Enhancing Bitmap Indices

Guadalupe Canahuate
The Ohio State University

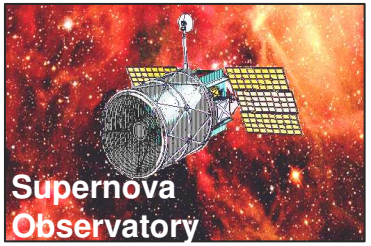
Advisor: Hakan Ferhatosmanoglu

Introduction

- Bitmap indices:
  - Widely used in data warehouses and read-only domains
  - Implemented in commercial DBMS such as Oracle, Informix and Sybase
  - Extremely efficient for execution of point and range queries

Applications

- Data warehouses
- Scientific data
  - **High-energy physics**: Simulations are continuously run, and notable events are stored with all the details.
  - **Climate modeling**: sensor data.
  - **Astro-physics**: telescopes devoted for observations.
- Visualization applications



Bitmap Indices

- Data is quantized into b categories using b bits
- Each tuple is encoded based on which category its attribute belongs to:
  - Bitmap Equality Encoding (BEE) (Projection Index, Binning): suited for point queries
  - Bitmap Range Encoding (BRE): suited for range queries

Value	EE			RE		
	1	2	3	1	2	3
1	1	0	0	1	1	1
2	0	1	0	0	1	1
3	0	0	1	0	0	1

- Fast Bitwise Operations over bit vectors (AND, OR, NOT, XOR)

Bitmap Compression

- Run-length encoders
  - Runs of 0s (0, run-count)
  - Byte-aligned Bitmap Code (BBC) [Oracle '94]
  - Word-Aligned Hybrid Code (WAH) [LBNL '02]

  - No need to decompress for query processing
  - Full scan of the bitmap is needed

Our Goal

- Enhance bitmap indices:
  - Reduce Index Size (ICDE'05, TKDE sub)
    - Improve compression
  - Direct access over compressed bitmaps (VLDB'06)
    - Bitmap Hashing
  - Handle Missing Data (EDBT'06)
    - Query support / semantics
  - Handle data updates (SSDBM'07)
    - Appends of new data

Our Goal

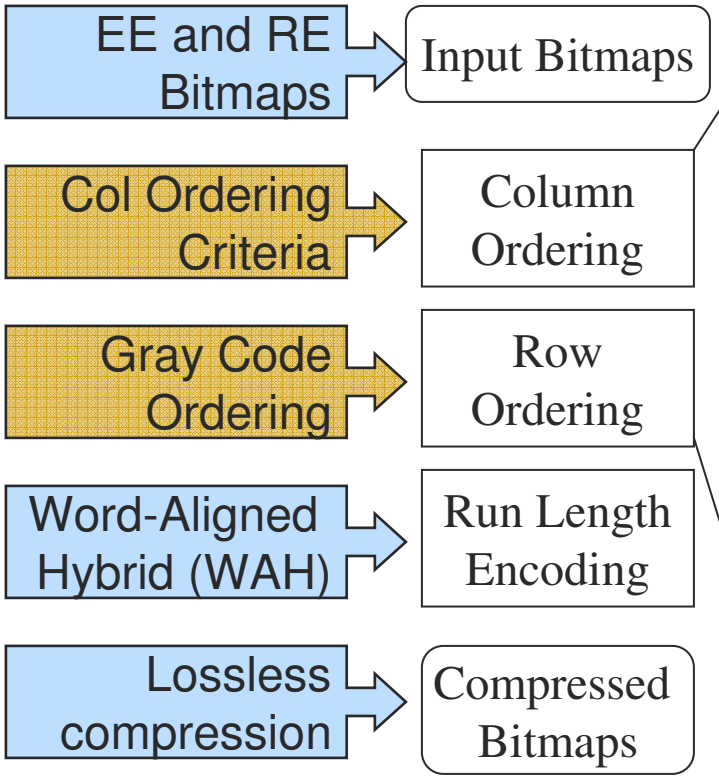
- Enhance bitmap indices:
  - Reduce Index Size (ICDE'05, TKDE sub)
    - Improve compression
  - Direct access over compressed bitmaps (VLDB'06)
    - Bitmap Hashing
  - Handle Missing Data (EDBT'06)
    - Query support / semantics
  - Handle data updates (SSDBM'07)
    - Appends of new data

Improving Compression by Data Reorganization

- NP-Complete
- Most TSP heuristics are ineffective
- Minimize the hamming distance of adjacent numbers
- Gray-code: A space filling-curve for hamming space
- A scalable, in-place algorithm to gray-sort the tuples

Submitted to TKDE

Reordering Framework



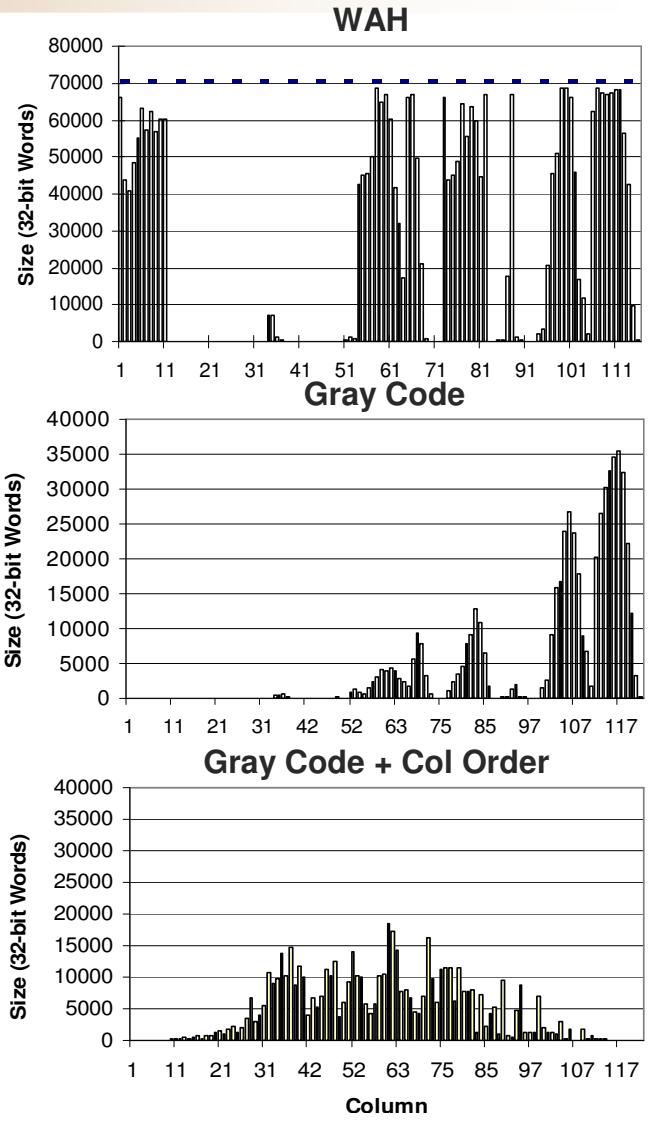

```
Reorder (A, start, end, b)
 1: i     start
 2: j     end
 3: while i < j do
 4: Decrement j until S(j,Cols[b]) = 0
 5:  Increment i until S(i,Cols[b]) = 1
 6:  if i < j then
 7:    Swap the ith and jth tuples on the table
 8:  end if
 9: end while
10: if b < no_of_bits then
11:  Cols[b+1] = nextCol();
12:  Reorder (A, start, j, b+1)
13: Reorder (A, j+1, end, b+1)
14: Reverse (j+1, end)
15: end if
```

Column Ordering Criteria

- Using Bitmap Data
  - Number of Set Bits
    - Static: Most 1s Asc/Desc
    - Dynamic: Most 1s in the Longest Run
  - Compressibility
    - Static: $C_i = \left| \frac{n}{2} - s_i \right|$
    - Dynamic: Weighted Sum of Compressibility over the Gray code segments
- Using Query Workloads
  - Most frequent accessed columns first

Experimental Results

- 2-10x improvement in compression over already compressed bitmaps
- 4-7x improvement in query execution time



Our Goal

- Enhance bitmap indices:
  - Reduce Index Size (ICDE'05, TKDE sub)
    - Improve compression
  - Direct access over compressed bitmaps (VLDB'06)
    - Bitmap Hashing
  - Handle Missing Data (EDBT'06)
    - Query support / semantics
  - Handle data updates (SSDBM'07)
    - Appends of new data

Compression and Direct Access

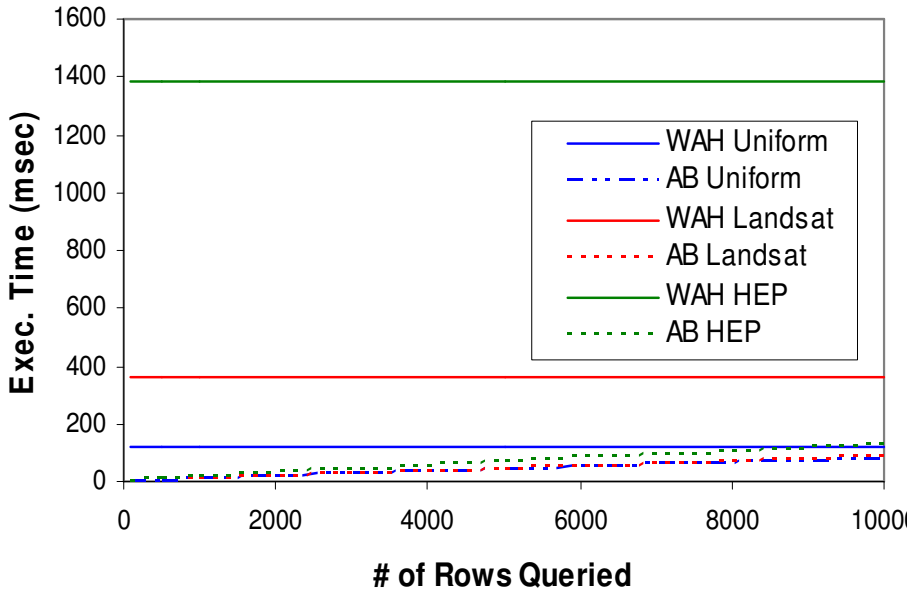
- Bitmaps need to be compressed:
  - Row numbers do not longer correspond to the bit position in the bitmap
- Cannot pin-point a row
  - Need to scan the bitmap
- Enable direct access over the compressed bitmaps
- Trade-off accuracy vs. efficiency
  - No false negatives

Direct Access Over Compressed Bitmaps

- Hashing/Bloom Filters
  - A 2^m bit array indexed using k independent hash functions
  - False positives can happen, but false negatives cannot
  - Only the set bits are inserted into the AB
  - Three levels of encoding:
    - Per table, per attribute, per bitmap column
  - Precision estimation

Direct Access Over Compressed Bitmaps

- Always set the array size to be smaller than WAH bitmap
- Execution time depends on the number of ***rows queried***, not in the size of the bitmaps



- For queries over less than ~15% of the rows, execution time is up to 3 orders of magnitude faster than WAH

Our Goal

- Enhance bitmap indices:
  - Reduce Index Size (ICDE'05, TKDE sub)
    - Improve compression
  - Direct access over compressed bitmaps (VLDB'06)
    - Bitmap Hashing
  - Handle Missing Data (EDBT'06)
    - Query support / semantics
  - Handle data updates (SSDBM'07)
    - Appends of new data

Handling missing data

- Incomplete databases are common in all research and industry domains
- Index performance degrades in the presence of missing data
- Missing data map to a distinguished category
- Query Semantics:
  - Missing is a match
    - In a patient database retrieve the records whose first name is “John” and middle name “Paul”.
  - Missing is not a match
    - In a census database retrieve the interviewee that answered “C” in question 4 and “A” in question 8

Handling Missing Data

		Equality Encoded						Range Encoded					
Record	**Value**	$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{1,5}$	$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{1,5}$
1	5	**0**	0	0	0	0	1	**0**	0	0	0	0	1
2	2	**0**	0	1	0	0	0	**0**	0	1	1	1	1
3	3	**0**	0	0	1	0	0	**0**	0	0	1	1	1
4	missing	**1**	0	0	0	0	0	**1**	1	1	1	1	1
5	4	**0**	0	0	0	1	0	**0**	0	0	0	1	1
6	5	**0**	0	0	0	0	1	**0**	0	0	0	0	1
7	1	**0**	1	0	0	0	0	**0**	1	1	1	1	1
8	3	**0**	0	0	1	0	0	**0**	0	0	1	1	1
9	missing	**1**	0	0	0	0	0	**1**	1	1	1	1	1
10	2	**0**	0	1	0	0	0	**0**	0	1	1	1	1

Query Processing with Missing Data

Equality Encoded

$v_1 \le A_i \le v_2$ =

Missing is a Match

$$\begin{cases} (\bigcup_{j=v_1}^{v_2} B_{i,j}) \vee B_{i,0} & \text{if} \quad v_2 - v_1 \le \lfloor C_i/2 \rfloor \\ \overline{\bigcup_{j=1}^{v_1-1} B_{i,j} \vee \bigcup_{j=v_2+1}^{C_i} B_{i,j}} & \textit{otherwise} \end{cases}$$

Missing is not a Match

$$\begin{cases} (\bigcup_{j=v_1}^{v_2} B_{i,j}) & \text{if} \quad v_2 - v_1 \le \lfloor C_i/2 \rfloor \\ \overline{\bigcup_{j=1}^{v_1-1} B_{i,j} \vee \bigcup_{j=v_2+1}^{C_i} B_{i,j}} \oplus B_{i,0} & \textit{otherwise} \end{cases}$$

Range Encoded

$$\begin{cases} B_{i,1} & \textit{if} \quad v_2 = v_1 = 1 \\ B_{i,v_1} \oplus B_{i,v_1-1} \vee B_{i,0} & \textit{if} \quad 1 < v_1 = v_2 < C_i \\ \overline{B_{i,C_i-1}} \vee B_{i,0} & \textit{if} \quad 1 < v_1 = v_2 = C_i \\ \overline{B_{i,v_1-1}} \vee B_{i,0} & \textit{if} \quad 1 < v_1 < C_i, \quad v_2 = C_i \\ B_{i,v_2} & \textit{if} \quad v_1 = 1, \quad 1 < v_2 < C_i \\ B_{i,v_2} \oplus B_{i,v_1-1} \vee B_{i,0} & \textit{otherwise} \end{cases}$$

$$\begin{cases} B_{i,1} \oplus B_{i,0} & \textit{if} \quad v_2 = v_1 = 1 \\ B_{i,v_1} \oplus B_{i,v_1-1} & \textit{if} \quad 1 < v_1 = v_2 < C_i \\ \overline{B_{i,C_i-1}} & \textit{if} \quad 1 < v_1 = v_2 = C_i \\ \overline{B_{i,v_1-1}} & \textit{if} \quad 1 < v_1 < C_i, \quad v_2 = C_i \\ B_{i,v_2} \oplus B_{i,0} & \textit{if} \quad v_1 = 1, \quad 1 < v_2 < C_i \\ B_{i,v_2} \oplus B_{i,v_1-1} & \textit{otherwise} \end{cases}$$

Our Goal

- Enhance bitmap indices:
  - Reduce Index Size (ICDE'05, TKDE sub)
    - Improve compression
  - Direct access over compressed bitmaps (VLDB'06)
    - Bitmap Hashing
  - Handle Missing Data (EDBT'06)
    - Query support / semantics
  - Handle data updates (SSDBM'07)
    - Appends of new data

Handling Data Updates

- Look for this project in the poster session… J

Conclusion

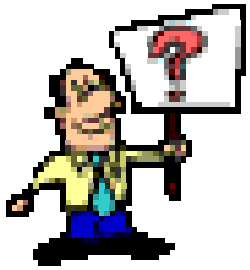
- Enhance bitmap indices:
  - Improve performance
    - Better compression
    - Direct access over compressed bitmap
  - Overcome limitations
    - Handle updates efficiently
    - Support more types of queries
- Extend the applicability of bitmaps to more domains that can benefit from bitmaps good performance

Ongoing Work

- Support new types of queries
  - Similarity Queries
- Integrate bitmaps to the coastal monitoring system (NSF Cyber Infrastructure Project)
- Bitmaps on emerging hardware technologies

Questions and Comments

- Thank you!



Email: canahuat@cse.ohio-state.edu