RLH: Bitmap Compression Technique Based on Run-Length and Huffman Encoding

Michał Stabno
Poznań University of Technology, Institute of Computing Science, Poland
QXL Poland
Michal.Stabno@allegro.pl
Robert Wrembel
Poznań University of Technology, Institute of Computing Science, Poland
Robert.Wrembel@cs.put.poznan.pl

Presentation Outline

- Bitmap index - concept and main characteristics
- Reducing size of bitmap index
- Run Length Huffman compression algorithm
- Run Length Huffman experimental evaluation
- Summary

Bitmap Index

- **Composed of bitmaps**
- **A bitmap is a vector of bits**
  - **every value from a domain has its own bitmap**
  - **the number of bits = the number of records**
  - **a given bit corresponds to a given record**

Clients

ID	sex
1	male
2	female
3	female
4	female
5	male
6	male
7	male
8	female
9	female
10	male
11	male
12	male
13	female
14	female
15	female
16	male
17	female
18	female
19	female

bitmap index

female	male
0	1
1	0
1	0
1	0
0	1
0	1
0	1
1	0
1	0
0	1
0	1
0	1
1	0
1	0
1	0
0	1
1	0
1	0
1	0

- **Basic characteristics**
  - **Efficient in answering equality and range queries**
  - **BI size depends on the cardinality of an indexed attribute**
    - **large for high cardinality attributes**

Reducing BI Size

- **Binning**
  - **[Kou00, SWS04, RSW05]**
- **Encoding**
  - **[WuBu98, ChIo99]**
- **Compressing**
  - **Byte-aligned Bitmap Compression [AnZi96]**
  - **Word-Aligned Hybrid [SWS02, WOS04]**
  - **Approximate Encoding [ACFT06]**
    - **may create false positives, additional verification**
  - **Reordering [JKCKV04, PTF05]**
    - **computationally very complex**
    - **reordering heuristics**

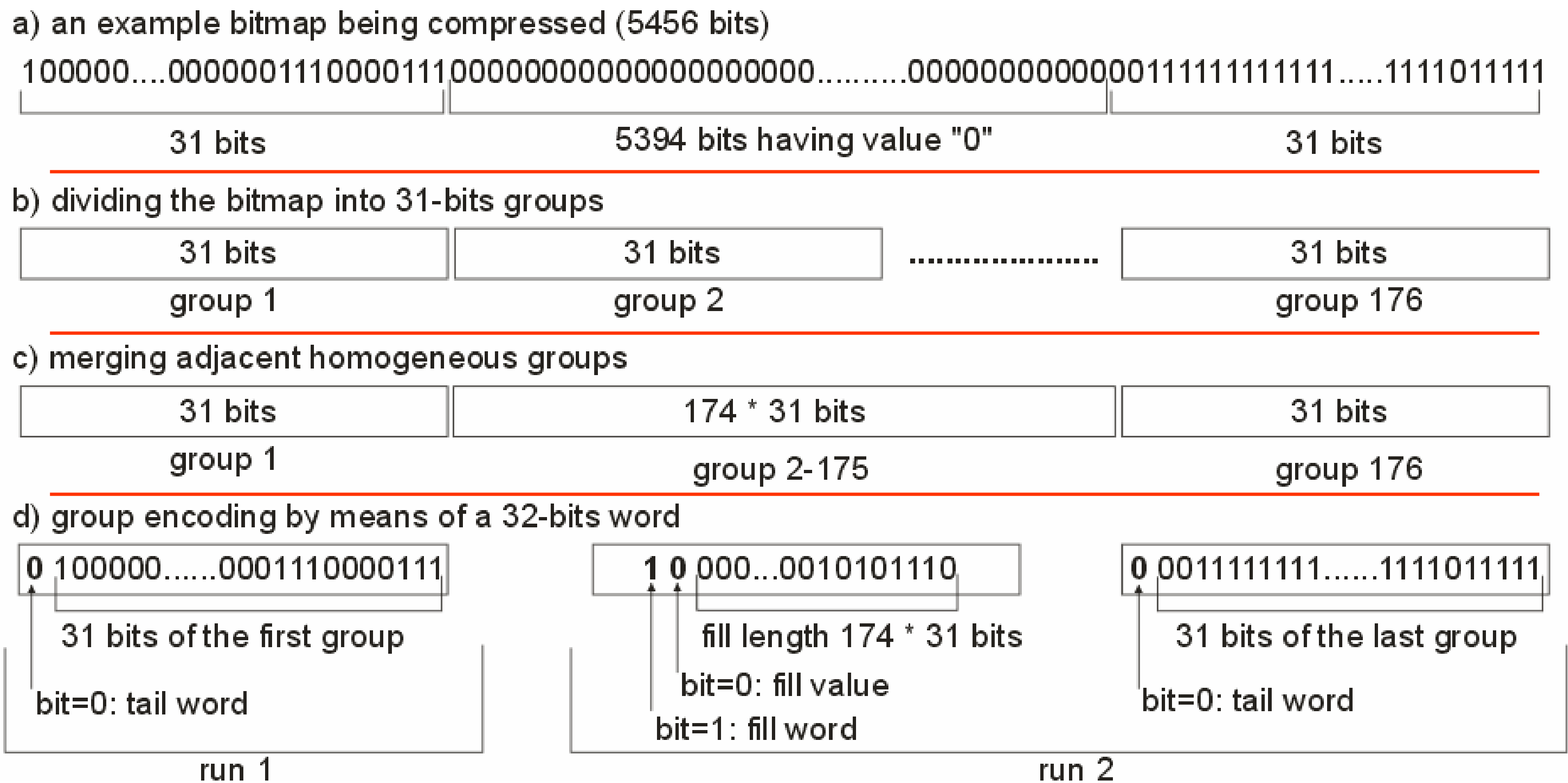
Bitmap Compression (1)

- Byte-aligned Bitmap Compression (BBC)
- Word-Aligned Hybrid (WAH)
- Based on the run-length encoding
  - homogeneous vectors of bits are replaced with a bit value (0 or 1) and the vector length
  - `0000000 1111111111 000` ⇨ 07 110 03
- BBC and WAH
  - a bitmap is divided into words
    - BBC uses 8-bit words
    - WAH uses 31-bit words

Bitmap Compression (2)

- WAH-compressed bitmaps are larger than BBC-compressed ones
- Operations on WAH-compressed bitmaps are faster than on BBC-compressed ones [SWS02, WOS02, WOS04]
- Our further focus is on comparing WAH to our approach

WAH (1)



- **Example taken from [StWu07]**

WAH (2)

1. **For low cardinality attributes bitmaps are dense**
   - **many homogeneous 31-bit words filled with 1**
2. **For high cardinality attributes bitmaps are sparse**
   - **many homogeneous 31-bit words filled with 0**
3. **For medium cardinality attributes**
   - **the number of homogeneous 31-bit words is lower**
   - **the compression ratio decreases**

➲ **A need for bitmap compression technique suitable for medium cardinality attributes**

Our Approach: RLH

- **RLH - the Run-Length Huffman Compression**
- **Based on**
  - **the Huffman encoding**
  - **a modified run-length encoding**

Huffman Encoding

➲ **Concept**

- **original symbols from a compressed file are replaced with bit strings**
- **the more frequently a given symbol appears in the compressed file the shorter bit string for representing the symbol**
- **encoded symbols and their corresponding bit strings are represented as a Huffman tree**
- **the Huffman tree is used for both compressing and decompressing**

RLH (1)

- **Modified run-length encoding**
  - **measures and encoded distances between bits of value 1**

1 0 0 3 0 3 0 0 1 0 0 0

1 0 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1

Clients

ID	sex
1	male
2	female
3	female
4	female
5	male
6	male
7	male
8	female
9	female
10	male
11	male
12	male
13	female
14	female
15	female
16	male
17	female
18	female
19	female

bitmap index

female	male
0	1
1	0
1	0
1	0
0	1
0	1
0	1
1	0
1	0
0	1
0	1
0	1
1	0
1	0
1	0
0	1
1	0
1	0
1	0

female: 100303001000

male: 030020033

- **Bitmaps encoded this way are input for the Huffman encoding**

RLH (2)

- **Huffman encoding**
  - **step1: computing frequencies of symbols (distances) in encoded bitmaps**

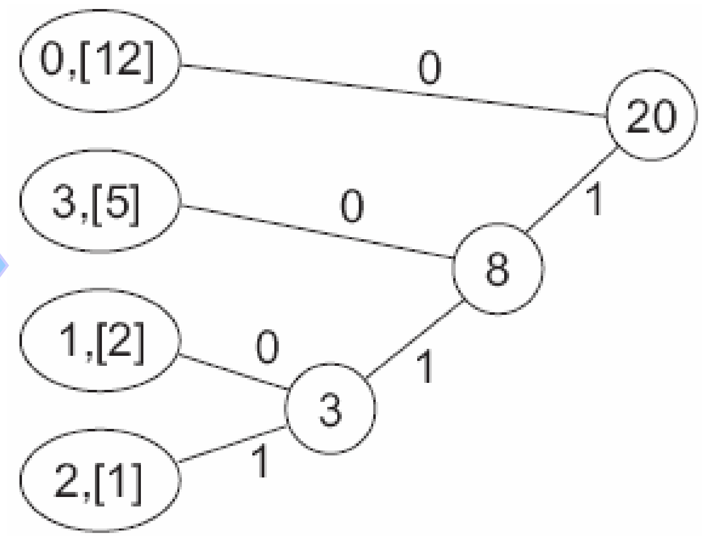
```
female: 100303001000
male:   030020033
```

distance	frequency
0	12
3	5
1	2
2	1

RLH (3)

- **Huffman encoding**
  - **step2: building a Huffman tree**

distance	frequency
0	12
3	5
1	2
2	1



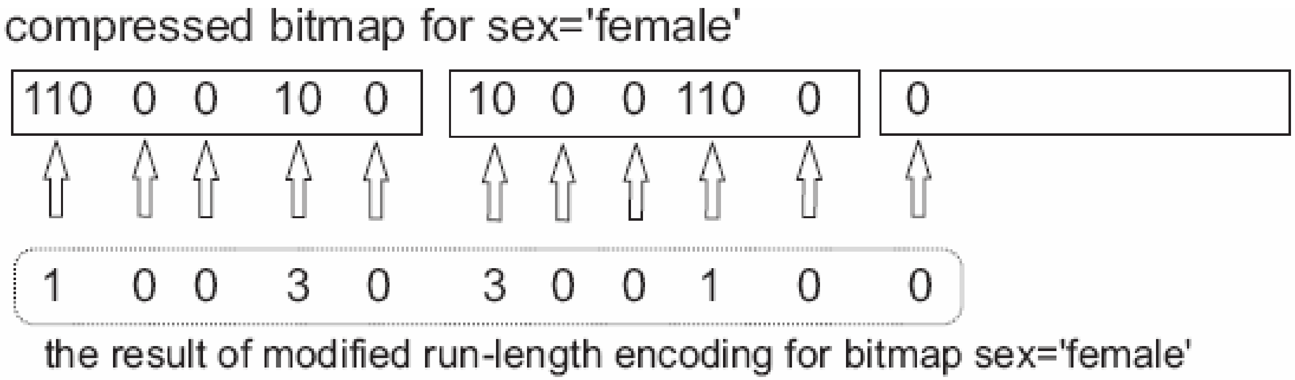
distance	code
0	0
3	10
1	110
2	111

- **an encoded symbol is represented by a path from the root to a leaf**

RLH (4)

- **Huffman encoding**
  - **step3: replacing distances with their Huffman codes**

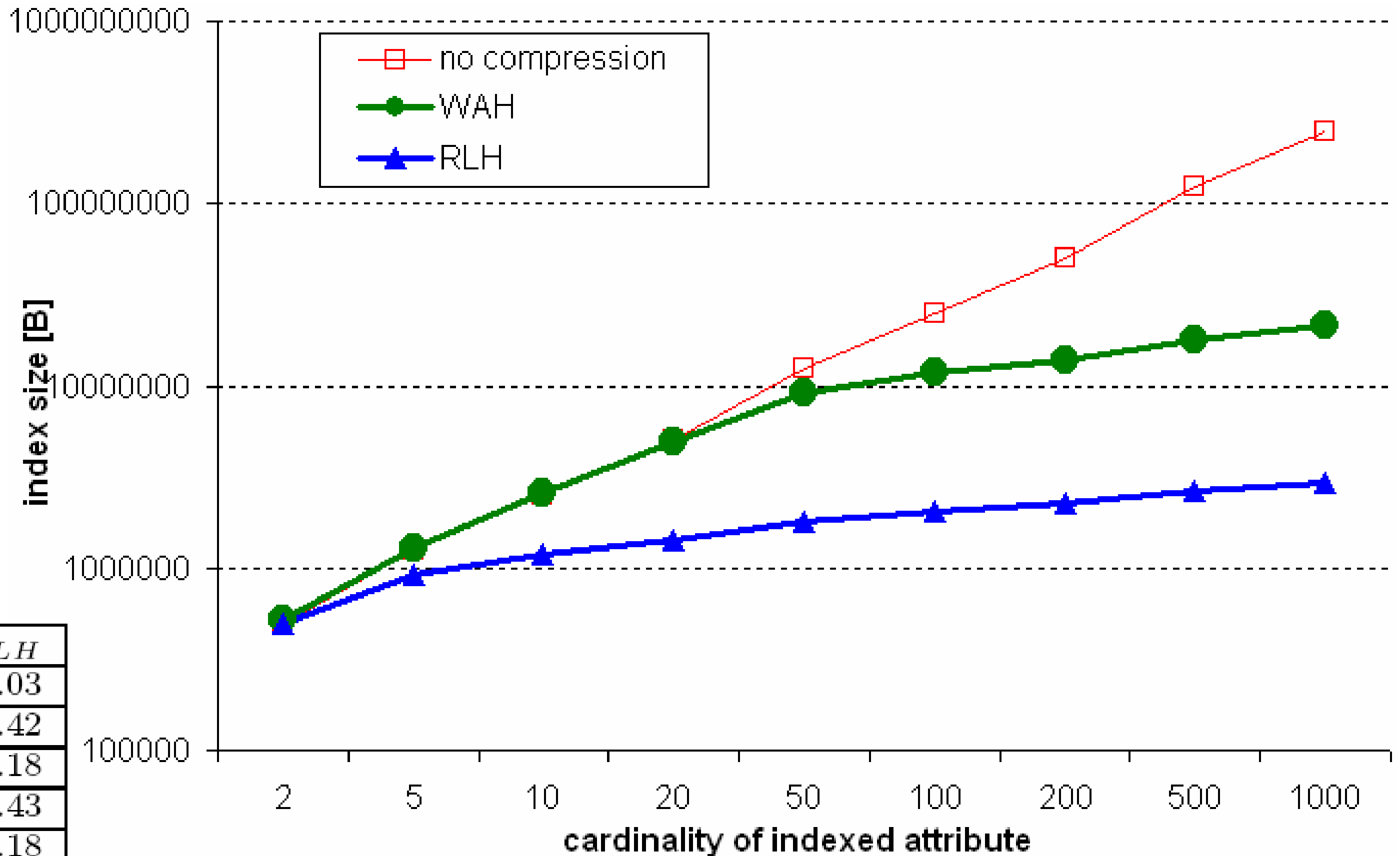
distance	code
0	0
3	10
1	110
2	111



Experimental Evaluation

- Comparing RLH, WAH, and uncompressed bitmaps with respect to
  - bitmap sizes
  - query response times
- Implementation in Java
  - data and bitmap indexes stored on disk in OS files
- Experiments run on
  - PC, AMD Athlon XP 2500+; 768 MB RAM; Windows XP
- Data
  - 2 000 000 indexed rows
  - indexed attribute of type integer
    - cardinality from 2 to 1000
    - randomly distributed values

WAH and RLH: index sizes



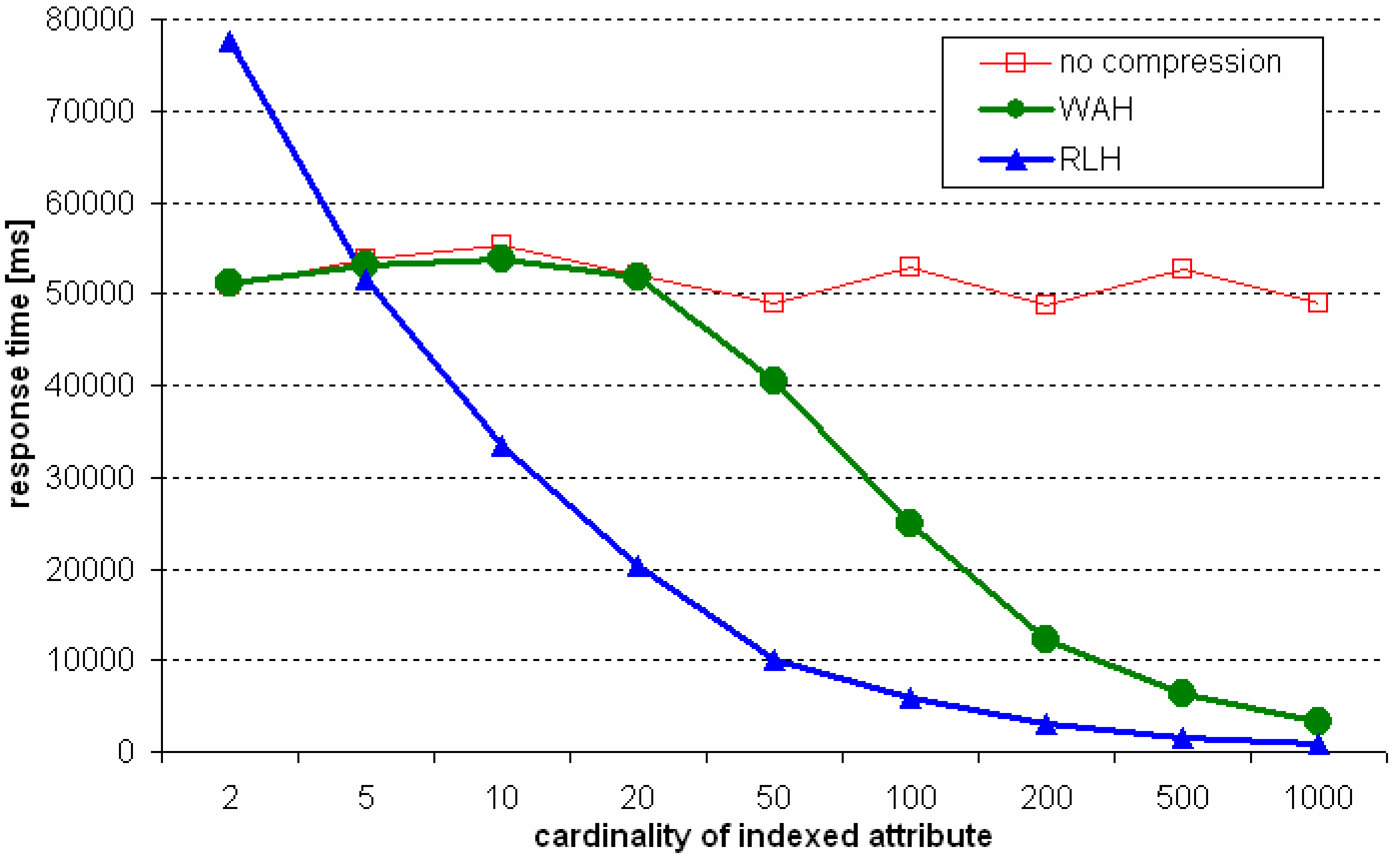
attr.cardinality	s_{WAH}/s_{RLH}
2	1.03
5	1.42
10	2.18
20	3.43
50	5.18
100	5.88
200	6.01
500	6.68
1000	7.36

➲ **cardinality increases ⇨ size of RLH bitmaps increases more slowly than WAH**

WAH and RLH: response times

➲ **Query**

```
select ... from ...
where ind_attribute in (v1, v2, ..., v100)
```



attr. card.	t(WAH)/t(RLH)
2	0,66
5	1,03
10	1,60
20	2,55
50	3,98
100	4,15
200	4,13
500	3,97
1000	3,54

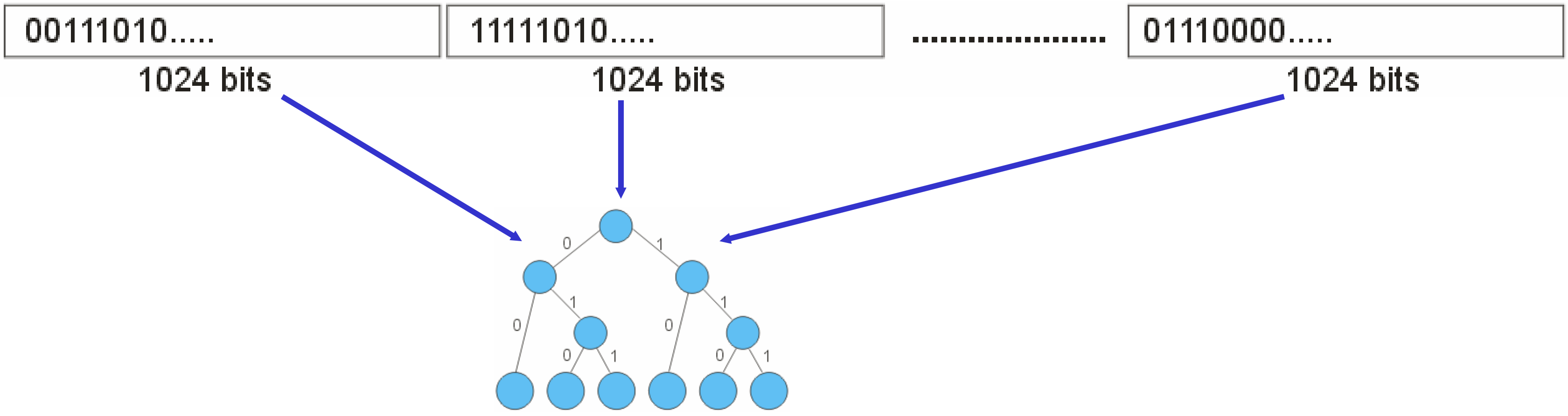
Updating RLH Bitmaps

- **Costly process**
  - **decompressing the whole bitmap**
  - **modifying the bitmap**
  - **compressing the bitmap**
- **Updating a RLH bitmap**
  - **changes frequencies of distances between 1 bits**
  - **creates new distances between 1 bits**
  - **requires building a new Huffman tree**
- **In a DW environment index structures**
  - **are dropped before loading a DW**
  - **are recreated after loading is finished**

RLH1024 Compression (1)

1. Dividing a bitmap into 1024-bit sections

- **constructing one Huffman tree based on frequencies of distances from all 1024-bit sections**



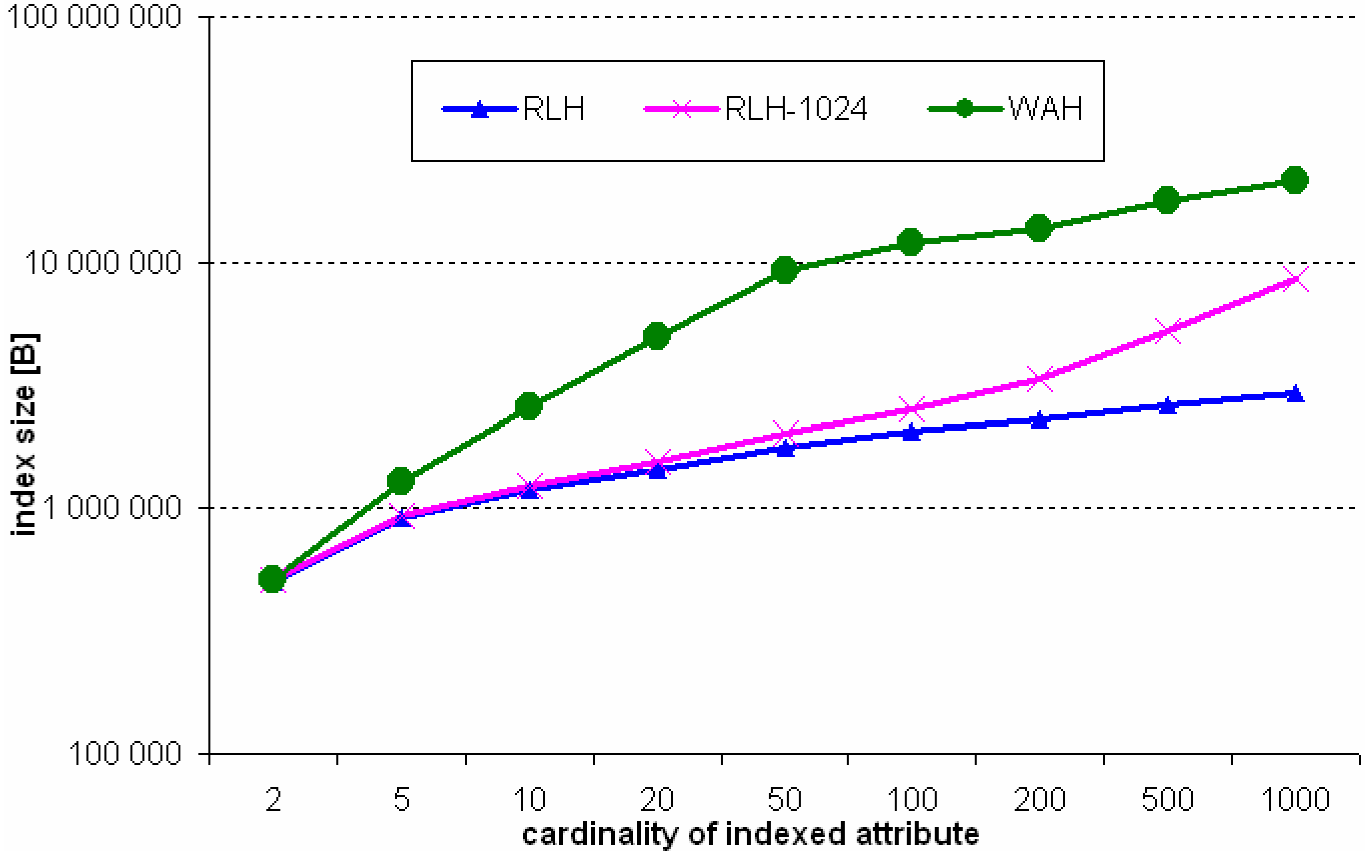
2. Including in the HT all possible distances that may appear in a 1024-bit section

- **non-existing distances have assigned the frequency of 1**

RLH1024 Compression (2)

- **Advantages**
  - including all the possible distances in the HT eliminates the need of rebuilding the HT after such a bitmap update that results in a new distance
  - 1024-bit sections can be read and processed in parallel
  - in order to update a bitmap, only an appropriate 1024-bit section has to be read and uncompressed

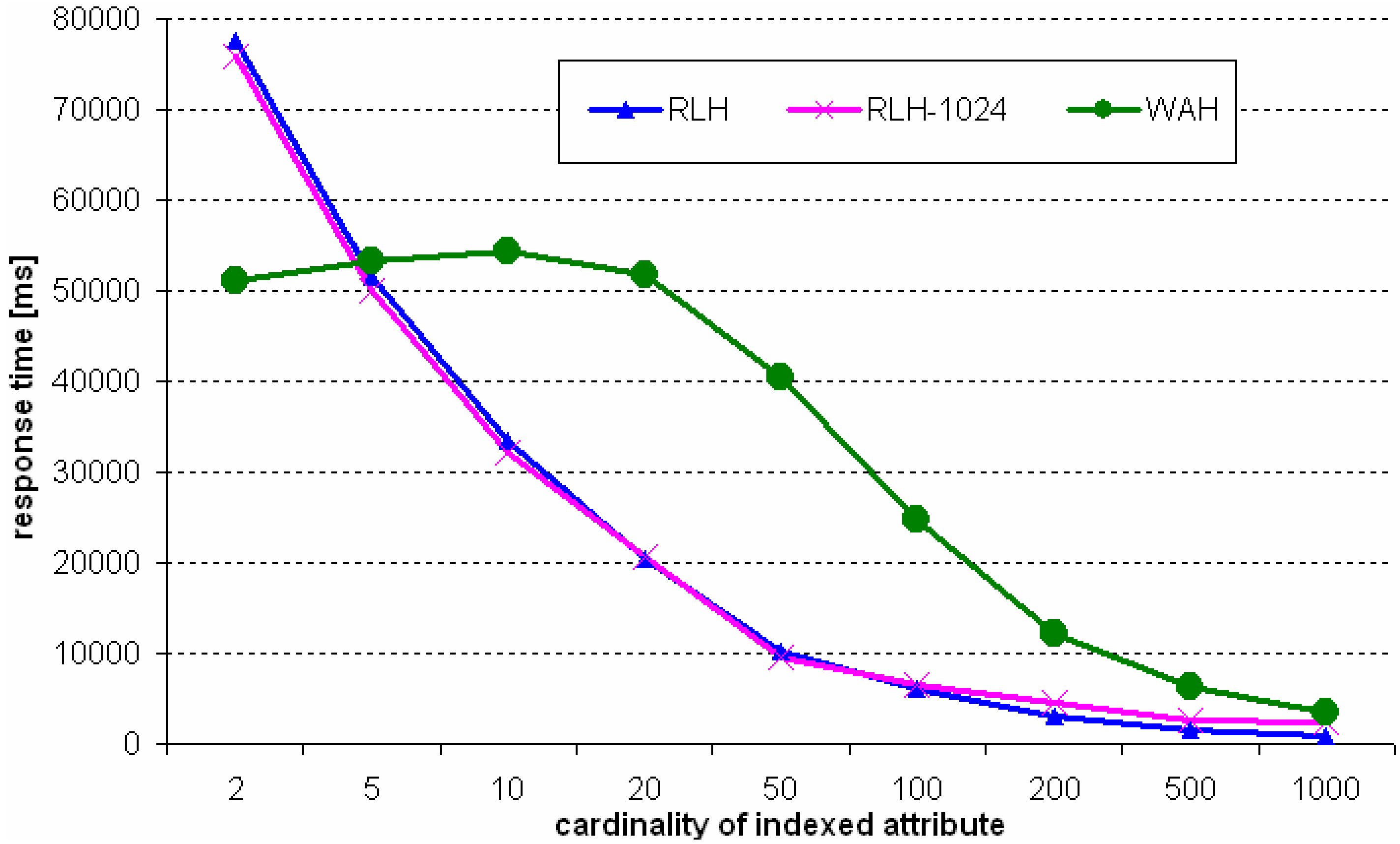
RLH and RLH1024: index sizes



RLH and RLH1024: response times

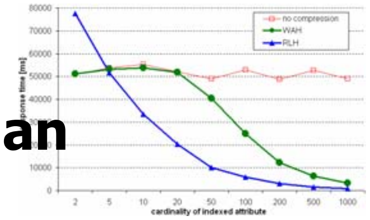
- **Query**

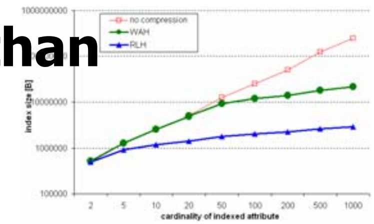
```
select ... from ...
where ind_attribute in (v1, v2, ..., v100)
```

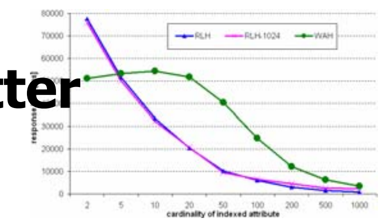


Summary

- **Alternative bitmap compression technique based on the run-length encoding and on Huffman encoding**
  - **RLH**
  - **RLH1024**
- **Observations**
  - **RLH offers a higher efficiency in accessing data than WAH for attribute cardinality from 5 to 1000**
  - **Bitmaps compressed with RLH are much smaller than corresponding bitmaps compressed with WAH for attribute cardinalities >10**
  - **RLH1024 offers a data access time characteristic similar to RLH, but additionally RLH1024 may better support bitmap updates**







Ongoing and Future Work

- Ongoing
  - evaluating the impact of values distribution on WAH and RLH
  - evaluating other than 1024-bit partition schemes
  - evaluating the efficiency of updating bitmaps in RLH and RLH1024
- Future
  - developing a cost model for RLH
  - developing a framework for selecting the most efficient bitmap partition scheme for RLH
  - developing a framework for selecting the most efficient bitmap compression technique for a given data characteristic
  - experimentally comparing BBC, WAH, RLH, and AE
  - integrating RLH into FastBit

References (1)

- **Binning**
  - [Kou00] Koudas N.: Space Efficient Bitmap Indexing. CIKM, 2000
  - [SWS04] Stockinger K., Wu K., Shoshani A.: Evaluation Strategies for Bitmap Indices with Binning. DEXA, 2004
  - [RSW05] Rotem D., Stockinger K., Wu K.: Optimizing Candidate Check Costs for Bitmap Indices. CIKM, 2005
- **Encoding**
  - [WuBu98] Wu M., Buchmann A.P.: Encoded Bitmap Indexing for Data Warehouses. ICDE, 1998
  - [ChIo99] Chan C.Y., Ioannidis Y.E.: An Efficient Bitmap Encoding Scheme for Selection Queries. SIGMOD, 1999
- **Compressing**
  - BBC
    - [AnZi96] Antoshenkov G., Ziauddin M.: Query Processing and Optimization in ORACLE RDB. VLDB Journal, 1996
  - WAH
    - [SWS02] Stockinger K., Wu K., Shoshani A.: Strategies for Processing ad hoc Queries on Large Data Sets. DOLAP, 2002
    - [WOS04] Wu K., Otoo E.J., Shoshani A. (2004): On the Performance of Bitmap Indices for High Cardinality Attributes. VLDB, 2004
    - [StWu07] Stockinger K., Wu K.: Bitmap Indices for Data Warehouses. In Wrembel R. and Koncilia C. (eds.): Data Warehouses and OLAP: Concepts, Architectures and Solutions. IGI Global, 2007

References (2)

- **Compressing**
  - **Approximate Compression with Bloom Filters**

    - [ACFT06] Apaydin T., Canahuate G., Ferhatosmanoglu H., Tosun, A. S.: Approximate encoding for direct access and query processing over compressed bitmaps. VLDB, 2006

  - **Reordering**

    - [JKCKV04] Johnson D., Krishnan S., Chhugani J., Kumar S., Venkatasubramanian S.: Compressing Large Boolean Matrices Using Reordering Techniques. VLDB, 2004
    - [PTF05] Pinar A., Tao T., Ferhatosmanoglu H.: Compressing Bitmap Indices by Data Reorganization. ICDE, 2005

- **WAH vs. BBC**

    - [SWS02] Stockinger K., Wu K., Shoshani A.: Strategies for processing ad hoc queries on large data warehouses. DOLAP, 2002
    - [WOS02] Wu K., Otoo E.J., Shoshani A.: Compressing bitmap indexes for faster search operations. SSDBM, 2002
    - [WOS04] Wu K., Otoo E.J., Shoshani A.: On the Performance of Bitmap Indices for High Cardinality Attributes. VLDB, 2004