CS 564: Database Management Systems

Lecture 30: Column Store

Xiangyao Yu

4/8/2024

Module B3 Query Processing

Relational operators I

Relational operators II

Query optimization I

Query optimization II

Column Store

Outline

Data layout

Data encoding

Operators

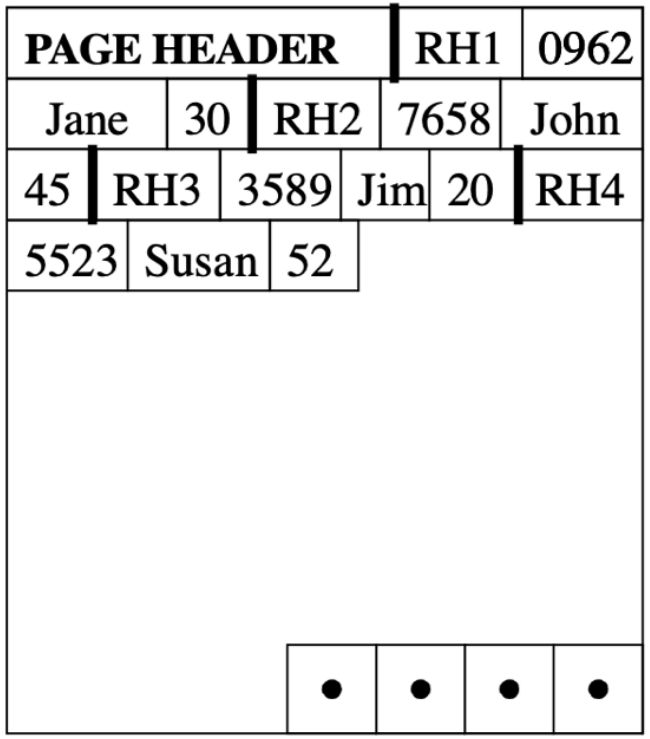
Zone map

Data Storage Format

RELATION R

RID	SSN	Name	Age
1	0962	Jane	30
2	7658	John	45
3	3859	Jim	20
4	5523	Susan	52
5	9743	Leon	43
6	0618	Dan	37

NSM PAGE



N-ary Storage Model (NSM)

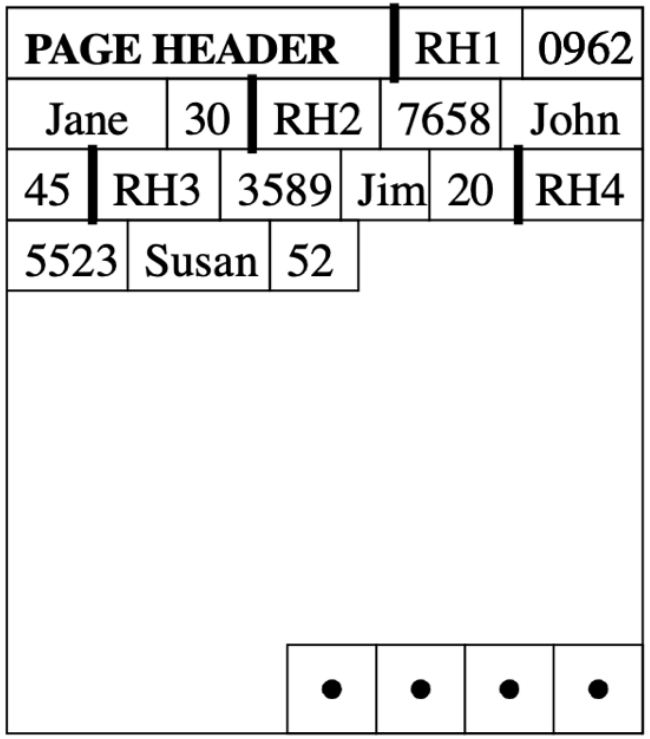
Source: *Data Page Layouts for Relational Databases on Deep Memory Hierarchies*, VLDB Journal, 2002

Data Storage Format

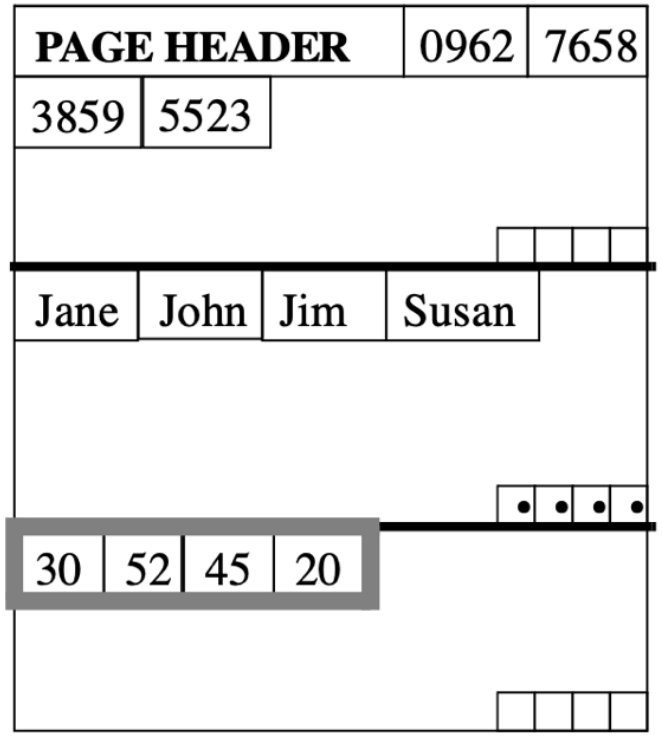
RELATION R

RID	SSN	Name	Age
1	0962	Jane	30
2	7658	John	45
3	3859	Jim	20
4	5523	Susan	52
5	9743	Leon	43
6	0618	Dan	37

NSM PAGE



PAX PAGE



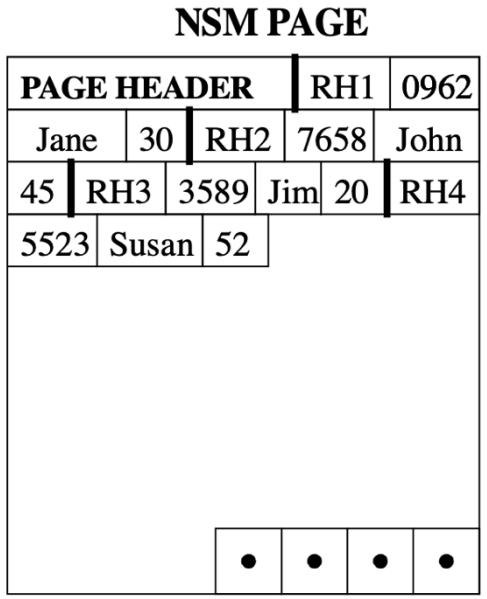
N-ary Storage Model (NSM) Partition Attributes Across (PAX)

Source: *Data Page Layouts for Relational Databases on Deep Memory Hierarchies*, VLDB Journal, 2002

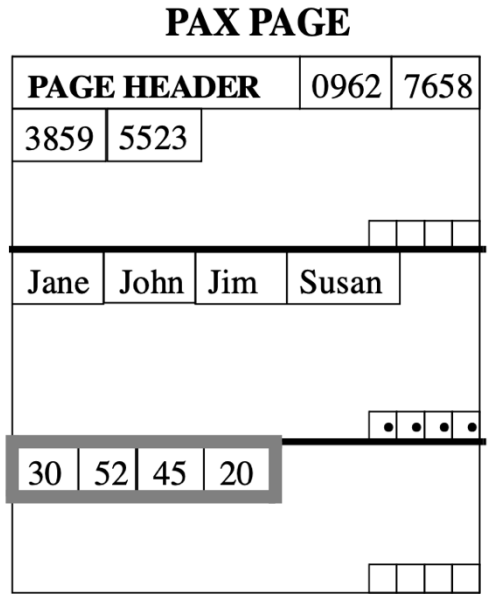
Column Store

Advantages of column store

- Encode data in compact layout
- Read relevant attributes only (projection is free!)



N-ary Storage Model (NSM)



Partition Attributes Across (PAX)

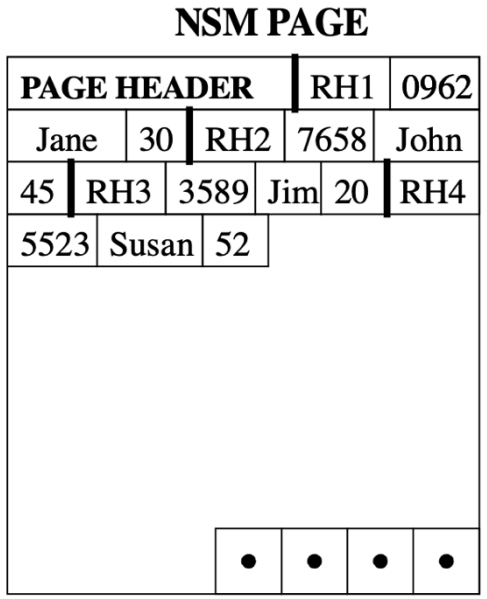
Column Store

Advantages of column store

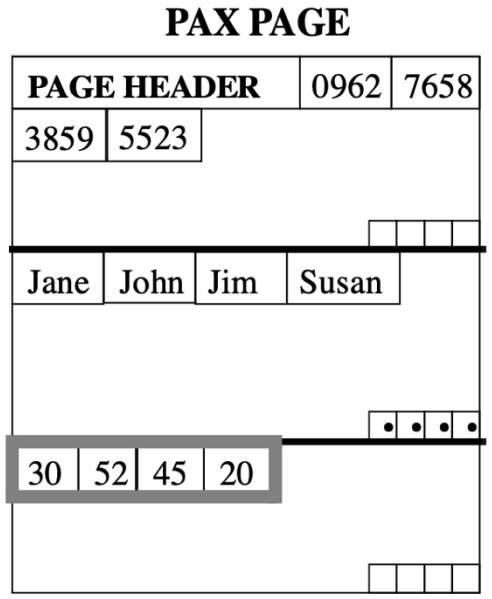
- Encode data in compact layout
- Read relevant attributes only (projection is free!)

Disadvantage of column store

- Update/Insertion is more expensive



N-ary Storage Model (NSM)



Partition Attributes Across (PAX)

Outline

Data layout

Data encoding

- Bit packing
- Run-length encoding (RLE)
- Delta encoding
- Dictionary encoding

Operators

Zone map

Data Encoding — Bit Packing

A: integer

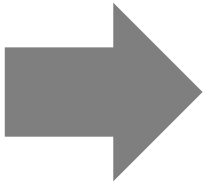
35	00000000	00000000	00000000	00100011
6	00000000	00000000	00000000	00000110
178	00000000	00000000	00000000	10110010
93	00000000	00000000	00000000	01011101
12	00000000	00000000	00000000	00001100
54	00000000	00000000	00000000	00110110

Data Encoding — Bit Packing

A: integer

35	00000000	00000000	00000000	00100011
6	00000000	00000000	00000000	00000110
178	00000000	00000000	00000000	10110010
93	00000000	00000000	00000000	01011101
12	00000000	00000000	00000000	00001100
54	00000000	00000000	00000000	00110110

Bit Packing



header
00100011
00000110
10110010
01011101
00001100
00110110

- Avoid storing leading zeros in a group of values
- Use bitwise operations to encode and decode

Data Encoding — Run-Length Encoding

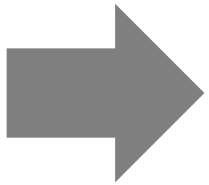
A: string

Madison
Madison
Madison
Madison
Chicago
Chicago

Data Encoding — Run-Length Encoding

A: string

Madison
Madison
Madison
Madison
Chicago
Chicago



Madison
Chicago

value array

4
2

count array

- Store data as (value, count) pairs
- Effective for compressing long sequences of repeated values

Data Encoding — Delta Encoding

A: integer

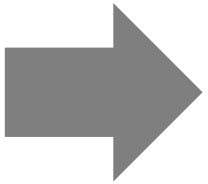
1000001
1000002
1000003
1000004
1000005
1000006

Data Encoding — Delta Encoding

A: integer

1000001
1000002
1000003
1000004
1000005
1000006

Delta Encoding



1000001
1
1
1
1
1

- Store the differences (or deltas) between consecutive values
- Ideal for data that changes gradually over time (e.g., timestamp, temperature, sensor values, etc.)
- Can apply bit packing and/or RLE on delta encoded data to further reduce data footprint

Data Encoding — Dictionary Encoding

A: string

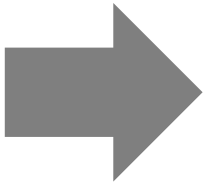
Wisconsin
Illinois
Minnesota
Minnesota
Wisconsin
Illinois

Data Encoding — Dictionary Encoding

A: string

Wisconsin
Illinois
Minnesota
Minnesota
Wisconsin
Illinois

Dictionary Encoding



Dictionary:
('Wisconsin': 0
'Illinois': 1
'Minnesota': 2)

Encoded data

0
1
2
2
0
1

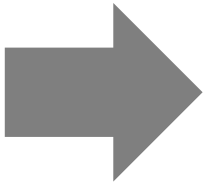
- Assign different integer identifier for all unique strings or symbols

Data Encoding — Dictionary Encoding

A: string

Wisconsin
Illinois
Minnesota
Minnesota
Wisconsin
Illinois

Dictionary Encoding



Dictionary:
('Wisconsin': 0
'Illinois': 1
'Minnesota': 2)

Encoded data

0
1
2
2
0
1

- Assign different integer identifier for all unique strings or symbols
- Can avoid decoding for certain operators
  - state='Wisconsin' → state=0
  - R.state=S.state

Data Encoding

Multiple encoding techniques can be applied in layers to further reduce data footprint

For example, dictionary-encoded data can be further compressed through bit packing, delta encoding, and/or run-length encoding

Layered encoding/decoding requires more computation

Data Encoding Example

A: char[10]

Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Minnesota
Minnesota
Minnesota
Minnesota
Illinois
Illinois
Illinois

Table size: 15*10=150 Bytes

Data Encoding Example

A: char[10]

Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Minnesota
Minnesota
Minnesota
Minnesota
Illinois
Illinois
Illinois

Table size: 15*10=150 Bytes

Dictionary encoding

Dictionary

‘Wisconsin’: 0
‘Illinois’: 1
‘Minnesota’: 2

0
0
0
0
0
0
0
0
0
2
2
2
2
1
1
1

Table size: 15*4=60 Bytes

Data Encoding Example

A: char[10]

Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Minnesota
Minnesota
Minnesota
Minnesota
Illinois
Illinois
Illinois

Table size: 15*10=150 Bytes

Dictionary encoding

Dictionary

'Wisconsin': 0
'Illinois': 1
'Minnesota': 2

0
0
0
0
0
0
0
0
2
2
2
2
1
1
1

Table size: 15*4=60 Bytes

Run-length encoding

value	count
0	8
2	4
1	3

Table size: 6*4=24 Bytes

Data Encoding Example

A: char[10]

Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Wisconsin
Minnesota
Minnesota
Minnesota
Minnesota
Illinois
Illinois
Illinois

Table size: 15*10=150 Bytes

Dictionary encoding

Dictionary

'Wisconsin': 0
'Illinois': 1
'Minnesota': 2

0
0
0
0
0
0
0
0
2
2
2
2
1
1
1

Table size: 15*4=60 Bytes

Run-length encoding

value	count
0	8
2	4
1	3

Table size: 6*4=24 Bytes

Bit packing

Table size: 6*1=6 Bytes

Outline

Data layout

Data encoding

Operators

- Projection
- Selection
- Join

Zone map

Projection

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

sid
22
29
31
32
58
64
71
74

sname
Dustin
Brutus
Lubber
Andy
Rusty
Horatio
Zorba
Horato

rating
7
1
8
8
10
7
10
9

age
45
33
55
25
35
35
16
35

Read the needed columns only

– Projection is free!

Selection

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

```
SELECT sname,
FROM Sailors
WHERE rating > 7;
```

Selection

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

rating > 7

0
0
1
1
1
0
1
1

Bitmap for rows that satisfy the predicate

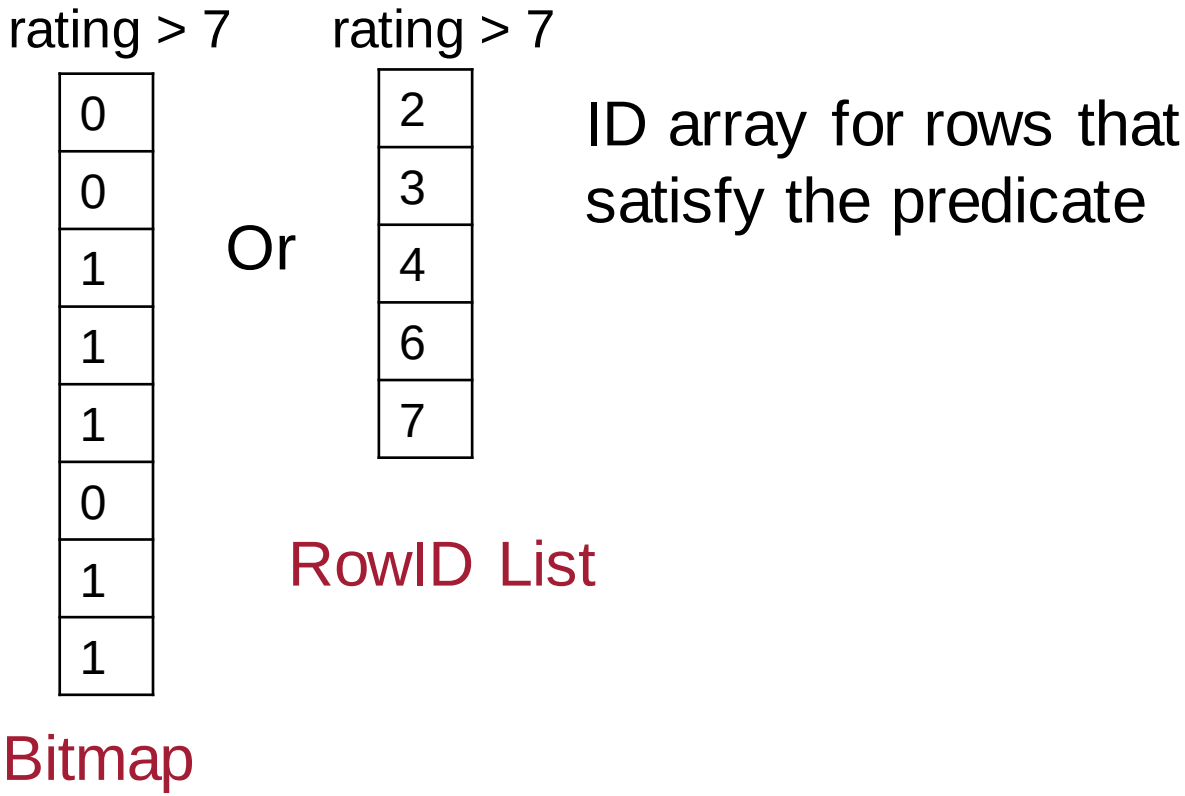
Bitmap

```
SELECT sname,
FROM Sailors
WHERE rating > 7;
```

Selection

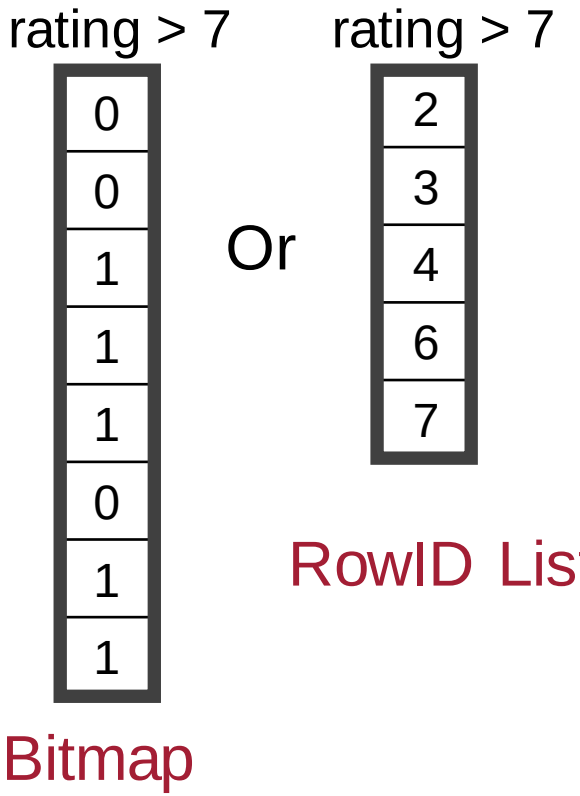
sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

```
SELECT sname,
FROM Sailors
WHERE rating > 7;
```



Selection

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35




```
SELECT sname,
FROM Sailors
WHERE rating > 7;
```

Produce the selection results using the `sname` column and the bitmap or the RowID list

Join

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Reserves

```
SELECT sname,
FROM Sailors S, Reserves R
WHERE S.sid = R.sid;
```

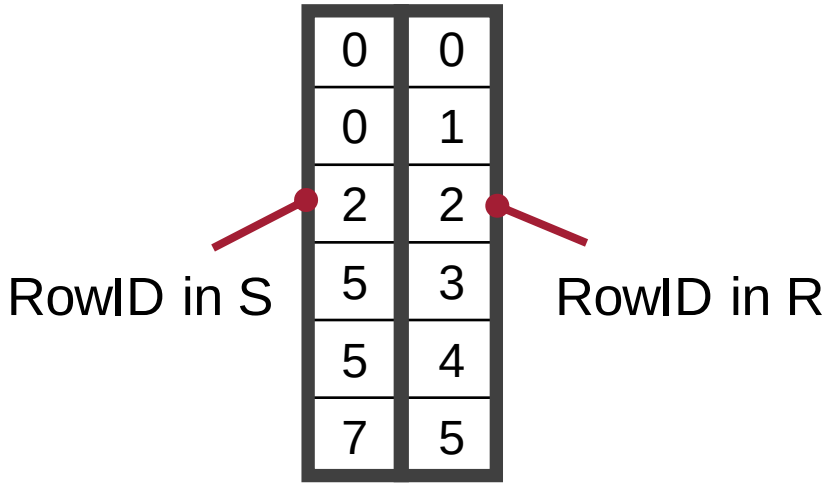
Join

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Reserves



RowID pairs

```
SELECT sname,
FROM Sailors S, Reserves R
WHERE S.sid = R.sid;
```

Read the join key columns and produce RowID pairs for matching tuples

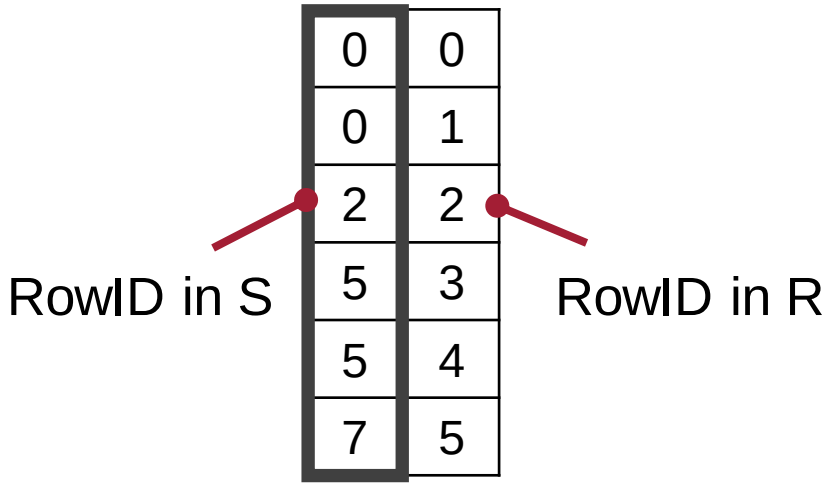
Join

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

sid	bid	day
22	101	10/10/98
22	102	10/10/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Reserves



RowID pairs

```
SELECT sname,
FROM Sailors S, Reserves R
WHERE S.sid = R.sid;
```

Late materialize other columns using the RowID arrays

Outline

Data layout

Data encoding

Operators

Zone map

Zone Map

Reduce read IO cost for columns that have no index

Zone Map

Reduce read IO cost for columns that have no index

Row Group 1

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25

Row Group 2

sid	sname	rating	age
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Zone Map

Reduce read IO cost for columns that have no index

Row Group 1

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25

Row Group 2

sid	sname	rating	age
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Zone Map

cname	type	val
sid	min	22
sid	max	32
rating	min	1
rating	max	8
…	…	…

cname	type	val
sid	min	58
sid	max	74
rating	min	7
rating	max	10
…	…	…

Zone Map

Reduce read IO cost for columns that have no index

Row Group 1

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25

Row Group 2

sid	sname	rating	age
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Zone Map

cname	type	val
sid	min	22
sid	max	32
rating	min	1
rating	max	8
…	…	…

cname	type	val
sid	min	58
sid	max	74
rating	min	7
rating	max	10
…	…	…

```
SELECT sname,
FROM Sailors  S
WHERE rating  < 6
```

Can skip row group 2 since min value for rating is 7

Summary

Data layout

Data encoding

- Bit packing
- Run-length encoding (RLE)
- Delta encoding
- Dictionary encoding

Operators

- Projection
- Selection
- Join

Zone map