3 Ideas for Big Data Exploration

Stratos Idreos
CWI, INS-1, Amsterdam
data is everywhere
daily data

years
**Eric Schmidt**: Every two days we create as much data as much we did from dawn of humanity to 2003.
**Eric Schmidt**: Every two days we create as much data as much we did from dawn of humanity to 2003
Found It!!!

Congratulations, it only took you 65,298 seconds.
not always sure what we are looking for (until we find it)
data exploration

not always sure what we are looking for (until we find it)
database systems great...
declarative processing, back-end to numerous apps
database systems great...
declarative processing, back-end to numerous apps

but databases are too heavy and blind!
database systems great...
declarative processing, back-end to numerous apps

but databases are too heavy and blind!
database systems great...
declarative processing, back-end to numerous apps

but databases are too heavy and blind!
database systems great...
declarative processing, back-end to numerous apps

but databases are too heavy and blind!
**database systems** great...
declarative processing, back-end to numerous apps

but databases are too heavy and blind!
expert users - idle time - workload knowledge

but databases are too heavy and blind!
systems tailored for data exploration
no workload knowledge

systems tailored for data exploration
no workload knowledge

no installation steps

systems tailored for data exploration
no workload knowledge

no installation steps

quick response times

systems tailored for data exploration
no workload knowledge

no installation steps

quick response times

minimize data-to-query time

systems tailored for data exploration
adaptive indexing

ACM SIGMOD Jim Gray Diss. Award - Cor Baayen Award

7 years, 7 papers

3 ideas
adaptive indexing

ACM SIGMOD Jim Gray Diss. Award - Cor Baayen Award

7 years, 7 papers

adaptive loading

In use in Hadapt and Tableau

3 years, 3 papers
**3 ideas**

**Adaptive Indexing**
- ACM SIGMOD Jim Gray Diss. Award - Cor Baayen Award
- 7 years, 7 papers

**Adaptive Loading**
- In use in Hadapt and Tableau
- 3 years, 3 papers

**dbTouch**
- VENI 2012
- 1 year, 1 paper
3 ideas

**adaptive indexing**

ACM SIGMOD Jim Gray Diss. Award - Cor Baayen Award

7 years, 7 papers

**adaptive loading**

In use in Hadapt and Tableau

3 years, 3 papers

**dbTouch**

VENI 2012

1 year, 1 paper

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

load  index  query

tune= create proper indexes offline

performance 10-100X
indexing

tune = create proper indexes offline
performance 10-100X

but it depends on workload!

which indices to build?
on which data parts?
and when to build them?
load index query

sample workload

timeline
load index query

timeline

sample workload analyze
timeline

sample workload   analyze   create indices

load   index   query
timeline

load index query

sample workload analyze create indices query

timeline
load

index

query

timeline

sample workload  analyze  create indices  query

complex and time consuming process
human administrators + auto-tuning tools

complex and time consuming process

sample workload → analyze → create indices → query
what can go wrong?
what can go wrong?

not enough idle time to finish proper tuning
what can go wrong?

not enough idle time to finish proper tuning

by the time we finish tuning, the workload changes
database cracking
database cracking
remove all tuning steps and need for human input
incremental, adaptive, partial indexing
database cracking

remove all tuning steps and need for human input

incremental, adaptive, partial indexing
**database cracking**
remove all tuning steps and need for human input
incremental, adaptive, partial indexing
database cracking
remove all tuning steps and need for human input
incremental, adaptive, partial indexing
every query is treated as an advice on how data should be stored

database cracking
remove all tuning steps and need for human input
incremental, adaptive, partial indexing

Continuous Improvement
cracking example

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

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

Q2:
select * 
from R
where R.A > 7
and R.A <= 16
### cracking example

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

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

**Column A**

<table>
<thead>
<tr>
<th>13</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td>14</td>
</tr>
</tbody>
</table>

**Cracker column of A**

<table>
<thead>
<tr>
<th>4</th>
<th>9</th>
<th>7</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>11</td>
<td>16</td>
</tr>
</tbody>
</table>

**Pieces**

- **Piece 1:** A <= 10
- **Piece 2:** 10 < A < 14
- **Piece 3:** 14 <= A
cracking example

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

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

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

Piece 1: A <= 7
Piece 2: 7 < A <= 10
Piece 3: 10 < A < 14
Piece 4: 14 <= A <= 16
cracking example

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

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

Piece 1: A <= 7
Piece 2: 7 < A <= 10
Piece 3: 10 < A < 14
Piece 4: 14 <= A <= 16
cracking example

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

<table>
<thead>
<tr>
<th>Column A</th>
<th>Cracker column of A</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Piece 1: A &lt;= 10</td>
</tr>
</tbody>
</table>

Piece 1: 16 < A
Piece 2: 10 < A < 14
Piece 3: 14 <= A
cracking example

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

Column A

<table>
<thead>
<tr>
<th>13</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Cracker column of A

<table>
<thead>
<tr>
<th>4</th>
<th>9</th>
<th>2</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Piece 1: A <= 10

Piece 2: 10 < A < 14

Piece 3: 14 <= A

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

<table>
<thead>
<tr>
<th>Column A</th>
<th>Cracker column of A</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>Piece 1: A &lt;= 10</td>
</tr>
<tr>
<td>3</td>
<td>Piece 2: 10 &lt; A &lt; 14</td>
</tr>
<tr>
<td>14</td>
<td>Piece 3: 14 &lt;= A</td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

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

Piece 1: 10 < A < 14
Piece 2: 10 < A < 14
Piece 3: 14 <= A
cracking example

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

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

Result tuples

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

Cracker column of A
| 4 | 9 | 2 | 7 | 1 | 3 | 8 | 6 |

Piece 1: A <= 10
Piece 2: 10 < A < 14
Piece 3: 14 <= A

Database Cracking CIDR 2007
cracking example

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

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

Dynamically/on-the-fly within the select-operator
gain knowledge on how data is organized

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

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

Dynamically/on-the-fly within the select-operator
cracking example

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

piece 1: A <= 7
piece 2: 7 < A <= 10
piece 3: 10 < A < 14
piece 4: 14 <= A <= 16

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Column A

Cracker column of A

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Column A
<table>
<thead>
<tr>
<th>13</th>
<th>16</th>
<th>4</th>
<th>9</th>
<th>12</th>
<th>7</th>
<th>1</th>
<th>19</th>
<th>3</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>10 &lt; A &lt; 14</td>
<td>1</td>
<td>19</td>
<td>3</td>
<td>14</td>
<td>11</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Cracker column of A

Piece 1: A <= 10
| 4  | 9  | 2  | 7  | 1  | 3  | 8  | 6  |

Piece 2: 10 < A < 14
| 4  | 9  | 2  | 7  | 1  | 3  | 8  | 6  |

Piece 3: 10 < A < 14
| 4  | 9  | 2  | 7  | 1  | 3  | 8  | 6  |

Piece 4: 14 <= A <= 16
| 4  | 9  | 2  | 7  | 1  | 3  | 8  | 6  |

Piece 5: 14 <= A <= 16
| 4  | 9  | 2  | 7  | 1  | 3  | 8  | 6  |

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
### Cracking Example

#### Q1:

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

#### Q2:

```
select *
from R
where R.A > 7
and R.A <= 16
```

---

#### Column A

<table>
<thead>
<tr>
<th>13</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
</tr>
</tbody>
</table>

#### Cracker column of A

- **Piece 1**: $A \leq 10$
- **Piece 2**: $7 < A \leq 10$
- **Piece 3**: $10 < A < 14$
- **Piece 4**: $14 \leq A \leq 16$
- **Piece 5**: $16 < A$

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
cracking example

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

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

Dynamically/on-the-fly within the select-operator
Q1: select * from R where R.A > 10 and R.A < 14
Q2: select * from R where R.A > 7 and R.A <= 16

Dynamically/on-the-fly within the select-operator
### Q1

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

### Q2

```
select *
from R
where R.A > 7
and R.A <= 16
```

---

#### Result tuples

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Dynamically/on-the-fly within the select-operator**
TPC-H

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

Response time (milli secs)

Query sequence

TPC-H Query 15

Sideways Cracking, SIGMOD 09
TPC-H

MonetDB  Sel. Crack  MySQL  Presorted
Presorted  Sid. Crack  Presorted

Response time (milli secs)

Query sequence

Normal MonetDB
selection cracking
TPC-H

Preparation cost 3-14 minutes

Normal MonetDB

Response time (milli secs)

Selection cracking

Query sequence
TPC-H

MonetDB • Sel. Crack • MySQL
Presorted • Sid. Crack • Presorted

Response time (milli secs)

Query sequence

764 TPC-H Query 15
420

Normal MonetDB
selection cracking
MonetDB with sideways cracking

Presorted MonetDB
Preparation cost 3-14 minutes
TPC-H

MonetDB  
Presorted

MySQL  
Presorted

Selection cracking

MonetDB with sideways cracking

Presorted MonetDB  
Preparation cost 3-14 minutes

TPC-H Query 15

Response time (milli secs)

Query sequence
TPC-H

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

Preparation cost: 3-14 minutes

Response time (milli secs)

Query sequence

Normal MonetDB
Selection cracking
MonetDB with sideways cracking
TPC-H

Presorted MonetDB
Preparation cost 3-14 minutes

MonetDB with sideways cracking

Selection cracking

Normal MonetDB

MySQL

Presorted

Sel. Crack

Sid. Crack

Response time (milli secs)

Query sequence
skyserver experiments

cracking answers 160,000 queries while full indexing is still half way creating the index
cracking databases

updates (SIGMOD07)

multiple columns (SIGMOD09)

storage restrictions (SIGMOD09)

benchmarking (TPCTC10)

algorithms (PVLDB11)

concurrency control (PVLDB12)

robustness (PVLDB12)
cracking databases

updates (SIGMOD07)

multiple columns (SIGMOD09)

storage restrictions (SIGMOD09)

benchmarking (TPCTC10)

algorithms (PVLDB11)

concurrency control (PVLDB12)

robustness (PVLDB12)

joins

disk based

multi-dimensional

holistic indexing

optimization

row-stores

multi-cores

aggregations

vectorwised
copy data inside the database

database now has full control
**loading**

- **load**
- **index**
- **query**

**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

single query cost (secs)

DB
Awk
database vs. unix tools

1 file, 4 attributes, 1 billion tuples

single query cost (secs)

break down db cost

- Loading: 93%
- Query Processing: 7%
database vs. unix tools

1 file, 4 attributes, 1 billion tuples

break down db cost
- Loading: 7%
- Query Processing: 93%

loading is a major bottleneck
1 file, 4 attributes, 1 billion tuples

break down db cost
- Loading: 7%
- Query Processing: 93%

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
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

db
zero loading cost
access raw data adaptively on-the-fly
query plan

scan

files cache

access raw data adaptively on-the-fly

zero loading cost
selective parsing - file indexing
file splitting - online statistics

query plan

scan

files

access raw data adaptively on-the-fly

cache

zero loading cost
NoDB SIGMOD 2012

Execution Time (sec)

MySQL
CSV Engine
MySQL
DBMS X
DBMS X w/ external files
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
reducing data-to-query time
querying
querying

load  index  query

declarative SQL interface
querying

declarative SQL interface
correct and complete answers
querying

load  index  query

declarative SQL interface

correct and complete answers
querying

complex and slow - not fit for exploration

declarative SQL interface

correct and complete answers
dbTouch

data
no expert knowledge - no set-up costs
interactive mode fit for exploration
challenge

design database systems tailored for touch exploration
challenge

design database systems tailored for touch exploration

user drives query processing actions
3 Ideas for Big Data Exploration

Stratos Idreos
CWI, INS-1, Amsterdam
3 Ideas for Big Data Exploration

Stratos Idreos
CWI, INS-1, Amsterdam

systems tailored for exploration
more in INS-I

data vaults  sciborg  cyclotron

sciql

holistic indexing

datacell  charles

hardware-conscious
more in INS-I

data vaults      sciborg      cyclotron

sciql             holistic indexing

Thank you!

datacell      charles

hardware-conscious