auto-tuning database kernels
prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
soon everyone will need to be a “data scientist”

hmm, my data is too big :(

how far away are we from a future where a data system sits in the critical path of everything we do?

new applications/requirements
Stratos’ theory #2

[IBM big data]

data system design, set-up, tune, use

[Stratos’ Guess]
not always sure what we are looking for (until we find it)

3 data exploration

data has always been big

volume  velocity  variety  veracity
data systems that are easy to:

design & build (years)

set-up & tune (months)

use (hours/days)
too many preparation options lead to complex installation

schema  storage  load  indexing  query

timeline

expert users - idle time - workload knowledge
users/applications
declarative interface
ask what you want
db system
DBA
need to choose the proper system & workloads/applications change rapidly
be able to query the data immediately & with good performance
be able to query the data immediately & with good performance

raw data

explore data and gain knowledge “immediately”
tune = create proper indices offline
performance 10-100X
indexing

storage  load  indexing  query

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?
timeline

storage load indexing query

dsampling workload

timeline
storage  load  indexing  query

sample workload  analyze  create indices

timeline
storage | load | indexing | query

sample workload | analyze | create indices | query

timeline
Stratos Idreos

timeline

storage load indexing 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

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
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

relation1/table1
A
B
C
D

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

sort

binary search

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

sort

binary search

result

1 2 3 4 5 6 7 8 9

11 12 13

11

14

16

19

time + knowledge

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

<table>
<thead>
<tr>
<th>column A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
Q1: select R.A from R where R.A > 10 and R.A < 14

piece1: A <= 10
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
Q1:
select R.A from R
where R.A>10
and R.A<14

piece1: A<=10
piece2: 10<A<14
piece3: A>=14
Q1:
select R.A from R
where R.A>10
and R.A<14

result

piece1: A<=10

piece2: 10<A<14

piece3: 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

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

Q2:
select R.A
from R
where R.A > 7
and R.A <= 16
Q1:
select R.A from R
where R.A>10
and R.A<14

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

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

**Result:**
- **piece1:** $A \leq 10$
- **piece2:** $7 < A < 14$
- **piece3:** $A \geq 14$

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]
select [15,55]

10 20 30 40 50 60

select [15,55]
select [15,55]

select [15,55]
touch at most two pieces at a time

pieces become smaller and smaller

select $[15,55]$
Stratos Idreos

# of queries

response time

standard

optimal

but requires time, workload knowledge & stability

# of queries
but requires time, workload knowledge & stability
Stratos Idreos

- "# of queries"
- "response time"
- "standard"
- "optimal"
- "adaptive"

but requires time, workload knowledge & stability
set-up

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

Database Cracking CIDR 2007
set-up

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

almost no
initialization overhead

of continuous adaptation

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

almost no
initialization overhead

continuous improvement

**Response time (secs)**

**Query sequence (x1000)**

- Scan
- Crack
- Full Index
set-up
100K random selections
random selectivity
random value ranges
in a 10 million integer column

almost no
initialization overhead

continuous improvement

Response time (secs)
Query sequence (x1000)

Database Cracking CIDR 2007
set-up

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

Continuous adaptation

Cumulative average response time (secs)

Query sequence

Full Index
Scan
Crack

Database Cracking CIDR 2007
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

set-up

Database Cracking CIDR 2007
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
updates
cracking indices are auxiliary data structures can be dropped any time
column of 10M tuples, random queries
1000 random insertions every 1000 queries

when updates arrive, drop the index

forgetting

crack (forget)

scan

Stratos Idreos
column of 10M tuples, random queries
1000 random insertions every 1000 queries

when updates arrive, drop the index

we do not exploit past cracking
log updates and apply on-line and on-demand during cracking
log updates and apply on-line and on-demand during cracking

updates

query

maintain
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>9</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>12</td>
<td>16</td>
<td>25</td>
<td>29</td>
</tr>
<tr>
<td>25</td>
<td>22</td>
<td>29</td>
<td>39</td>
<td>31</td>
</tr>
<tr>
<td>39</td>
<td>31</td>
<td>33</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

pending inserts
pending deletes
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

column

pending inserts

pending deletes
goal: minimize physical actions

arrays are dense
pieces are ordered
values in a piece are not ordered
goal: minimize physical actions

arrays are dense

pieces are ordered

values in a piece are not ordered

<table>
<thead>
<tr>
<th>pending inserts</th>
<th>pending deletes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
column of 10M tuples, random queries
1000 random insertions every 1000 queries

merge all updates
merge only qualifying updates
column of 10M tuples, random queries
1000 random insertions every 1000 queries

merge all updates
merge only qualifying updates

gradual merging avoids high picks
Stratos Idreos

The ripple

Pending updates

Query result column
the ripple
the ripple
the ripple

column

query result

pending updates

doing work not relevant for the current query
the ripple

doing work not relevant for the current query
Stratos Idreos

the ripple

doing work not relevant for the current query
The diagram illustrates the concept of "the ripple." It shows two columns representing query results and a central column labeled "pending updates." The text notes that the system might be doing work not relevant for the current query. The diagram is accompanied by the name "Stratos Idreos."
pending updates

doing work not relevant for the current query
the ripple

doing work not relevant for the current query
the ripple

doing work not relevant for the current query
the ripple

doing work not relevant for the current query
the ripple

query result

pending updates

doing work not relevant for the current query

query result
the ripple

column

pending updates

query result

doing work not relevant for the current query

column

query result
the ripple

query result

pending updates

update only the hot tuples

doing work not relevant for the current query
Stratos Idreos

queries (x1000)

merge all updates

merge only qualifying updates

Cost per query (microseconds)

Cost per query (microseconds)

forget

ripple

maintains adaptive behavior
sideways cracking
how to process >1 columns
positional alignment

positional lookups

\[ A(i) = A + i \times \text{width}(A) \]
select min(C) from R where A<10 & B<20
**select** \( \text{max}(D), \text{min}(E) \) **from** \( R \) **where** \((A>10 \text{ and } A<40) \text{ and } (B>20 \text{ and } B<60)\)**

- **binary search** for 10 & 40
- For all B values between pos1 & 2: if B>20 and B<60
- Mark bit vector at pos \( i \)
- For each marked position
- **max(D)**

**avoid scan** of A  
**avoid TR** on B 
Work on a **restricted area** across all columns 
Good for memory hierarchy

---

*Stratos Idreos*
RowIDs

without cracking

ordered

sequential access

RowIDs

with cracking

unordered

random access
sideways cracking
sideways cracking
sideways cracking

A

B

C

D

Stratos Idreos

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09

query

A      B      C      D

Sideways Cracking
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Element" /></td>
<td><img src="image2.png" alt="Element" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image3.png" alt="Element" /></td>
<td><img src="image4.png" alt="Element" /></td>
</tr>
</tbody>
</table>
Sideways Cracking, SIGMOD 09

Stratos Idreos
sideways cracking

query
sideways cracking

query

Stratos Idreos

Sideways Cracking, SIGMOD 09
sideways cracking

A

B

C

D

query
Sideways Cracking, SIGMOD 09

A
B
C
D

query
Sideways Cracking, SIGMOD 09

query
log crack actions and replay to align columns dynamically
replace tuple reconstruction with cracking actions
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td>15</td>
<td>b6</td>
</tr>
<tr>
<td>22</td>
<td>b7</td>
<td>22</td>
<td>b8</td>
</tr>
<tr>
<td>7</td>
<td>b9</td>
<td>4</td>
<td>b10</td>
</tr>
<tr>
<td>26</td>
<td>b11</td>
<td>24</td>
<td>b12</td>
</tr>
<tr>
<td>11</td>
<td>b13</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Initial state**

`select B from R where 10<A<15`
<table>
<thead>
<tr>
<th>Initial state</th>
<th>select B from R where 10&lt;A&lt;15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

Piece 1: Position 1, value <=10
Piece 2: Position 7, value >10
Piece 3: Position 9, value >=15

Sideways Cracking, SIGMOD 09
Sideways Cracking, SIGMOD 09

<table>
<thead>
<tr>
<th>Initial state</th>
<th>select B from R where 10 &lt; A &lt; 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

Cracker Map

<table>
<thead>
<tr>
<th>Cracker index</th>
<th>M_{AB}</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>b9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
<td></td>
</tr>
</tbody>
</table>

Cracker index

- Position 1: value <= 10
  - Piece 1
  - Position 7: value > 10
    - Piece 2
  - Position 9: value >= 15
    - Piece 3

Stratos Idreos
Stratos Idreos

Sideways Cracking, SIGMOD 09

```
select B from R where 10 < A < 15
```

**Cracker Map**

- **Head**
- **Tail**

<table>
<thead>
<tr>
<th>Initial state</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12</td>
<td>b1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>b5</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>b6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>b8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>b11</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>b12</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>b13</td>
</tr>
</tbody>
</table>

**Cracker index**

- **M_{AB}**

**Pieces**

- **Piece 1**
  - **Position 1**
  - value ≤ 10
  - **Cracker Map**
    - Piece 1: 12, b1
    - Piece 2: 11, b12
    - Piece 3: 15, b5

- **Piece 2**
  - **Position 7**
  - value > 10
  - **Cracker Map**
    - Piece 1: 12, b1
    - Piece 2: 11, b12
    - Piece 3: 15, b5

- **Piece 3**
  - **Position 9**
  - value ≥ 15
  - **Cracker Map**
    - Piece 1: 12, b1
    - Piece 2: 11, b12
    - Piece 3: 15, b5

**Crack based on head, carry tail**
sideways cracking

Cracker Map

Head Tail

select B from R where 10<A<15

Cracker index

M_{AB}

Initial state

A B

12 b1
3 b2
5 b3
9 b4
15 b5
22 b6
7 b7
26 b8
4 b9
2 b10
24 b11
11 b12
16 b13

Cracking knowledge

Piece 1

Position 1

value <=10

Piece 2

Position 7

value >10

Piece 3

Position 9

value >=15

crack based on head, carry tail

Stratos Idreos

Sideways Cracking, SIGMOD 09
sideways cracking

Cracker Map

Position 1
value <=10
Piece 1

Position 7
value >10
Piece 2

Position 9
value >=15
Piece 3

Cracking knowledge

No tuple reconstruction
sideways cracking

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cracker index</th>
<th>$M_{AB}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>b1</td>
<td>4</td>
<td>b9</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
<td>3</td>
<td>b2</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td>5</td>
<td>b3</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
<td>9</td>
<td>b4</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td>2</td>
<td>b10</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
<td>7</td>
<td>b7</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Cracking knowledge**
- No tuple reconstruction
- Dynamically/on-the-fly within the select-operator

**Initial state**
- select B from R where 10 < A < 15

**Cracker Map**
- Head
- Tail
- Crack based on head, carry tail

**Position**
- Position 1
  - value <= 10
  - Piece 1
- Position 7
  - value > 10
  - Piece 2
- Position 9
  - value >= 15
  - Piece 3
sideways cracking

Cracker Map
Head | Tail
--- | ---

<table>
<thead>
<tr>
<th>Initial state</th>
<th>select B from R where 10&lt;A&lt;15</th>
<th>select B from R where 5=&lt;A&lt;17</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>Cracker index</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>12</td>
<td>b1</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>b2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>b3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>b4</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td>b5</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>b6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>b7</td>
<td>12</td>
</tr>
<tr>
<td>26</td>
<td>b8</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>b9</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>b10</td>
<td>22</td>
</tr>
<tr>
<td>24</td>
<td>b11</td>
<td>24</td>
</tr>
<tr>
<td>11</td>
<td>b12</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>b13</td>
<td>16</td>
</tr>
</tbody>
</table>

Cracking knowledge
No tuple reconstruction
Dynamically/on-the-fly within the select-operator

Cracker Map
Head | Tail
--- | ---

<table>
<thead>
<tr>
<th>Cracker index</th>
<th>M&lt;sub&gt;AB&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piece 1</td>
<td>Position 1 value &lt;=10</td>
</tr>
<tr>
<td>Piece 2</td>
<td>Position 7 value &gt;10</td>
</tr>
<tr>
<td>Piece 3</td>
<td>Position 9 value &gt;=15</td>
</tr>
<tr>
<td>Piece 4</td>
<td>Position 11 value &gt;=17</td>
</tr>
<tr>
<td>Piece 5</td>
<td>Position 4 value &gt;=5</td>
</tr>
<tr>
<td>Piece 6</td>
<td>Position 7 value &gt;=10</td>
</tr>
<tr>
<td>Piece 7</td>
<td>Position 11 value &gt;=17</td>
</tr>
</tbody>
</table>
Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

select B from R where A < 3

Crack A < 3

M_{AB}

v < 3

\[ \begin{array}{l}
2 & b4 \\
1 & b3 \\
4 & b2 \\
7 & b1 \\
8 & b5 \\
3 & b6 \\
6 & b7 \\
\end{array} \]

Result

\[ b4, b3 \]

Wrong alignment

Correct alignment

adaptive alignment

Sideways Cracking, SIGMOD 09
Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

select B from R where A<3

Crack A<3

<table>
<thead>
<tr>
<th>v&lt;3</th>
<th>2</th>
<th>b4</th>
</tr>
</thead>
<tbody>
<tr>
<td>v&gt;=3</td>
<td>1</td>
<td>b3</td>
</tr>
</tbody>
</table>

select C from R where A<5

Crack A<5

<table>
<thead>
<tr>
<th>v&lt;5</th>
<th>3</th>
<th>c6</th>
</tr>
</thead>
<tbody>
<tr>
<td>v&gt;=5</td>
<td>4</td>
<td>c2</td>
</tr>
</tbody>
</table>

Result

<table>
<thead>
<tr>
<th>M_{AB}</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>b4</td>
<td>b3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M_{AC}</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>c6</td>
<td>c2</td>
</tr>
</tbody>
</table>

Wrong alignment

Correct alignment
Select B from R where $A < 3$

Select C from R where $A < 5$

Select B, C from R where $A < 4$

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

Crack $A < 3$

Crack $A < 5$

Crack $A < 4$

Result

$M_{AB}$

$M_{AC}$

$M_{BC}$

Sideways Cracking, SIGMOD 09
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

**Initial state**

**select B from R where A<3**

**Crack A<3**

\[ M_{AB} \]

\[ v<3 \]

\[ v>=3 \]

**Result**

\[ b4 \]

\[ b3 \]

**select C from R where A<5**

**Crack A<5**

\[ M_{AC} \]

\[ v<5 \]

\[ v>=5 \]

**Result**

\[ c6 \]

\[ c2 \]

\[ c3 \]

\[ c4 \]

**select B,C from R where A<4**

**Crack A<4**

\[ M_{AB} \]

\[ M_{AC} \]

\[ v<4 \]

\[ v>=4 \]

\[ v>=5 \]

**Result**

\[ c6 \]

\[ c2 \]

\[ c3 \]

\[ c4 \]

\[ b4 \]

\[ c6 \]

\[ c2 \]

\[ c3 \]

**Wrong alignment**

**Correct alignment**

Adaptive alignment
perform the same cracks and in the same order on all maps with the same head
perform the same cracks and in the same order on all maps with the same head

on-line alignment touch/load everything, always
perform the same cracks and in the same order on all maps with the same head.
perform the same cracks and in the same order on all maps with the same head
on-line alignment

touch/load everything, always

remember and replay cracks across columns
select B from R where $A < 3$

select C from R where $A < 5$

select B, C from R where $A < 4$

Wrong alignment
adaptive alignment

Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

select B from R where A < 3

Crack A < 3

\[ M_{AB} \]

Result

b4
b3

v < 3

v > 3

select C from R where A < 5

Crack A < 5

\[ M_{AC} \]

Result

c6

v < 5

v > 5

select B, C from R where A < 4

Crack A < 4

\[ M_{AB} \]

Result

Wrong alignment

Correct alignment

Wrong alignment

Wrong alignment

Sideways Cracking, SIGMOD 09
### Initial state

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>b5</td>
<td>c3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>b6</td>
<td>c4</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c6</td>
<td></td>
</tr>
</tbody>
</table>

### Select B from R where \( A < 3 \)

- \( M_{AB} \)
- \( \text{Result: } \{ b4, b3 \} \)

### Select C from R where \( A < 5 \)

- \( M_{AC} \)
- \( \text{Result: } \{ c6, c2, c3 \} \)

### Select B,C from R where \( A < 4 \)

- \( M_{AC} \)
- \( \text{Result: } \{ c4, c2, c3 \} \)

### Incorrect alignment

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>b5</td>
<td>c3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>b6</td>
<td>c4</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c6</td>
<td></td>
</tr>
</tbody>
</table>

### Correct alignment

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>b5</td>
<td>c3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>b6</td>
<td>c4</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c6</td>
<td></td>
</tr>
</tbody>
</table>

### Sideways Cracking, SIGMOD 09
### Initial state

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>c7</td>
</tr>
</tbody>
</table>

### select B from R where A<3

#### Crack A<3

<table>
<thead>
<tr>
<th>v&lt;3</th>
<th>v&gt;=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b4</td>
</tr>
<tr>
<td>2</td>
<td>b3</td>
</tr>
<tr>
<td>4</td>
<td>b2</td>
</tr>
<tr>
<td>7</td>
<td>b1</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
</tr>
</tbody>
</table>

**MA_B**

Result: b4 b3

### select C from R where A<5

#### Crack A<5

<table>
<thead>
<tr>
<th>v&lt;5</th>
<th>v&gt;=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>c6</td>
</tr>
<tr>
<td>4</td>
<td>c2</td>
</tr>
<tr>
<td>1</td>
<td>c3</td>
</tr>
<tr>
<td>2</td>
<td>c4</td>
</tr>
<tr>
<td>8</td>
<td>c5</td>
</tr>
<tr>
<td>7</td>
<td>c1</td>
</tr>
<tr>
<td>6</td>
<td>c7</td>
</tr>
</tbody>
</table>

**MA_C**

Result: c6 c2 c3 c4

### select B,C from R where A<4

#### Crack A<4

<table>
<thead>
<tr>
<th>v&lt;4</th>
<th>v&gt;=4</th>
<th>v&gt;=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>b4</td>
<td>b3</td>
</tr>
<tr>
<td>1</td>
<td>b3</td>
<td>b6</td>
</tr>
<tr>
<td>3</td>
<td>b6</td>
<td>b2</td>
</tr>
<tr>
<td>8</td>
<td>b5</td>
<td>b1</td>
</tr>
<tr>
<td>7</td>
<td>b1</td>
<td>b4</td>
</tr>
<tr>
<td>6</td>
<td>b7</td>
<td>b5</td>
</tr>
</tbody>
</table>

**MA_B**

Wrong alignment

**MA_C**

Correct alignment

---

Stratos Idreos

Sideways Cracking, SIGMOD 09
replace tuple reconstruction with cracking

Wrong alignment

Correct alignment

Sideways Cracking, SIGMOD 09
wider maps...but too many combinations
maps of different maps sets lead to alignment problems

select D from R where 3<A<10 and 4<B<8 and 1<C<7
wider maps...but too many combinations
maps of different maps sets lead to alignment problems
use a single map set and exploit bit-vectors

select D from R where 3<A<10 and 4<B<8 and 1<C<7
select D from R where 3<A<10 and 4<B<8 and 1<C<7
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector
multi-selections

select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector

Align 3<A<10
Grab tail values
select D from R where 3<A<10 and 4<B<8 and 1<C<7

Crack 3<A<10
Analyze tail 4<B<8
Create bit vector

Align 3<A<10
Analyze tail 1<C<7
Refine bit vector

Align 3<A<10
Grab tail values

Use histogram-like info from maps to choose map set
TPC-H Query 15

MonetDB - Sel. Crack - MySQL
Presorted - Sid. Crack

Response time (milli secs)
Query sequence

Sideways Cracking, SIGMOD 09
TPC-H Query 15

Response time (milli secs)

Query sequence

normal MonetDB
selection cracking

MonetDB
Presorted
Sel. Crack
Sid. Crack
MySQL
Presorted
presorted MonetDB preparation cost 3-14 minutes per query
presorted MonetDB preparation cost 3-14 minutes per query

normal MonetDB

selection cracking

MonetDB with sideways cracking

Response time (milli secs)

Query sequence

TPC-H Query 15

Sel. Crack

Sid. Crack

MySQL

Presorted

MonetDB

Presorted

3-14 minutes per query
MonetDB • Presorted

Sel. Crack • Presorted

MySQL • Presorted

Response time (milli secs)

TPC-H Query 15

Query sequence

MonetDB with sideways cracking

selection cracking

presorted MonetDB preparation cost 3-14 minutes per query

presorted MonetDB

Sel. Crack

Sid. Crack

MySQL

normal MonetDB

Sideways Cracking, SIGMOD 09

Stratos Idreos
Presorted MonetDB preparation cost 3-14 minutes per query.
Presorted MonetDB preparation cost 3-14 minutes per query.

TPC-H Query 15

MonetDB

Presorted

Sel. Crack

Sid. Crack

MySQL

Presorted

Response time (milli secs)

Query sequence
Partial Sideways Cracking

Initial state

Select B from R where 9<A<=15

Select B from R where 9<A<13

Select B from R where 5=A<8

Select C from R where 8=A<15

Chunk map \( H(A,\text{id}) \)

Partial map \( MAB \)

Partial map \( MAC \)

Keys

<table>
<thead>
<tr>
<th>Keys</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>b7</td>
<td>c7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>b8</td>
<td>c8</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>b9</td>
<td>c9</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>b10</td>
<td>c10</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>b11</td>
<td>c11</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>b12</td>
<td>c12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>b13</td>
<td>c13</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>b14</td>
<td>c14</td>
</tr>
</tbody>
</table>

Select B from R where v<9

Select B from R where v>9

Select B from R where v>15

Select C from R where v<9

Select C from R where v>9

Select C from R where v>15

Keys

<table>
<thead>
<tr>
<th>Keys</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>b1</td>
<td>c1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>b2</td>
<td>c2</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>b3</td>
<td>c3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>b4</td>
<td>c4</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>b5</td>
<td>c5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>b6</td>
<td>c6</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>b7</td>
<td>c7</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>b8</td>
<td>c8</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>b9</td>
<td>c9</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>b10</td>
<td>c10</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>b11</td>
<td>c11</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>b12</td>
<td>c12</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>b13</td>
<td>c13</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>b14</td>
<td>c14</td>
</tr>
</tbody>
</table>

Initial state

Select B from R where 9<A<=15

Select B from R where v<=9

Select B from R where v>9

Select B from R where v>15

Select B from R where 9<A<13

Select B from R where v<=9

Select B from R where v>9

Select B from R where v>15

Select B from R where 5=A<8

Select B from R where v<5

Select B from R where v>5

Select B from R where v>8

Select B from R where v>15

Select C from R where v<8

Select C from R where v>8

Select C from R where v>15
crack

crack
no alignment

crack

no alignment

crack
concurrency control

problem: read queries become write queries! (?)

goal: be able to crack for multiple queries in parallel
write queries

traditional indexing

read queries

adaptive indexing
traditional indexing
change index contents and structure
write queries

adaptive indexing
only index structure changes
read queries
no need for traditional \textcolor{red}{locks} = too heavy

short term \textcolor{red}{latches} = fast and release quickly

\begin{itemize}
  \item change index contents and structure
  \item write queries
  \item traditional indexing
  \item only index structure changes
  \item read queries
  \item adaptive indexing
\end{itemize}
**traditional indexing**
- all or nothing
- change index contents and structure
- write queries

**adaptive indexing**
- incremental and optional
- only index structure changes
- read queries

Concurrency Control, PVLDB 12
Stratos Idreos

Concurrent Control, PVLDB 12
traditional indexing

- all or nothing
- change index contents and structure
- write queries

adaptive indexing

- incremental and optional
- only index structure changes
- read queries

Concurrency Control, PVLDB 12
all or nothing
change index contents and structure
write queries

traditional indexing

incremental and optional
only index structure changes
read queries

adaptive indexing
all or nothing

change index contents and structure

write queries

traditional indexing

incremental and optional

only index structure changes

read queries

adaptive indexing
traditional indexing

adaptive indexing

all or nothing

incremental and optional

change index contents and structure

only index structure changes

write queries

read queries

Concurrency Control, PVLDB 12

Stratos Idreos
all or nothing

change index contents and structure

write queries

traditional indexing

incremental and optional

only index structure changes

read queries

adaptive indexing

Concurrency Control, PVLDB 12
Stratos Idreos

traditional indexing

impact stable storage
all or nothing
change index contents and structure
write queries

stable storage optional
incremental and optional
only index structure changes
read queries

adaptive indexing

Concurrency Control, PVLDB 12
impact stable storage

all or nothing

change index contents and structure

write queries

traditional indexing

stable storage optional

incremental and optional

only index structure changes

read queries

adaptive indexing
**traditional indexing**

- need to serialize
- impact stable storage
- all or nothing
- change index contents and structure
- write queries

**adaptive indexing**

- can execute in any order
- stable storage optional
- incremental and optional
- only index structure changes
- read queries
fewer conflicts as we adapt

![Graph showing the relationship between conflicts and query sequence](chart.png)

- **Conflicts vs. Query Sequence**:
  - The graph illustrates the decrease in conflicts with an increasing query sequence.

- **Response Time vs. Query Sequence**:
  - The chart displays the response time in seconds against the query sequence (x1000).
  - Three methods are compared:
    - **Scan**: Shows a steady increase in response time.
    - **Crack**: Demonstrates a significant drop in response time.
    - **Full Index**: Indicates a moderate response time pattern.

---

**Concurrency Control, PVLDB 12**

Stratos Idreos
select min(C) from R where A<10 & B<20
select min(C) from R where A<10 & B<20
late reconstruction

```
select min(C) from R where A<10 & B<20
```

disk

memory

A  B  C  D

A<10  IDs
late reconstruction

select min(C) from R where A < 10 & B < 20

disk

memory

A B C D

A<10 IDs B
select min(C) from R where A<10 & B<20
late reconstruction

select min(C) from R where A<10 & B<20

disk

memory

A<10 IDs B B<20 IDs C
late reconstruction

select min(C) from R where A<10 & B<20

disk

memory

A B C D

A<10 IDs

B B<20 IDs

C minC

Concurrency Control, PVLDB 12
late reconstruction

select min(C) from R where A<10 & B<20

disk

memory

A<10 IDs B B<20 IDs C minC

column lock and release
as soon as an operator completes
select \([a,b]\)
need to latch only to be cracked pieces (max 2 per select)

select [a,b]
piece locking

avl-tree  

<table>
<thead>
<tr>
<th>crack column</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
piece locking

avl-tree  crack column  crack select
Concurrency Control, PVLDB 12
piece locking

avl-tree  crack column  crack select

wlock
piece locking

avl-tree  crack column
piece locking
piece locking

avl-tree

rack column

rlock

max

Concurrency Control, PVLDB 12
piece locking

avl-tree  crack column

max

rlock
piece locking

avl-tree \(ightarrow\) crack column

max

rlock
piece locking

avl-tree → crack column → max

rlock

Concurrent Control, PVLDB 12
piece locking

avl-tree  crack column

10
90
140
200
300
Stratos Idreos

avl-tree

10
90
140
200
300

crack column

65
230

crack select

piece locking
piece locking

avl-tree

crack column

wlock

wlock

10
90
140
200
300
65
230

crack select
Stratos Idreos

piece locking

avl-tree

crack column

wlock

10
90
140
200
300

r/wlock

wlock

65

230

crack select
Stratos Idreos

avl-tree

piece locking

crack column

wlock

q1, q2, q3... qn

wlock

230

65

10

90

140

200

300

HARVARD
School of Engineering and Applied Sciences

Stratos Idreos
piece locking
Stratos Idreos

piece locking

avl-tree

crack column

crack select

q1, q2, q3... qn

10

90

140

200

300

wlock

65

230

wlock
Rows: 100M
Query: select sum(A) from R where v1 < A1 < v2
Selectivity: 0.01%, Random, # of queries: 1024
Clients: 1-32, Machine: 4 cores

Adaptive indexing maintains its performance advantage

Concurrent execution

Concurrency Control
- Enabled
- Disabled

Total Time for 1024 Queries (secs)

Throughput over 1024 Queries (Queries/sec)

(Sequential Execution)
Query: select sum(A) from R where v1 < A1 < v2
Selectivity: 0.01%, Random, # of queries: 1024
Clients: 8, Machine: 4 cores

adaptive indexing maintains the adaptive behavior

adaptive behavior also for conflicts
data systems today
allow us to answer queries fast

data systems tomorrow
should allow us to find fast which queries to ask
Overview of Data Exploration Techniques
S. Idreos, O. Papaemmanouil, and S. Chaudhuri
ACM SIGMOD International Conference on Management of Data, 2015
(explore citations)
a semester of quizzes and brainstorming

NoSQL key-value stores
& research with DASlab
(only for CS165 students or otherwise advanced students)
learn the concepts, not just the techniques
learn to adapt
it all starts with how we store the data
yes you can do research

The End

DASLab productions 2017

/*test server until Dec 13
labs continue until Dec 8*/