**Indexing for Analytical Workloads**

<table>
<thead>
<tr>
<th>Column A</th>
<th>A=10</th>
<th>A=20</th>
<th>A=30</th>
<th>A=40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Compact representation of query result**
- **Query result is readily available**

**Bitvectors**
- Can leverage fast Boolean operators
- Bitwise operations over bitvectors
- More space-efficient for low domain cardinality
- Space-inefficient for larger domains
- Addressed by bitwise encoding/compression
- Core idea: run-length encoding in prior work
- Updating encoded bitvectors is very inefficient

**Goal:** Bitmap Indexing for Reads & Updates

**Existence Bitvector is the bottleneck**

**Update Conscious Bitmaps (UCB), SSDBM 2007**
- **deletes** by invalidation
- **reads**: bitwise AND with EB
- **updates**: delete-then-append

**Challenge:** read cost increases with updates

**UpBit Design & Operations**

**Bottleneck 1:** single auxiliary bitvector
- **Design Element 1:** multiple auxiliary bitvectors to distribute the cost
- **Design Element 2:** efficient random access in bitvectors

**Bottleneck 2:** repetitive bitwise operations
- **Design Element 1:** update bitvectors initialized to 0s
  - Current value is the XOR of every update flips a bit
- **Design Element 2:** fence pointers allow access to the compressed word at a known position of the uncompressed bitvector
- **Design Element 3:** query-driven merging
  - Use past bitwise operations to merge updates

**Scalable Updates and Efficient Reads with UpBit**

**15-115x faster updates**

- n=100M tuples, d=100 domain values, 10% updates (varying % of updates)

**Competitive reads**

- 3x faster than UCB
- 8% overhead over read-optimized

**More in the paper ...**

- Multi-threaded support for reads and updates
- Impact of data skew
- Experiments with scientific data
- Full query analysis (including TPCH)

**Tuning:** merging and fence pointers

- Merge every 10-20 updates
- Fence pointers every 10^8-10^10 bits

**Scalable Updates**

**UpBit vs. Scan**

- n = 1B, d = 1000 distinct domain values (range)
- n = 1B, d vary for equality: 1000, 100, 10, 1