POZNAN UNIVERSITY OF TECHNOLOGY

Data Warehouse Physical Design: Part I

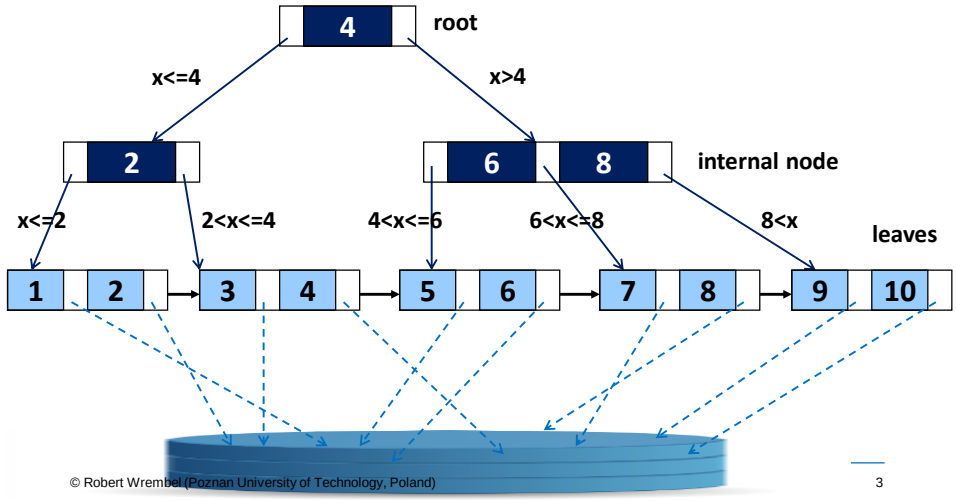
Robert Wrembel
Poznan University of Technology
Institute of Computing Science
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

Lecture outline

- **Basic index structures**
  - **B-tree**
  - **bitmap**
  - **characteristics of the bitmap index**

B-tree

- Foundation for other indexes (join, bitmap, bitmap join, clustering, MDC)



Bitmap index - definitions

- Attribute cardinality ⇨ domain size
  - card(make)=4, card(color)=4
- Bitmap ⇨ vector of bits
  - a bit corresponds to a row in a table

make	...	color	B: red
Subaru		red	1
Mercedes		green	0
Mercedes		blue	0
Subaru		blue	0
BMW		blue	0
BMW		green	0
BMW		red	1
Audi		red	1
Audi		black	0
BMW		black	0
Subaru		red	1
Mercedes		green	0

Bitmap index - definitions

- **Bitmap index**
  - **collection of bitmaps**
  - **one bitmap for one value from attribute domain**

make	...	color
Subaru		red
Mercedes		green
Mercedes		blue
Subaru		blue
BMW		blue
BMW		green
BMW		red
Audi		red
Audi		black
BMW		black
Subaru		red
Mercedes		green

B: red	B: green	B: blue	B: black
1	0	0	0
0	1	0	0
0	0	1	0
0	0	1	0
0	0	1	0
0	1	0	0
1	0	0	0
1	0	0	0
0	0	0	1
0	0	0	1
1	0	0	0
0	1	0	0

Bitmap index in queries

```
select count(*) from CarSales
where  make in ('Audi', 'Mercedes', 'BMW')
and    type = 'sport'
and    color ='red';
```

B:Audi	OR	B:Mercedes	OR	B: BMW	AND	B:sport	AND	B: red	=	
0		0		0		1		1		0
0		1		0		0		0		0
0		1		0		0		0		0
0		0		0		0		0		0
0		0		1		1		0		0
0		0		1		1		0		0
0		0		1		1		1		1
1		0		0		1		1		1
1		0		0		0		0		0
0		0		1		0		0		0
0		0		0		1		1		0
0		1		0		1		0		0

Implementation

- Organizing bitmaps
  - 2-dimensional table [rowNO][attrValue]
  - list of bit vectors (bitmaps), e.g., Python list
  - key-value (key: attrValue, value: bitmap), e.g., Python dictionary
  - B-tree index (tree structure for finding a leaf, leaf: bitmap or pointer to bitmap)
  - ...
- Problem: mapping between bitNO and rowNO
  - fixed size records
  - variable size records

make	...	color	B: red
Subaru		red	1
Mercedes		green	0
Mercedes		blue	0
Subaru		blue	0
BMW		blue	0
BMW		green	0
BMW		red	1
Audi		red	1
Audi		black	0
BMW		black	0
Subaru		red	1
Mercedes		green	0

Mapping bits to ROWIDs

- Easy solution:
  - fixed size records
  - fixed number of rows per DB block → rpb

block 1	1	slot1
	2	slot2
	3	slot3
	4	slot4
	5	slot5
block 2	6	slot1
	7	slot2
	8	slot3
	9	slot4
	10	slot5
block 3	11	slot1
	12	slot2
	13	slot3
	14	slot4
	15	slot5

$$blockNO = \left\lceil \frac{bitNO}{rpb} \right\rceil$$

rpb=5
bitNO=2 → blockNO=1
...
bitNO=7 → blockNO=2

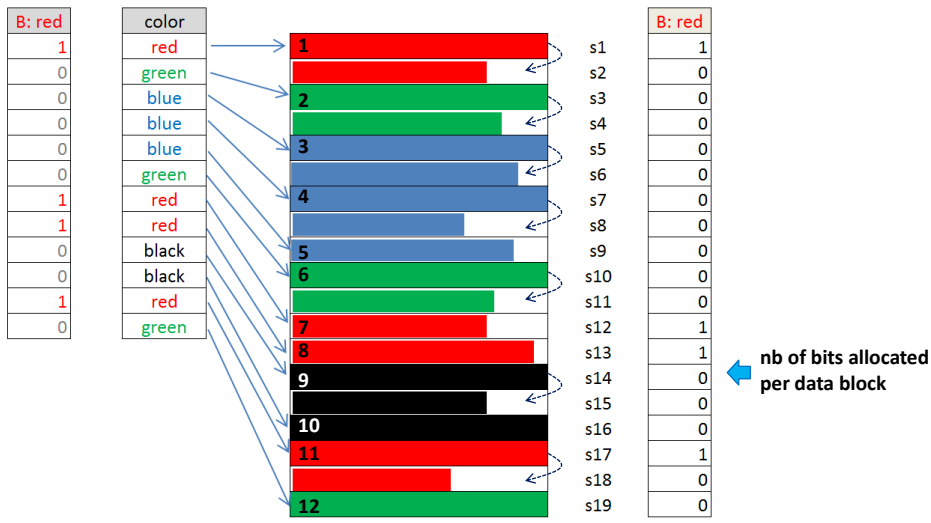
IF MOD(bitNo;rpb) !=0
  THEN slotNO = MOD(bitNO; rpb)
ELSE slotNO=bitNo/blockNO

bitNO=2 → MOD(2;5)=2
bitNO=5 → MOD(5;5)=0 slotNO=5/1
...
bitNO=7 → MOD(7;5)=2
bitNO=10 → MOD(10;5)=0 slotNO=10/2

Mapping bits to ROWIDs

- **Real approach**
  - **estimate the average length L of a row in a table**
  - **compute the number of slots in a DB block:** $\lfloor \frac{BlockSize}{L} \rfloor$
  - **allocate more slots than** $\lfloor \frac{BlockSize}{L} \rfloor$
  - **real row length differs from L → some slots are wasted → a bitmap index is larger than it could be**

Mapping bits to ROWIDs



Estimating row length: Oracle

```
create table TEST_BI1
(id number(7),
 no number(7),
 txt1 varchar2(20),
 txt2 varchar2(20),
 txt3 varchar2(20),
 txt4 varchar2(20),
 txt5 varchar2(20));
```

```
create table TEST_BI2
(id number(7) not null,
 no number(7) not null,
 txt1 varchar2(20) not null,
 txt2 varchar2(20) not null,
 txt3 varchar2(20) not null,
 txt4 varchar2(20) not null,
 txt5 varchar2(20) not null);
```

```
select o.name, o.obj#, t.spare1
from sys.obj$ o, sys.tab$ t
where o.obj#=t.obj# and o.name in ('TEST_BI1', 'TEST_BI2')

NAME                             OBJ#     SPARE1
------------------------------ ---------- ----------
TEST_BI1                          73450        736
TEST_BI2                          73452        506
```

max number of rows per data block

Bitmap index size: Oracle

- Populating tables TEST_BI1 and TEST_BI2 with 10^6 of rows
- Computing column statistics
- Creating bitmap indexes

```
create bitmap index NO_BI_INDX1 on TEST_BI1(NO) pctfree 0;
```

```
create bitmap index NO_BI_INDX2 on TEST_BI2(NO) pctfree 0;
```

- Getting index sizes form data dictionary

```
select index_name, leaf_blocks
from dba_indexes
where index_name in ('NO_BI_INDX1', 'NO_BI_INDX2');
```

BI characteristics

- **Reasonably small size for attributes of low cardinality**
- **Example**
  - **#rows = 1 000 000**
  - **card(A) = 4**
  - **ROWID = 10B (Oracle)**
  - **bitmap index on attribute A**
    - **4 bitmaps: 4 x (1 000 000 / 8) = 4 x 125kB = 500kB**
  - **B^+-tree on attribute A**
    - **1 000 000 x 10B = 10MB**

BI characteristics

- **Efficient processing of bitmaps**
  - **logical operations AND, OR, NOT, COUNT**
  - **64 bits processed in one CPU clock cycle**
- **Size**
  - **small index ⇨ small #I/O**
  - **processing in RAM**

BI characteristics

- **Large size for attributes of large cardinality**
- **Example**
  - **#rows = 1 000 000**
  - **card(A) = 1024**
  - **ROWID = 10B (Oracle)**
  - **bitmap index on attribute A**
    - **1024 bitmaps: 1024 x (1 000 000 / 8) = 1024 x 125kB = 128MB**
  - **B$^+$-tree on A**
    - **1 000 000 x 10B = 10MB**

BI characteristics

- **Index maintenance**
  - **inserting rows ⇨ increasing length of bitmaps**

Sales

ProductID	SalesPrice	...
200	45	
400	50	
100	40	
200	55	
500	75	
100	65	
400	70	

100	200	400	500
0	1	0	0
0	0	1	0
1	0	0	0
0	1	0	0
0	0	0	1
1	0	0	0
0	0	1	0

Sales

ProductID	SalesPrice	...	100	200	400	500
200	45		0	1	0	0
400	50		0	0	1	0
100	40		1	0	0	0
200	55		0	1	0	0
500	75		0	0	0	1
100	65		1	0	0	0
400	70		0	0	1	0
200	45		0	1	0	0
400	55		0	0	1	0
400	55		0	0	1	0

BI characteristics

- **Index maintenance**
  - **deleting rows ⇨**
    - **decreasing length of bitmaps**
    - **OR bitmap of deleted rows**

Sales

ProductID	SalesPrice	...
200	45	
400	50	
100	40	
~~200~~	~~55~~	
~~500~~	~~75~~	
100	65	
400	70	

100	200	400	500
0	1	0	0
0	0	1	0
1	0	0	0
1	0	0	0
0	0	1	0

Sales

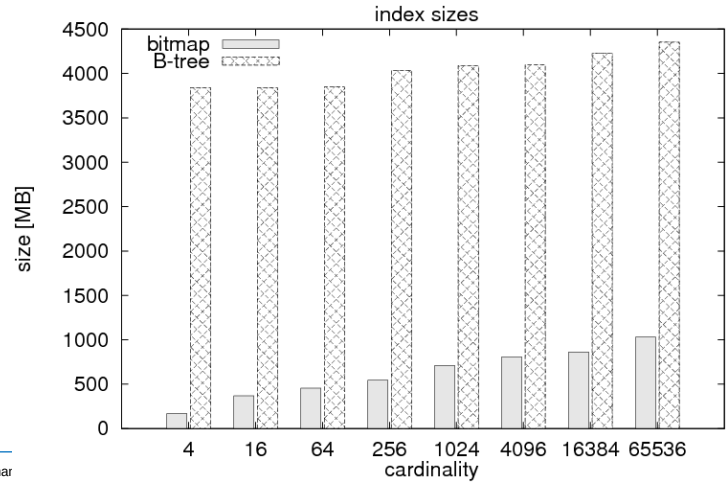
ProductID	SalesPrice	...	100	200	400	500	Exists
200	45		0	1	0	0	1
400	50		0	0	1	0	1
100	40		1	0	0	0	1
~~200~~	~~55~~		0	1	0	0	0
~~500~~	~~75~~		0	0	0	1	0
100	65		1	0	0	0	1
400	70		0	0	1	0	1

BI characteristics

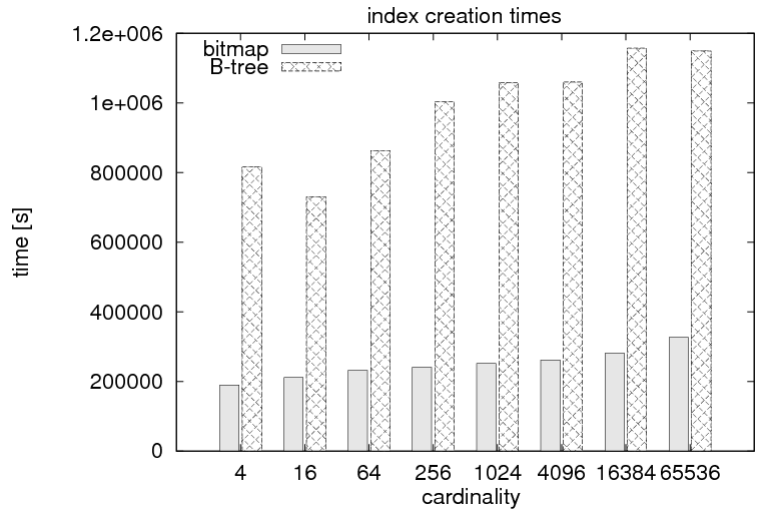
- **Index maintenance**
  - **updating rows ⇨ updating 2 bitmaps**
  - **locking contiguous segments of bitmaps ⇨ concurrency decreasing**

Experiment (1)

- **Oracle11g**
  - **data cache: 1.7GB, SGA: 3.4GB**
  - **#rows: 250 000 000 (DB size: 10.8GB)**

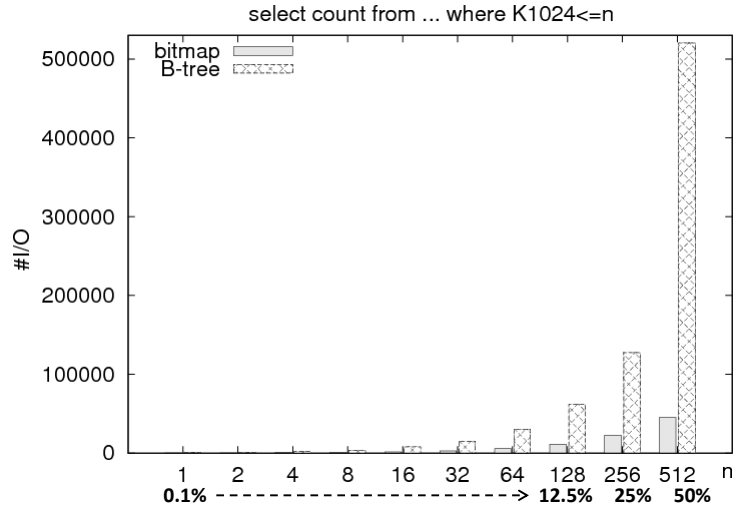


Experiment (2)



Experiment (3)

➲ **cardinality of indexed attribute: 1024**



Experiment (4)

WHERE K1024 <= 128

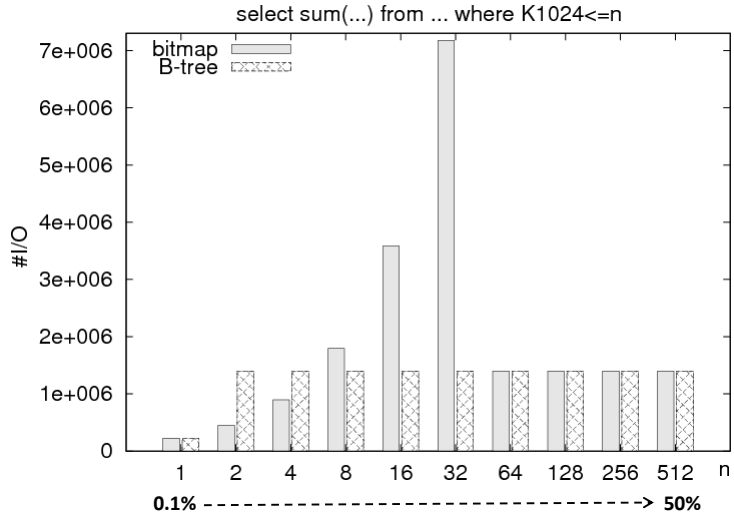
bitmap index

```
-------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |     1 |     4 | 11315   (1)| 00:02:16 |
|   1 |  SORT AGGREGATE              |             |     1 |     4 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |             |    31M|   119M| 11315   (1)| 00:02:16 |
|*  3 |    BITMAP INDEX RANGE SCAN   | BMP_5K1024  |       |       |            |          |
-------------------------------------------------------------------------------------------
```

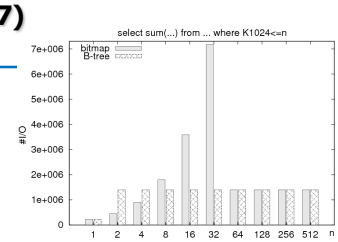
B*-tree index

```
-----------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |     1 |     4 | 65441   (1)| 00:13:06 |
|   1 |  SORT AGGREGATE    |             |     1 |     4 |            |          |
|*  2 |   INDEX RANGE SCAN | BMP_5K1024  |    31M|   119M| 65441   (1)| 00:13:06 |
-----------------------------------------------------------------------------------
```

Experiment (5)



Experiment (7)



WHERE K1024 <= 4

bitmap index

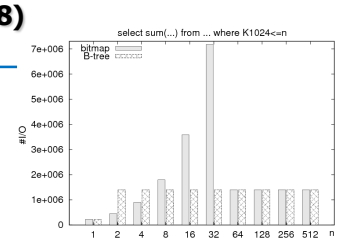
```
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |     1 |    10 |   149K  (1)| 00:29:51 |
|   1 |  SORT AGGREGATE               |             |     1 |    10 |            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID | BMTEST      |   977K|  9543K|   149K  (1)| 00:29:51 |
|   3 |    BITMAP CONVERSION TO ROWIDS|             |       |       |            |          |
|*  4 |     BITMAP INDEX RANGE SCAN   | BMP_5K1024  |       |       |            |          |
-------------------------------------------------------------------------------------------
```

B*-tree index

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |         |     1 |    10 |   387K  (2)| 01:17:26 |
|   1 |  SORT AGGREGATE    |         |     1 |    10 |            |          |
|*  2 |   TABLE ACCESS FULL| BMTEST  |   977K|  9543K|   387K  (2)| 01:17:26 |
-------------------------------------------------------------------------------
```

Experiment (8)

WHERE K1024 <= 32



bitmap index

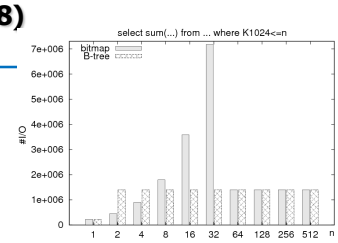
Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	10	344K (1)	01:08:57
1	SORT AGGREGATE		1	10		
2	TABLE ACCESS BY INDEX ROWID	BMTEST	7819K	74M	344K (1)	01:08:57
3	BITMAP CONVERSION TO ROWIDS					
* 4	BITMAP INDEX RANGE SCAN	BMP_5K1024				

B*-tree index

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	10	387K (2)	01:17:26
1	SORT AGGREGATE		1	10		
* 2	TABLE ACCESS FULL	BMTEST	7819K	74M	387K (2)	01:17:26

Experiment (8)

WHERE K1024 <= 64



bitmap index

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	10	387K (2)	01:17:26
1	SORT AGGREGATE		1	10		
* 2	TABLE ACCESS FULL	BMTEST	15M	149M	387K (2)	01:17:26

B*-tree index

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	10	387K (2)	01:17:26
1	SORT AGGREGATE		1	10		
* 2	TABLE ACCESS FULL	BMTEST	15M	149M	387K (2)	01:17:26