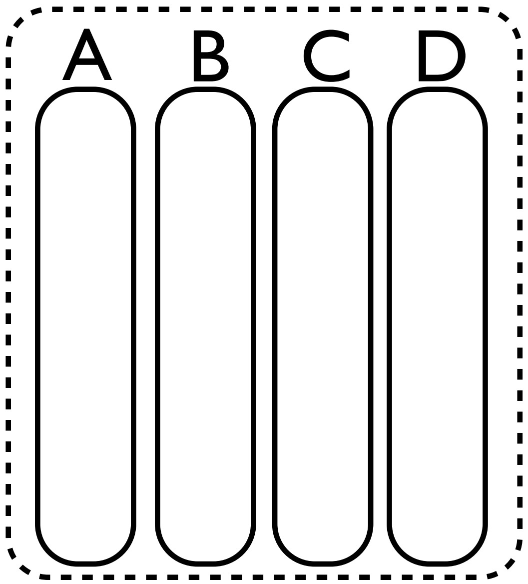
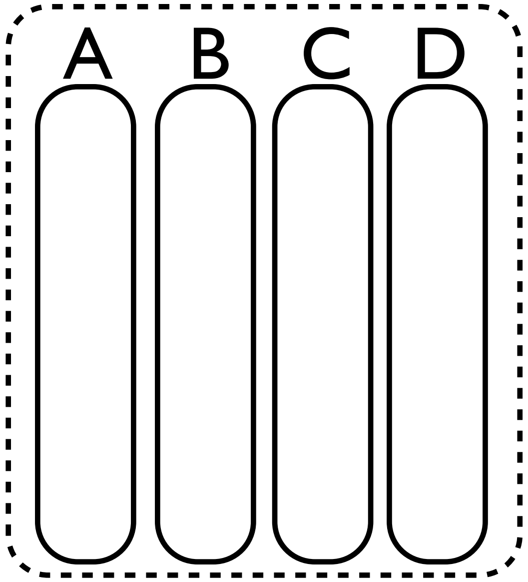
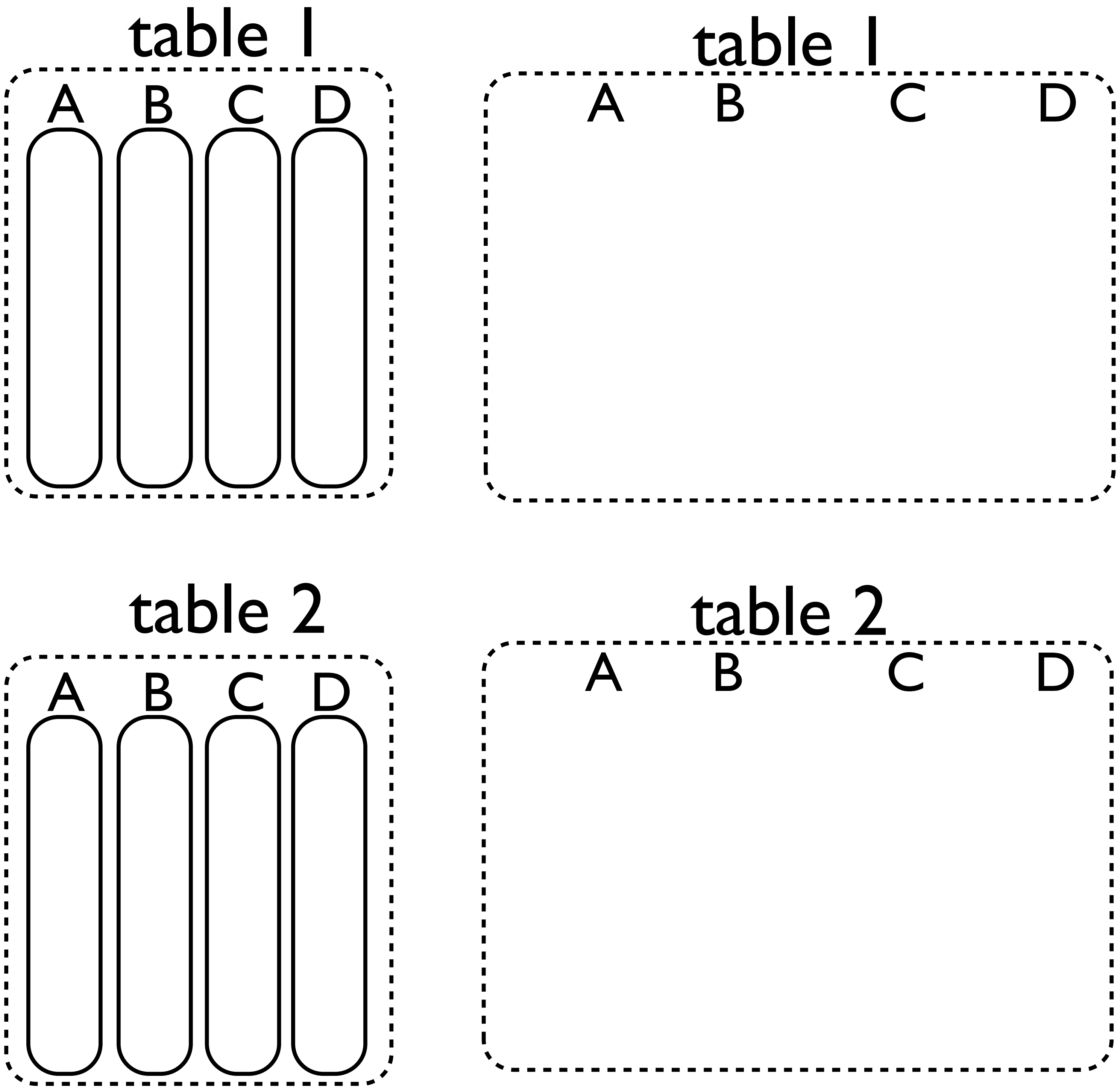


table 2



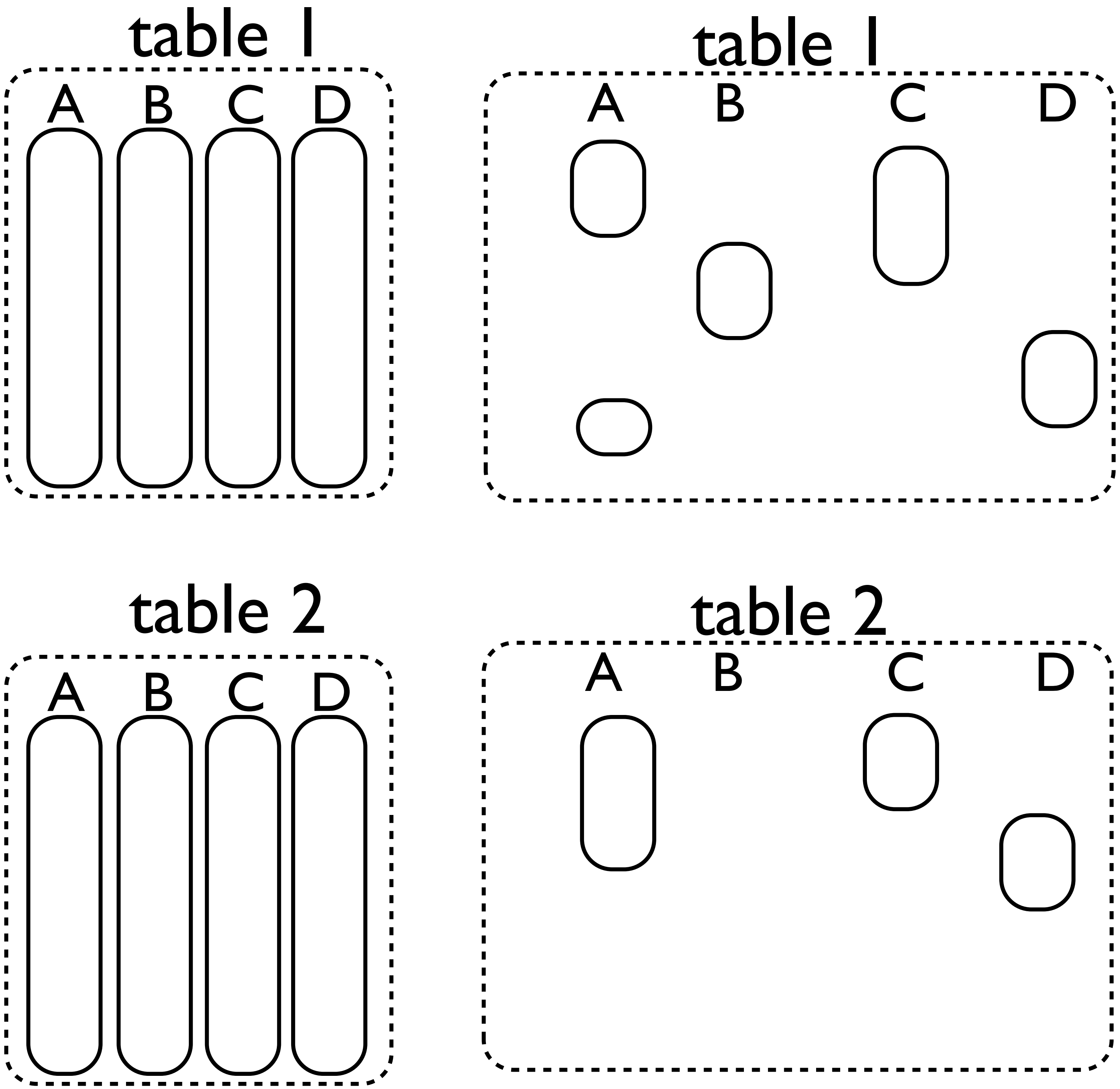
cracking tangram

base data ***as queries arrive...***



cracking tangram

base data ***as queries arrive...***

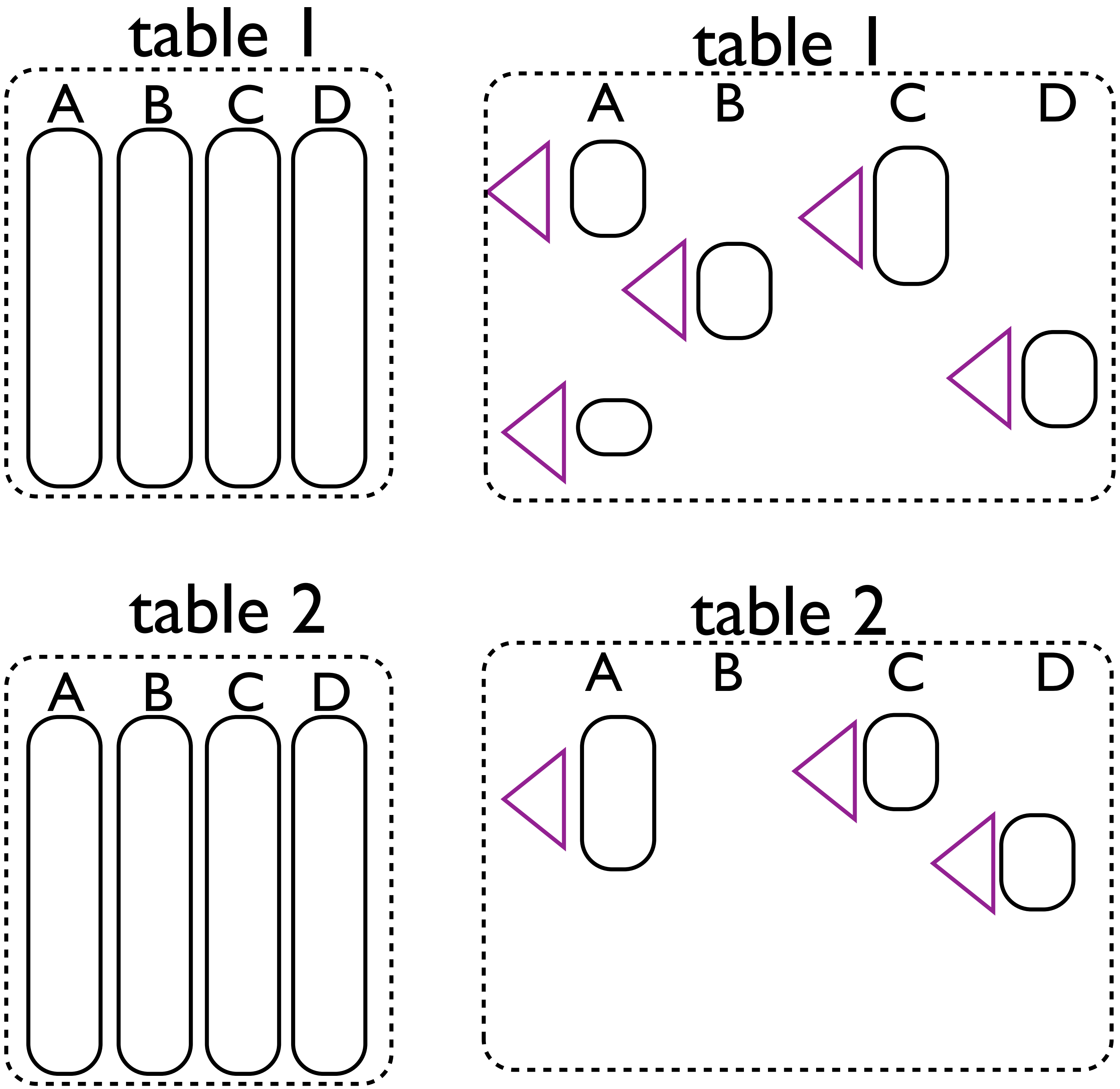


partial materialization

cracking tangram

base data

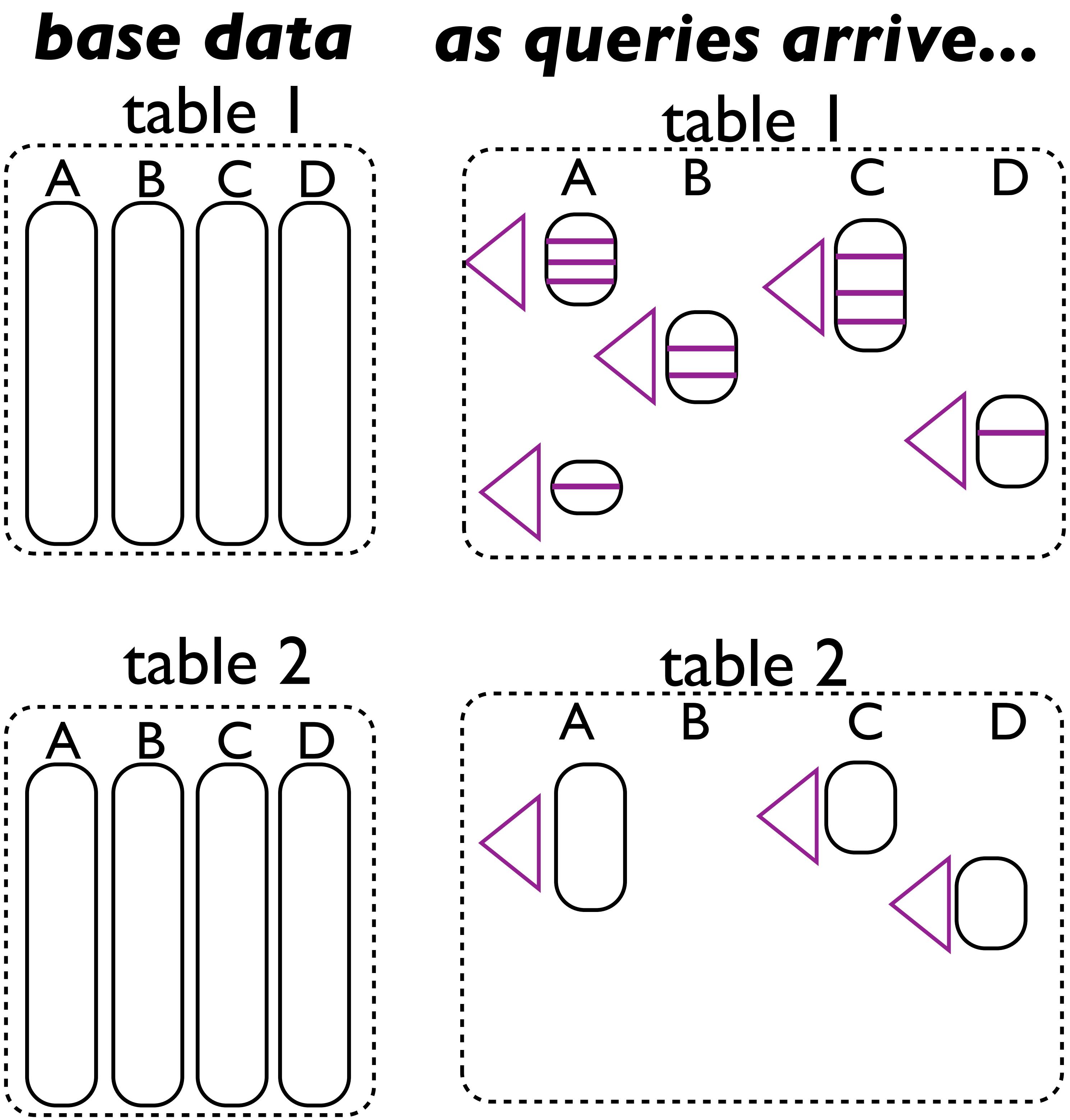
as queries arrive...



partial materialization

partial indexing

cracking tangram

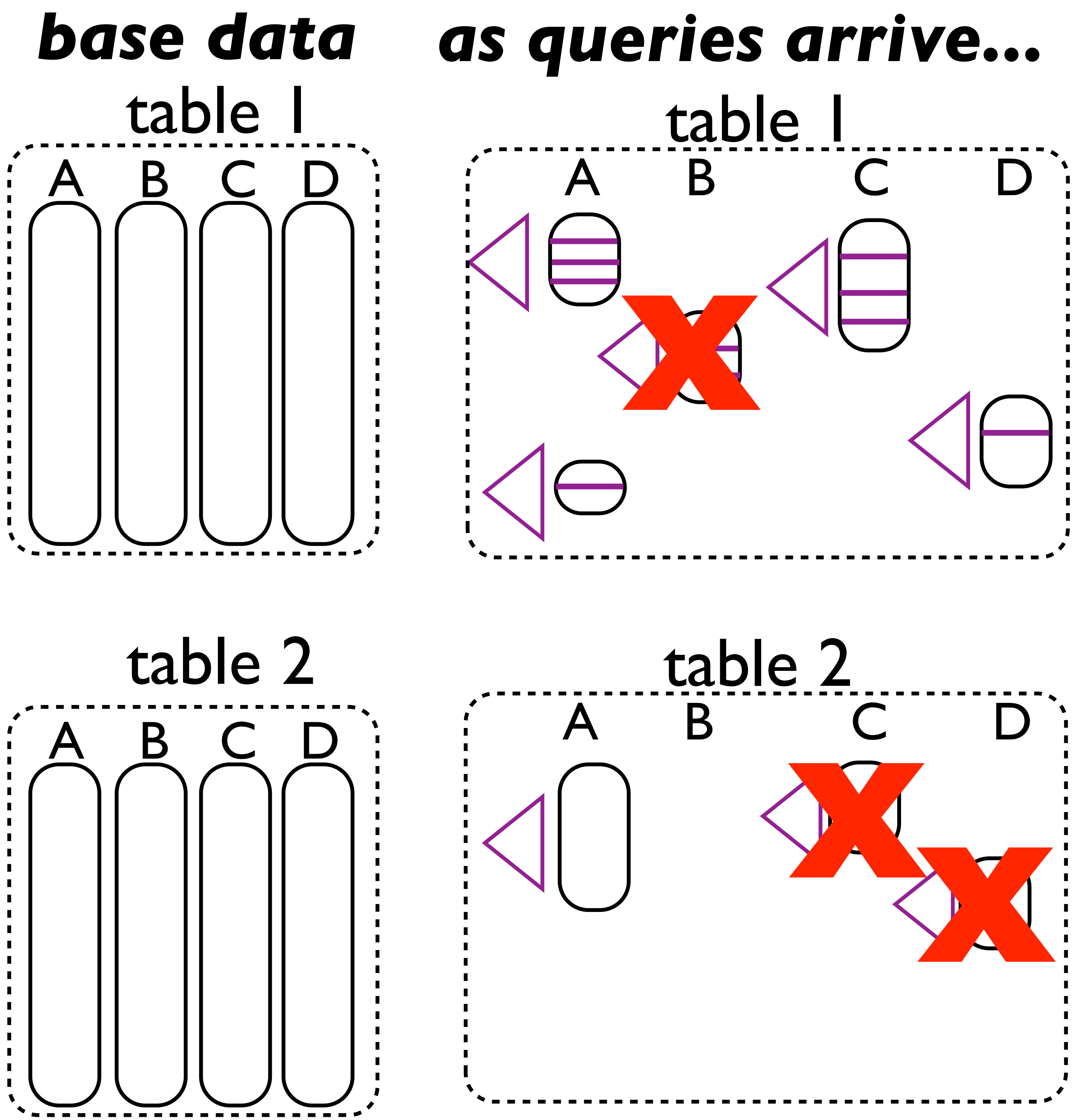


partial materialization

partial indexing

continuous adaptation

cracking tangram



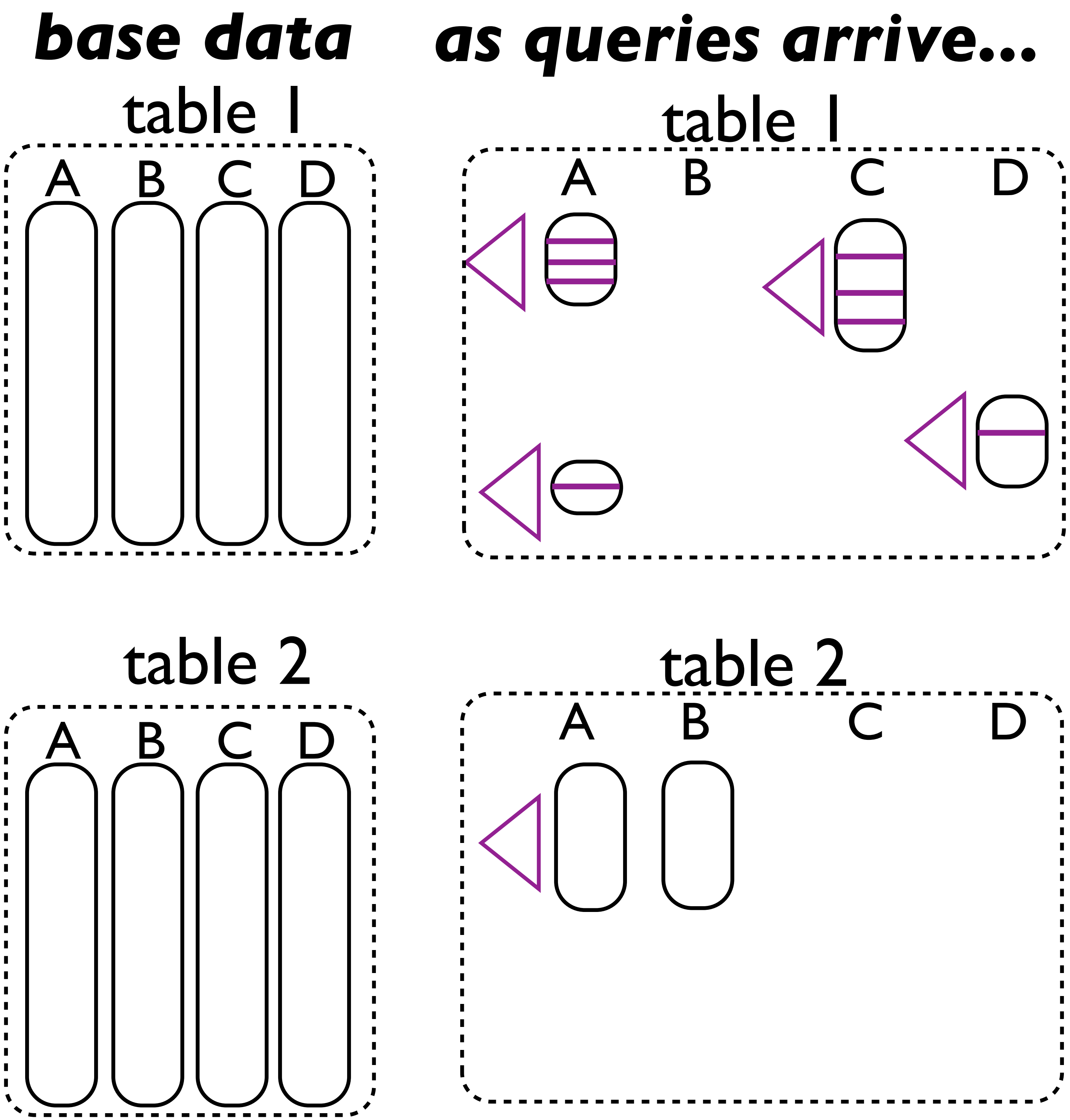
partial materialization

partial indexing

continuous adaptation

storage adaptation

cracking tangram



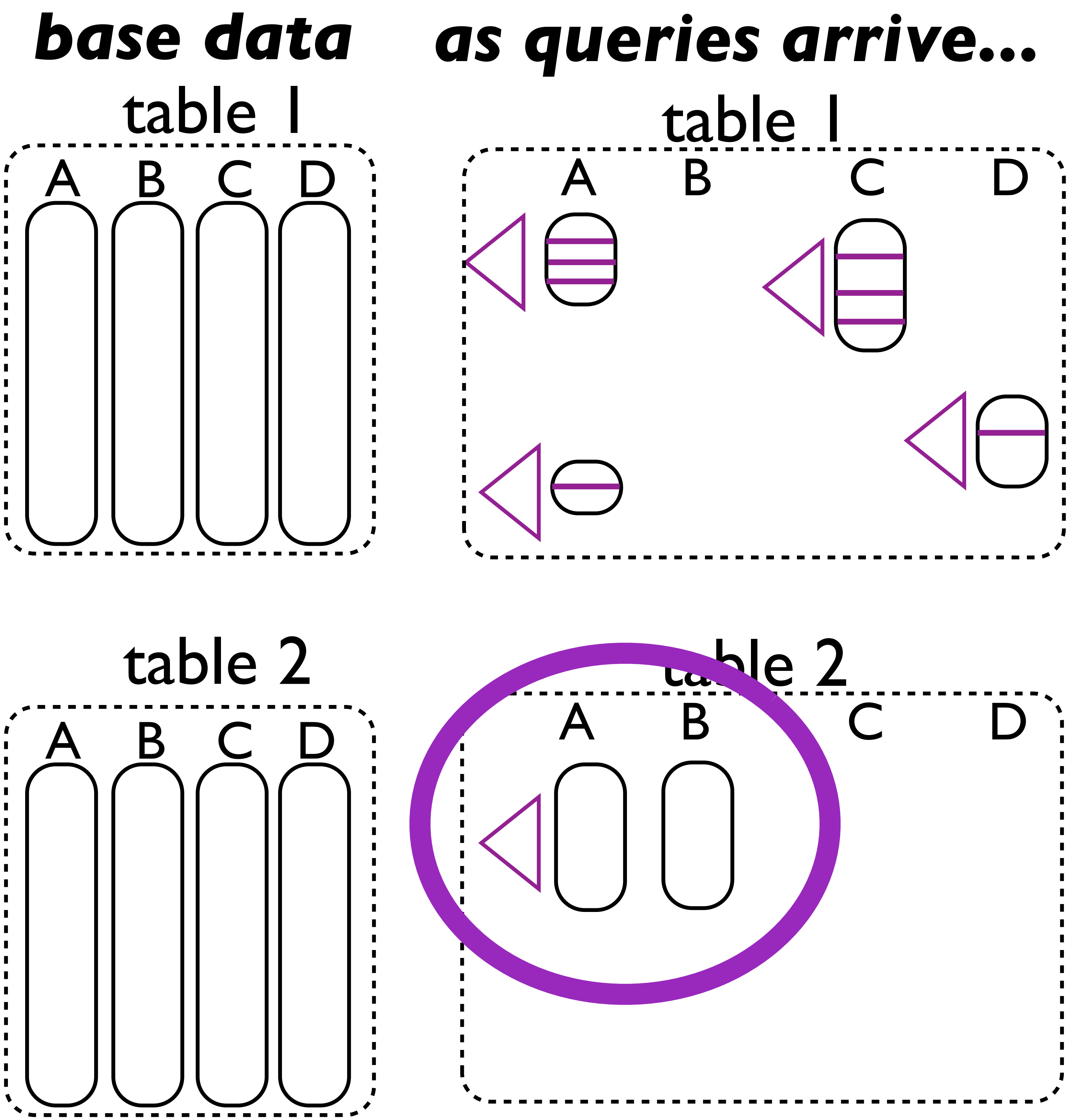
partial materialization

partial indexing

continuous adaptation

storage adaptation

cracking tangram



partial materialization

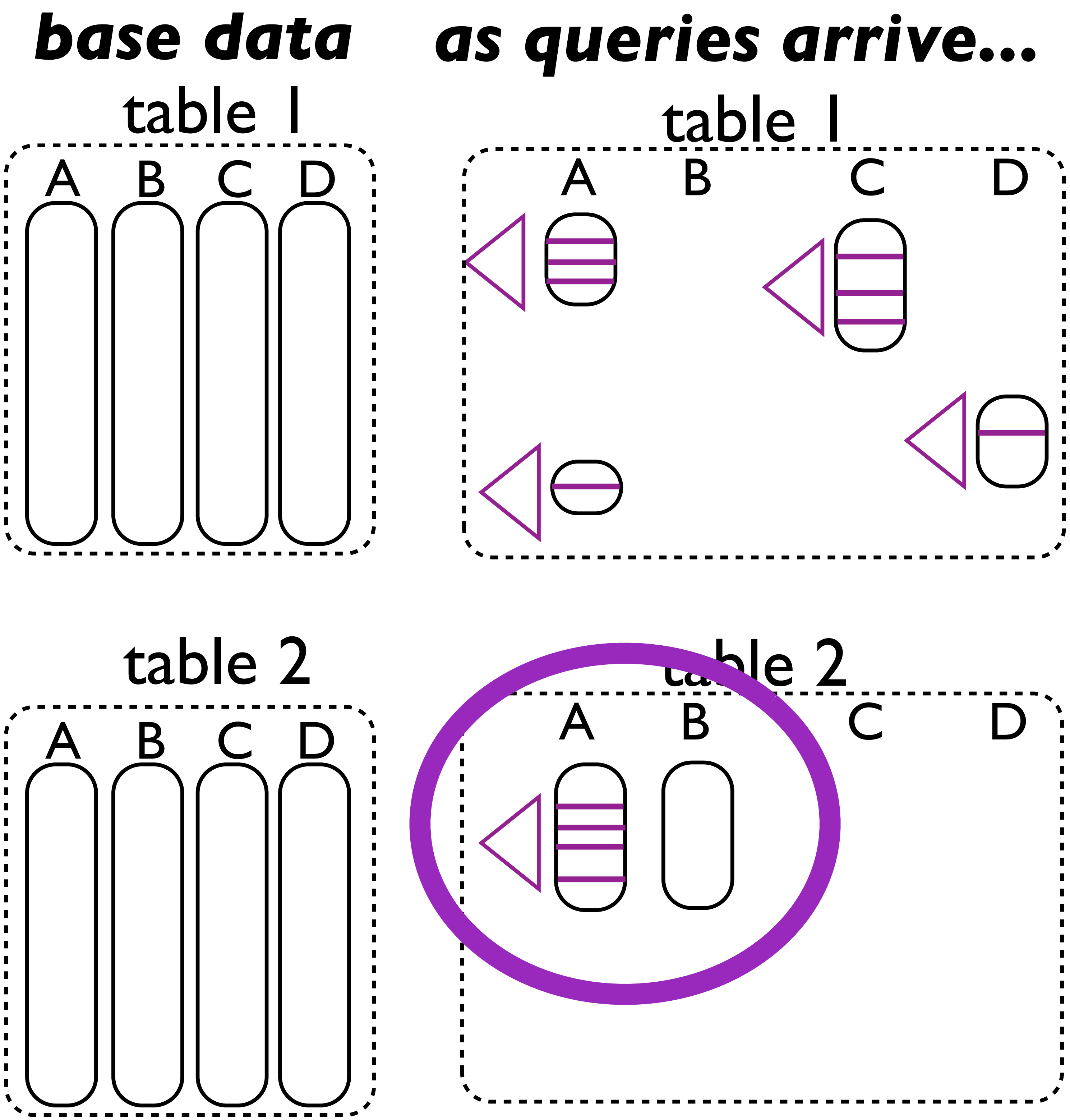
partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

cracking tangram



partial materialization

partial indexing

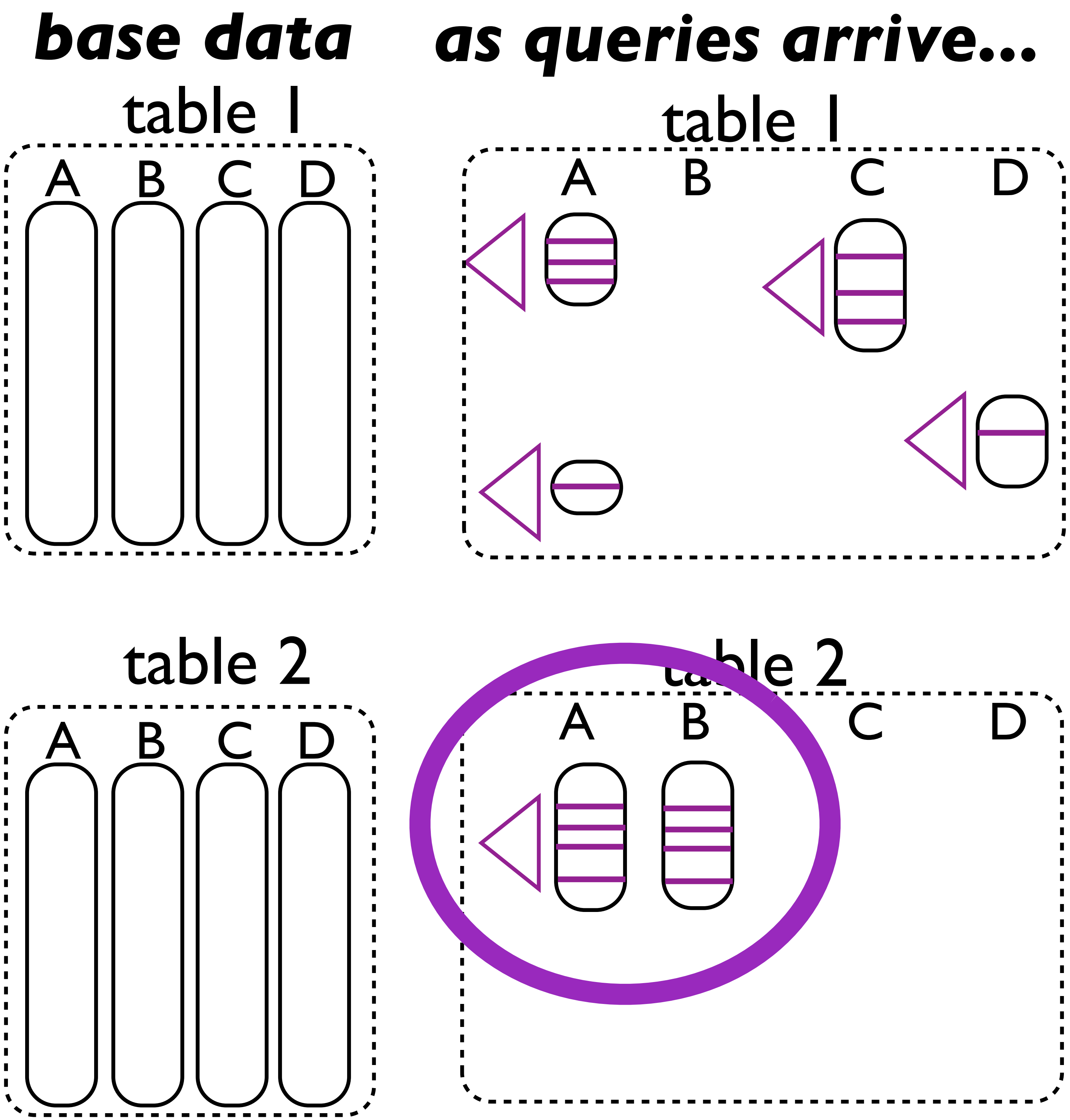
continuous adaptation

storage adaptation

no tuple reconstruction

adaptive alignment

cracking tangram



partial materialization

partial indexing

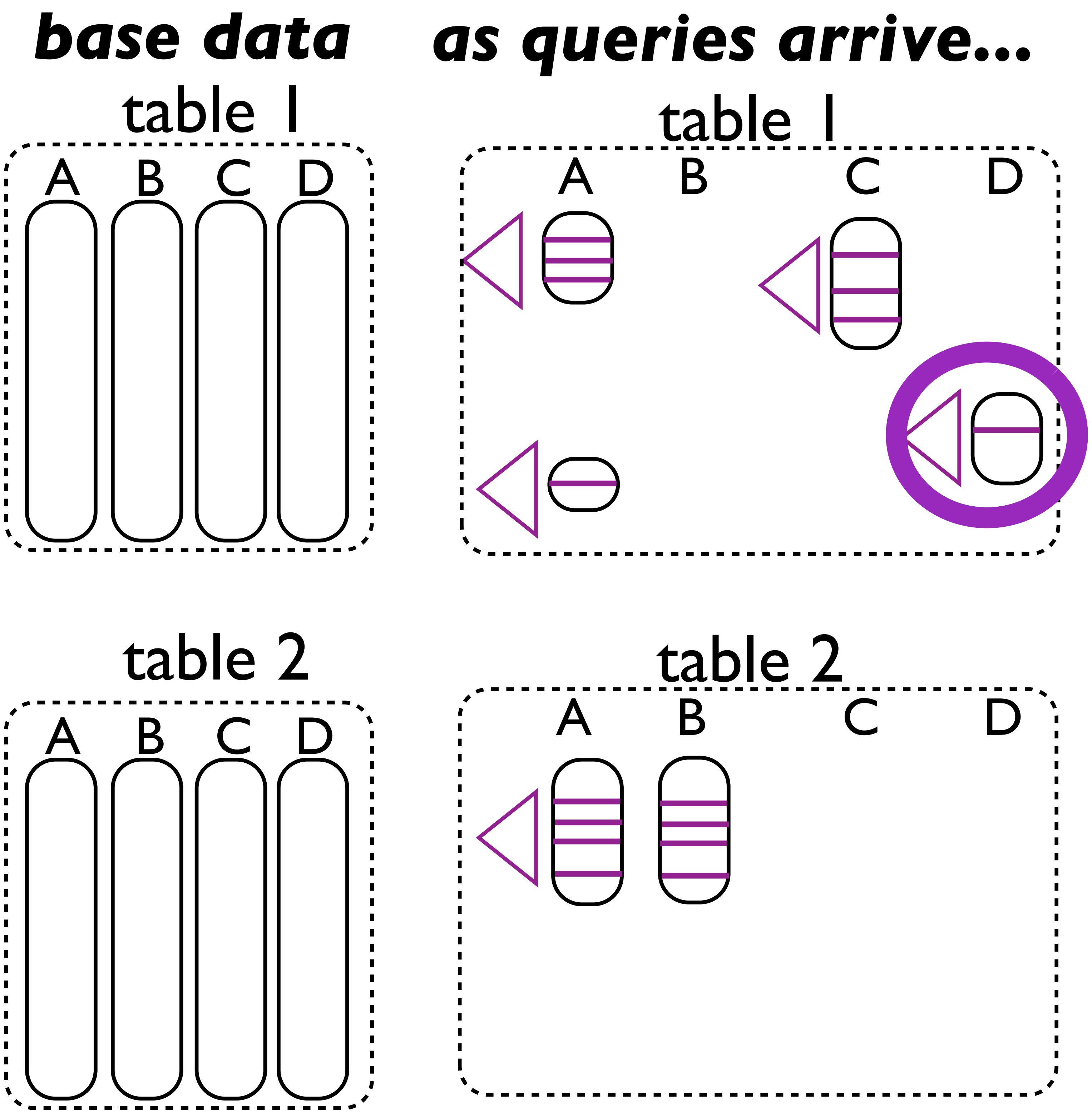
continuous adaptation

storage adaptation

no tuple reconstruction

adaptive alignment

cracking tangram



partial materialization

partial indexing

continuous adaptation

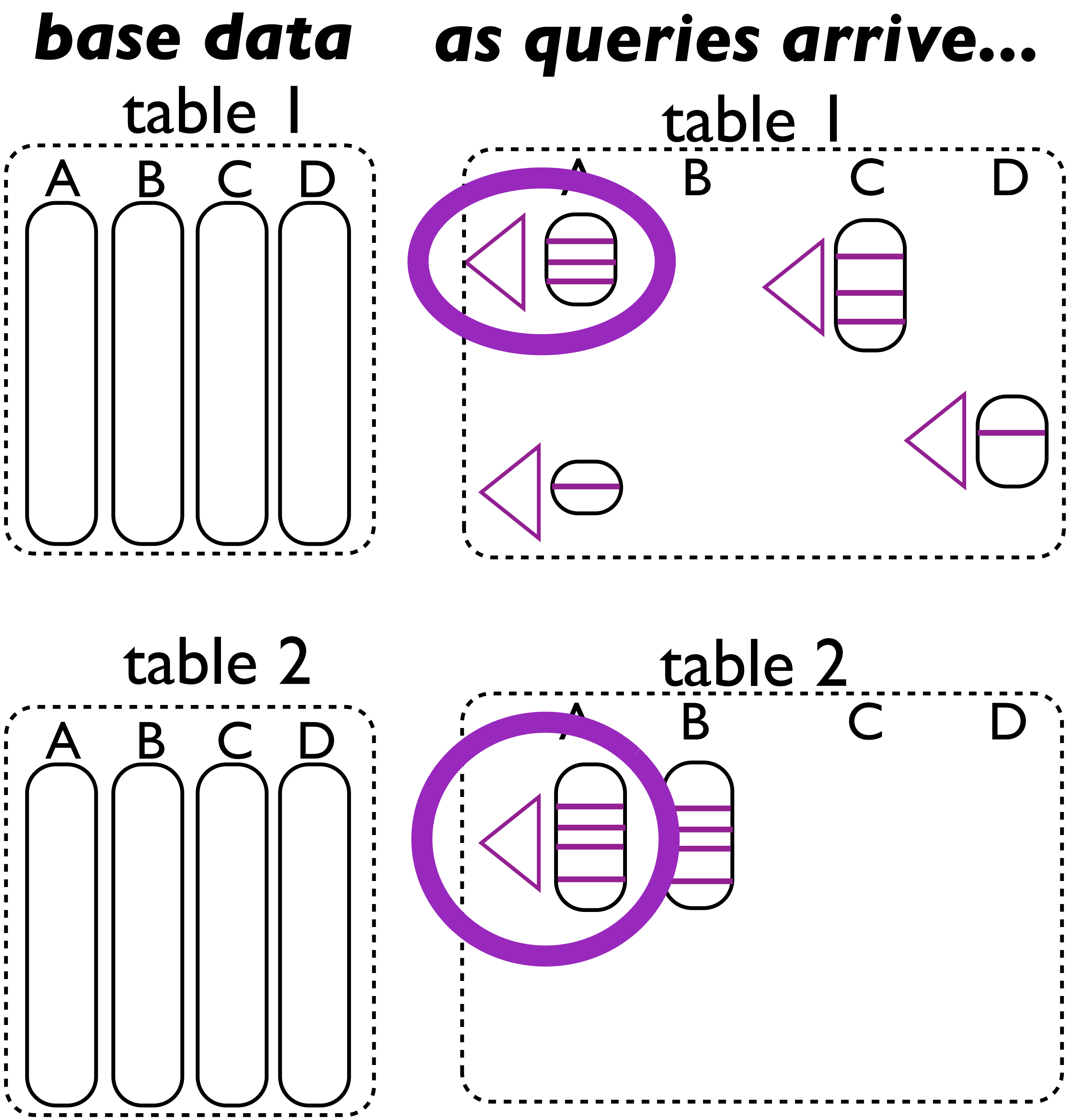
storage adaptation

no tuple reconstruction

adaptive alignment

sort in caches

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

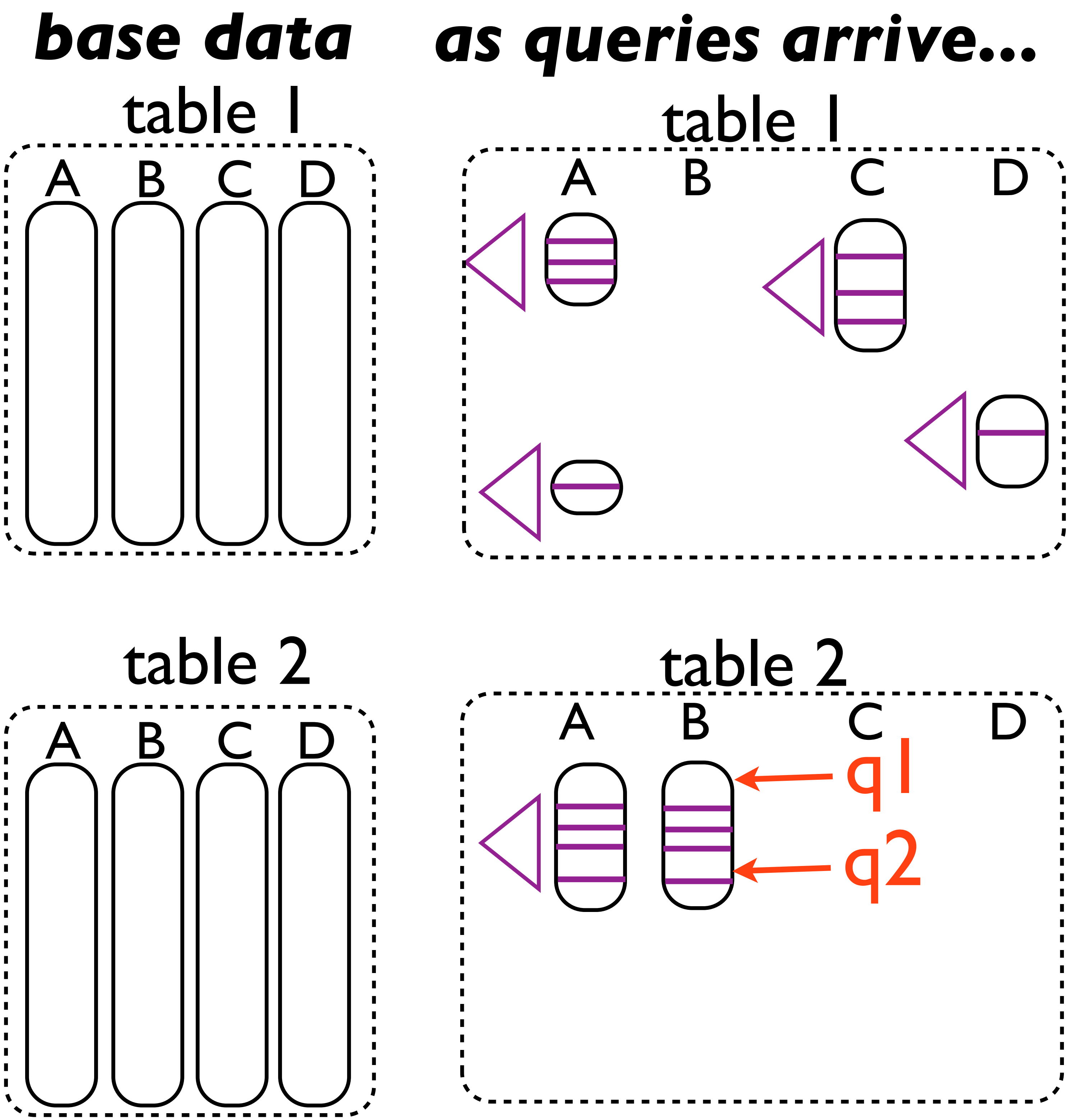
no tuple reconstruction

adaptive alignment

sort in caches

crack joins

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

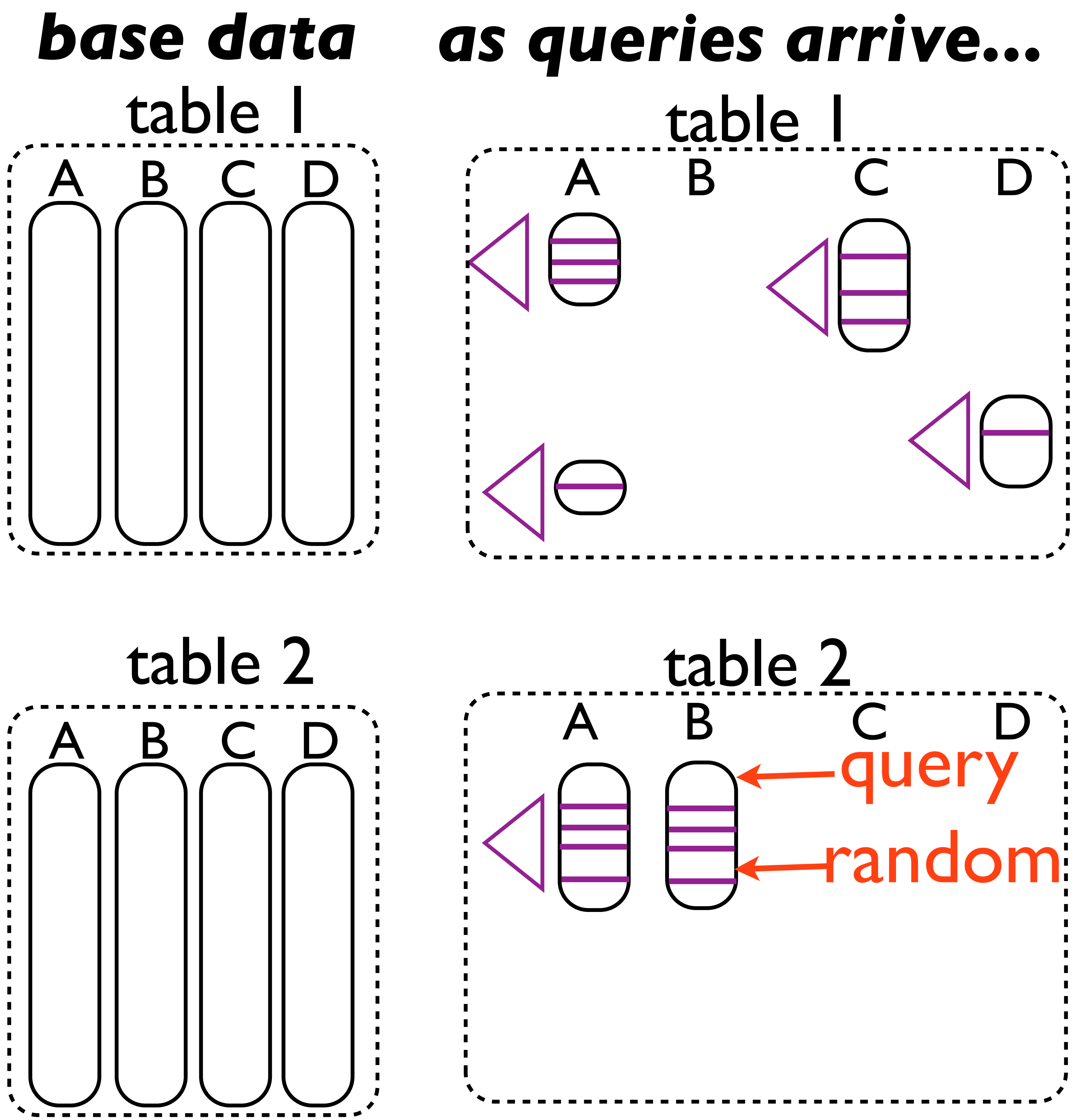
adaptive alignment

sort in caches

crack joins

lightweight locking

cracking tangram



partial materialization

partial indexing

continuous adaptation

storage adaptation

no tuple reconstruction

adaptive alignment

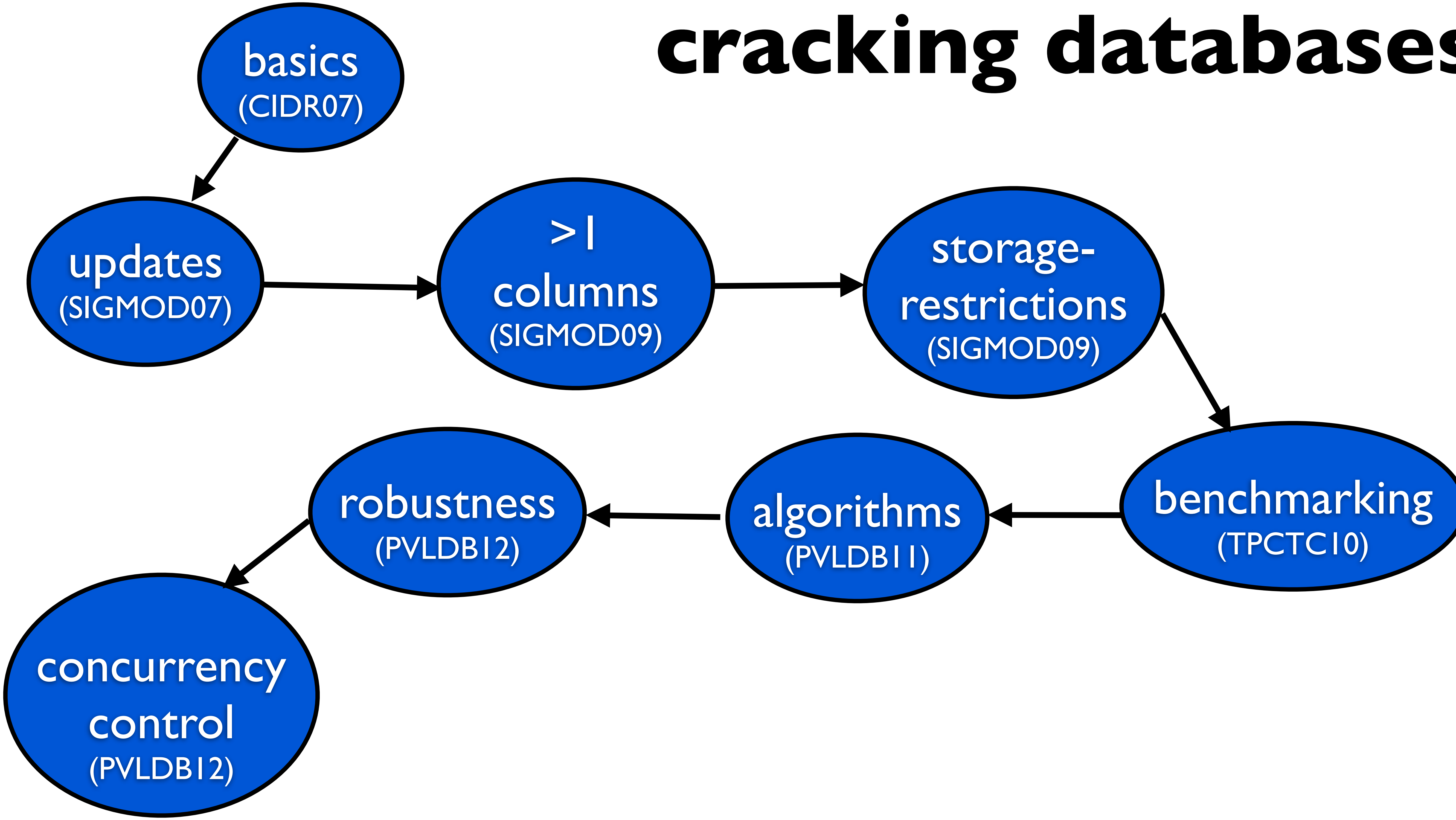
sort in caches

crack joins

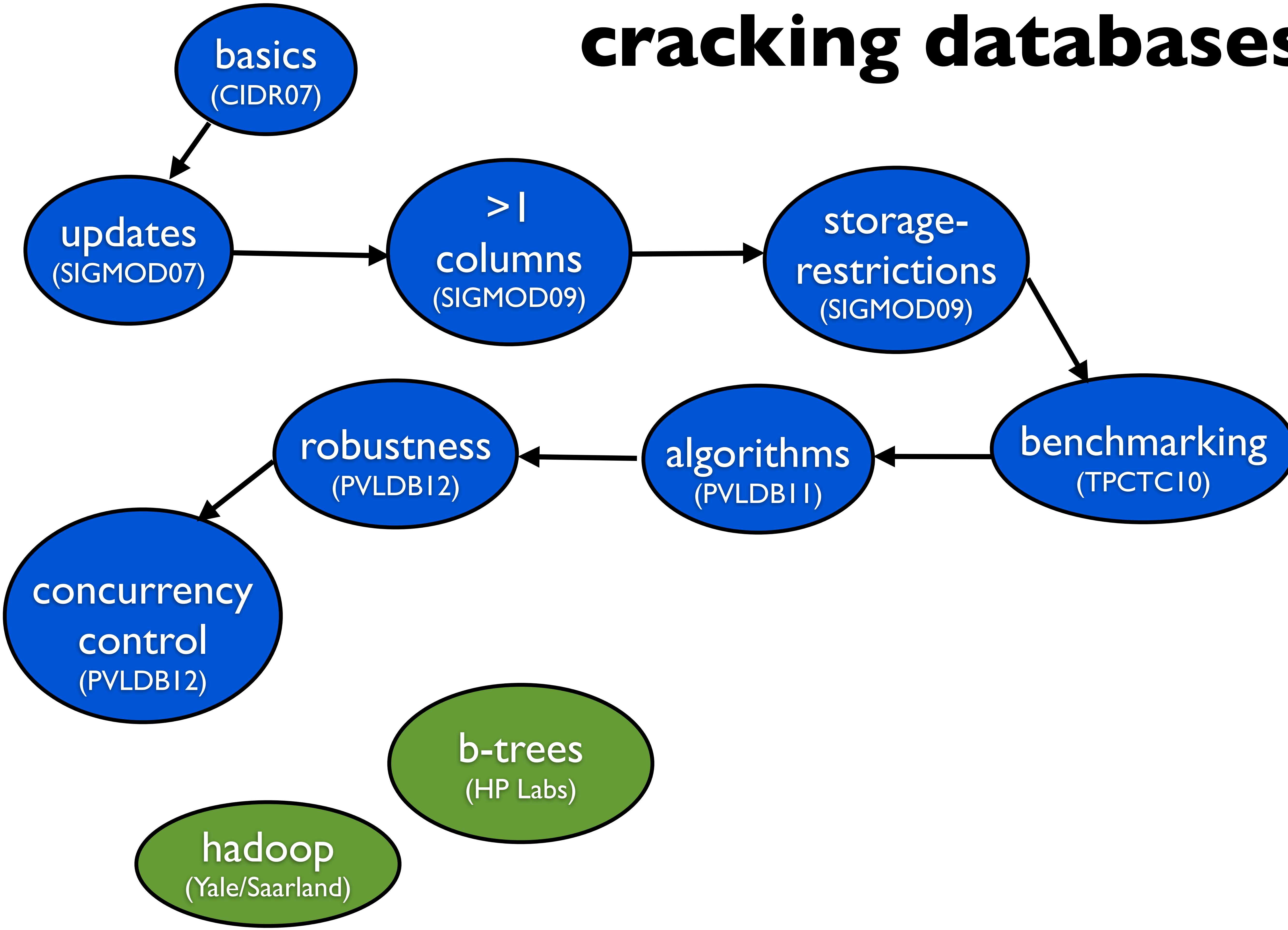
lightweight locking

stochastic cracking

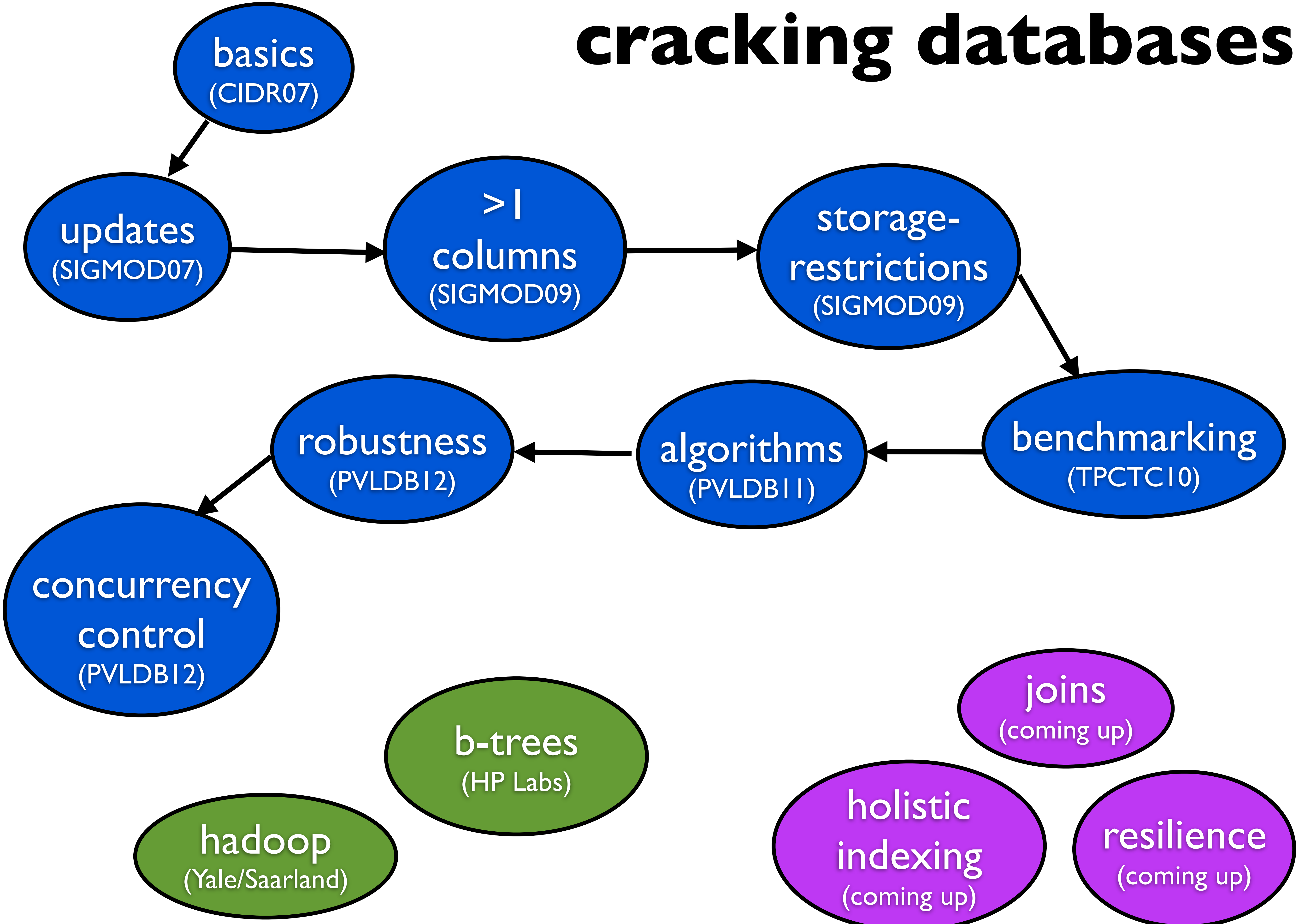
cracking databases



cracking databases



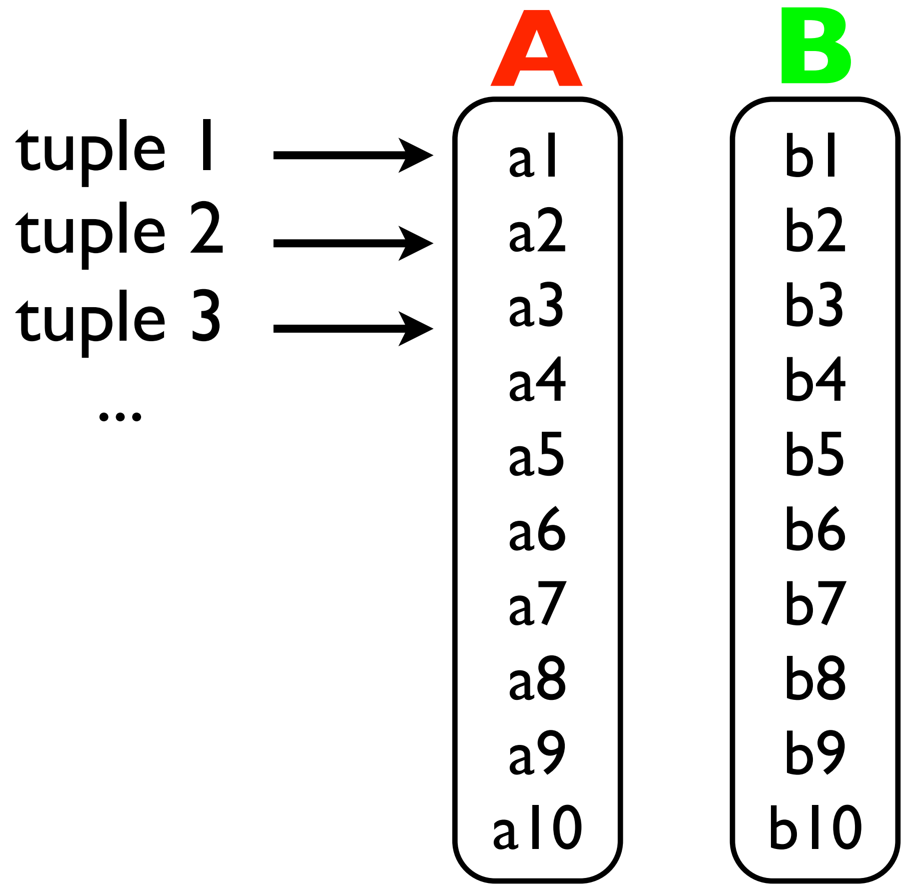
cracking databases



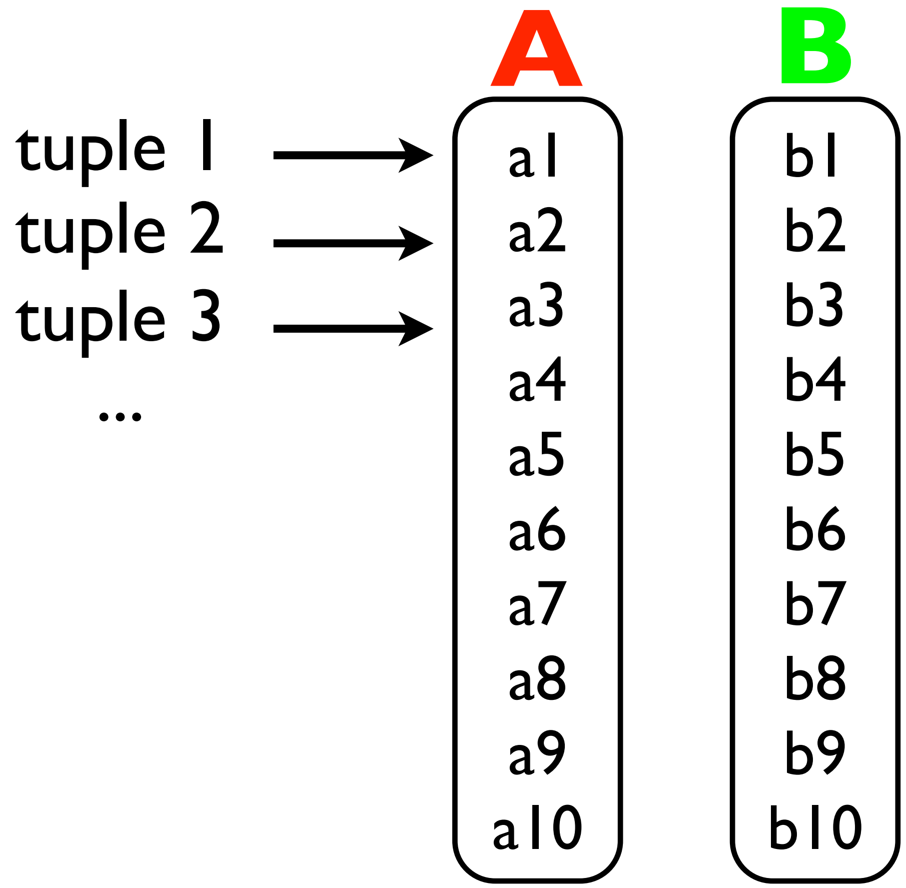
positional alignment

lookup

$A(i) = A + i * width(A)$



positional alignment



lookup

A(i) = A + i * width(A)

query

max(B) where A<10

positional alignment ➜

tuple 1 → a1
tuple 2 → a2
tuple 3 → a3
...

A	B
a1	b1
a2	b2
a3	b3
a4	b4
a5	b5
a6	b6
a7	b7
a8	b8
a9	b9
a10	b10

lookup

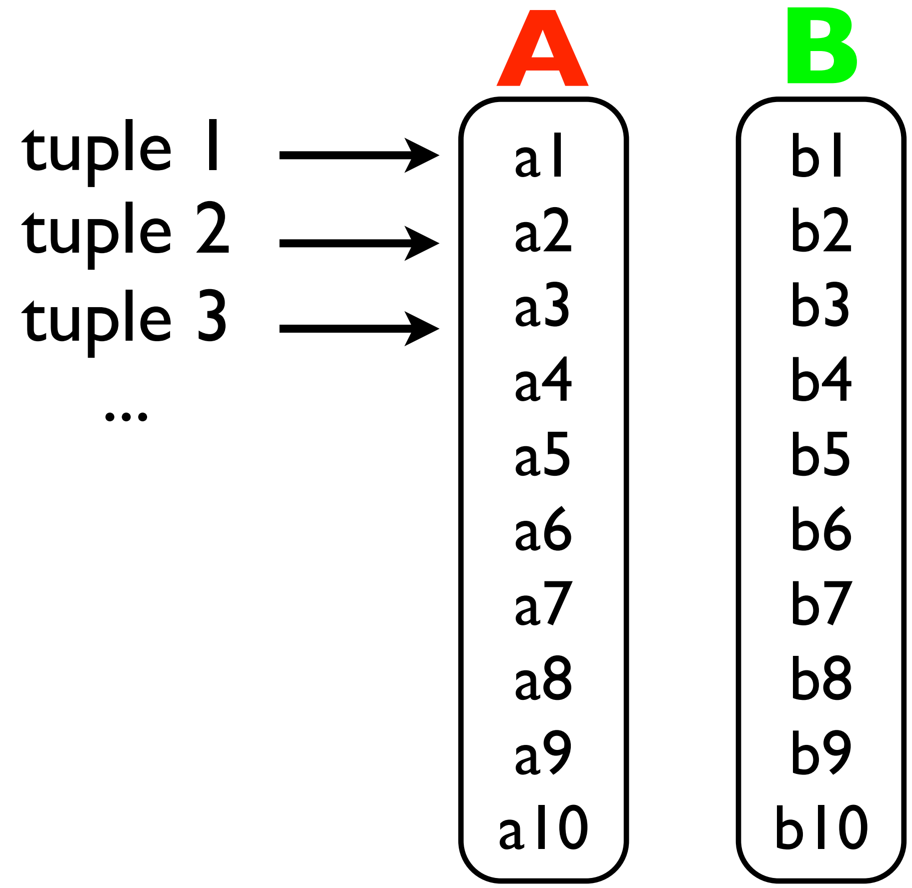
A(i) = A + i * width(A)

query

max(**B**) where **A**<10

A
a1
a2
a3
a4
a5
a6
a7
a8
a9
a10

positional alignment

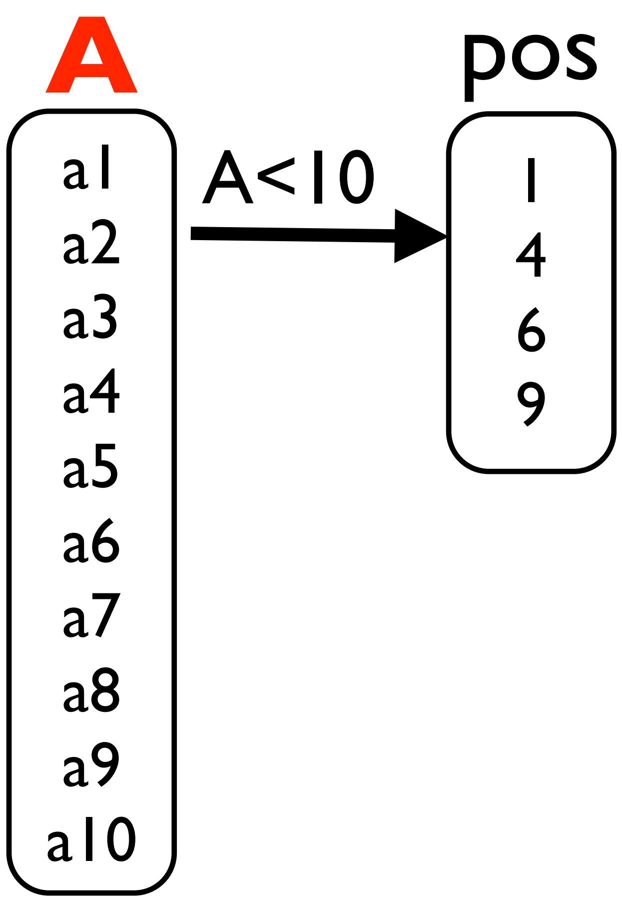


lookup

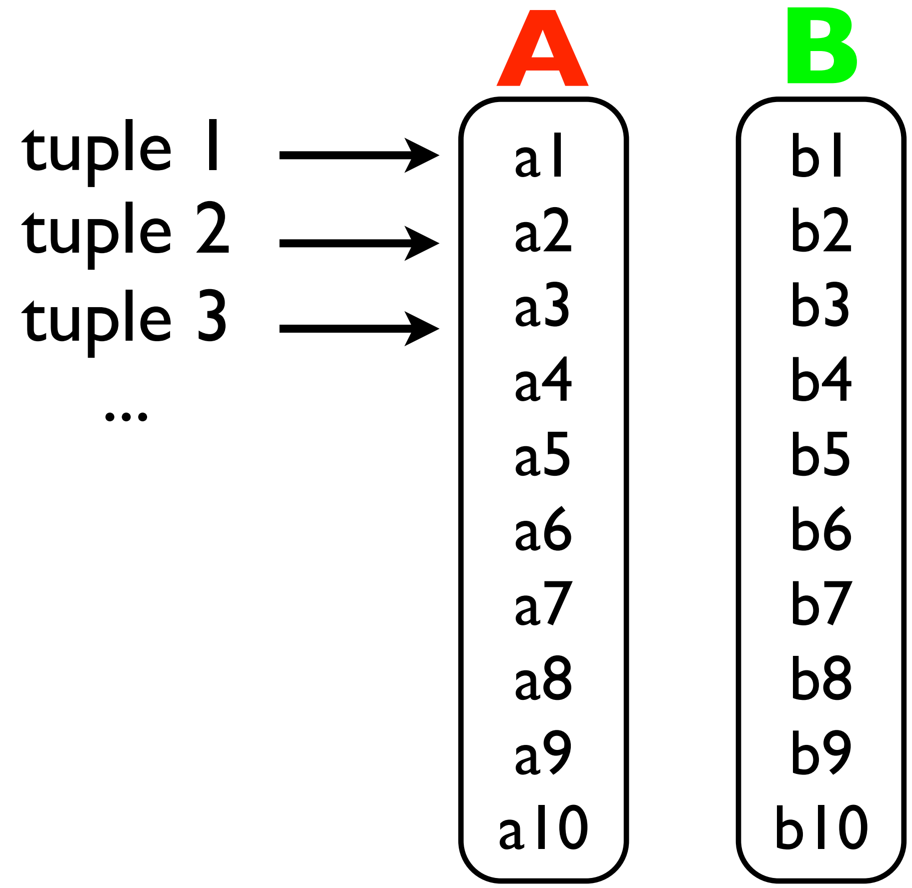
A(i) = A + i * width(A)

query

max(B) where A<10



positional alignment



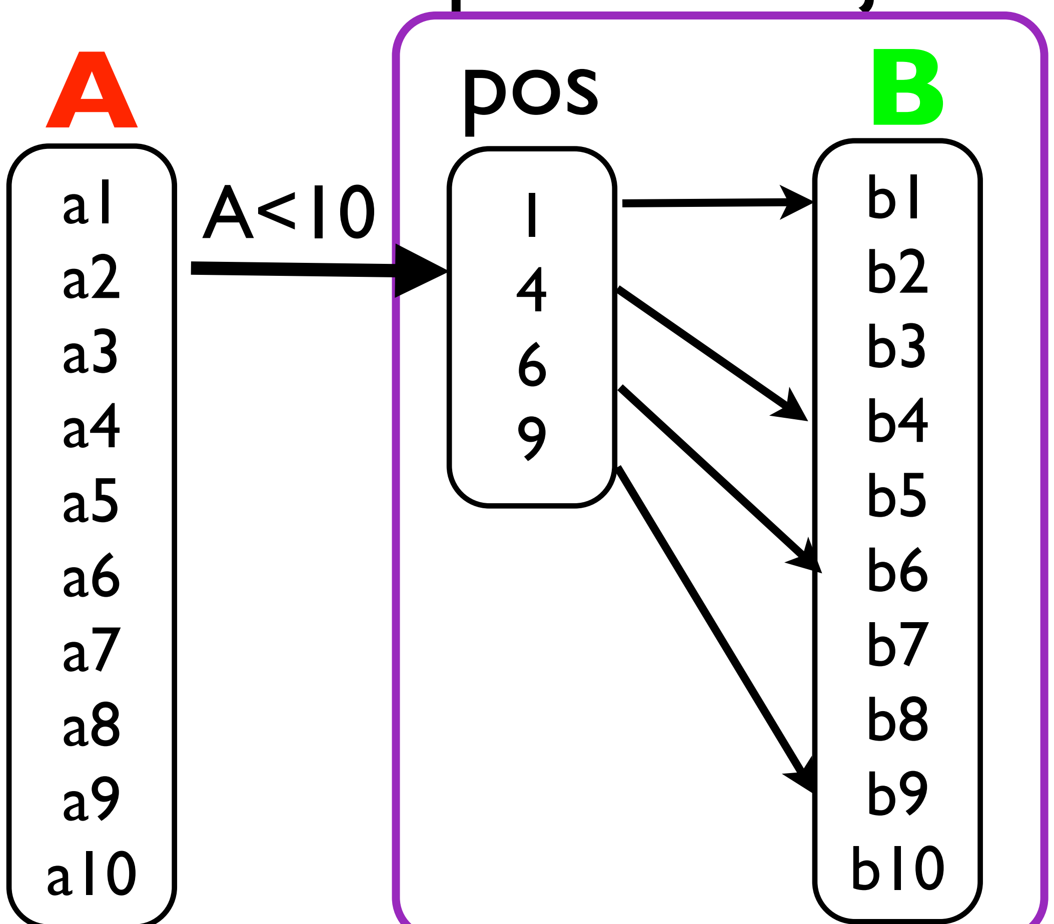
lookup

A(i) = A + i * width(A)

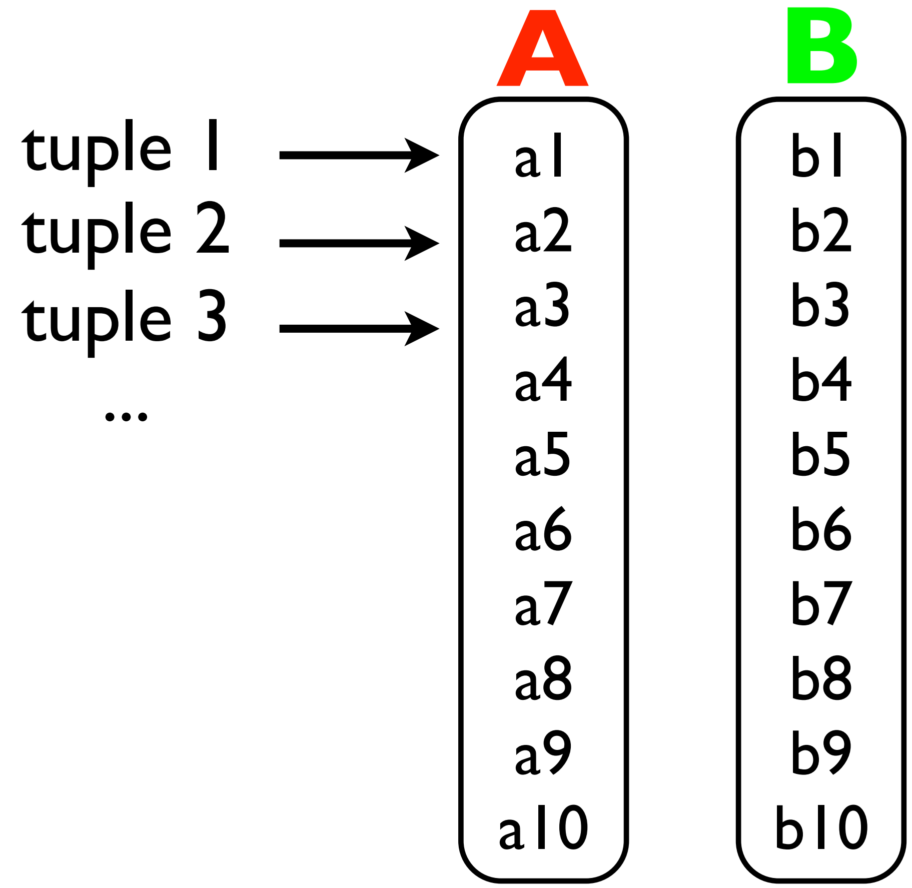
query

max(B) where A<10

positional join



positional alignment

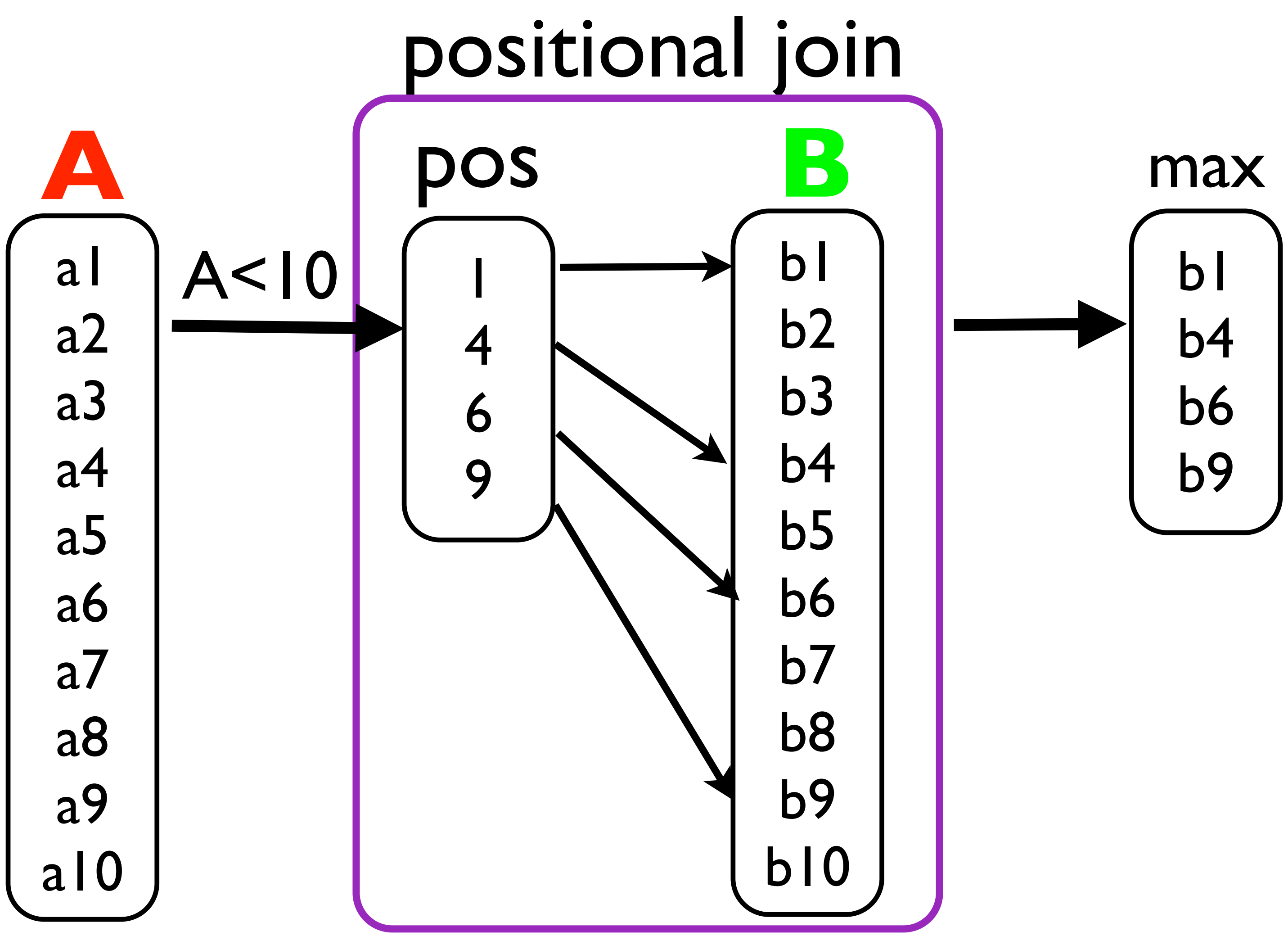


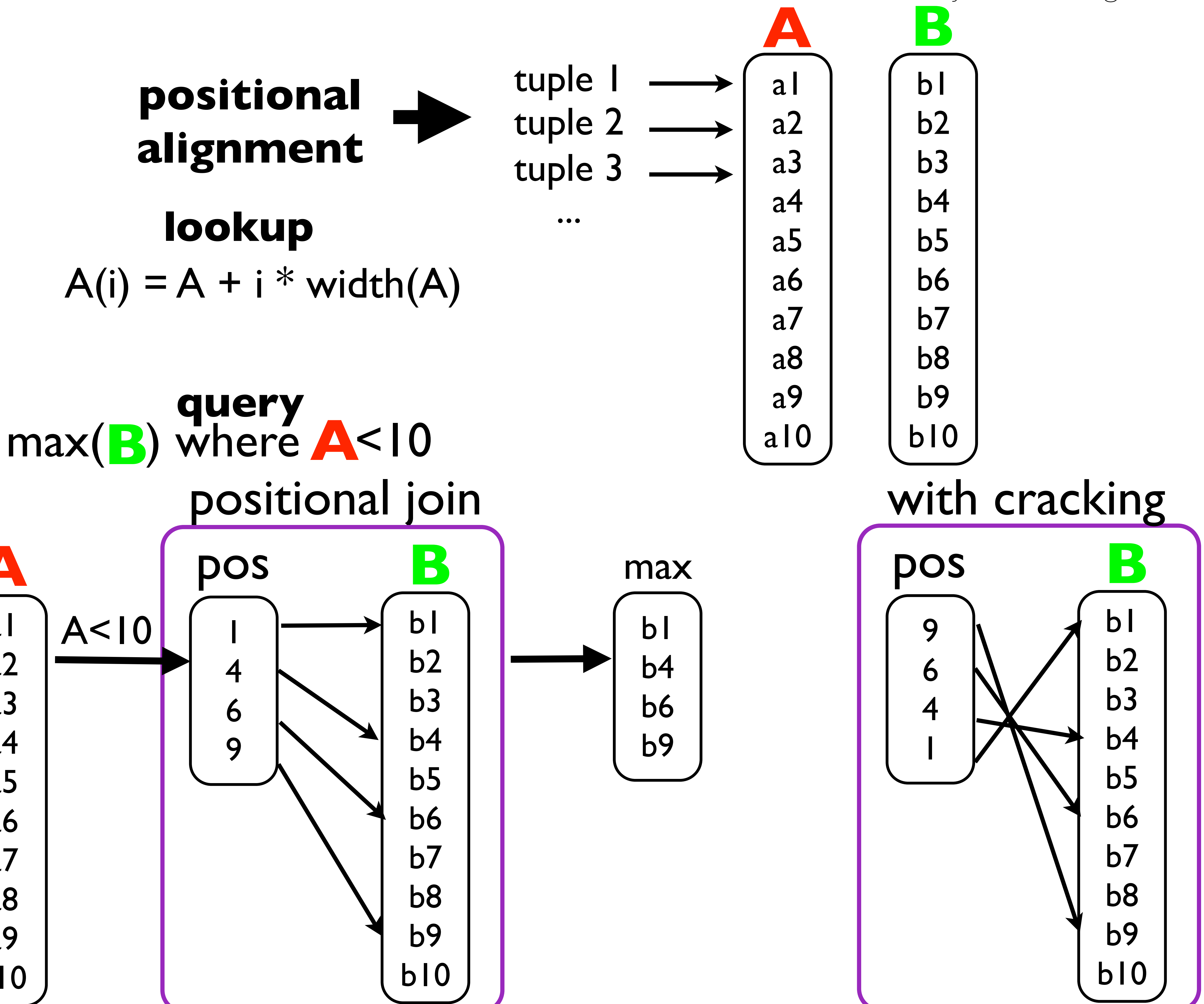
lookup

A(i) = A + i * width(A)

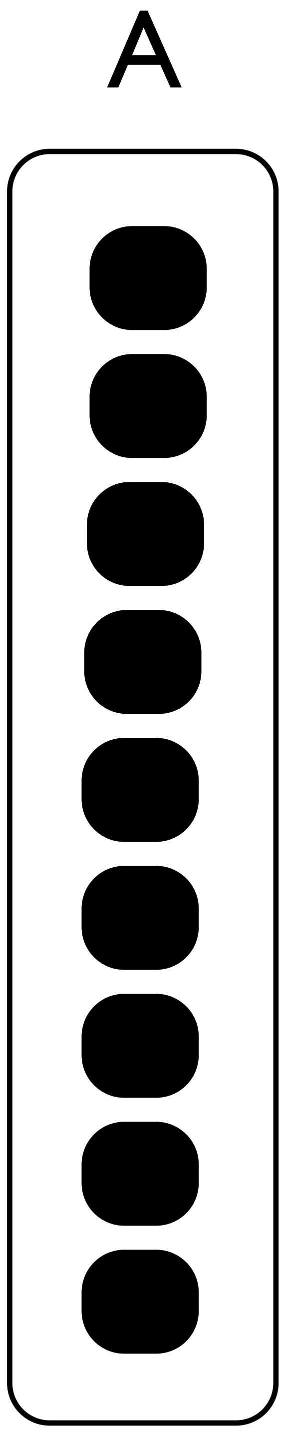
query

max(B) where A<10

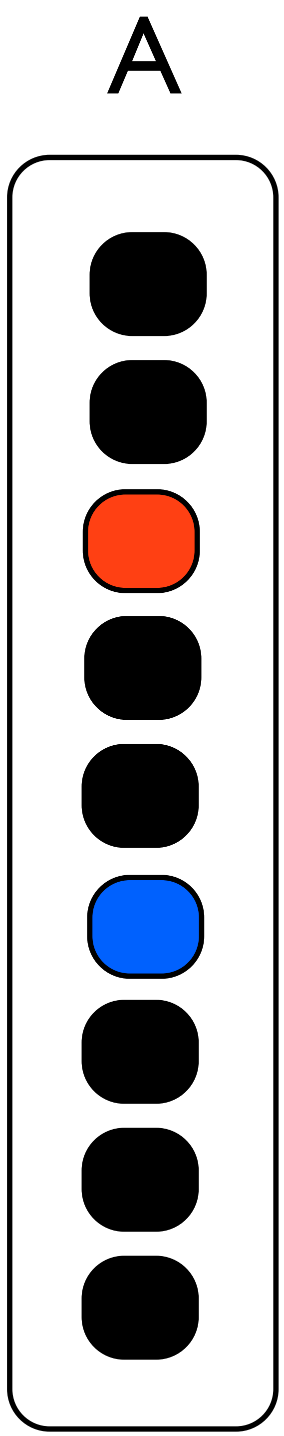



positional
alignment
tuple 1
tuple 2
tuple 3
...
lookup
A(i) = A + i * width(A)
A
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
B
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10
query
max(B) where A<10
positional join
A
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
A<10
pos
1 4 6 9
B
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10
max
b1 b4 b6 b9
with cracking
pos
9 6 4 1
B
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10

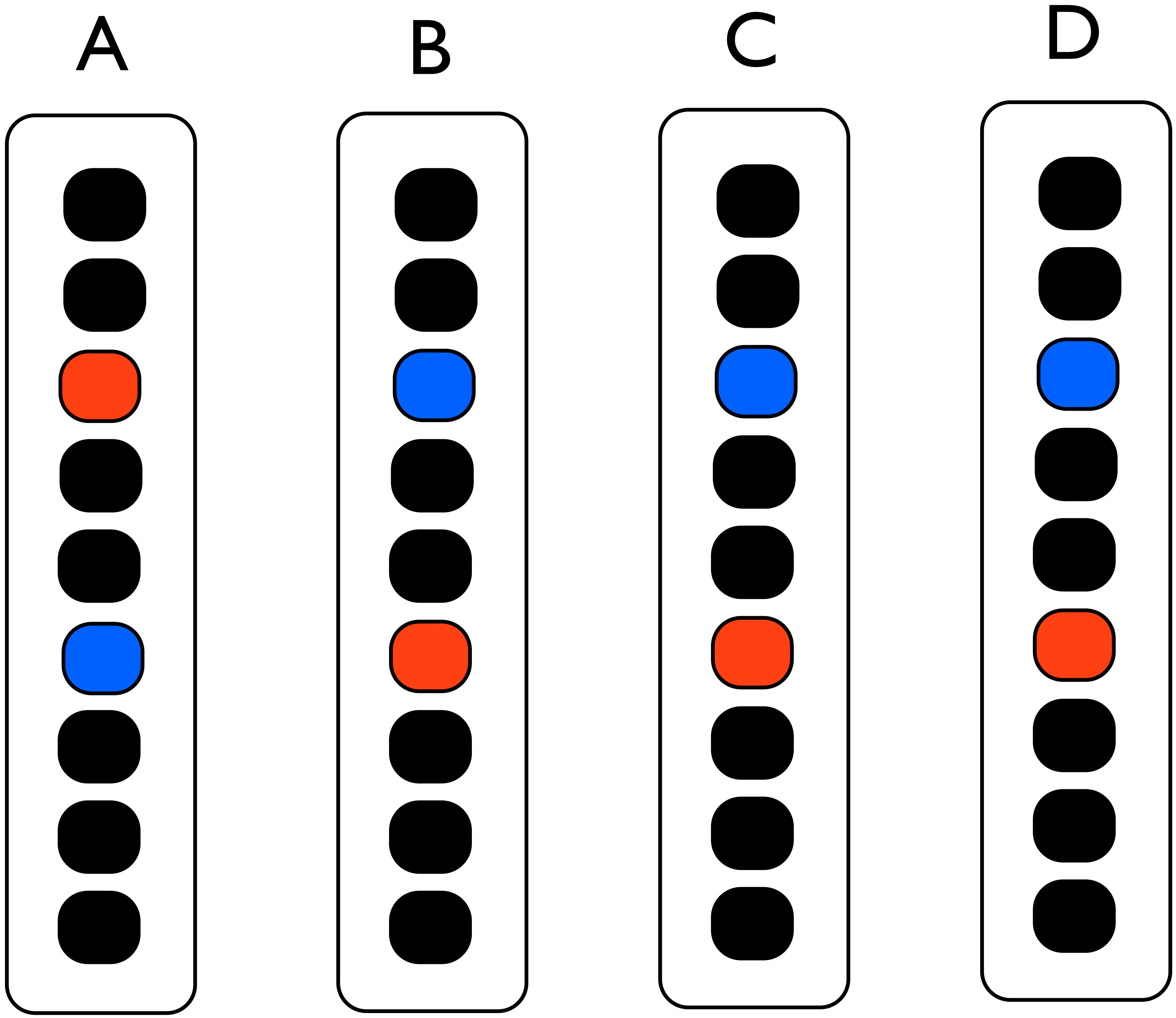
sideways cracking



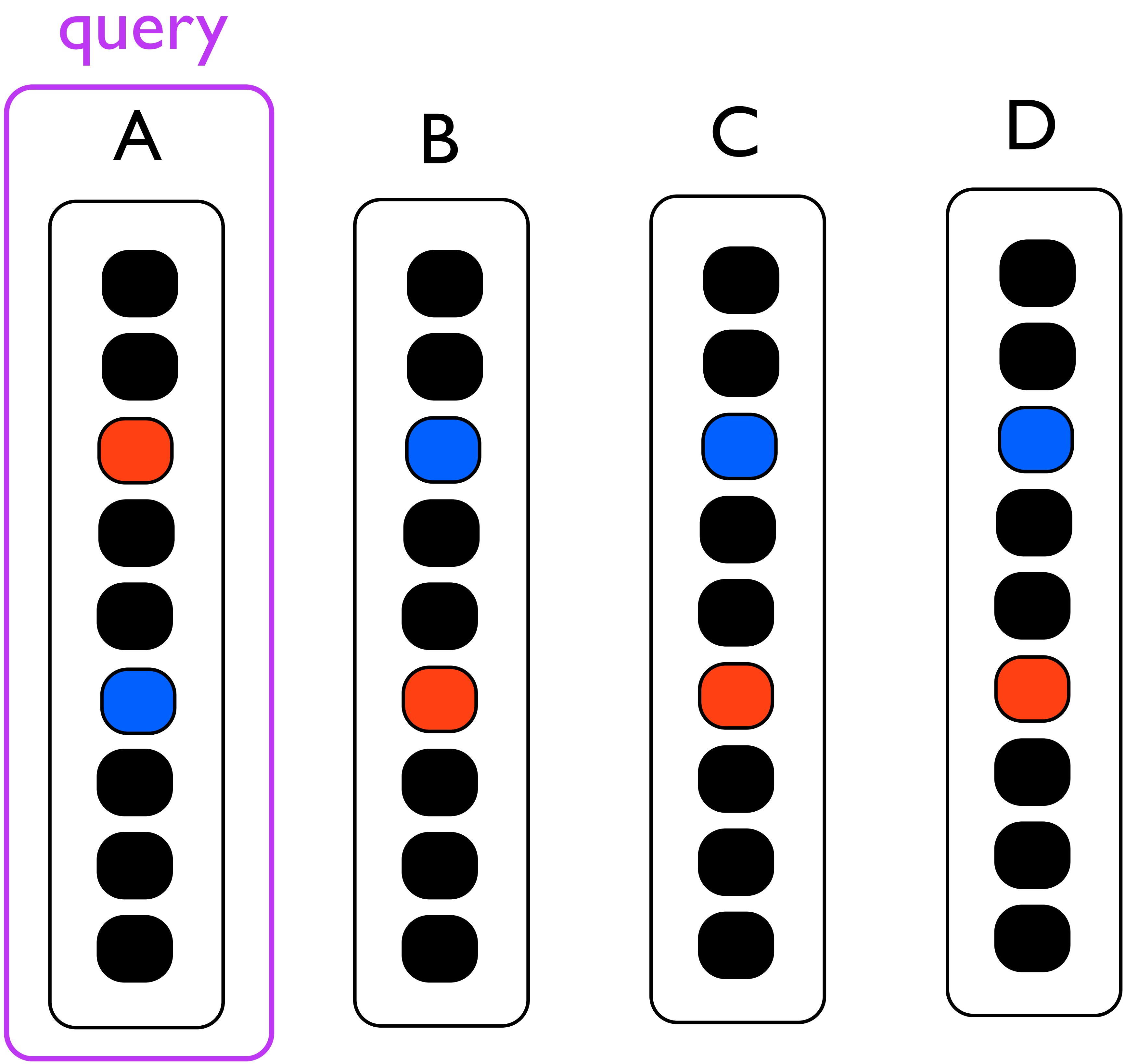
sideways cracking



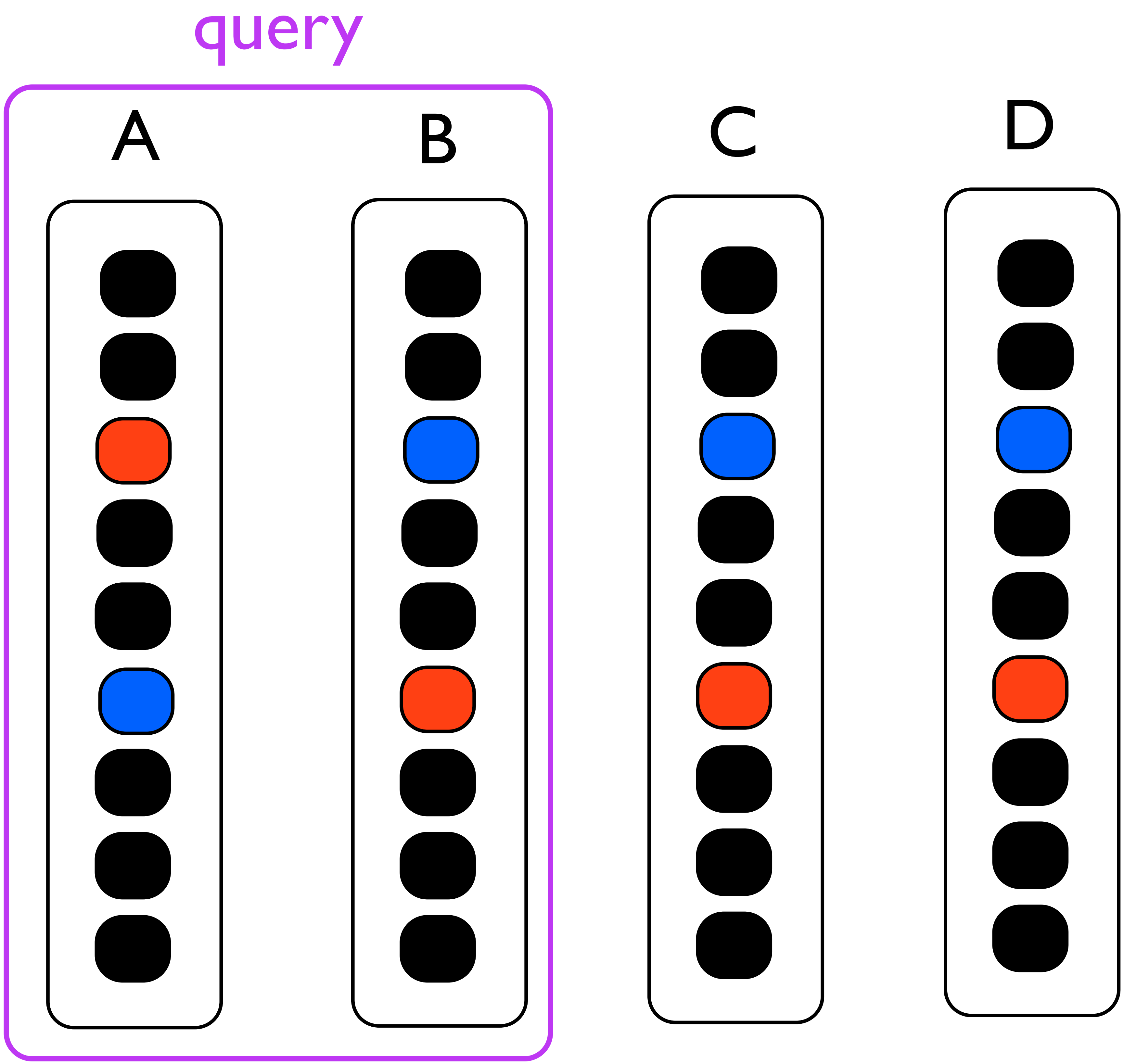
sideways cracking



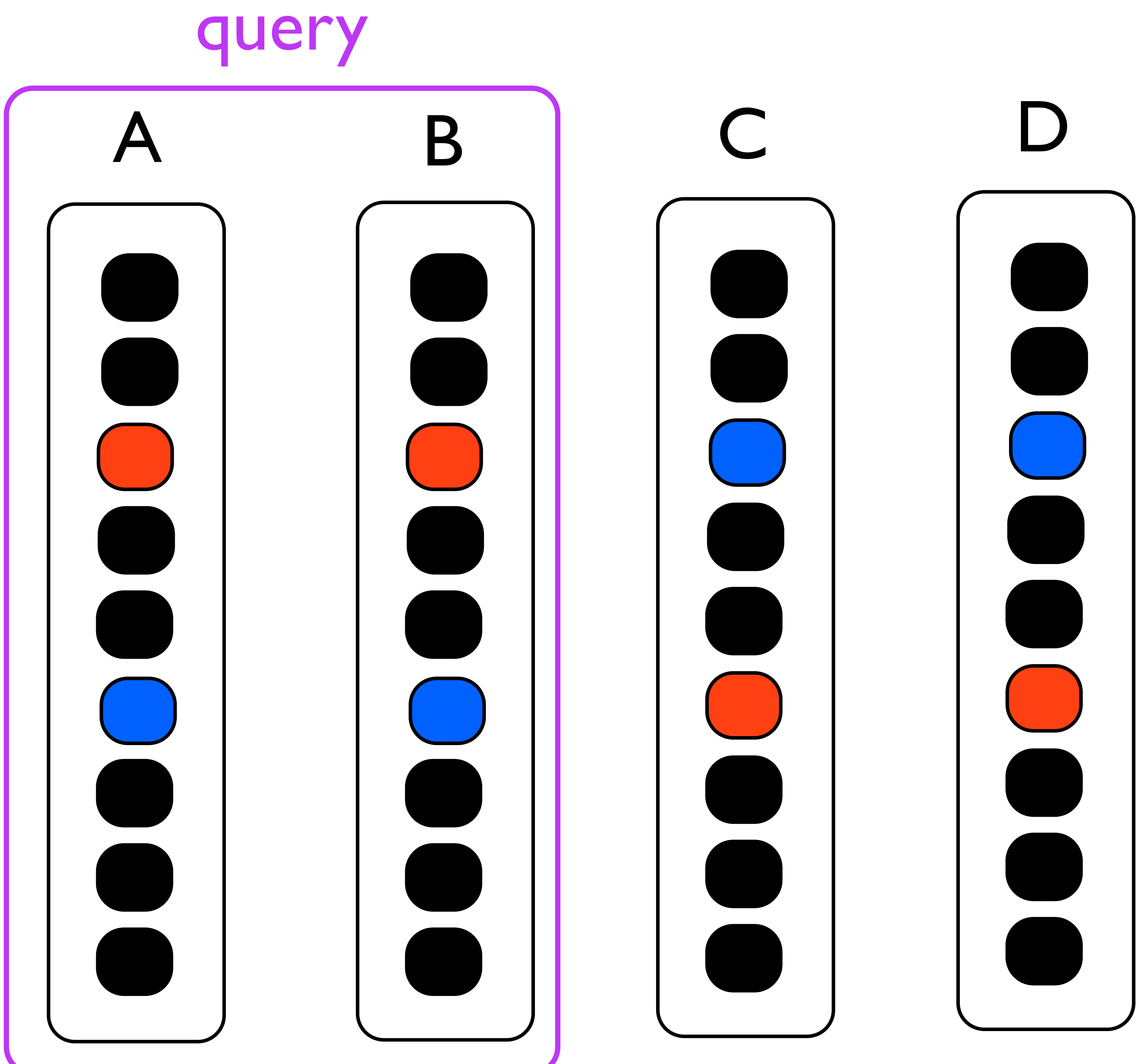
sideways cracking



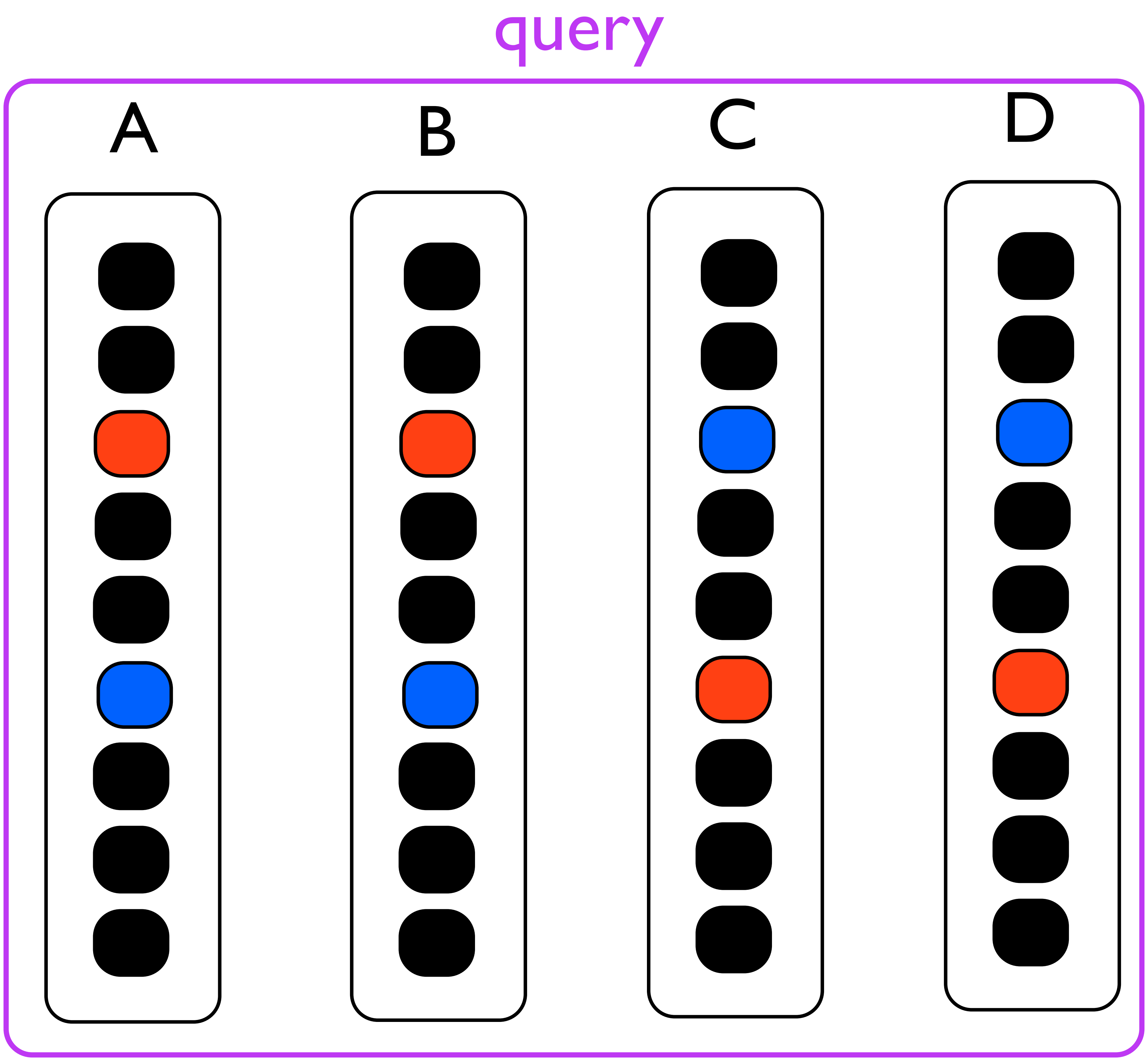
sideways cracking



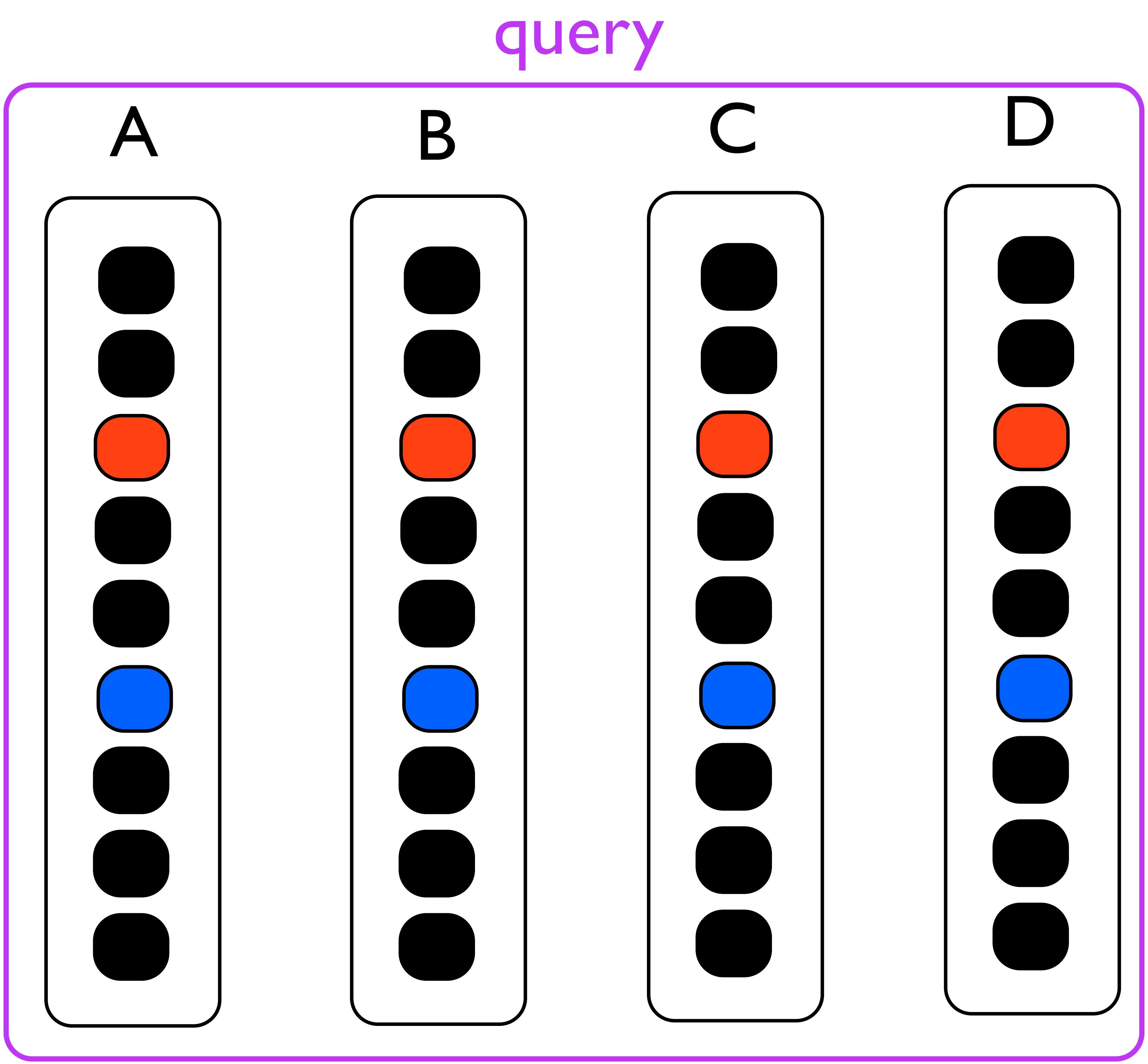
sideways cracking



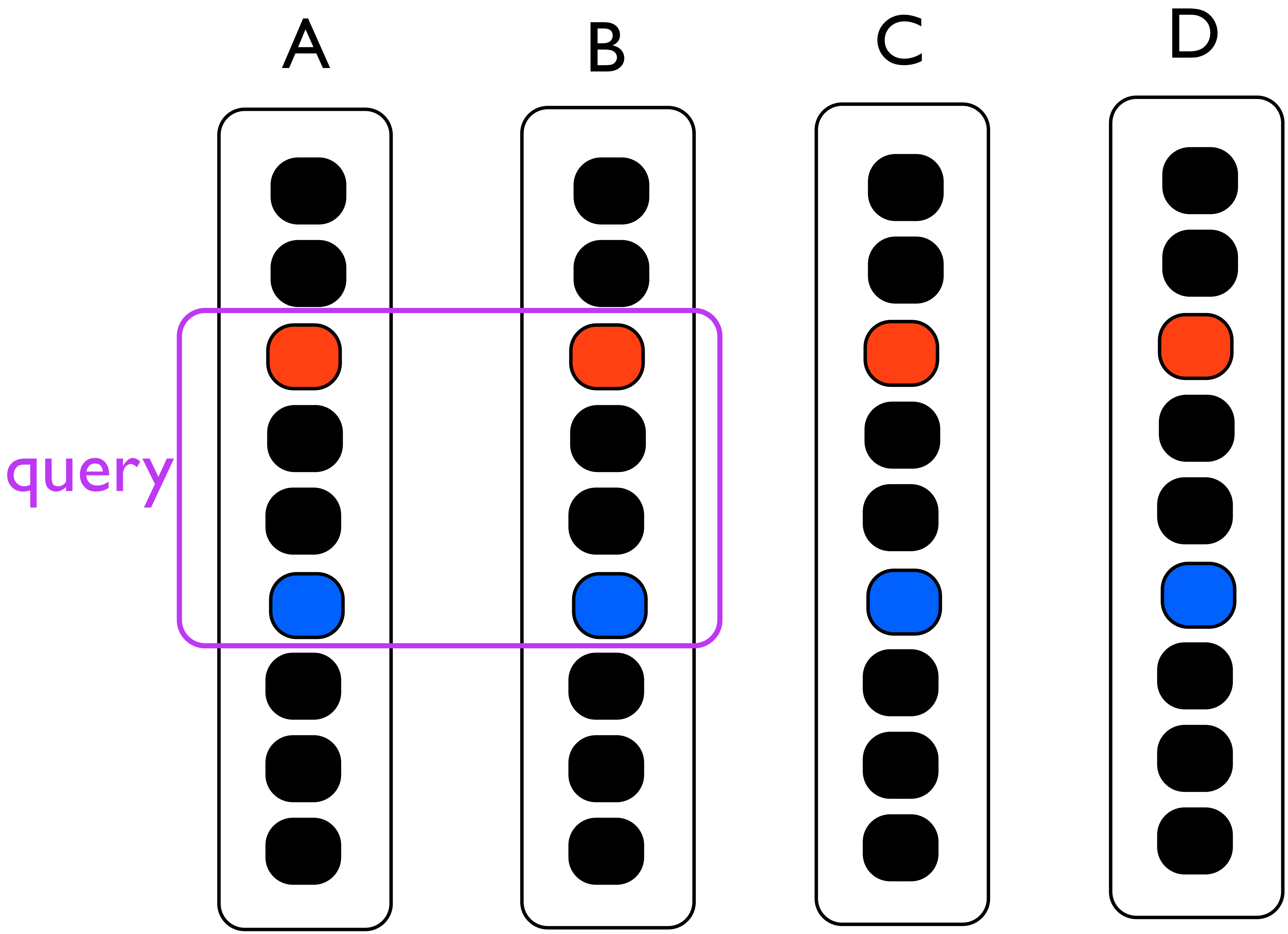
sideways cracking



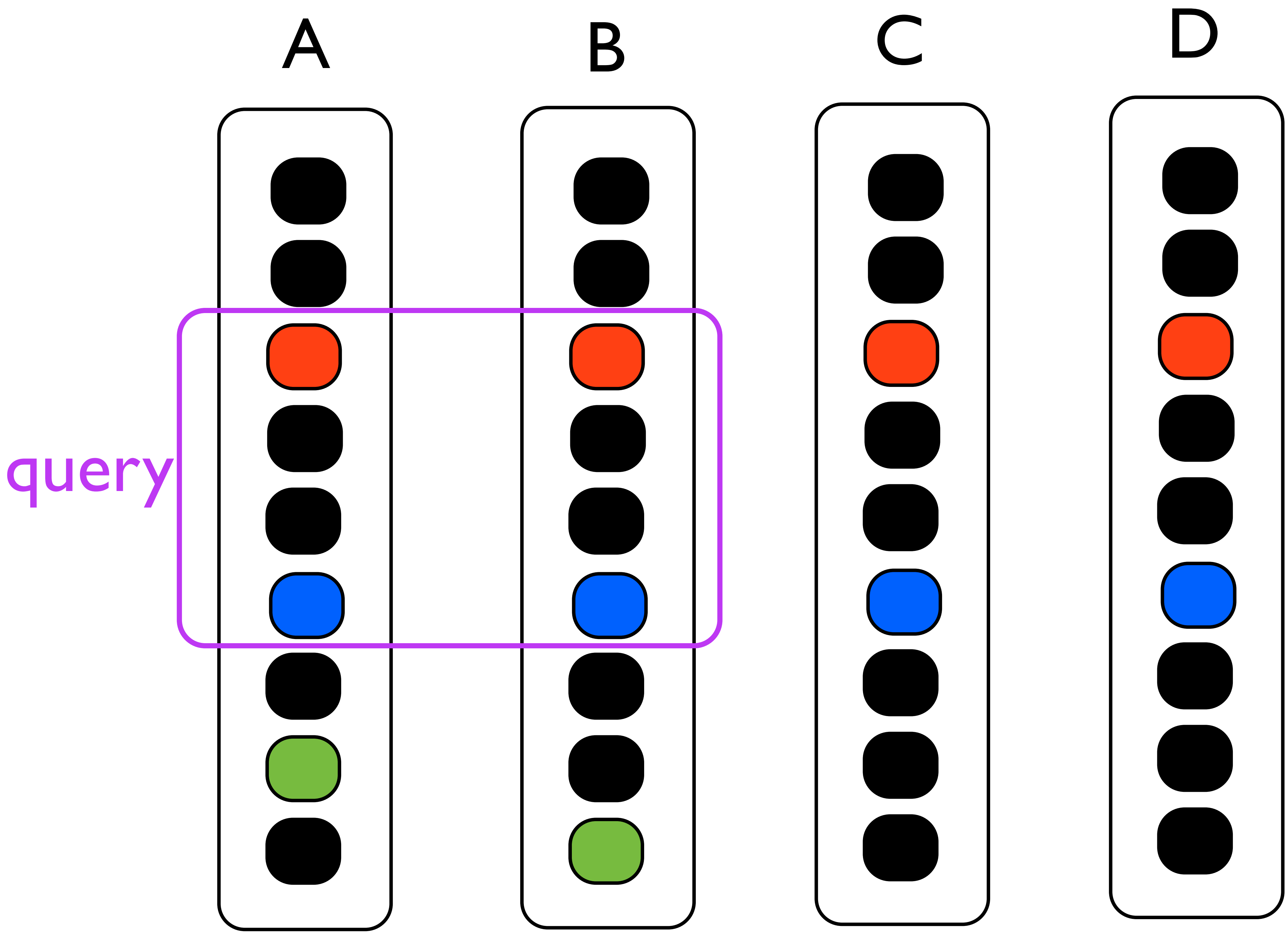
sideways cracking



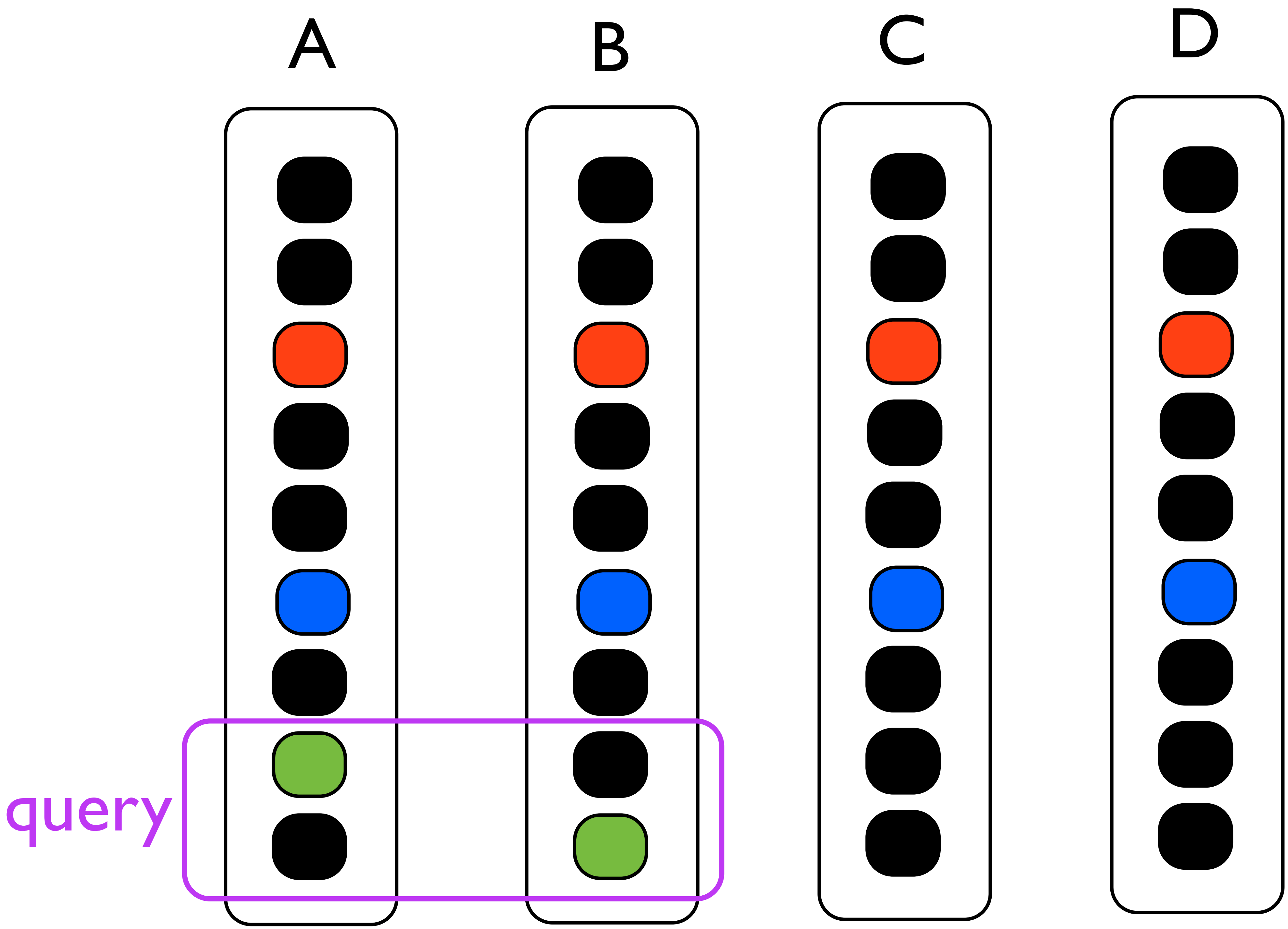
sideways cracking



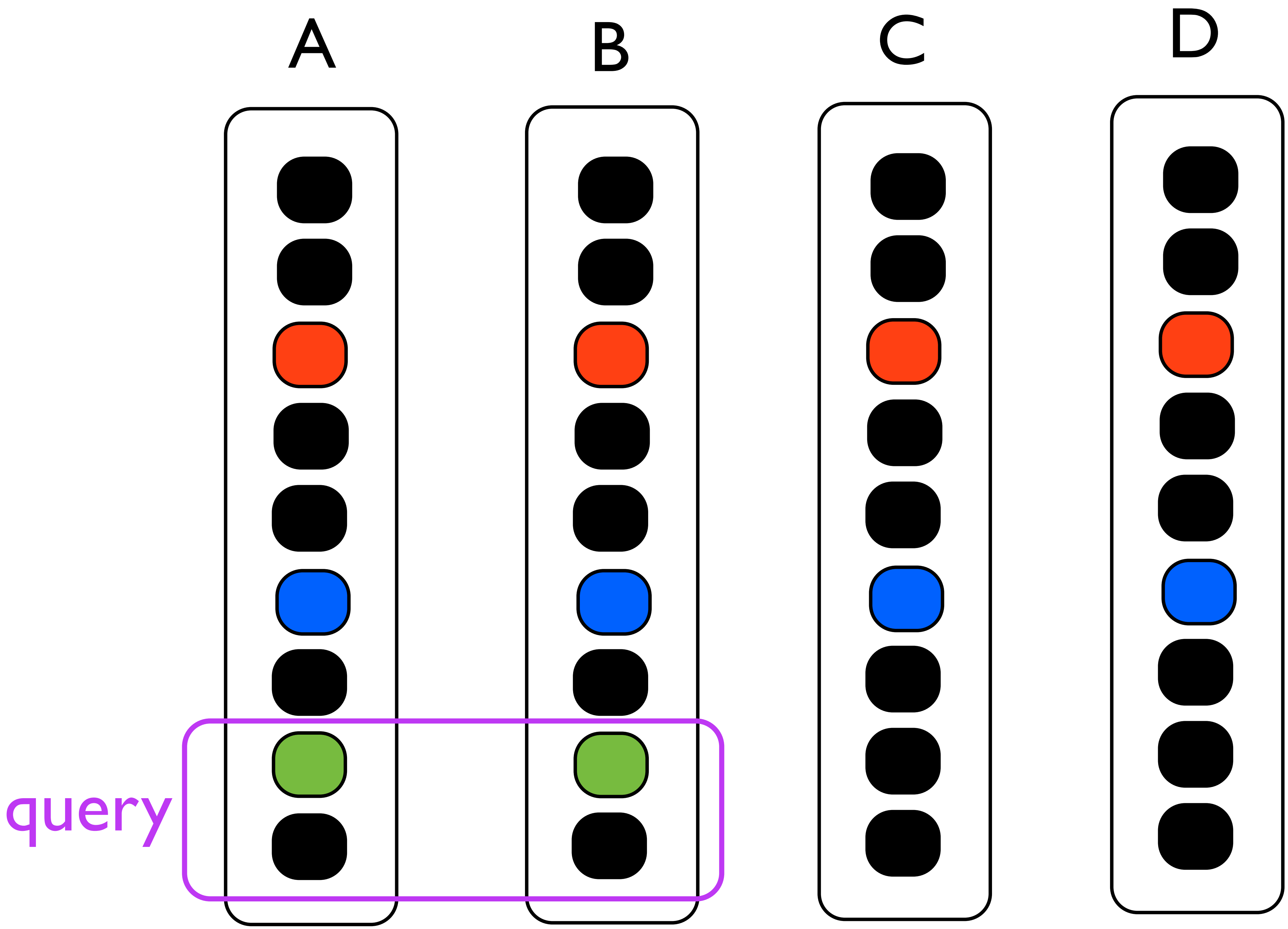
sideways cracking



sideways cracking

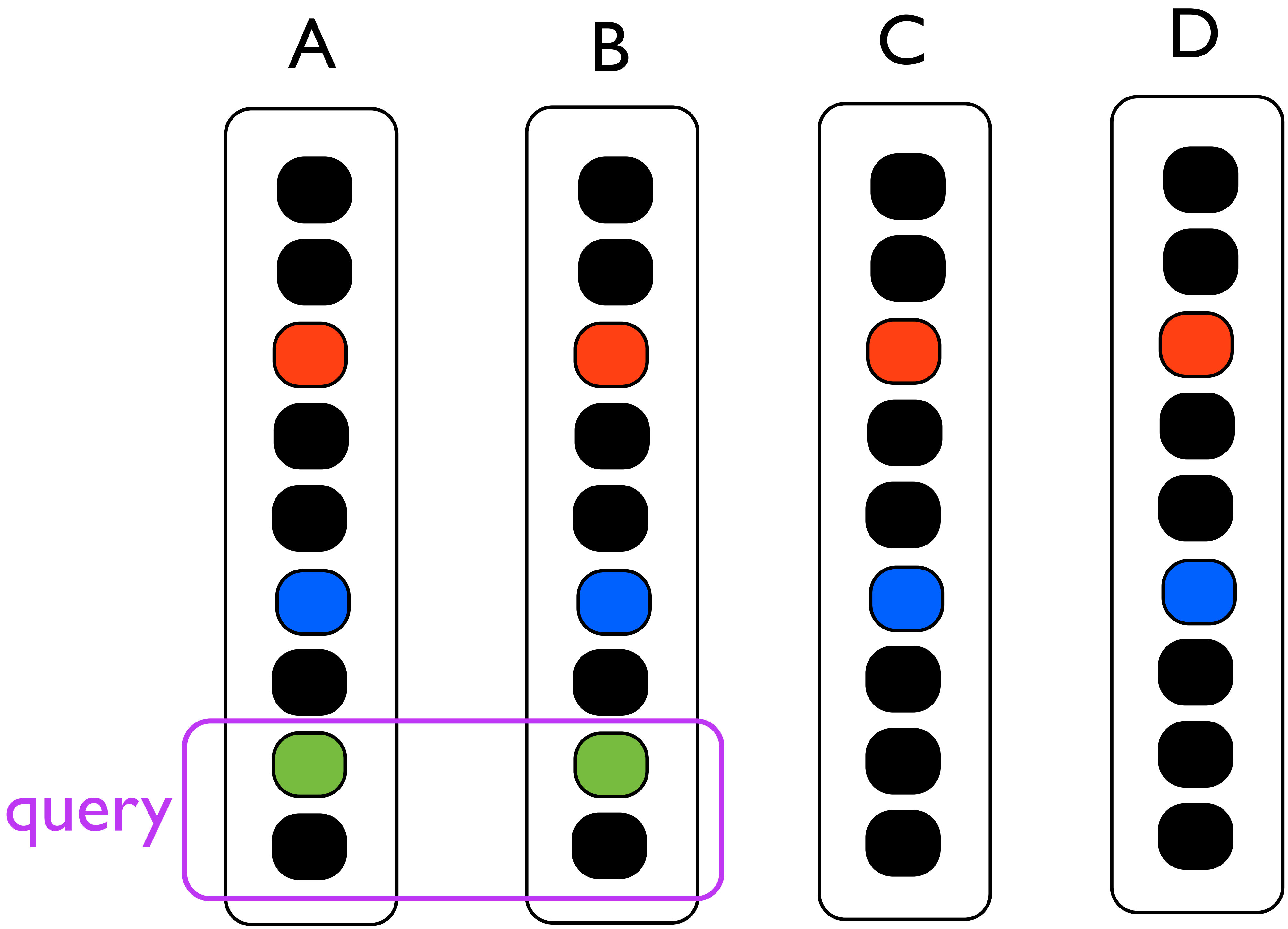


sideways cracking

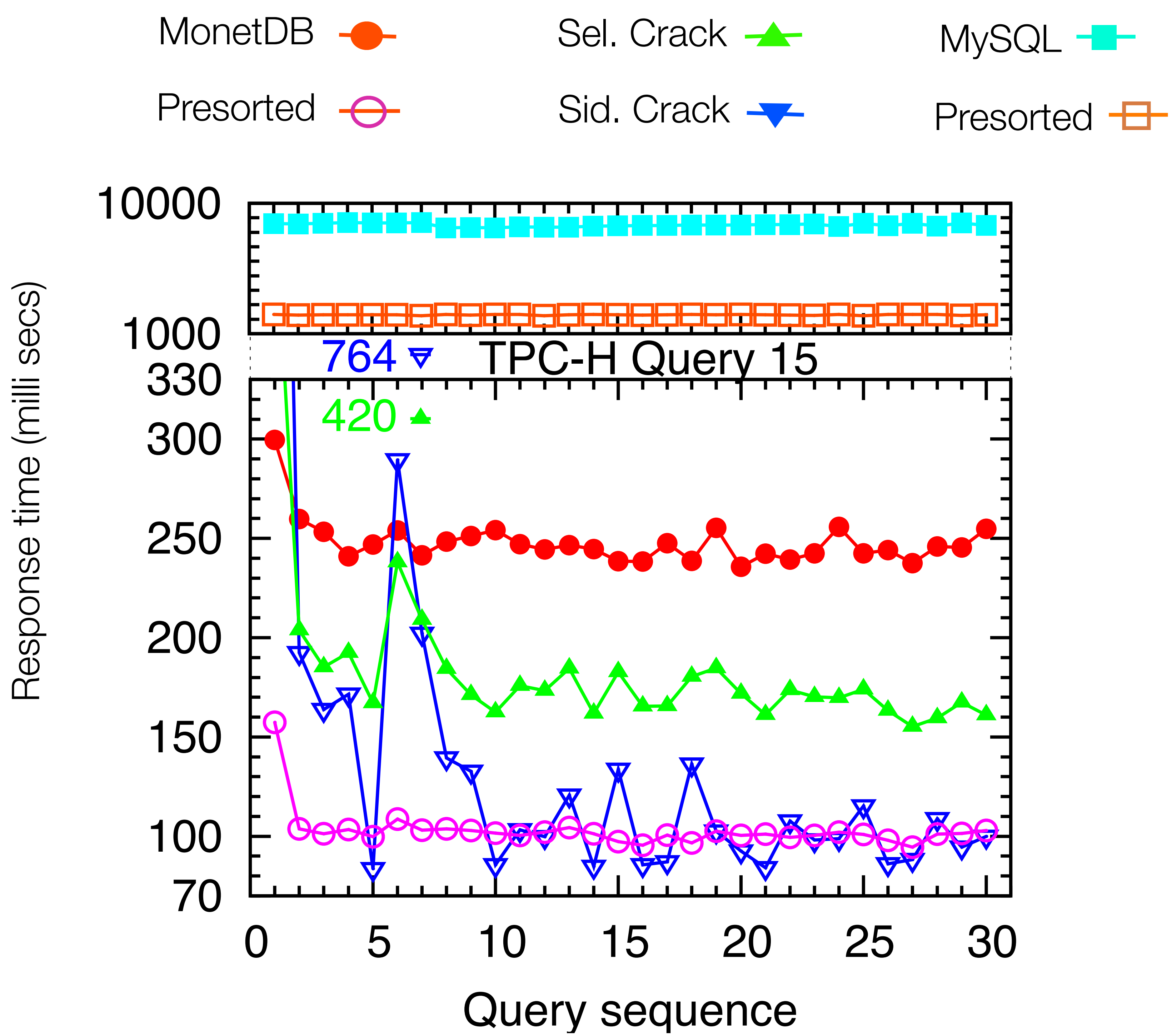


replace tuple reconstruction with cracking

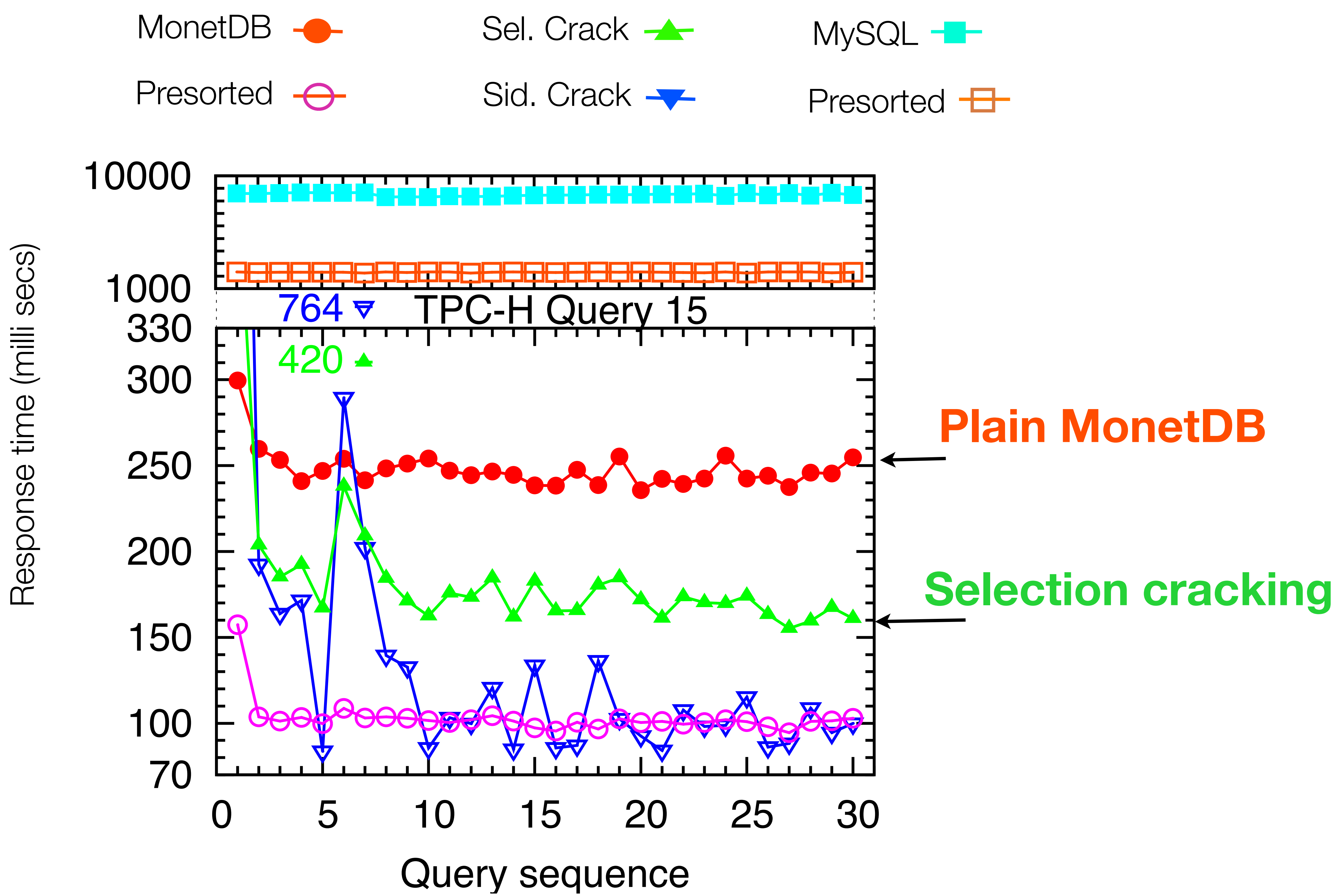
log crack actions and replay to align columns dynamically



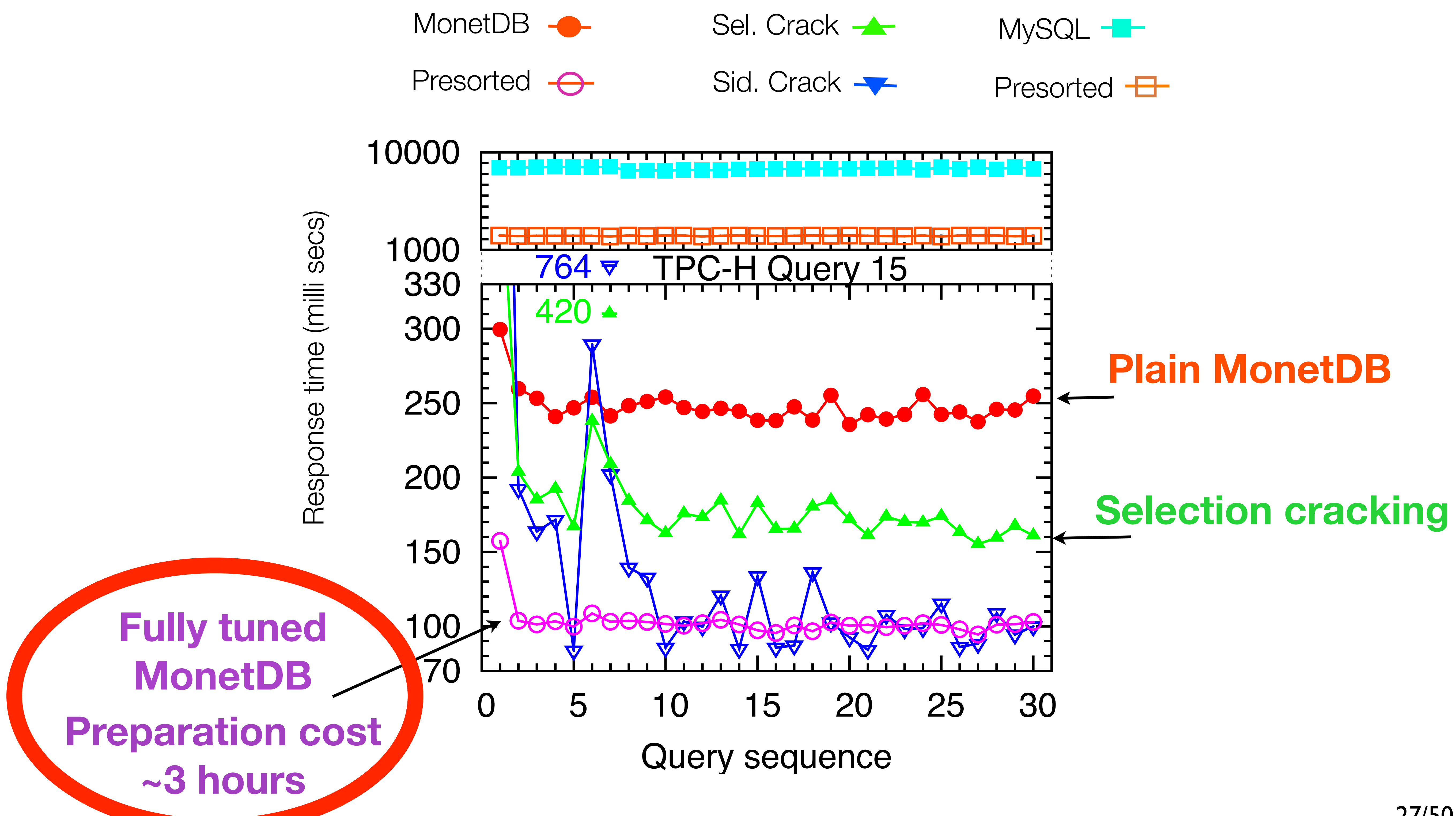
TPC-H


MonetDB
Sel. Crack
MySQL
Presorted
Sid. Crack
Presorted
TPC-H Query 15
764
420
Response time (milli secs)
10000
1000
330
300
250
200
150
100
70
0
5
10
15
20
25
30
Query sequence

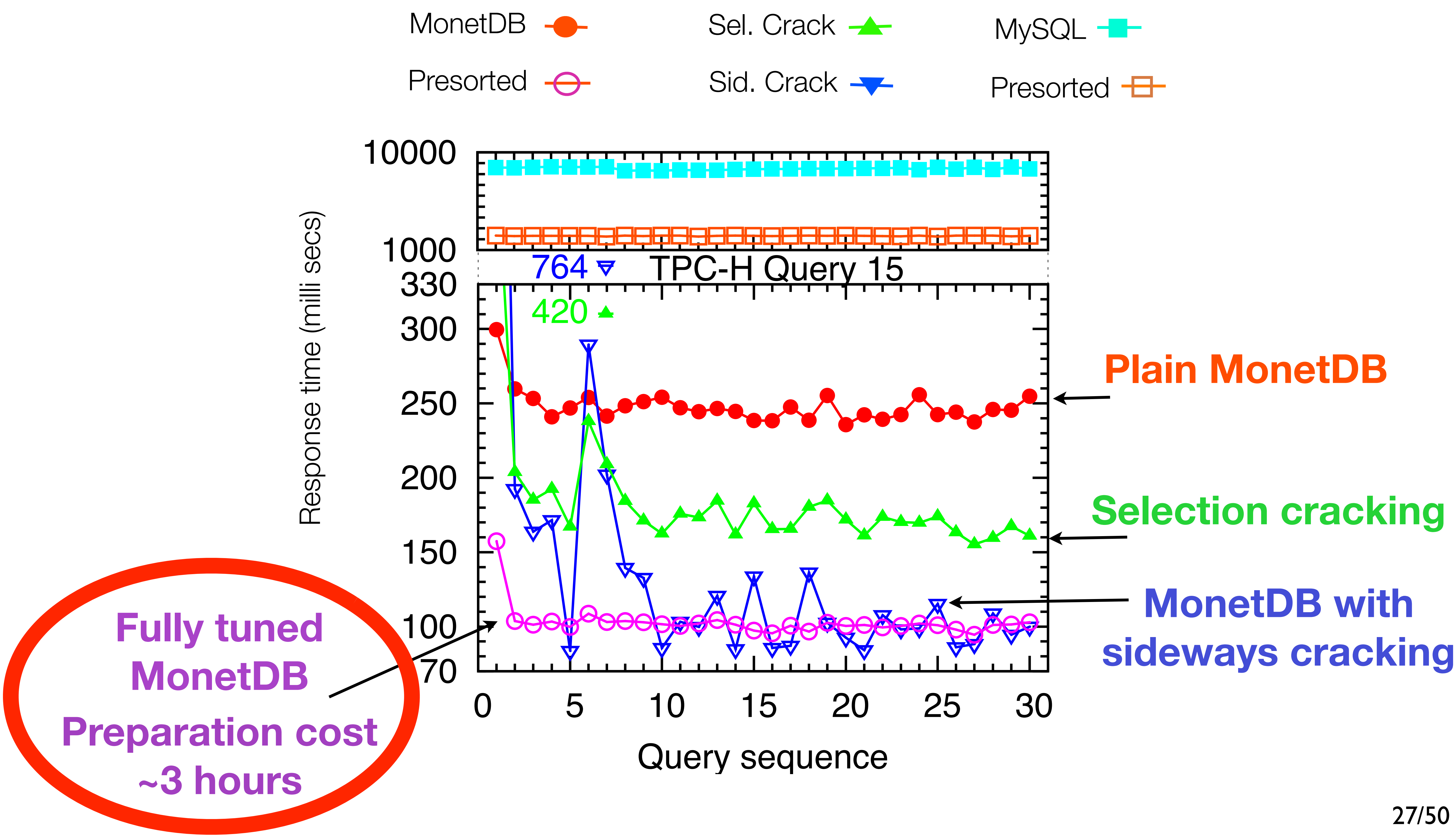
TPC-H


MonetDB
Sel. Crack
MySQL
Presorted
Sid. Crack
Presorted
764
TPC-H Query 15
420
Response time (milli secs)
Query sequence
10000
1000
330
300
250
200
150
100
70
0
5
10
15
20
25
30
Plain MonetDB
Selection cracking

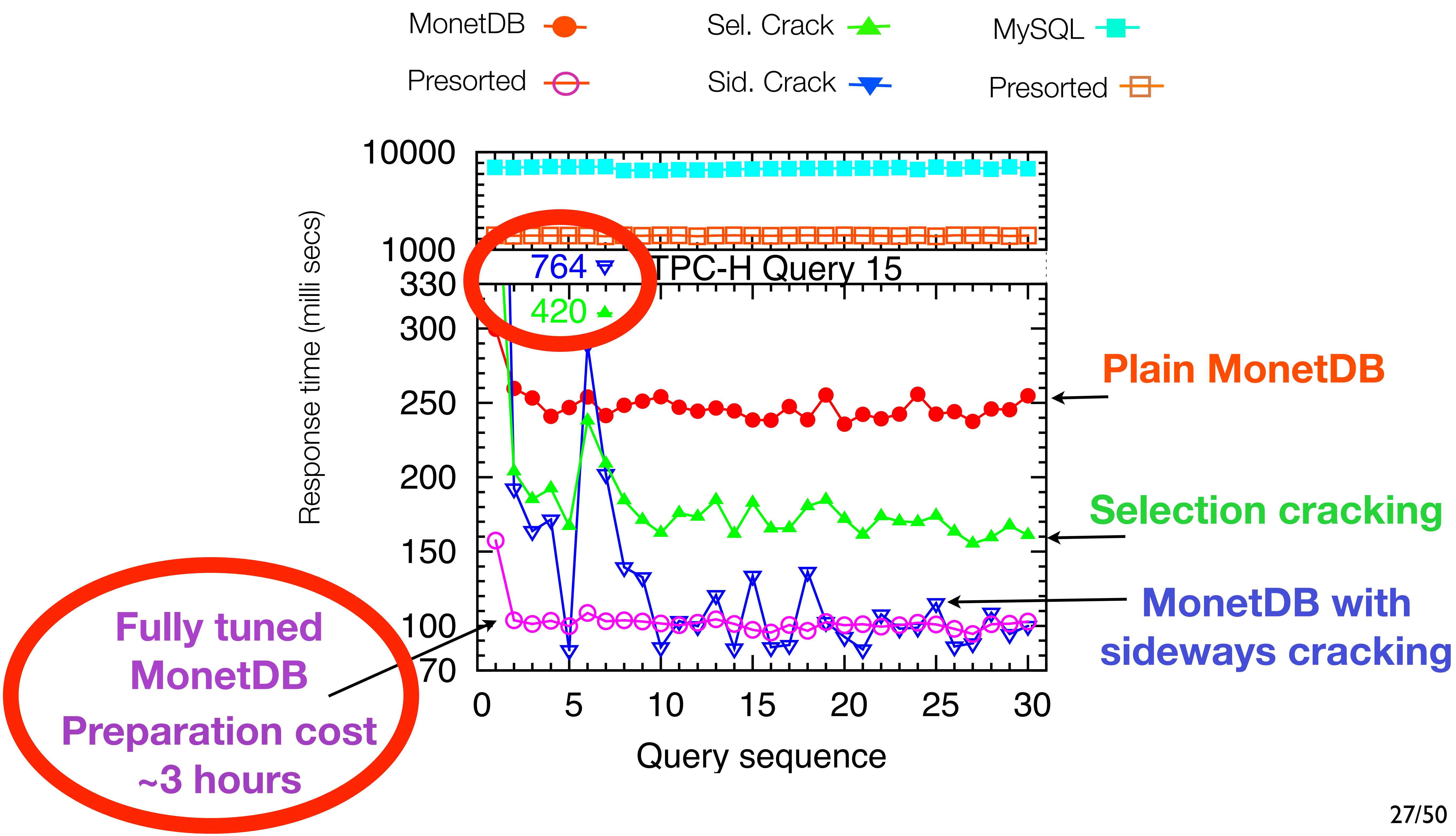
TPC-H


MonetDB
Presorted
Sel. Crack
Sid. Crack
MySQL
Presorted
764
420
TPC-H Query 15
Response time (milli secs)
Query sequence
10000
1000
330
300
250
200
150
100
70
0
5
10
15
20
25
30
Plain MonetDB
Selection cracking
Fully tuned MonetDB
Preparation cost ~3 hours

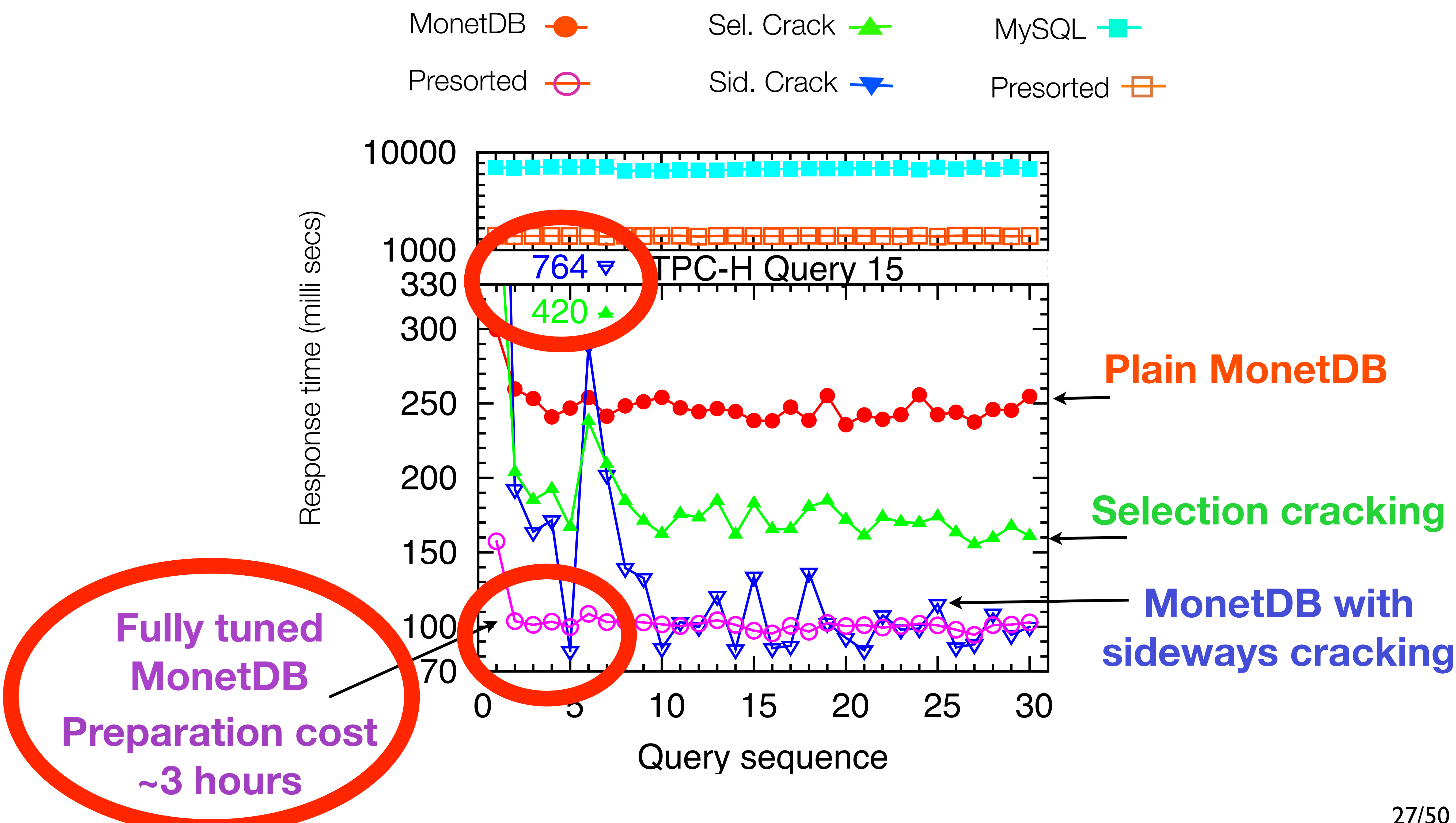
TPC-H


MonetDB
Presorted
Sel. Crack
Sid. Crack
MySQL
Presorted
764
TPC-H Query 15
420
Response time (milli secs)
10000
1000
330
300
250
200
150
100
70
0
5
10
15
20
25
30
Query sequence
Plain MonetDB
Selection cracking
MonetDB with sideways cracking
Fully tuned MonetDB
Preparation cost ~3 hours

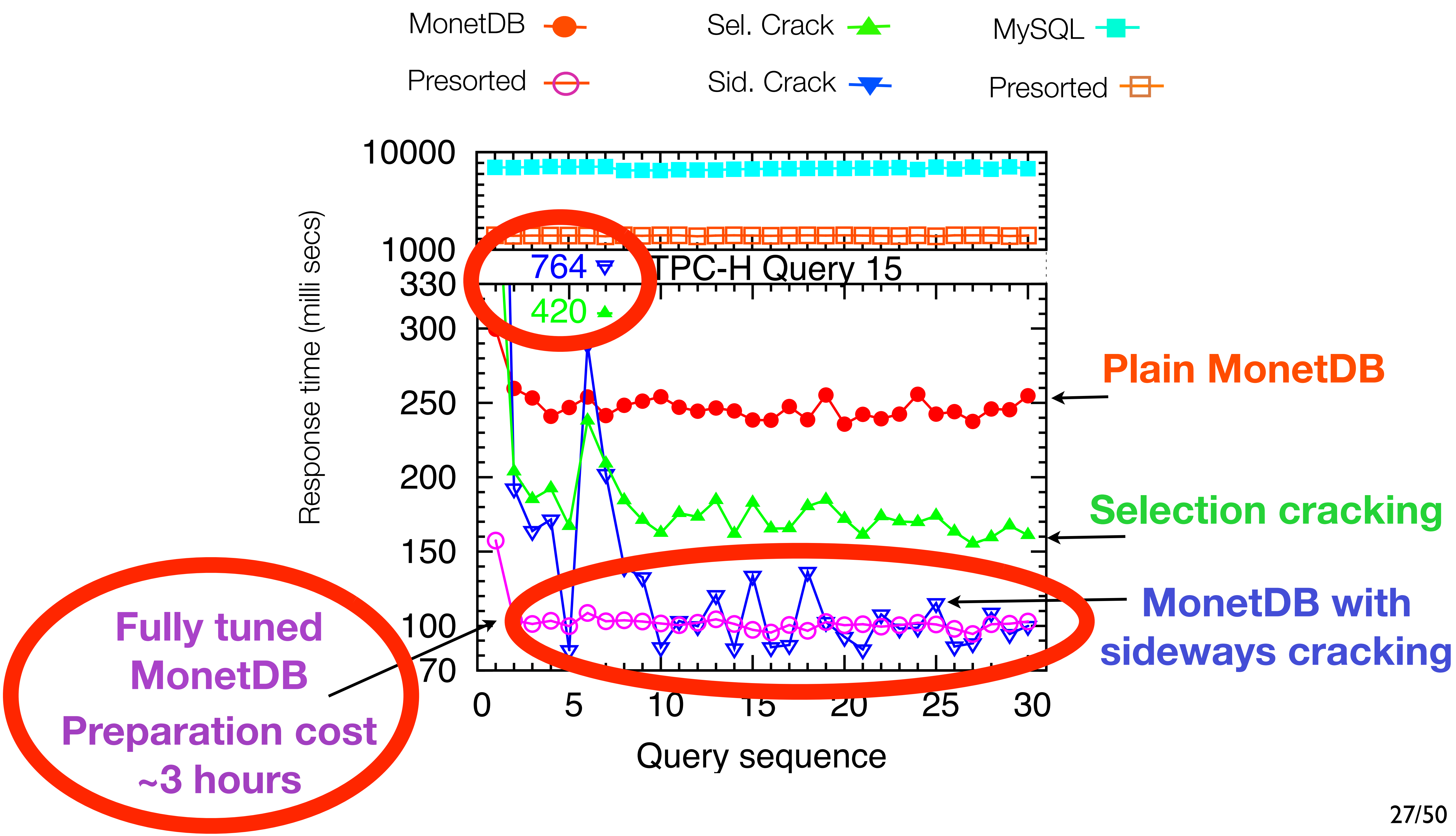
TPC-H


MonetDB
Sel. Crack
MySQL
Presorted
Sid. Crack
Presorted
TPC-H Query 15
764
420
Response time (milli secs)
10000
1000
330
300
250
200
150
100
70
0
5
10
15
20
25
30
Query sequence
Plain MonetDB
Selection cracking
MonetDB with sideways cracking
Fully tuned MonetDB
Preparation cost ~3 hours

TPC-H

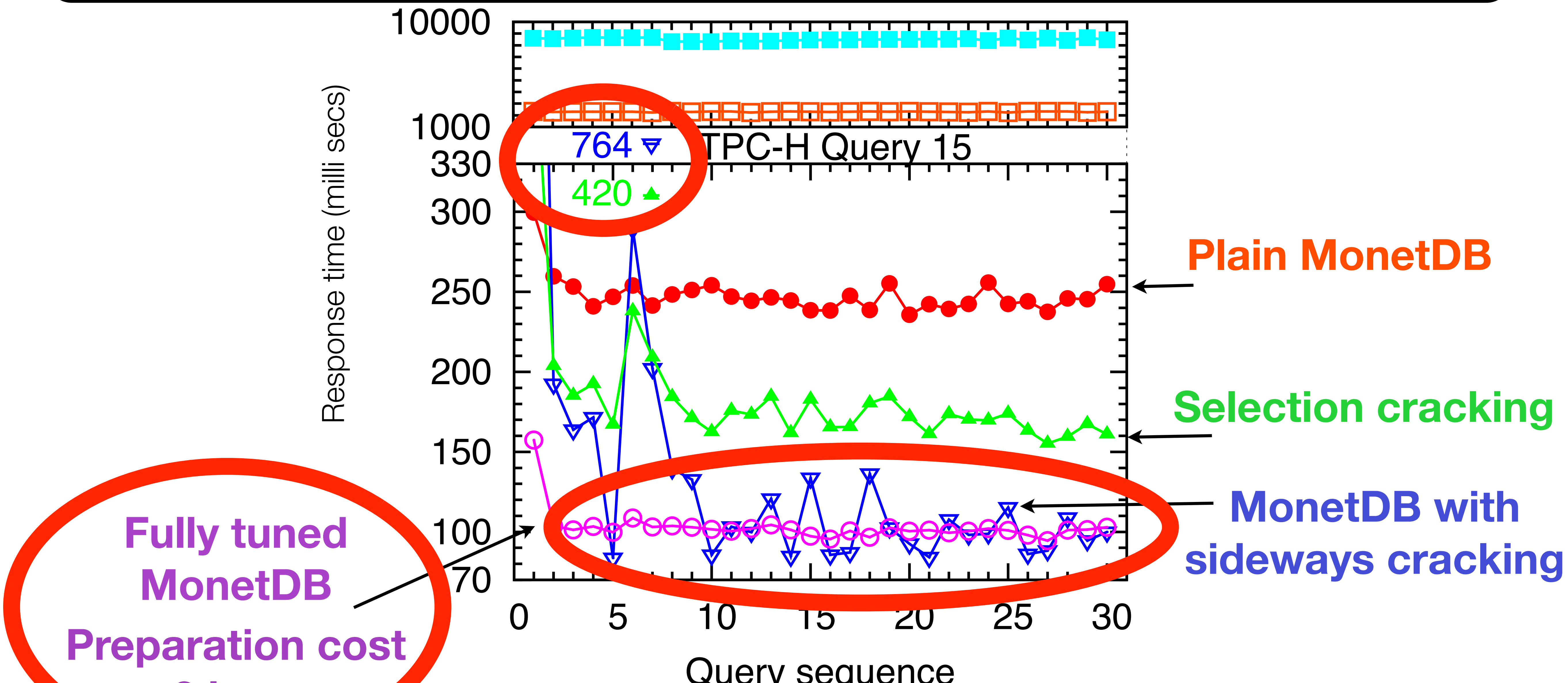

MonetDB
Sel. Crack
MySQL
Presorted
Sid. Crack
Presorted
Response time (milli secs)
10000
1000
330
300
250
200
150
100
70
764
420
TPC-H Query 15
0
5
10
15
20
25
30
Query sequence
Plain MonetDB
Selection cracking
MonetDB with sideways cracking
Fully tuned MonetDB
Preparation cost ~3 hours

TPC-H


MonetDB
Sel. Crack
MySQL
Presorted
Sid. Crack
Presorted
764
420
TPC-H Query 15
Response time (milli secs)
Query sequence
Plain MonetDB
Selection cracking
MonetDB with sideways cracking
Fully tuned MonetDB
Preparation cost ~3 hours

TPC-H

reducing data-to-query time



cracking on Skyserver (4TB)

(Sloan Digital Sky Survey, www.sdss.org)

cracking answers 160.000 queries while full indexing is still half way creating one index

reducing data-to-query time

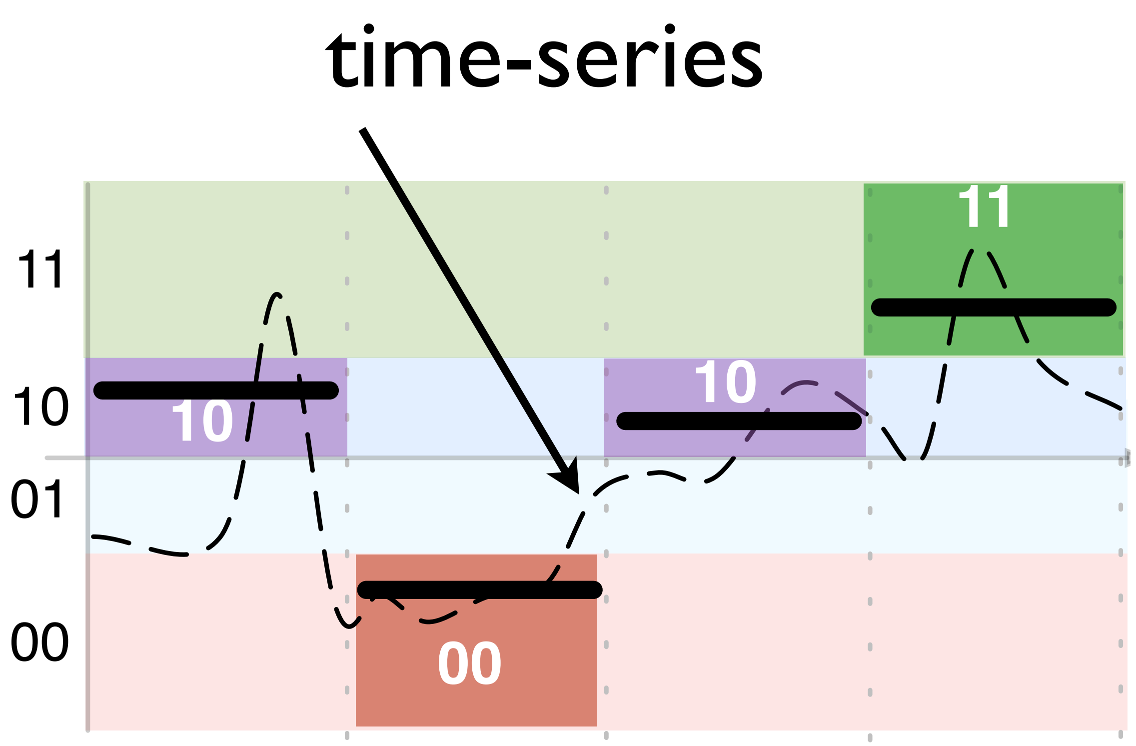
cracking on Skyserver (4TB)

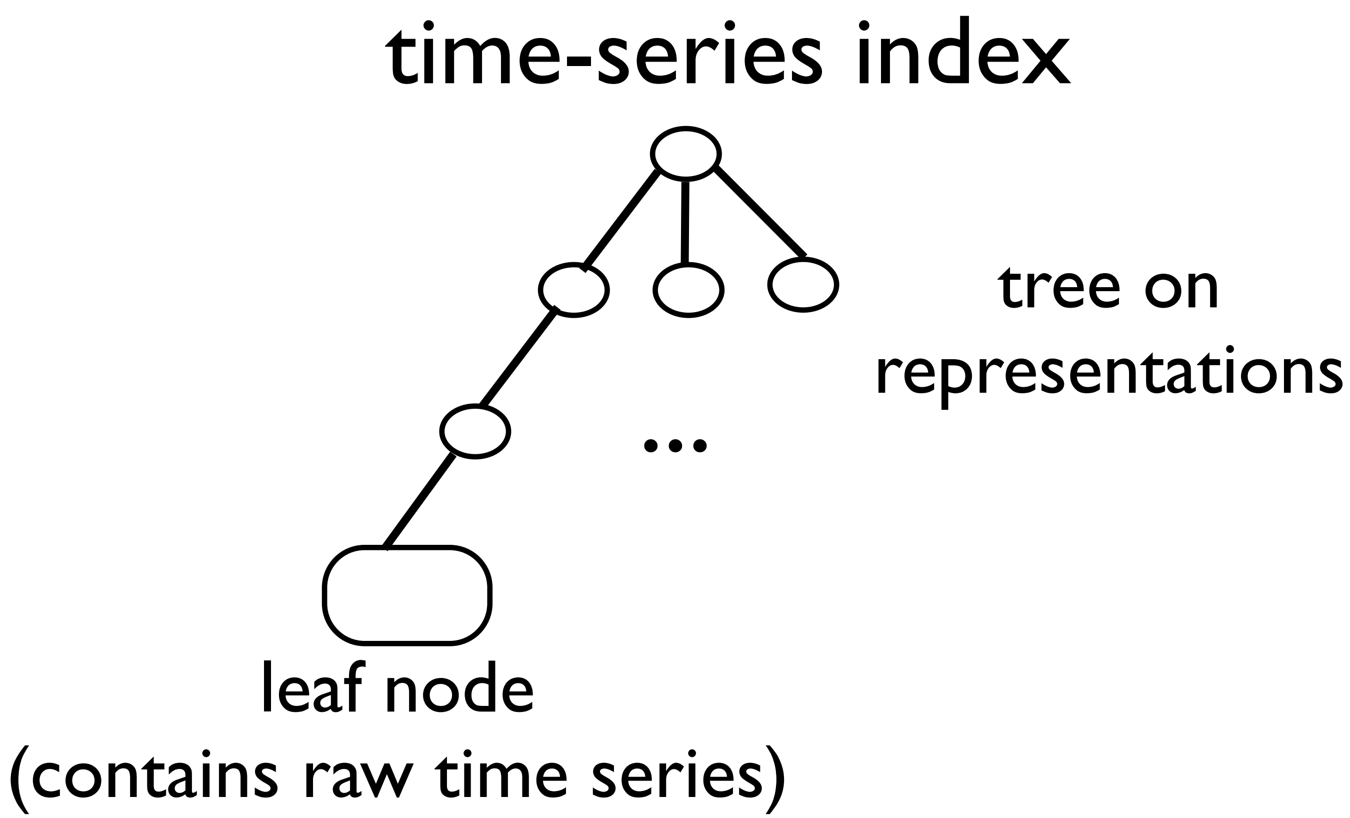
(Sloan Digital Sky Survey, www.sdss.org)

cracking answers 160.000 queries while full indexing is still half way creating one index

time-series indexing

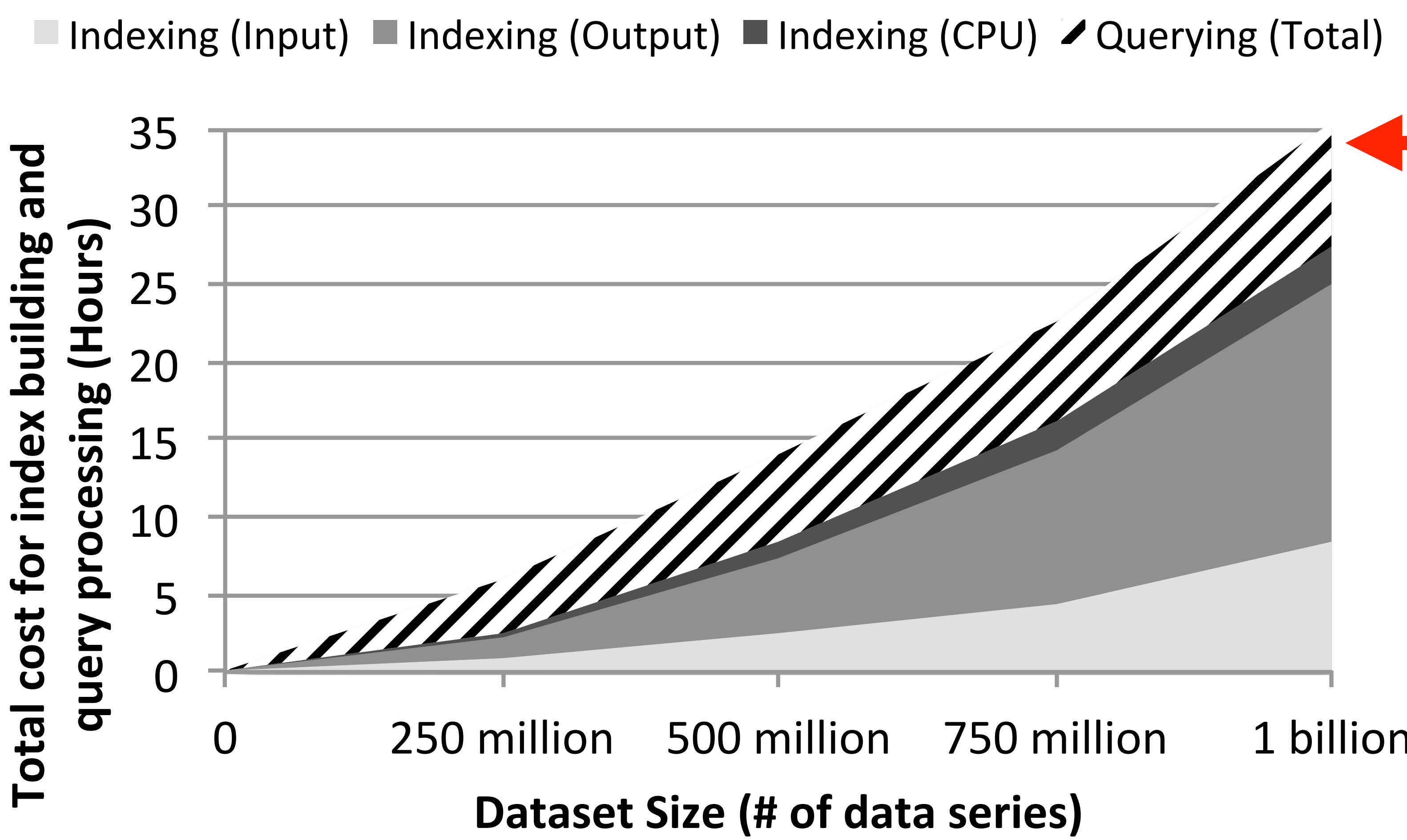
time-series={a1,a2,...,an}
typical query: find a time-series which is similar to time series x





state of the art in time series

does not scale - non interactive

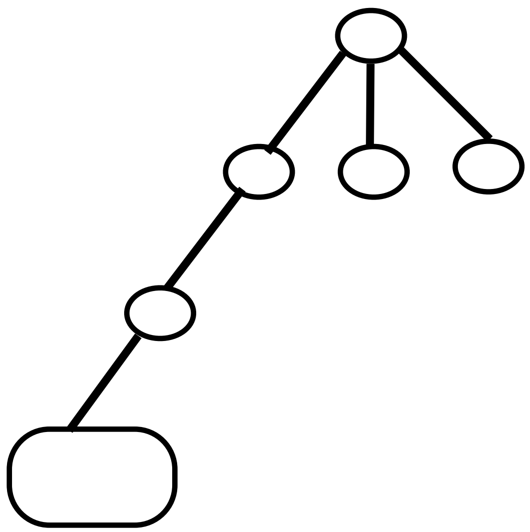


35 hours to index and query (10K queries)

ADS: Adaptive Data Series indexing

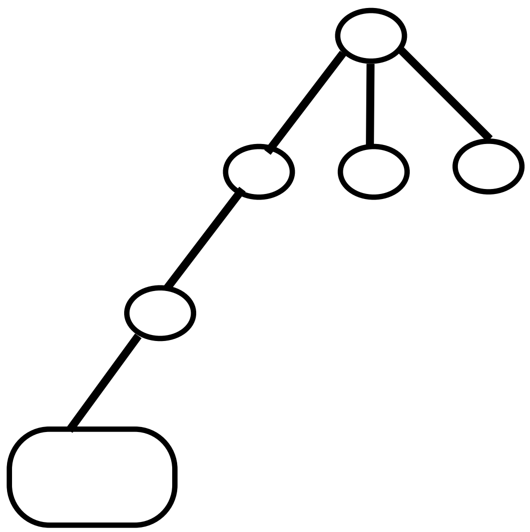
built the index tree as queries arrive

ADS: Adaptive Data Series indexing

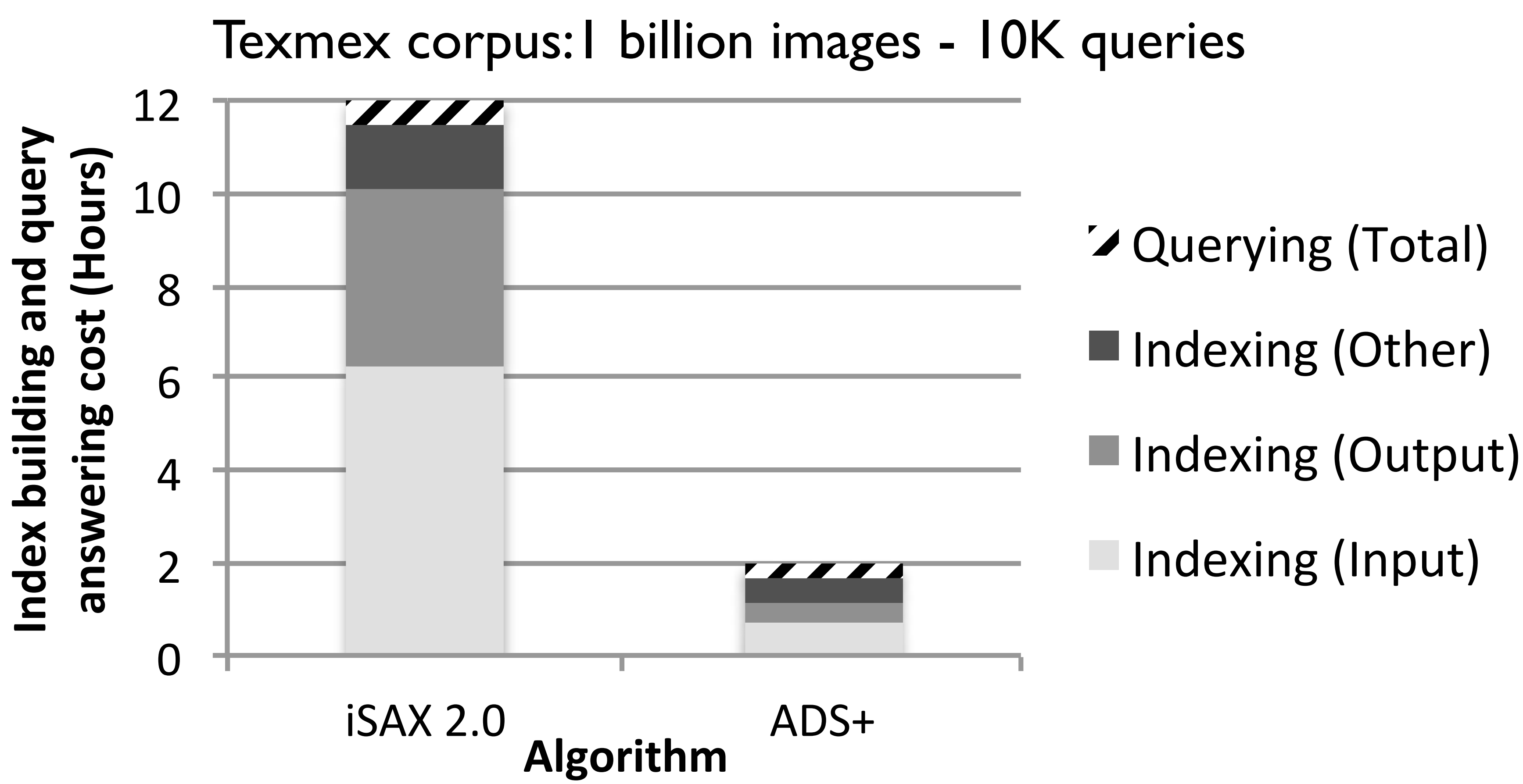


built the index tree as queries arrive

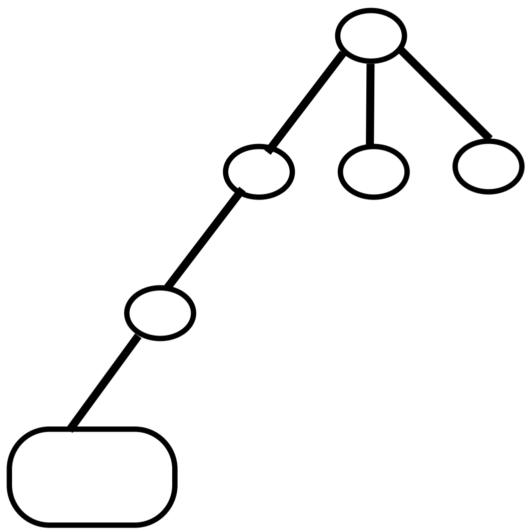
ADS: Adaptive Data Series indexing



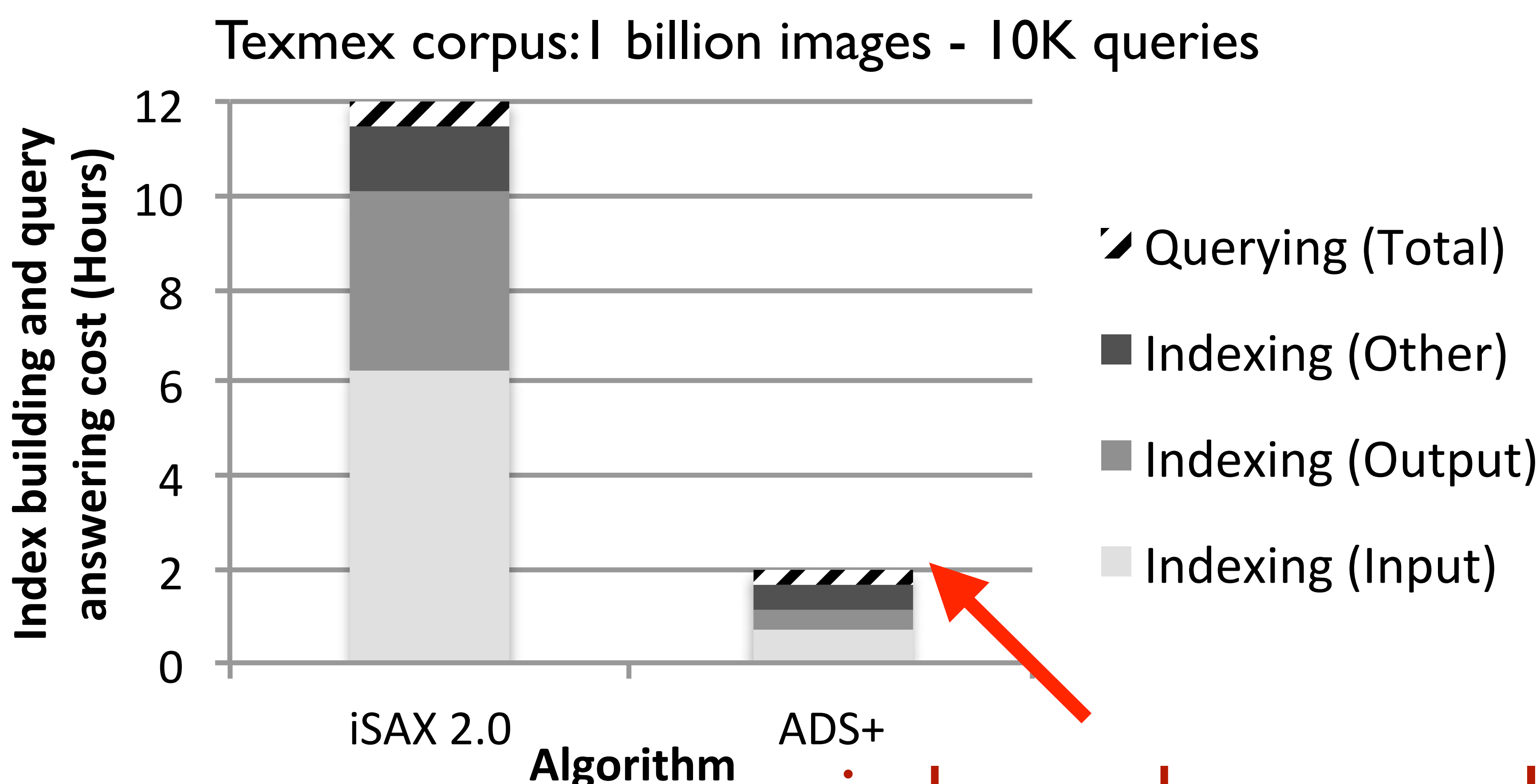
built the index tree as queries arrive



ADS: Adaptive Data Series indexing

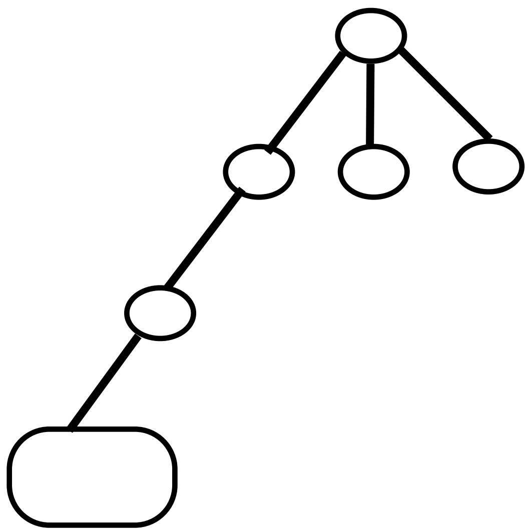


built the index tree as queries arrive

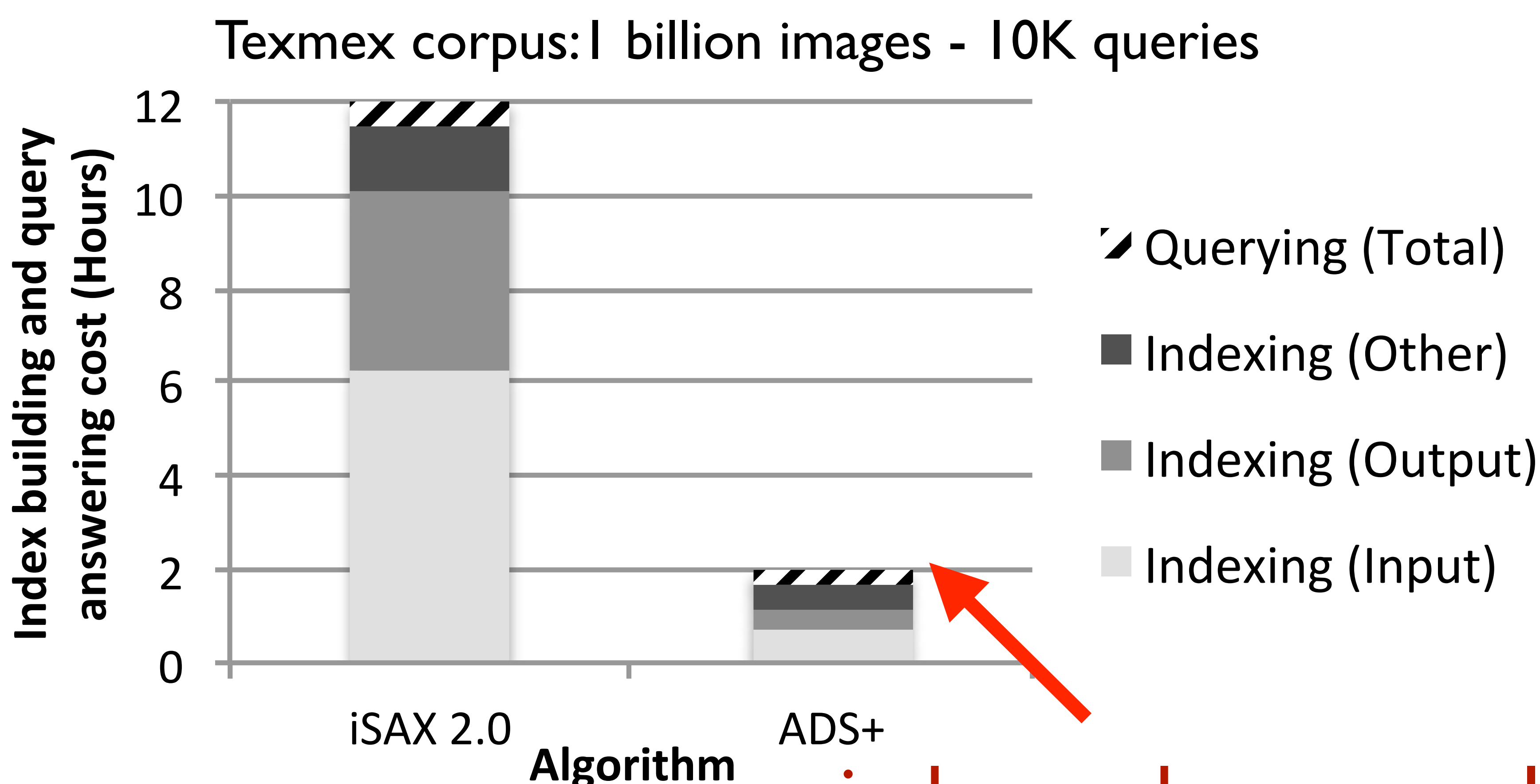


index and process all queries in 2 hours instead of 12

reducing data-to-query time



built the index tree as queries arrive



index and process all queries in 2 hours instead of 12

loading

loading

copy data inside the database

database now has full control

loading

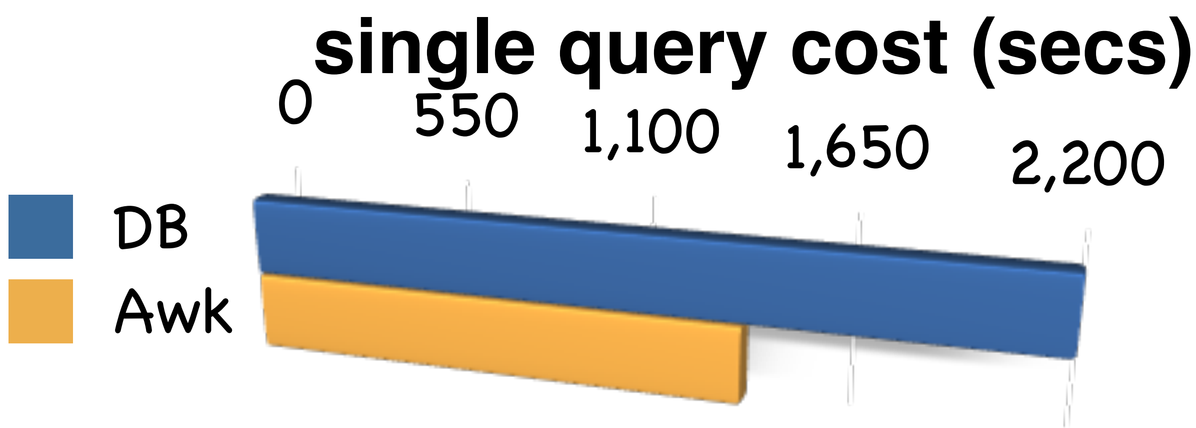
copy data inside the database

database now has full control

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

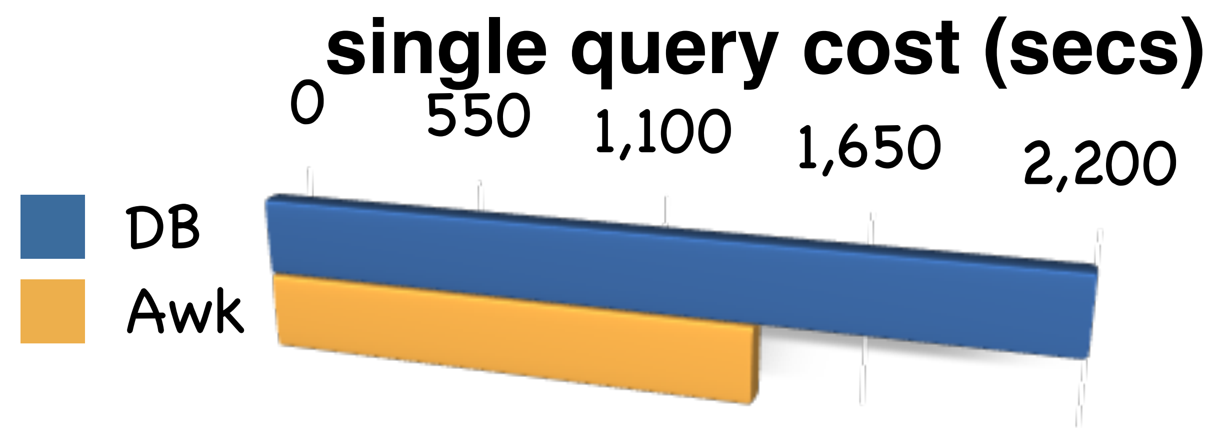
database vs. unix tools

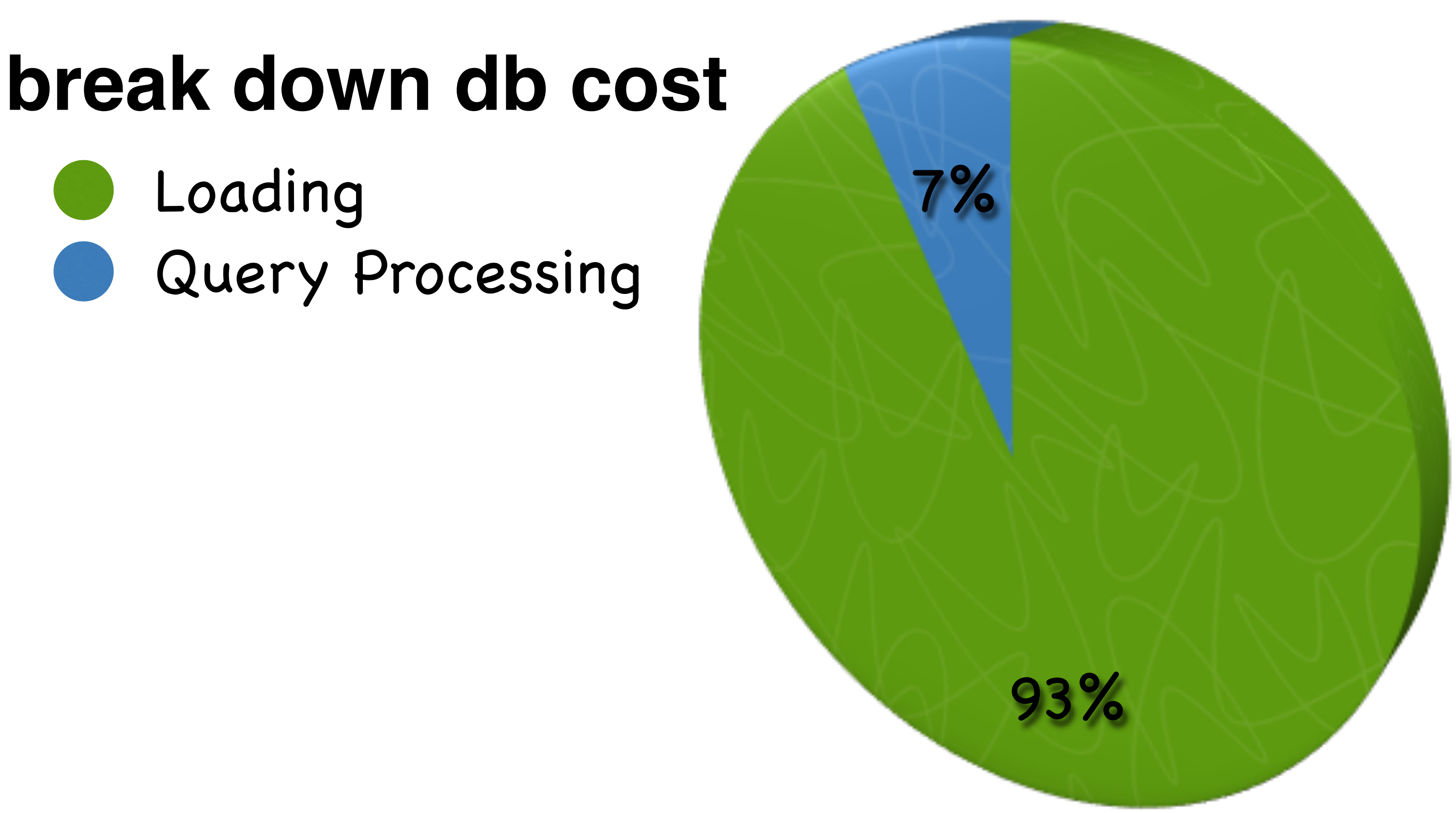
1 file, 4 attributes,
1 billion tuples



database vs. unix tools

1 file, 4 attributes,
1 billion tuples

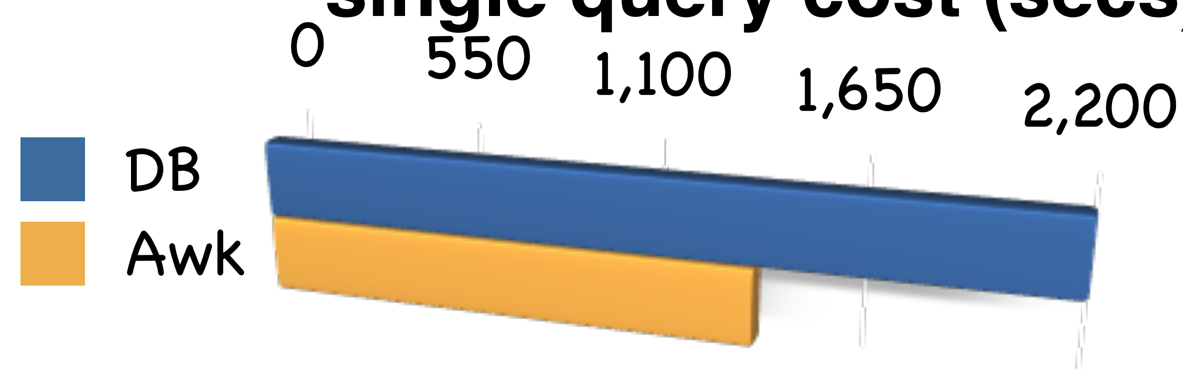




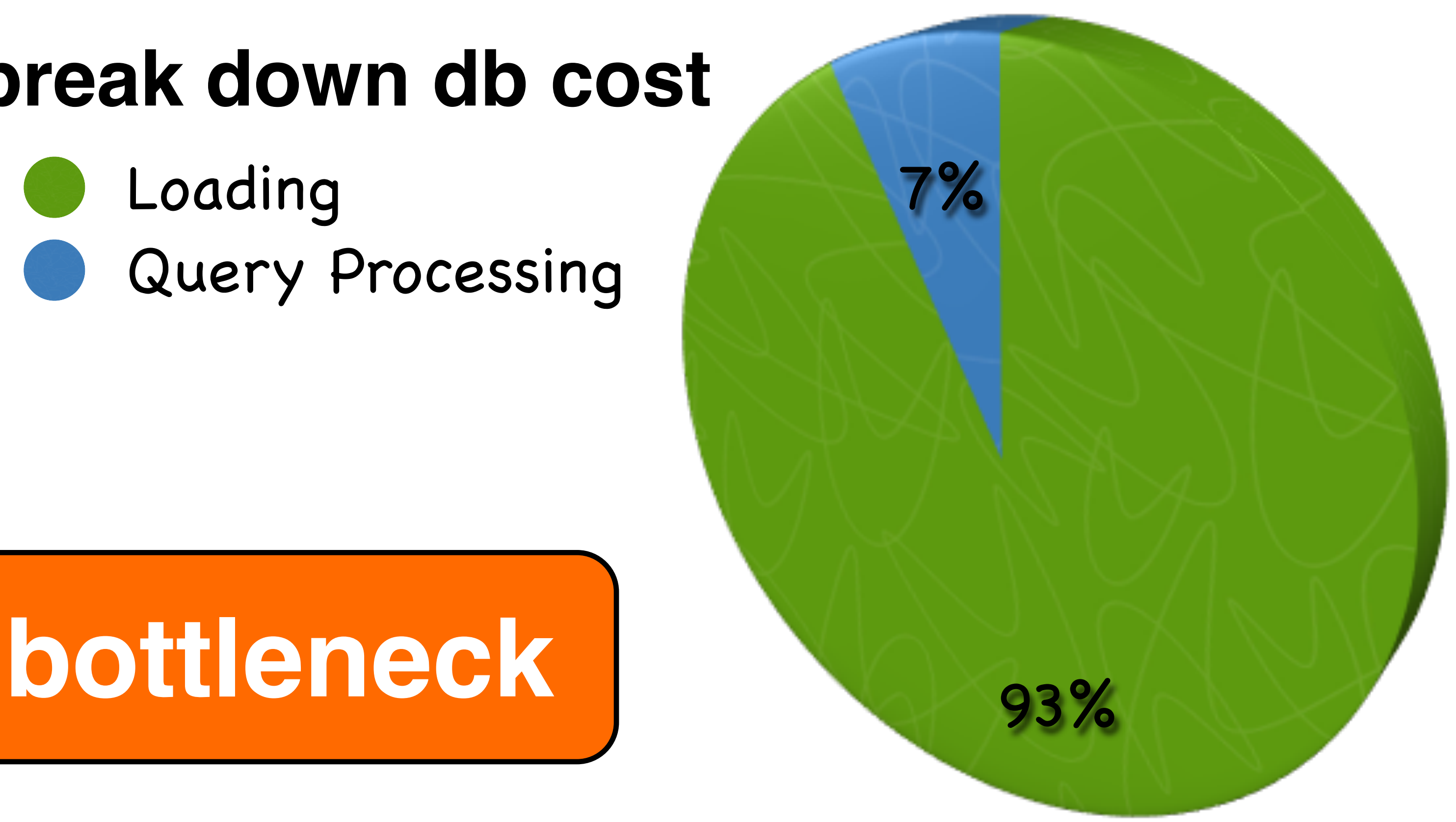
database vs. unix tools

1 file, 4 attributes,
1 billion tuples

single query cost (secs)



break down db cost

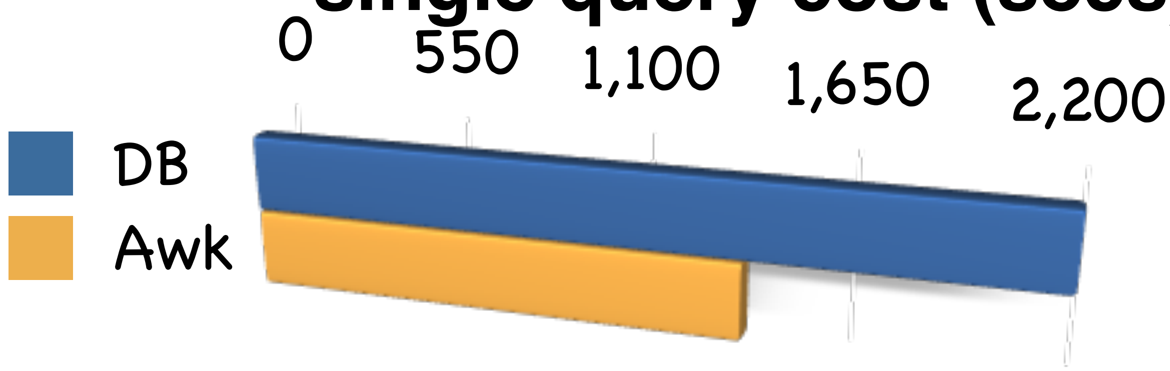


loading is a major bottleneck

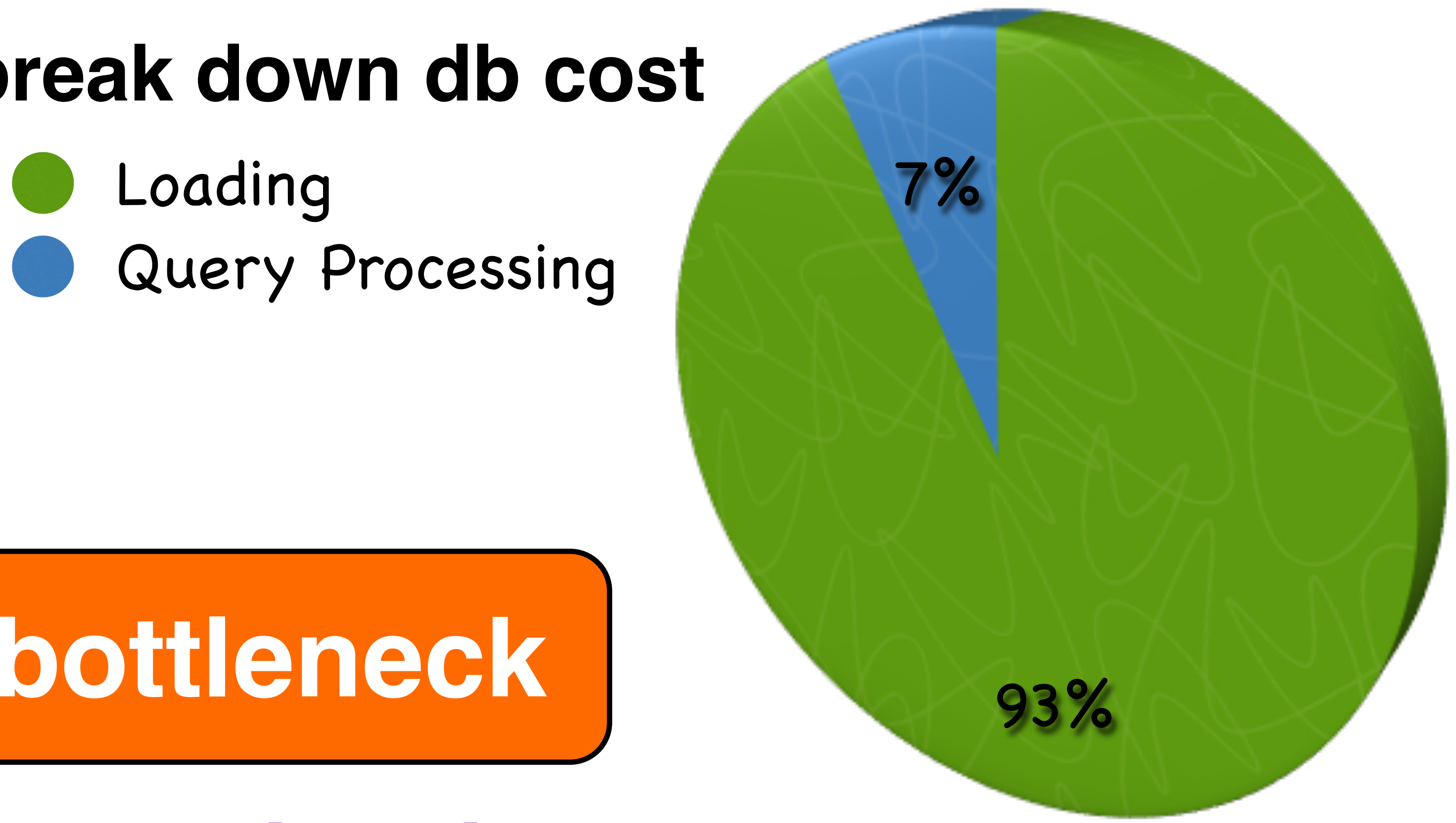
database vs. unix tools

1 file, 4 attributes,
1 billion tuples

single query cost (secs)



break down db cost



loading is a major bottleneck

... but writing/maintaining scripts is hard too

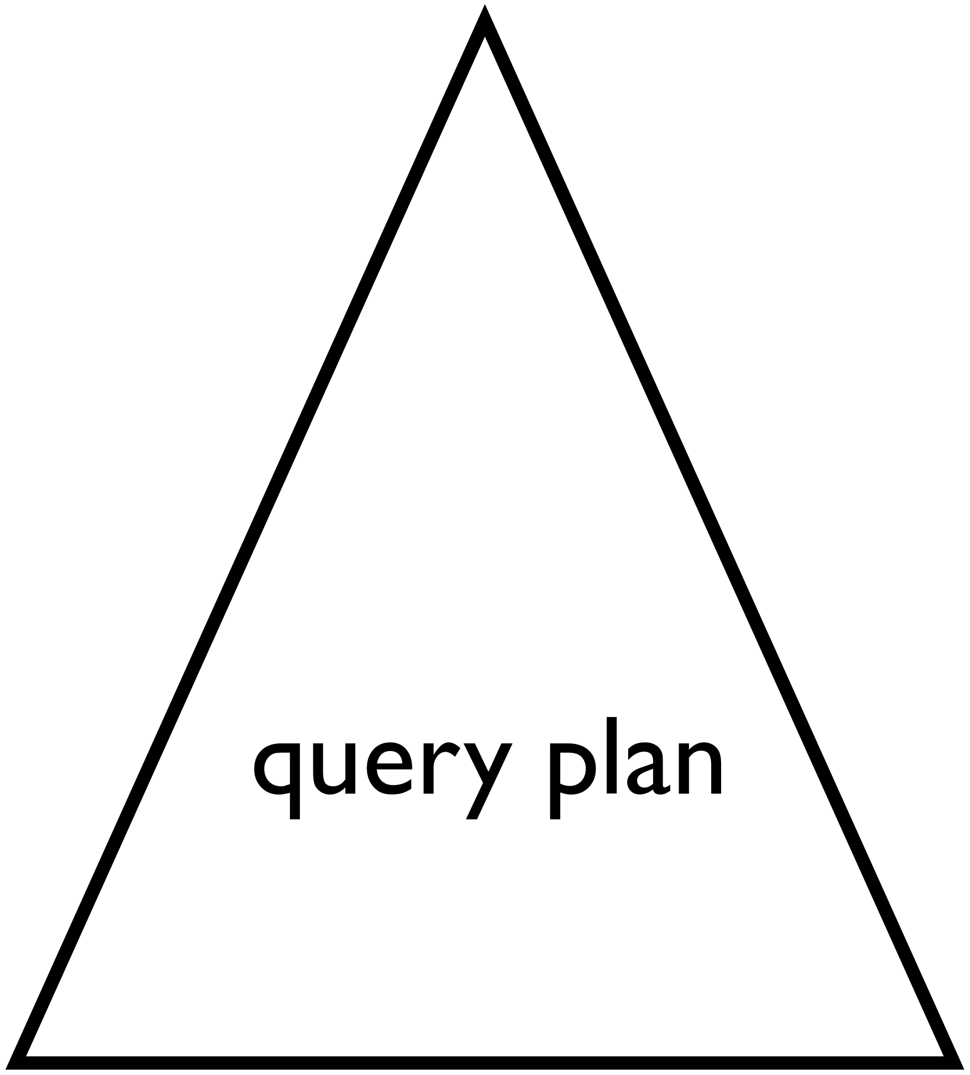
adaptive loading

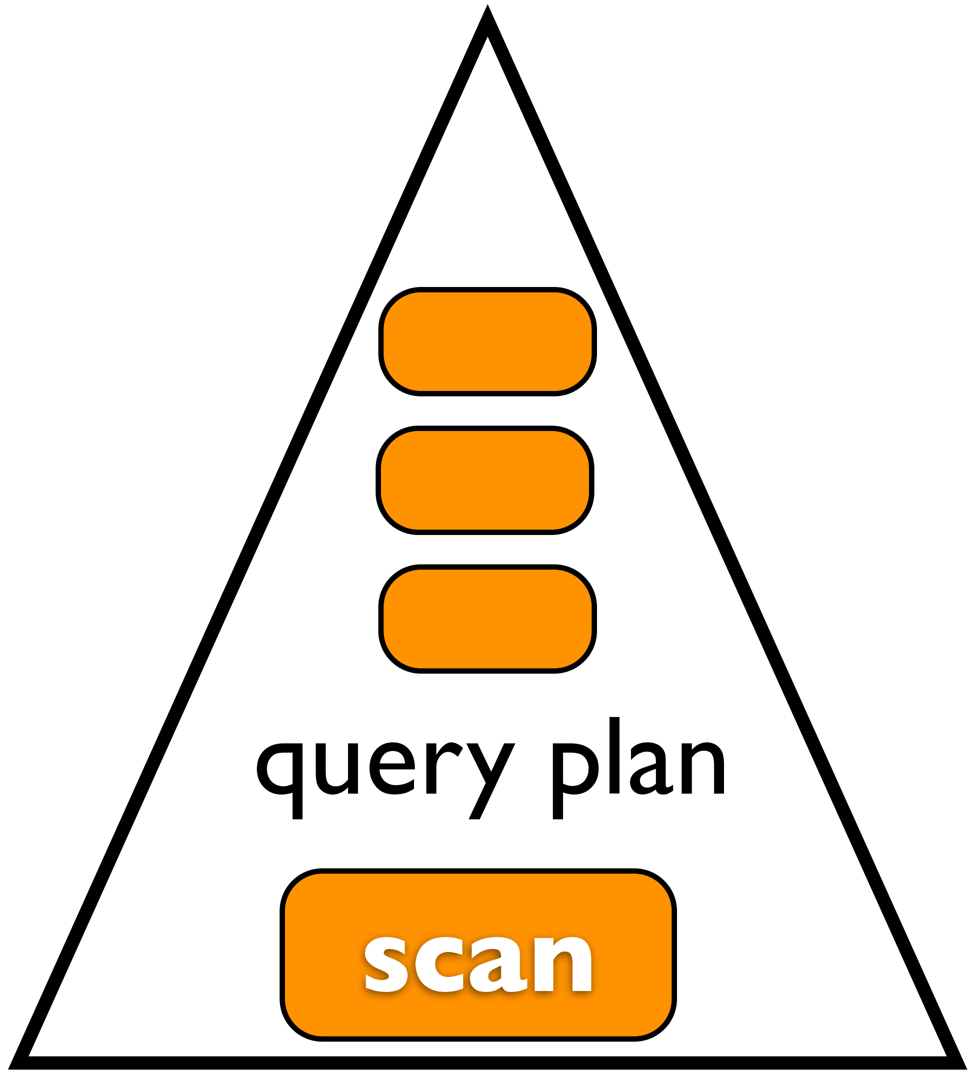
load/touch only what is needed
and only when it is needed

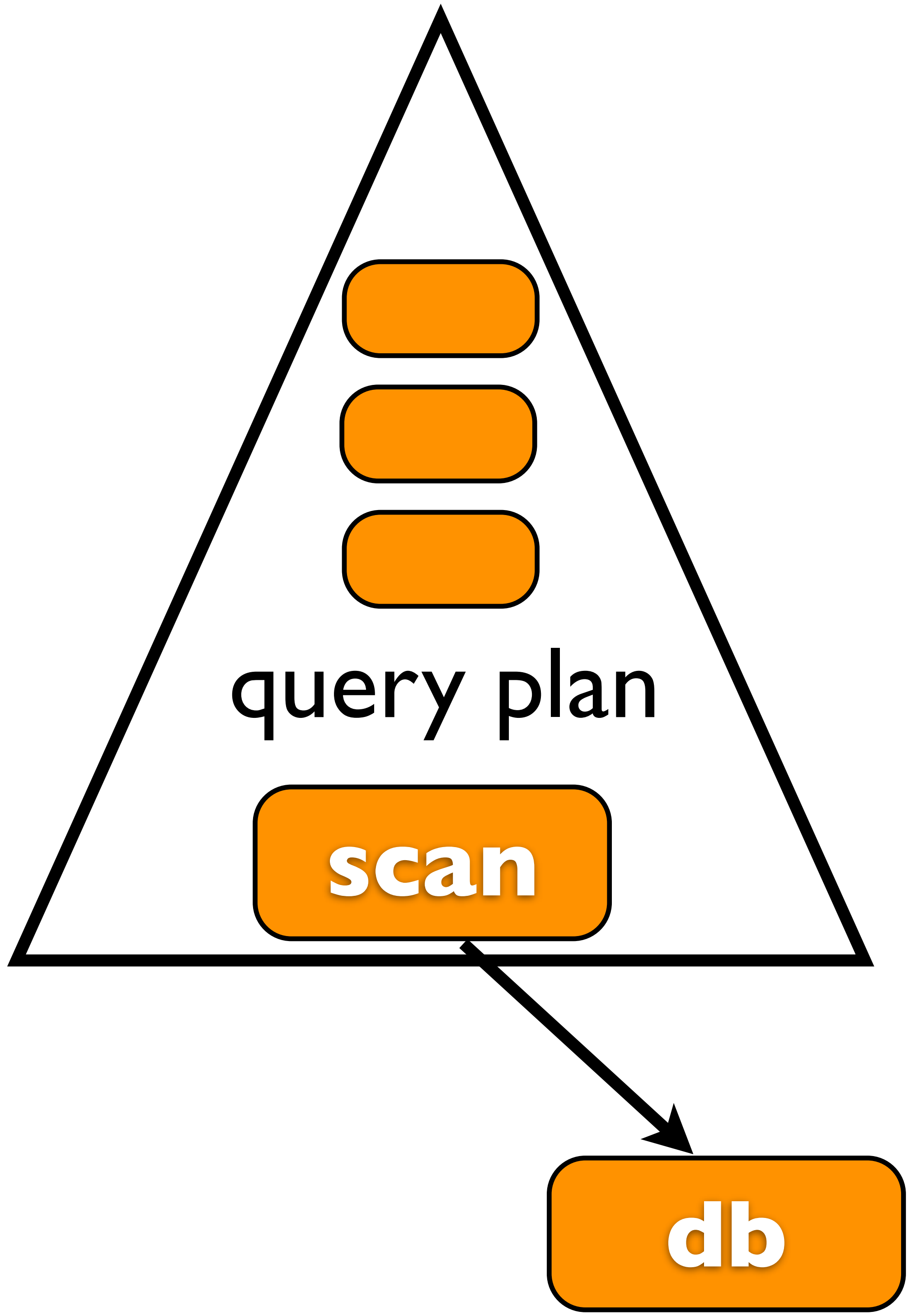
but raw data access is expensive

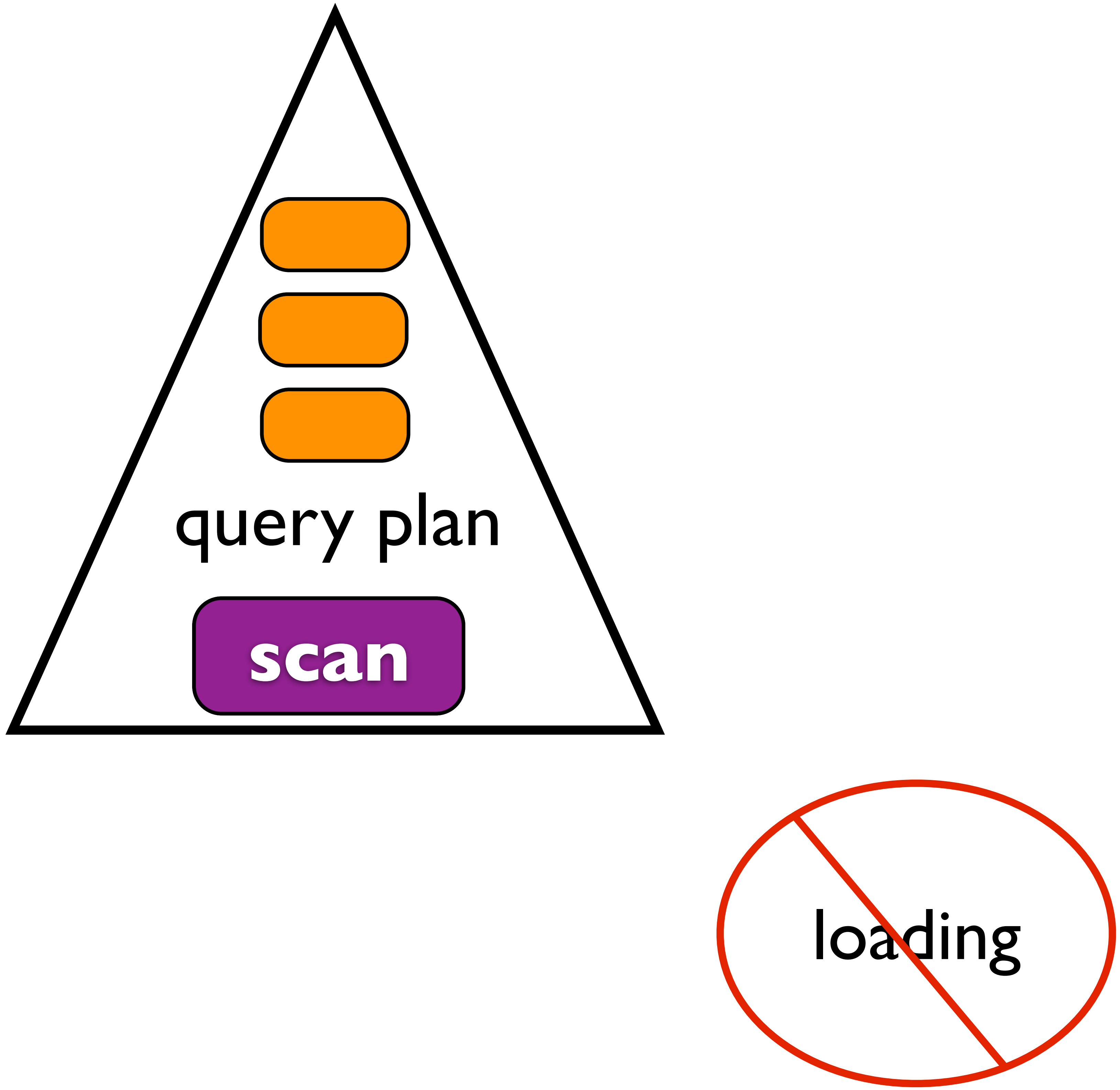
tokenizing - parsing - no indexing - no statistics

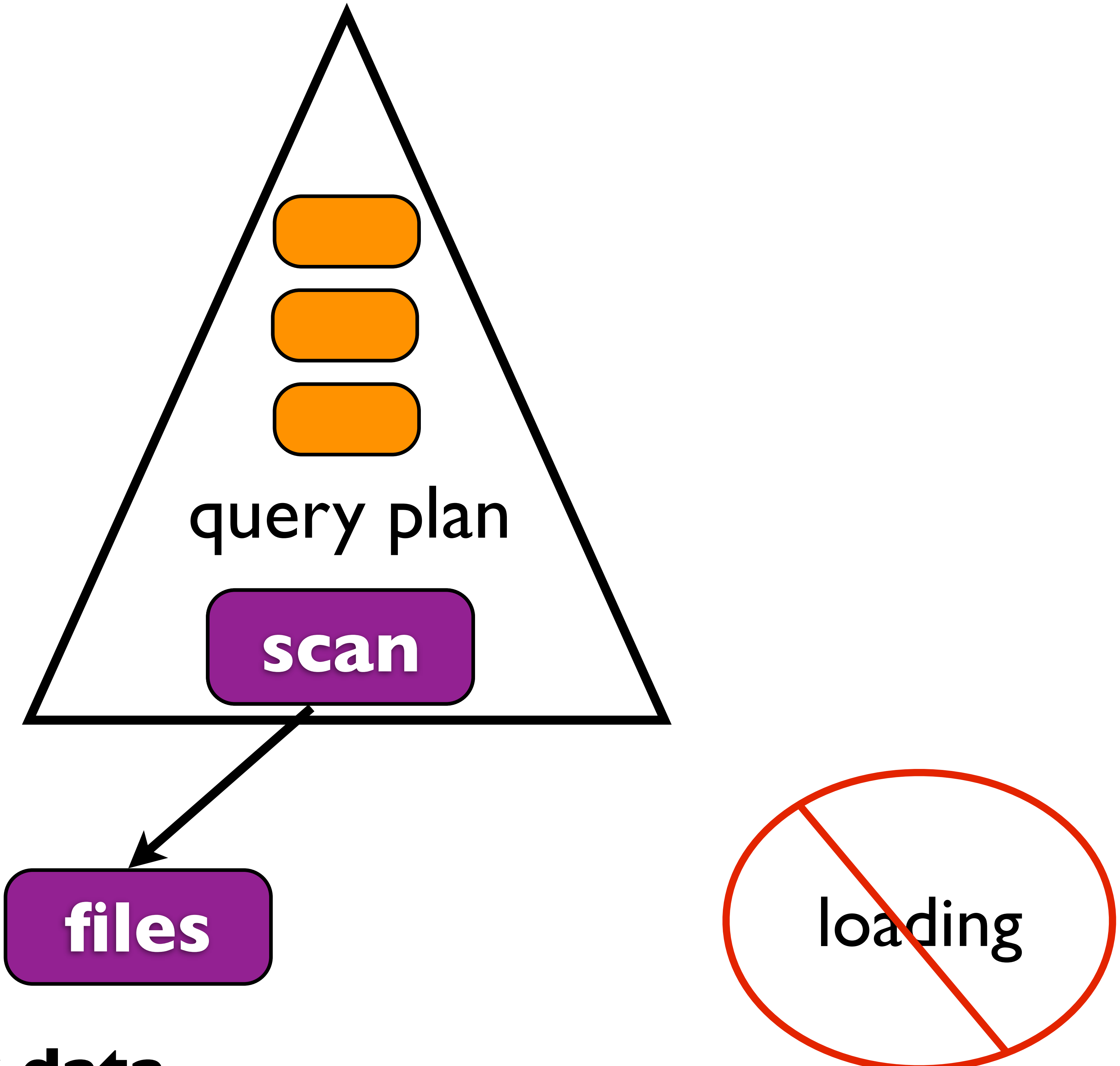
challenge: fast raw data access


query plan

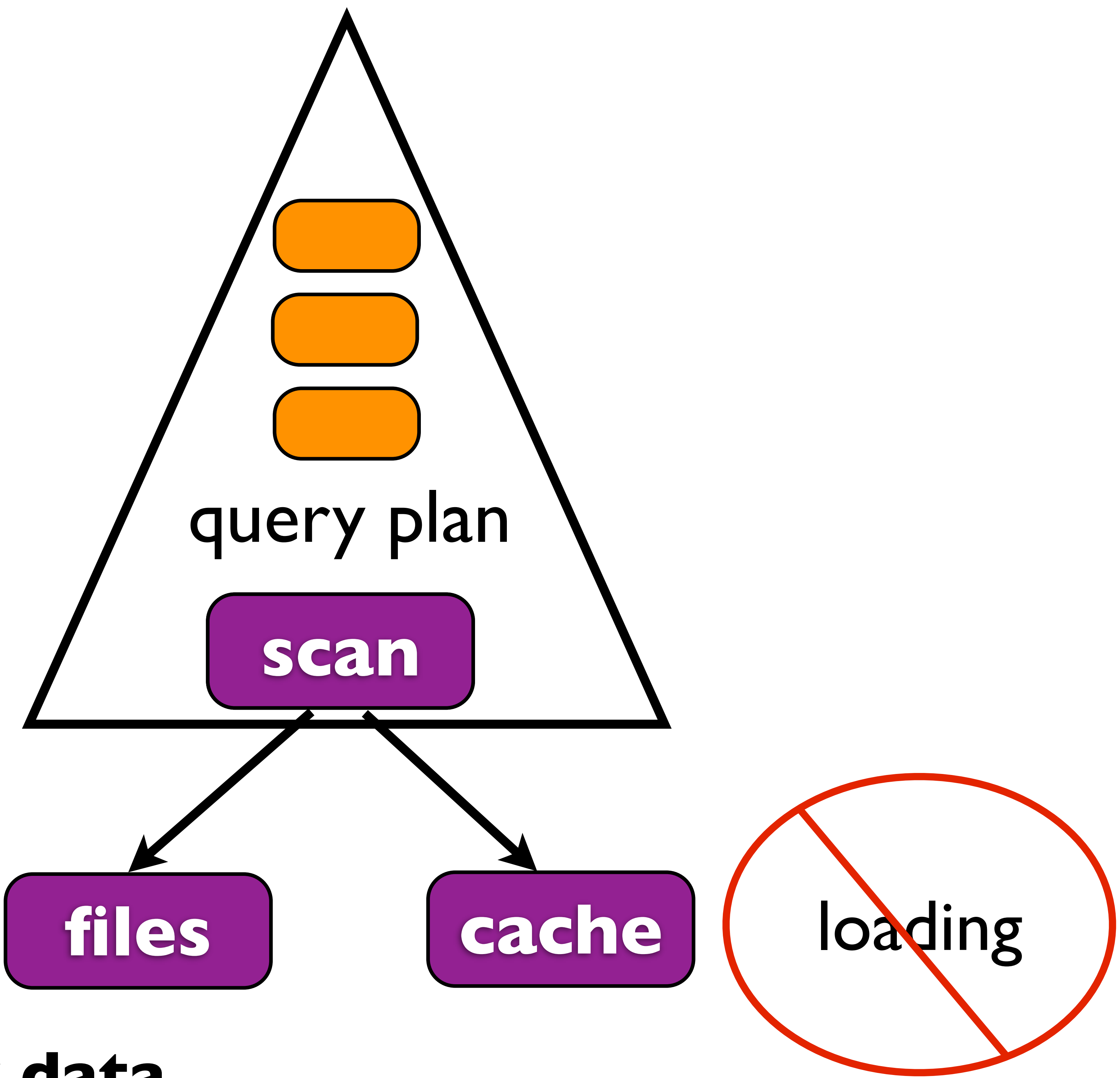

query plan
scan


query plan
scan
db

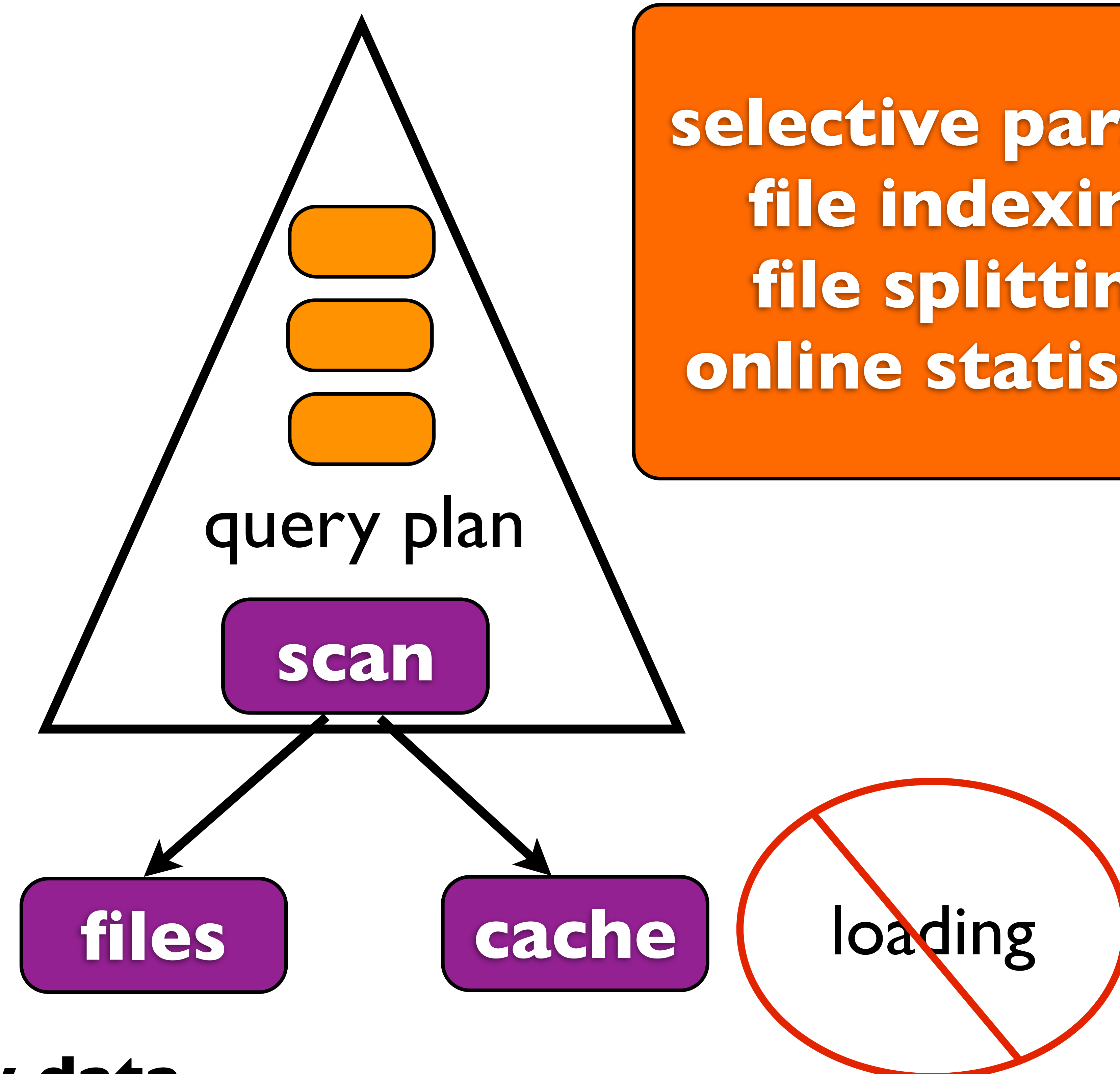

query plan
scan
loading



access raw data adaptively on-the-fly

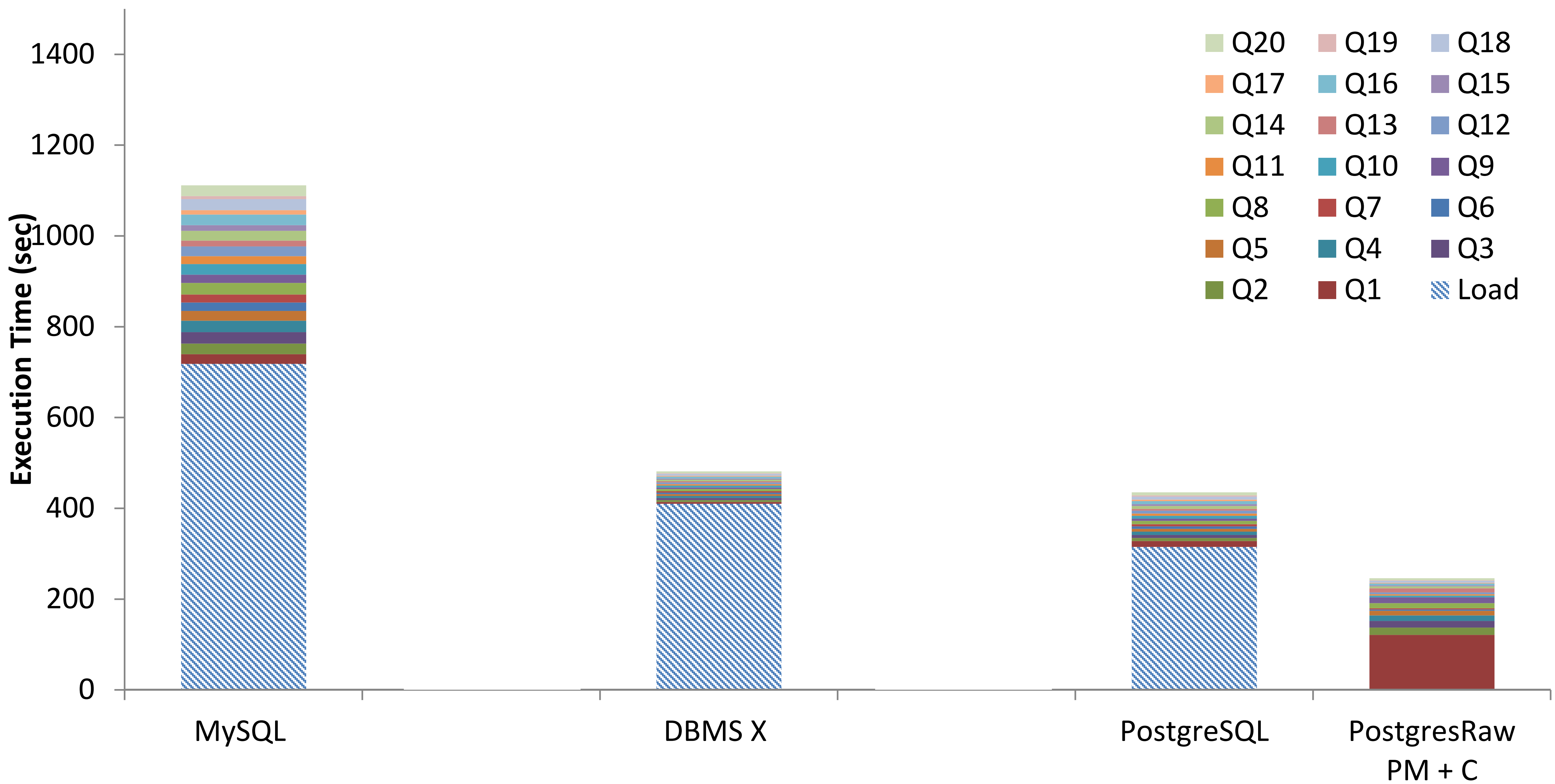


access raw data adaptively on-the-fly

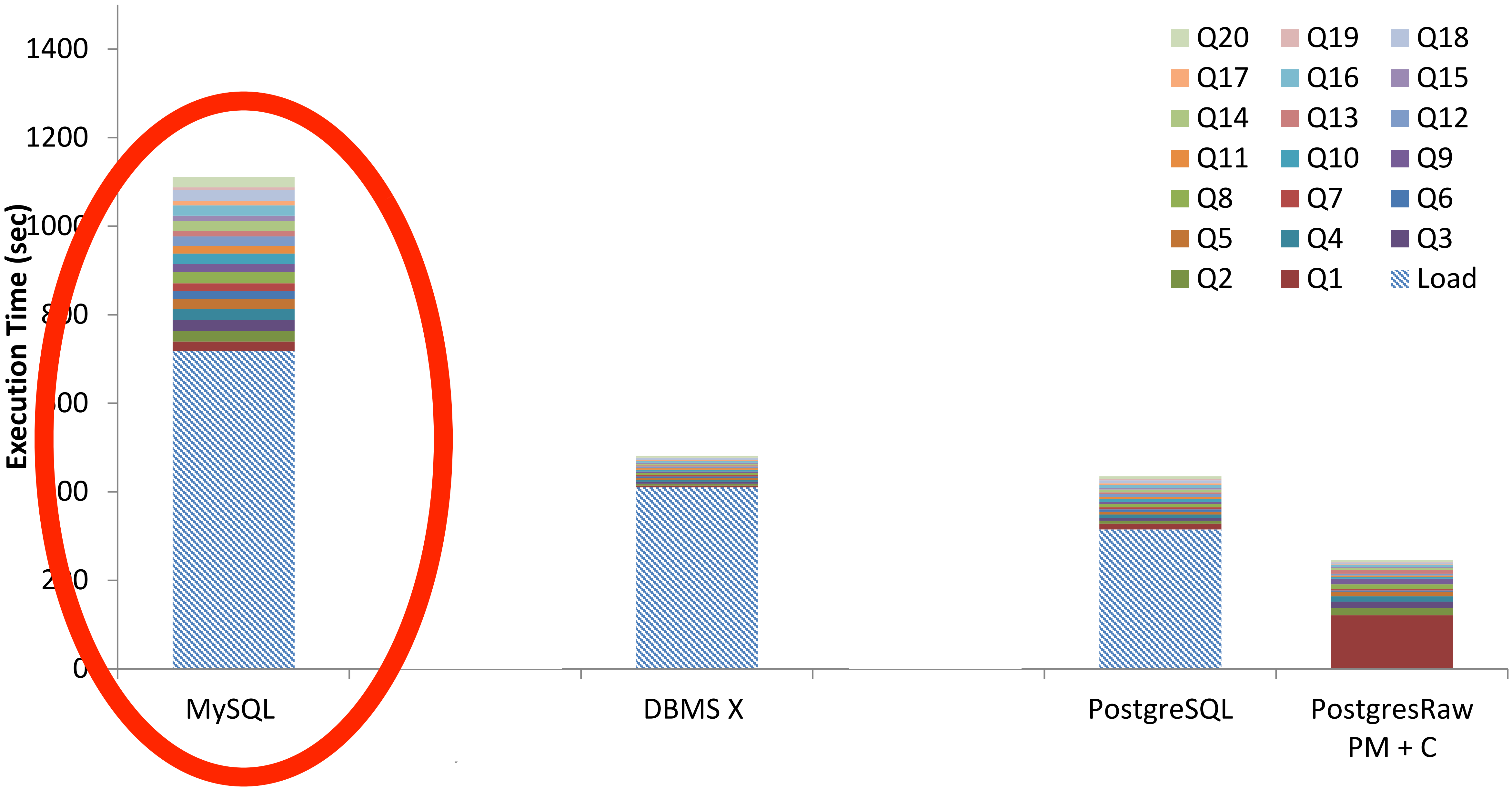


selective parsing
file indexing
file splitting
online statistics

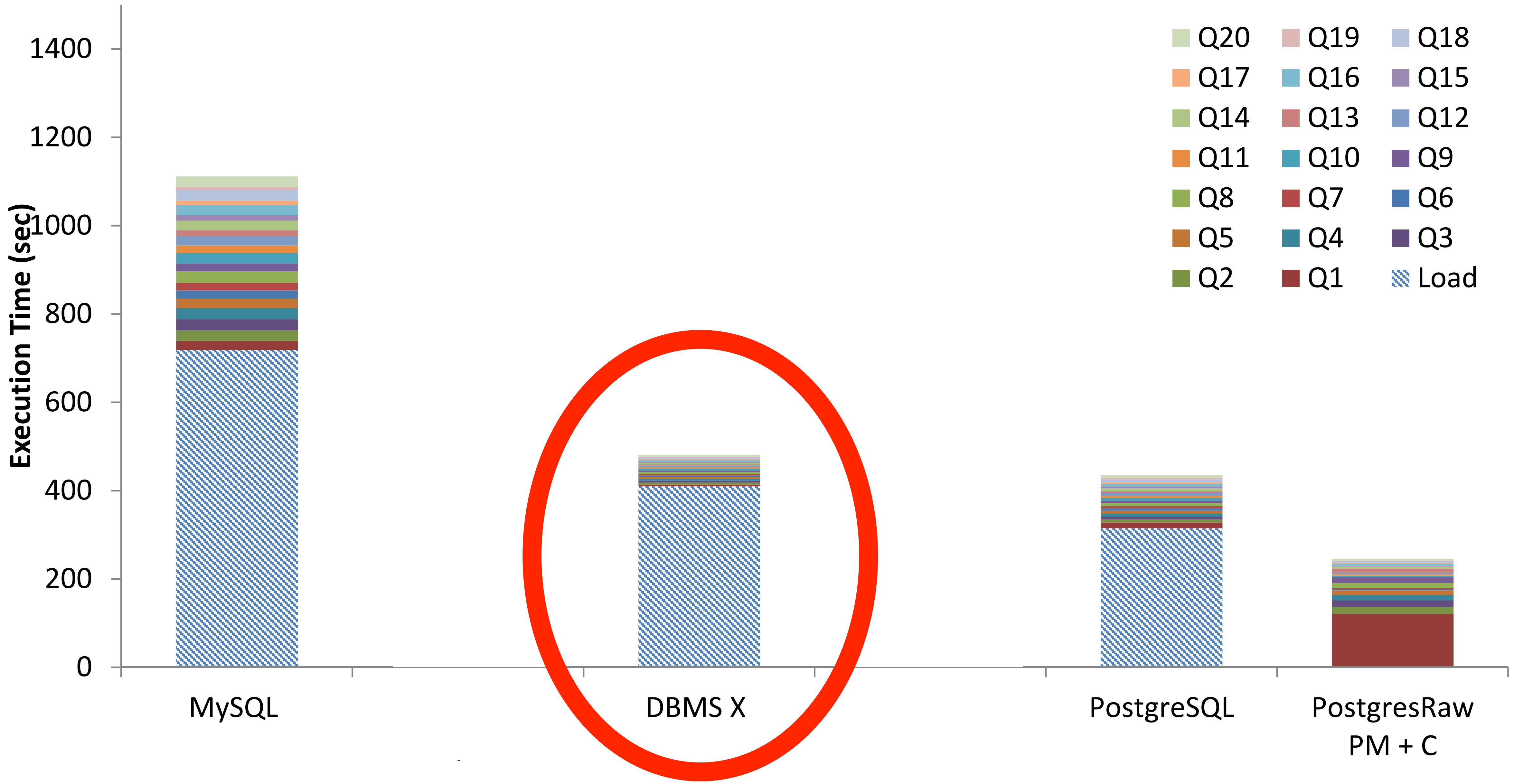
access raw data
adaptively on-the-fly

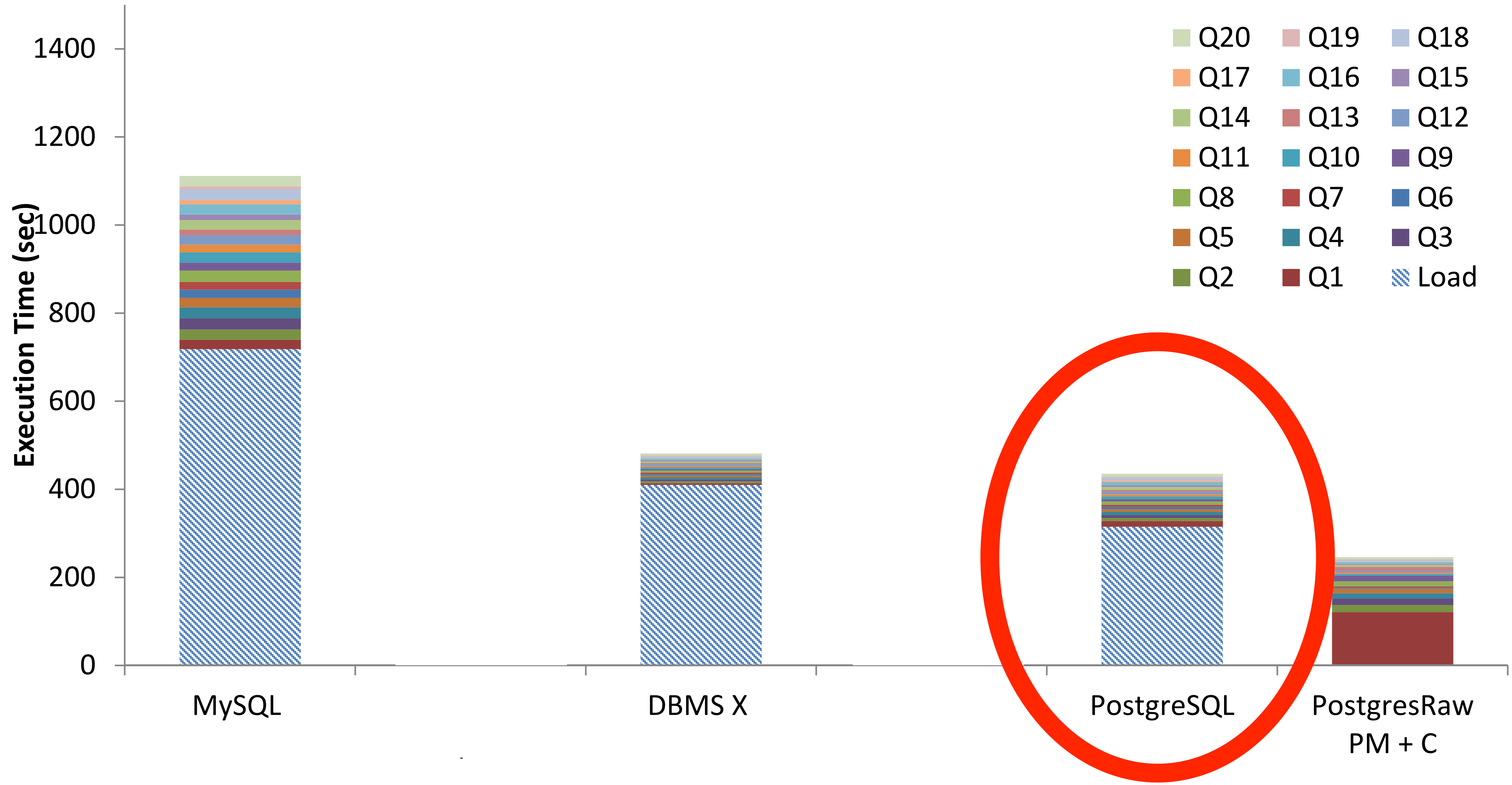


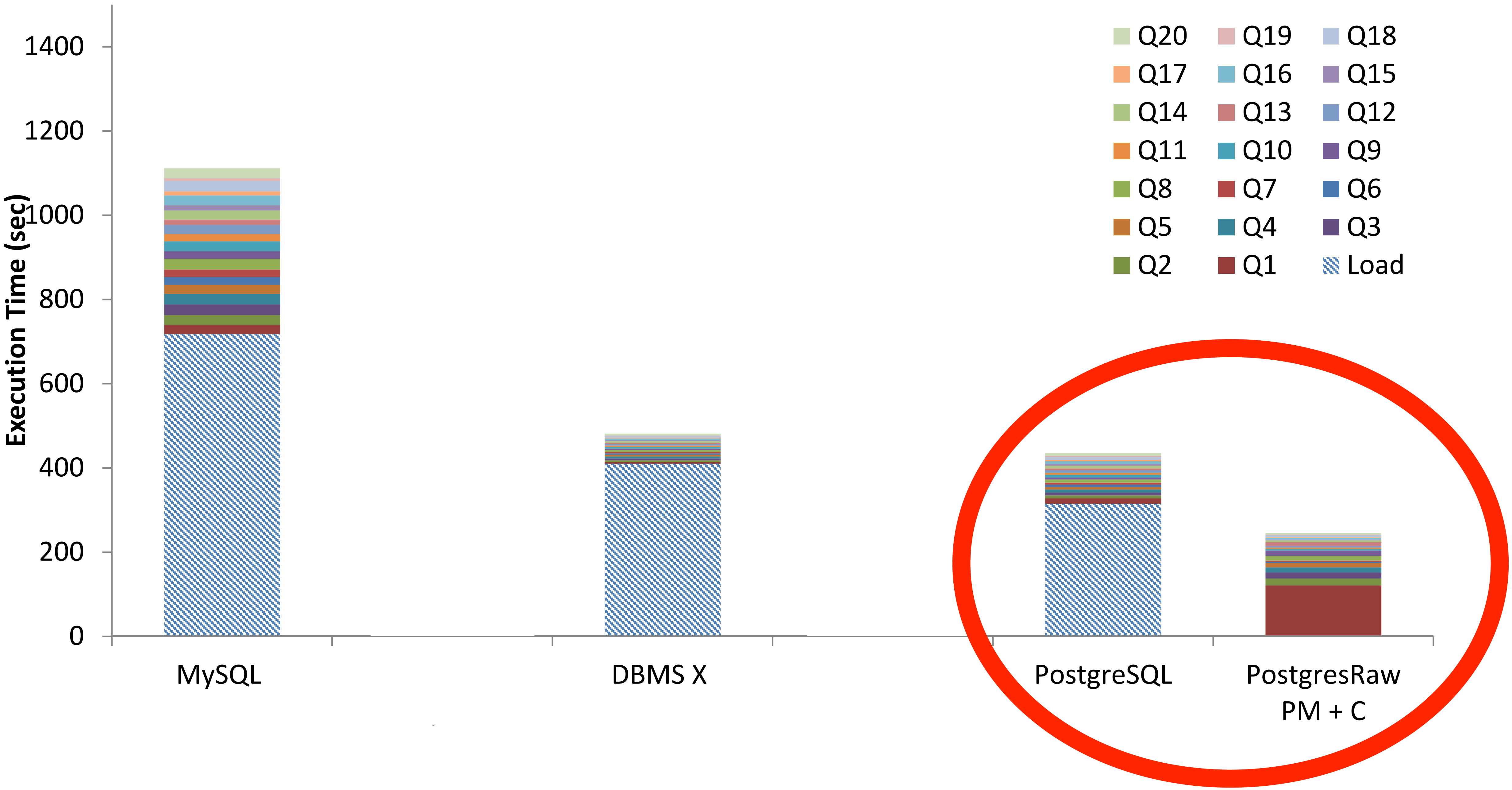
Execution Time (sec)
1400
1200
1000
800
600
400
200
0
MySQL
DBMS X
PostgreSQL
PostgresRaw PM + C
Q20
Q19
Q18
Q17
Q16
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Load



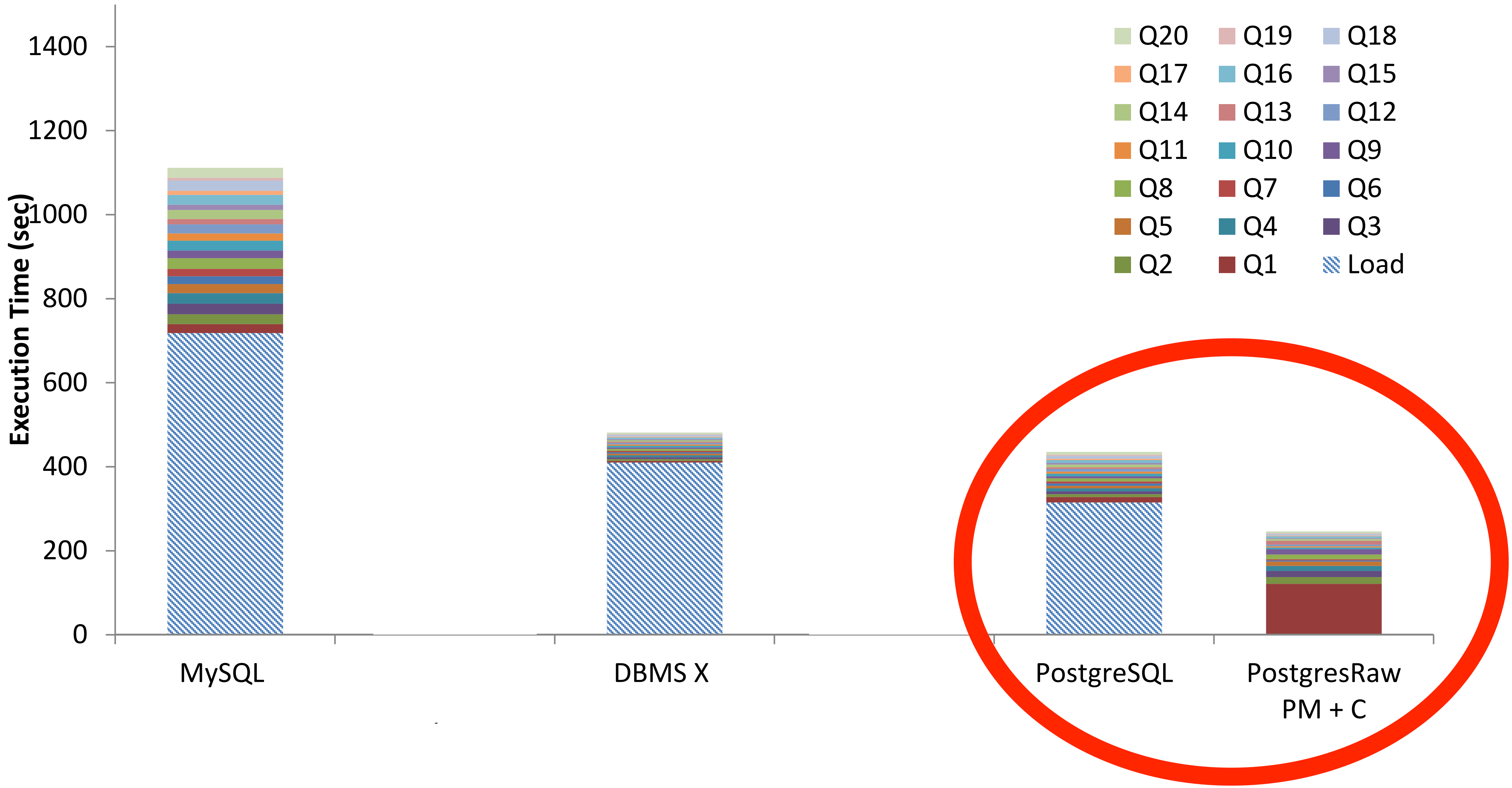
Execution Time (sec)
1400
1200
1000
800
600
400
200
0
MySQL
DBMS X
PostgreSQL
PostgresRaw PM + C
Q20
Q19
Q18
Q17
Q16
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Load


Execution Time (sec)
0
200
400
600
800
1000
1200
1400
Q20
Q19
Q18
Q17
Q16
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Load
MySQL
DBMS X
PostgreSQL
PostgresRaw PM + C


Execution Time (sec)
0
200
400
600
800
1000
1200
1400
MySQL
DBMS X
PostgreSQL
PostgresRaw PM + C
Q20
Q19
Q18
Q17
Q16
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Load


Execution Time (sec)
1400
1200
1000
800
600
400
200
0
Q20
Q19
Q18
Q17
Q16
Q15
Q14
Q13
Q12
Q11
Q10
Q9
Q8
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Load
MySQL
DBMS X
PostgreSQL
PostgresRaw PM + C

reducing data-to-query time



towards auto-tuning data kernels

towards auto-tuning data kernels

so what's next?

adaptive (load-store-execute)

cracking(+ AI, + OS, +ML)

compression

multidimensional cracking

disk based cracking

multi-core cracking

row-store cracking

aggregations

...and many more...

interactive data systems

querying

querying

load tune query

SQL interface

querying

SQL interface

correct and complete answers

querying

complex and slow - not fit for exploration

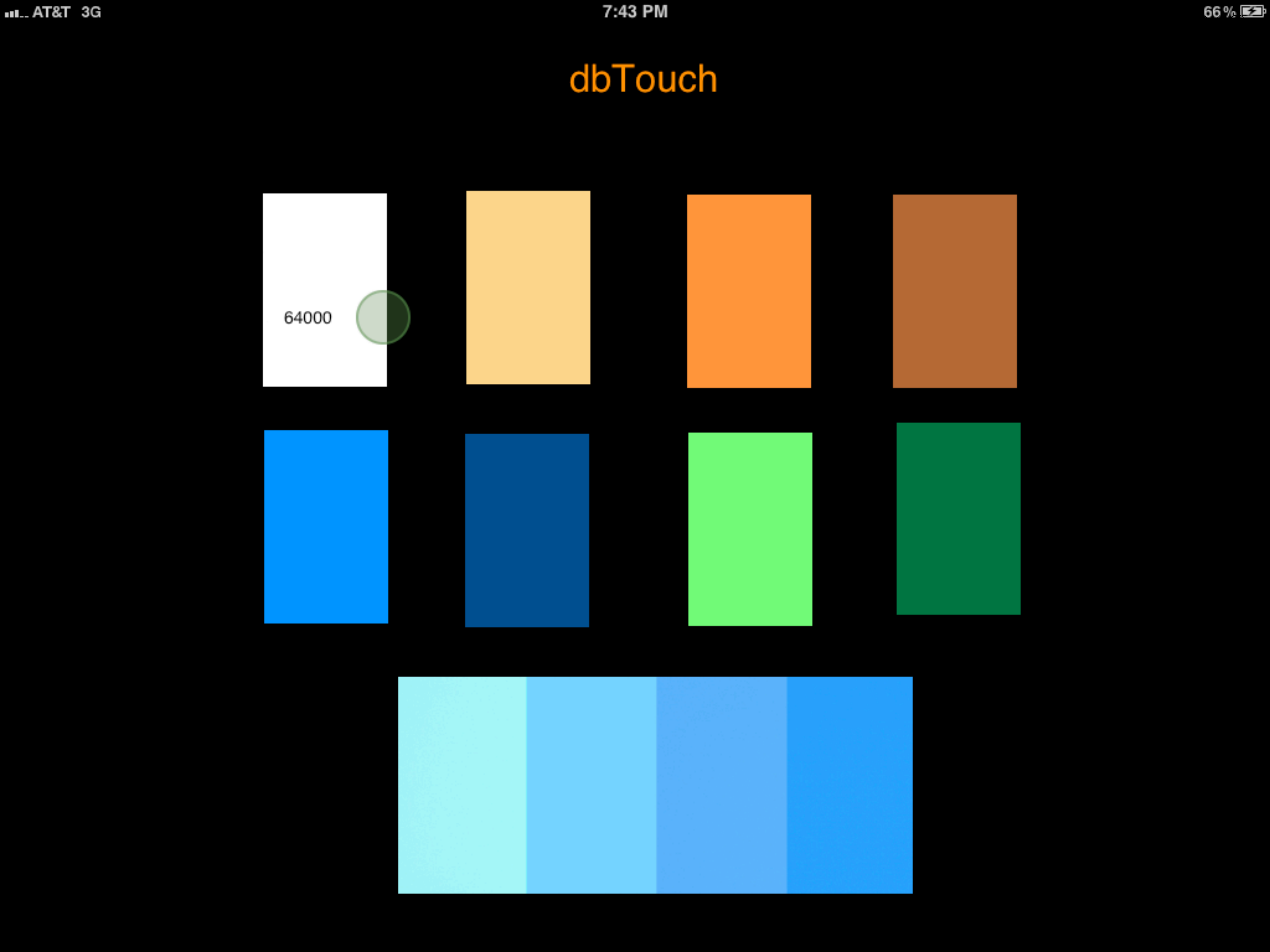
SQL interface

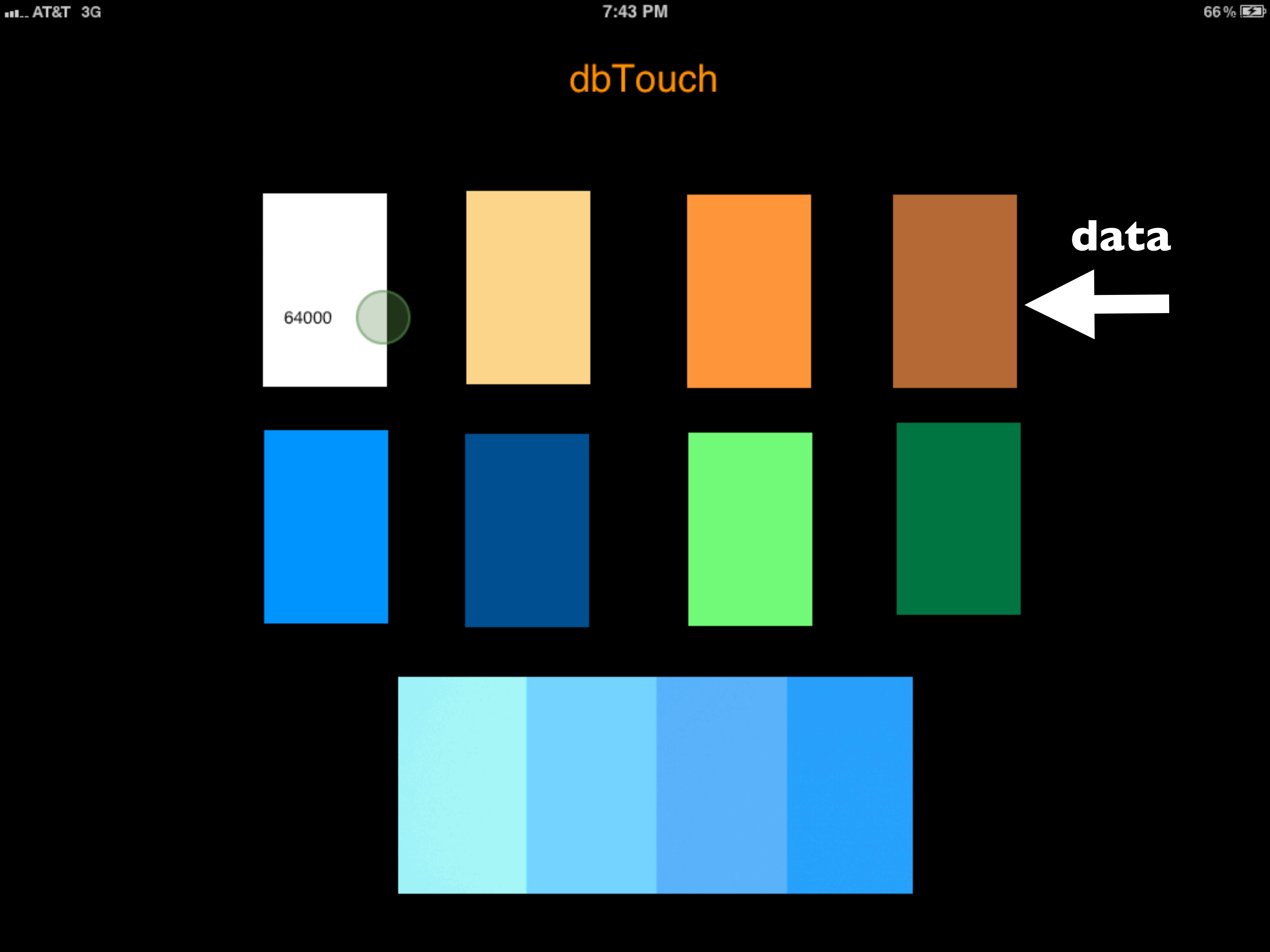
correct and complete answers

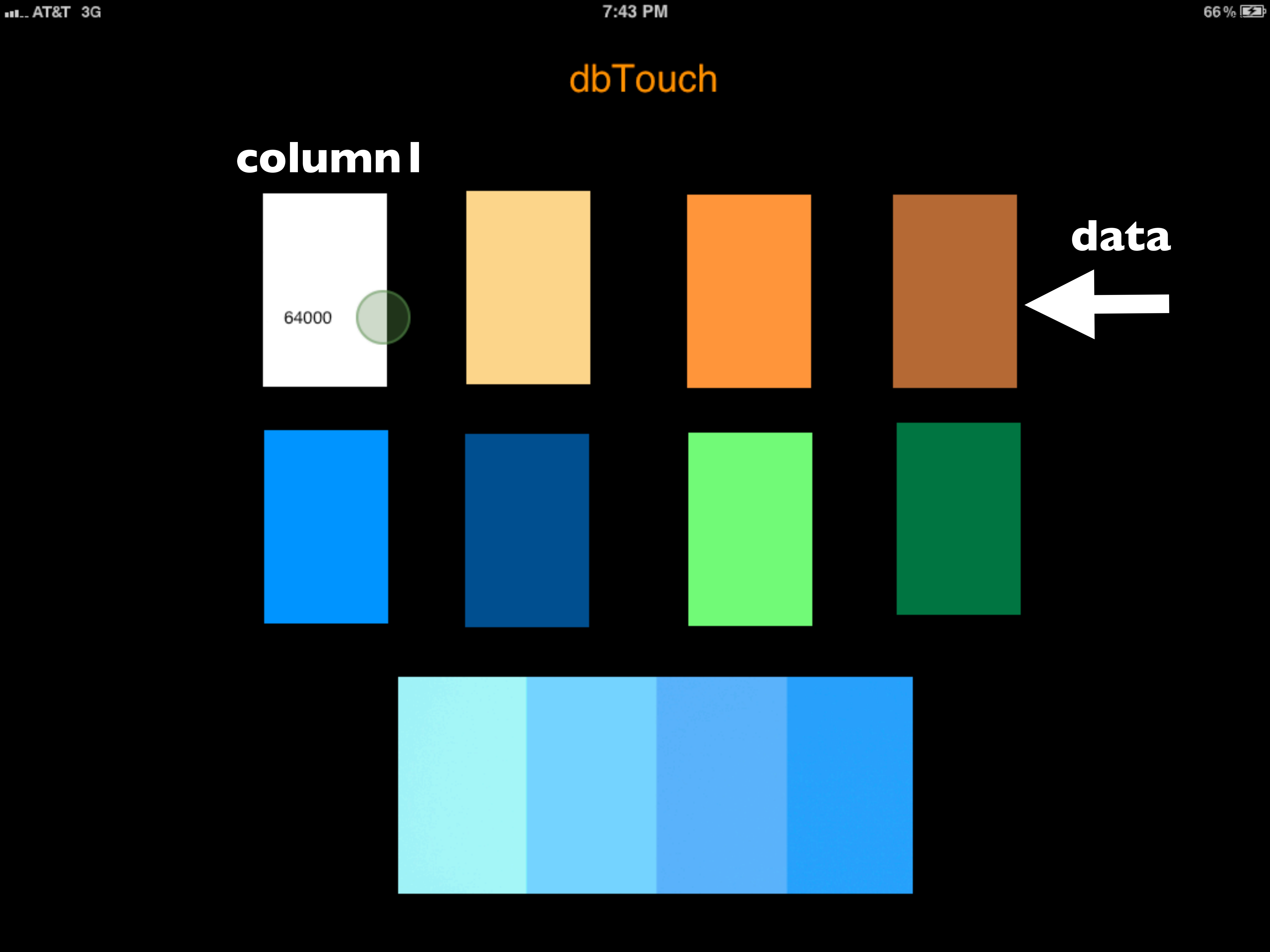
just touch the data you need

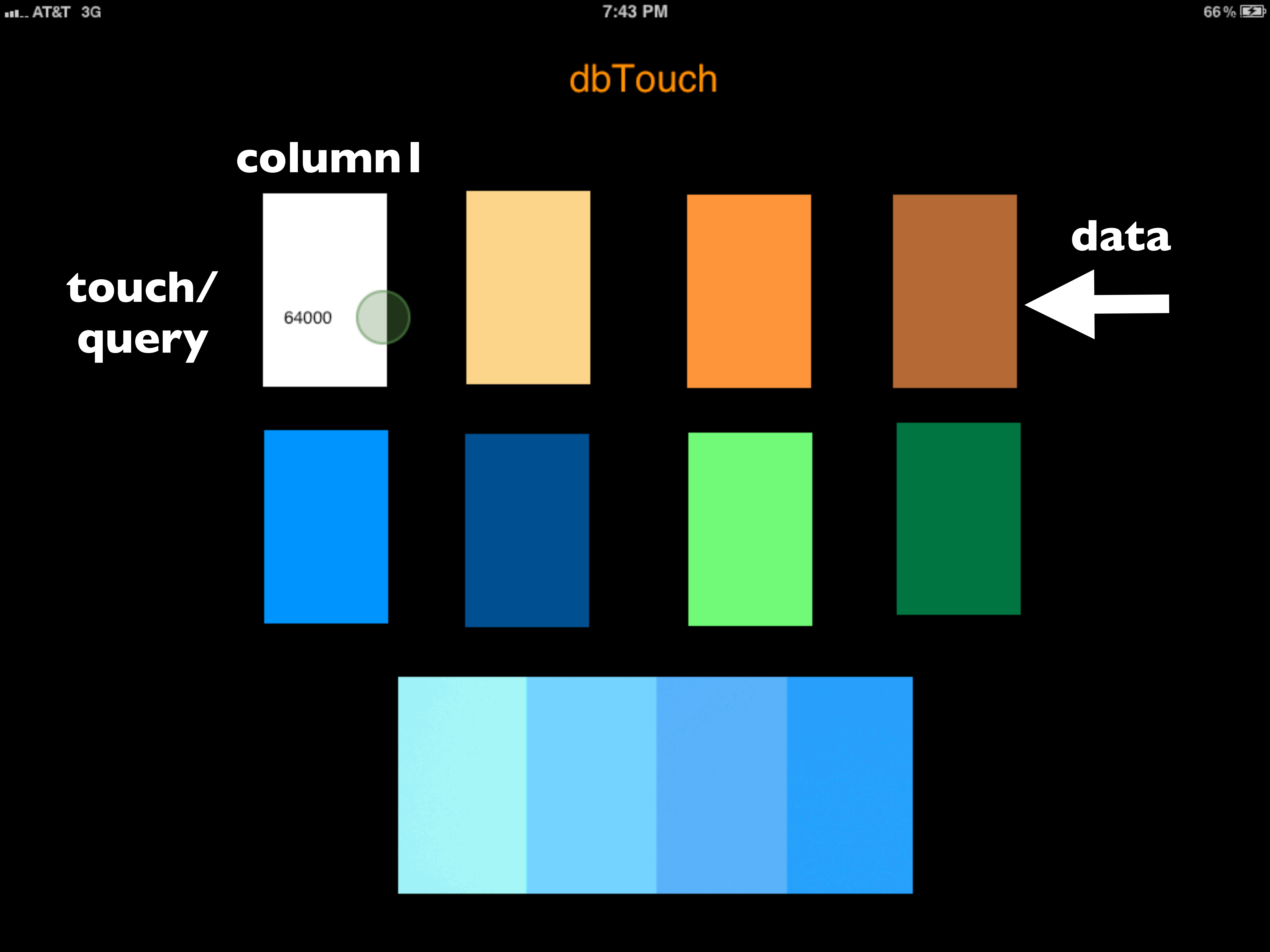
just touch the data you need

this is not about query building
it is about query processing


AT&T 3G
7:43 PM
66 %
dbTouch
64000


AT&T 3G
7:43 PM
66 %
dbTouch
64000
data

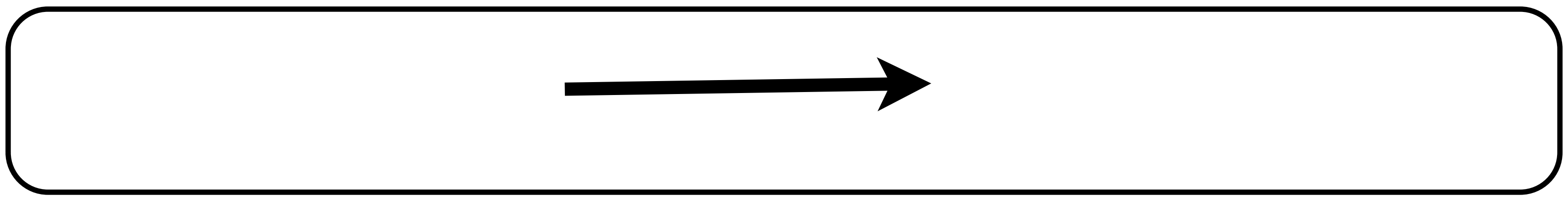

AT&T 3G
7:43 PM
66 %
dbTouch
column1
64000
data


AT&T 3G
7:43 PM
66 %
dbTouch
column1
touch/
query
64000
data

dbTouch demo (ICDE 2014)

what does this mean for db kernels?

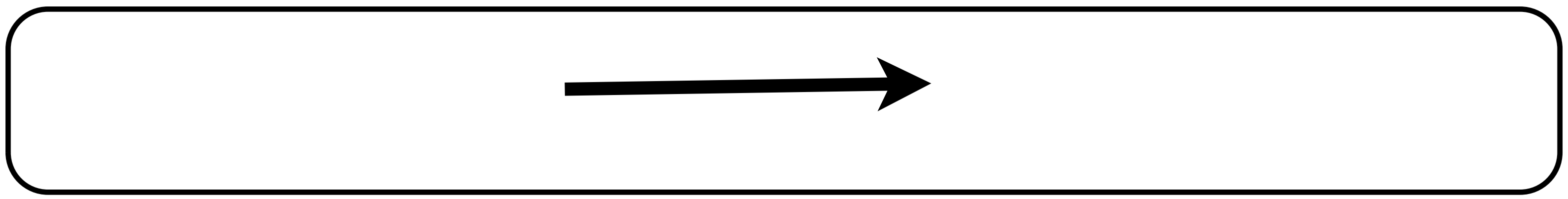
db



select R.a from R

what does this mean for db kernels?

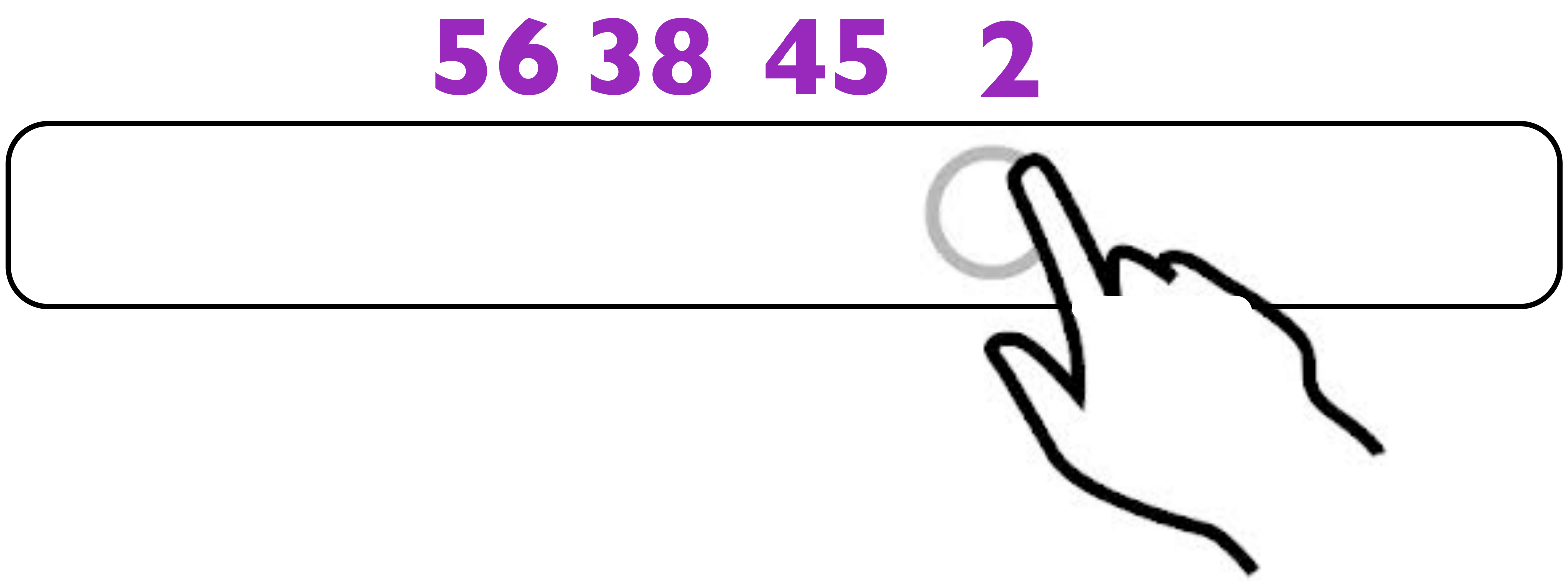
db



select R.a from R

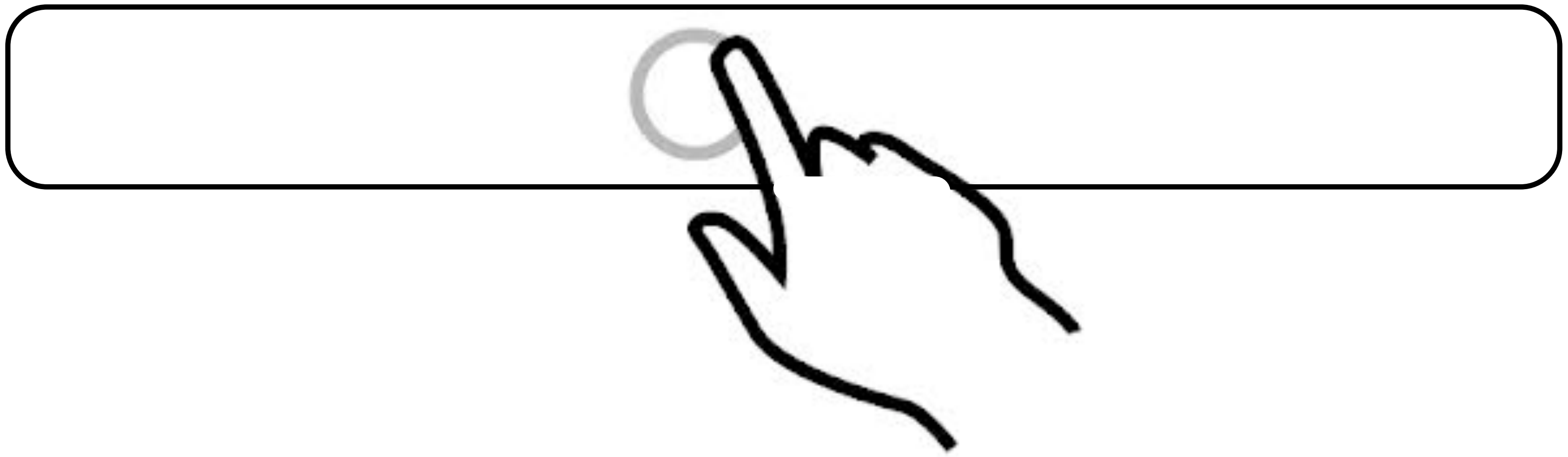
what does this mean for db kernels?

dbTouch



process only what you touch

explore: touch,observe and react

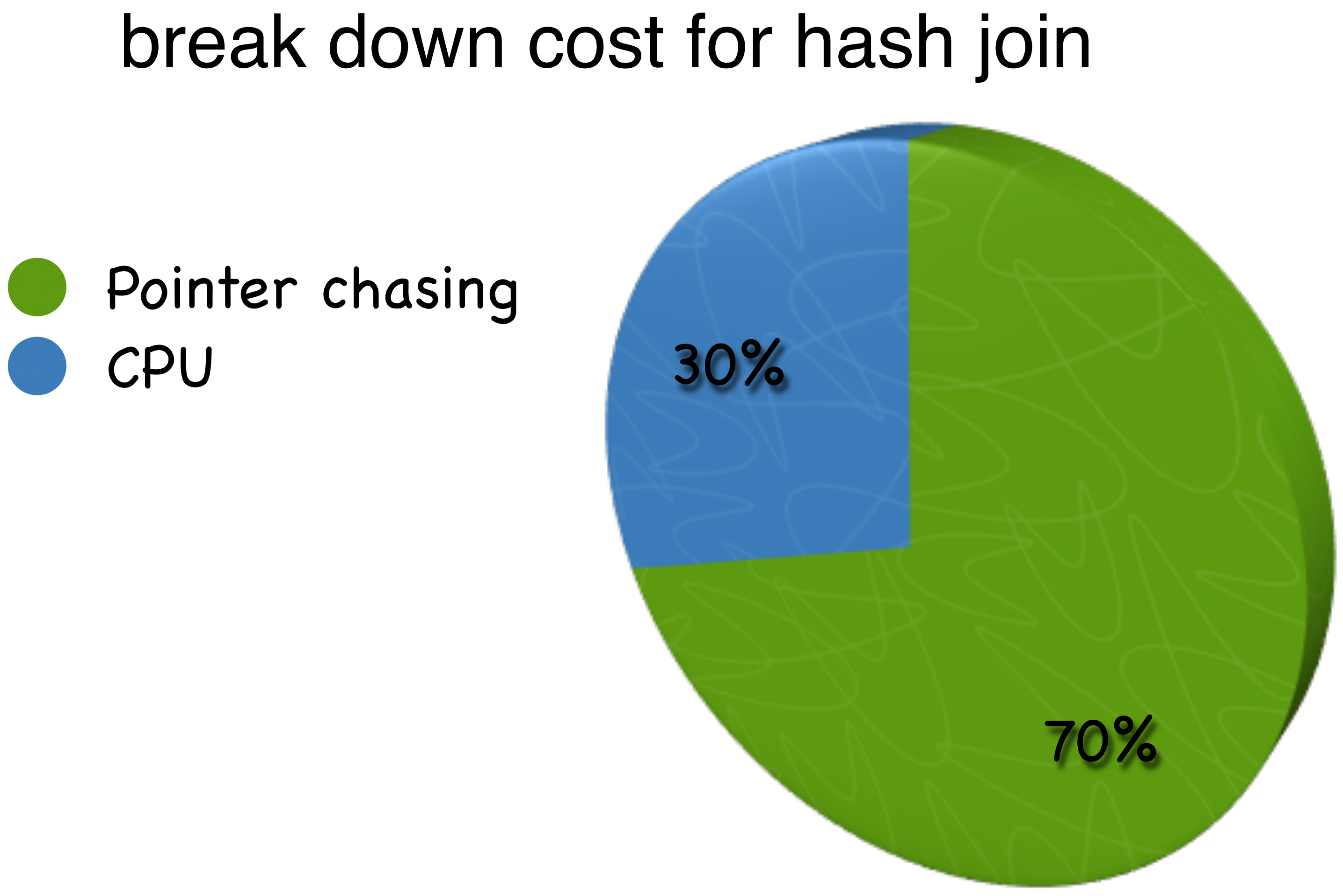


the system does not have control of the data flow
the user dictates which is the next tuple

hierarchies of samples
incremental and adaptive operators
adaptive indexing - adaptive loading

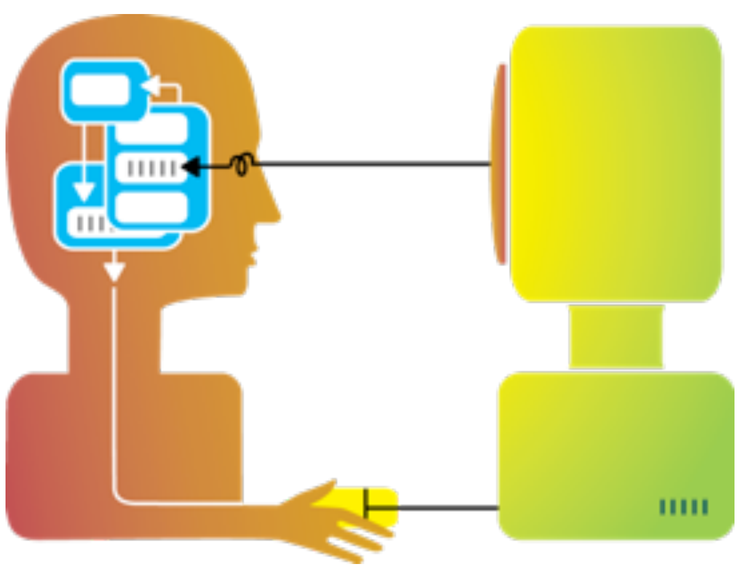
rethink db kernels: correct Vs. interactive

rethink db kernels: correct Vs. interactive



an exploration tool

get a quick feeling about your data
focus on interesting areas

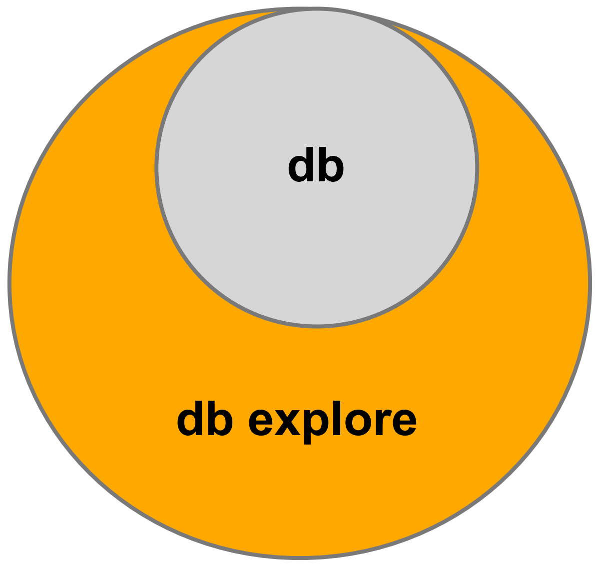


+

HCI + databases

a database system

allows you to answer queries fast



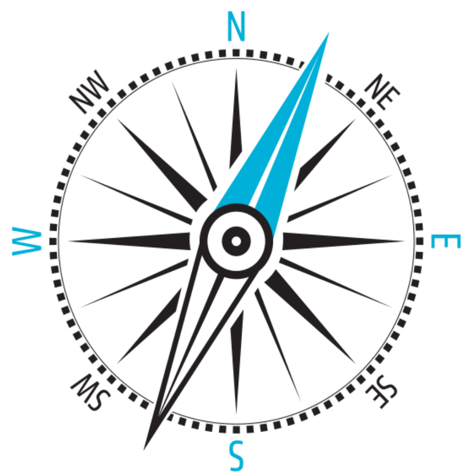
a data exploration system

allows you to find fast which queries to ask

properties of data exploration systems

easy to use
(no tuning, no set-up)

interactive navigation
(no need for correct/complete answers)



adaptive indexing

dbTouch

adaptive loading

3 Ideas for Big Data Exploration

adaptive systems - tailored for exploration

it is not the strongest species that survive, nor the most intelligent, but the ones most responsive to change

[Darwin, Megginson]

Stratos Idreos

adaptive indexing

dbTouch

adaptive loading

3 Ideas for Big Data Exploration

adaptive systems - tailored for exploration

it is not the strongest species that survive, nor the most intelligent, but the ones most responsive to change

[Darwin, Megginson]

Thank you!

Stratos Idreos