Database Cracking
Languages and Runtimes for Big Data
Row Stores

Traditional DB: Lay out data on disk in rows
Column Stores

Columnar DB: Lay out data on disk in columns
## Column Stores

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Row1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Row2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>Row3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Row4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B1</td>
<td>Row1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>Row2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td>Row3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td>Row4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>Row1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>Row2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>Row3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Row4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>Row1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>Row2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>Row3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>Row4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Store with Row ID to recover original table
Why use a Column Store?
Immediate Data Access

**Problem:** Data is initially unsorted

**Query:** Find all rows where $100 < A \leq 200$

What is the fastest way to answer this query?
Immediate Data Access

**Problem:** Data is initially unsorted

What if you get 2 queries?

... 3 queries?

... 100 queries?
Immediate Data Access

**Problem:** Data is initially unsorted

**Strategy 1:** Index the data then run queries

First few queries are much slower (upfront indexing cost)

**Strategy 2:** Linear scans over the data

Last few queries are much slower (no indexing!)
Immediate Data Access

**Problem:** Data is initially unsorted

**Strategy 3:** Index *while* you run queries!

Re-use compute effort of scans.
Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: > 4 and ≤ 4

```
Low

0 9 3 5 4 8 7 1 2 6
```

**Current**

Query 1: Find 4 < X ≤ 7
**Cracking**

**Step 1:** Split into 2 bins: \( > 4 \) and \( \leq 4 \)

Query 1: Find \( 4 < X \leq 7 \)
Cracking

**Step 1:** Split into 2 bins: $> 4$ and $\leq 4$

Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: > 4 and ≤ 4

Query 1: Find 4 < X ≤ 7
**Cracking**

**Step 1:** Split into 2 bins: $> 4$ and $\leq 4$

Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: $> 4$ and $\leq 4$

Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: > 4 and ≤ 4

Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: $> 4$ and $\leq 4$

Query 1: Find $4 < X \leq 7$
Cracking

**Step 1:** Split into 2 bins: > 4 and ≤ 4

![Index Array](image)

Low

Query 1: Find 4 < X ≤ 7
Cracking

**Step 1:** Split into 2 bins: $> 4$ and $\leq 4$

```
0 3 4 1 9 8 7 5 2 6
```

**Query 1:** Find $4 < X \leq 7$
CSE 662 - Database Languages & Runtimes

Cracking

**Step 1:** Split into 2 bins: > 4 and ≤ 4

Low

Current

0 3 4 1 2 8 7 5 9 6

Query 1: Find 4 < X ≤ 7
Cracking

**Step 2:** Split into 2 bins: > 7 and ≤ 7

![number_list](0 3 4 1 2 8 7 5 9 6)

**Query 1:** Find 4 < X ≤ 7
Cracking

**Step 2:** Split into 2 bins: > 7 and ≤ 7

Query 1: Find 4 < X ≤ 7
Cracking

**Step 2:** Split into 2 bins: > 7 and ≤ 7

<table>
<thead>
<tr>
<th>0</th>
<th>3</th>
<th>4</th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>8</th>
<th>5</th>
<th>9</th>
<th>6</th>
</tr>
</thead>
</table>

Current

**Query 1:** Find 4 < X ≤ 7
Cracking

**Step 2:** Split into 2 bins: > 7 and ≤ 7

0 3 4 1 2 7 5 6 9 8

Current

**Query 1:** Find $4 < X \leq 7$
Cracking

**Step 2:** Split into 2 bins: > 7 and ≤ 7

Low  High

| 0 | 3 | 4 | 1 | 2 | 7 | 5 | 6 | 9 | 8 |

Result

**Query 1:** Find 4 < X ≤ 7
Query 1: Find $4 < X \leq 7$
Query 2: Find $2 < X \leq 5$
**Query 2:** Find $2 < X \leq 5$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
3-Way Cracking

Query 1: Find $4 < X \leq 7$
Does cracking work with a row-oriented database?
Discussion Questions…

How would one crack a multi-attribute index?

(e.g., a spatial index?)
Discussion Questions…

Can updates be performed efficiently on a cracker index?
Discussion Questions...

Can updates be performed efficiently on a cracker index?

What constraints are required?
Discussion Questions…

What applications would cracking work well on?
What applications would cracking work poorly on?
Discussion Questions...

Upfront Indexing vs Sequential Scan vs Cracking...

Where is the cutoff?