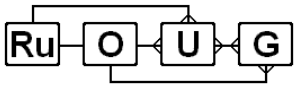

INTRA-BLOCK ROW CHAINING

Sayan Malakshinov

Oracle ACE Associate

Oracle certified performance tuning expert

http://orasql.org

The documentation states:

When a table has more than 255 columns, rows that have data after the 255th column are likely to be chained within the same block. This is called intra-block chaining. A chained row’s pieces are chained together using the rowids of the pieces. With intra-block chaining, users receive all the data in the same block. If the row fits in the block, users do not see an effect in I/O performance, because no extra I/O operation is required to retrieve the rest of the row.

Example 1

- 1. Create table TEST with 355 columns (c_1, c_2, …, c_355)
- 2. insert into test(**c_300**) values(2)
- 3. dump data blocks

```
bdba: 0x018019f7
    block_row_dump:
        tl: 260 fb: -----L-- lb: 0x1  cc: 255
            col  0: *NULL*
            col  1: *NULL*
            ...
            col 252: *NULL*
            col 253: *NULL*
            col 254: [ 2]  c1 03

        tl: 54 fb: --H-F--- lb: 0x1  cc: 45
            col  0: *NULL*
            col  1: *NULL*
            ...
            col 43: *NULL*
            col 44: *NULL*
```

tl – row piece length
fb – flags:
- H - header
- F – first row piece
- L – last piece

lb – "lock byte" (# of ITL entry)
cc – number of columns

The resulted dump file shows us:
1. Both row pieces are in the same block 0x018019f4
2. They contain only first 300 columns (trailing 55 columns are NULLs)
3. First row piece contains columns c_46 – c_300,
4. Second row piece contains columns c_1 – c_45 (they all are NULLs)

Example 2

- 1. Create table TEST with 355 columns (c_1, c_2, …, c_355)
- 2. insert into test(c_1) values(null)
- 3. update test set c_300=2
- 4. dump data blocks

```
bdba: 0x018019f3
    block_row_dump:
        tl: 260 fb: -----L-- lb: 0x1  cc: 255
            col  0: *NULL*
            col  1: *NULL*
         ...
         col 253: *NULL*
            col 254: [ 2]  c1 04

bdba: 0x018019f4
block_row_dump:

bdba: 0x018019f5
block_row_dump:
```

```
bdba: 0x018019f6
block_row_dump:

bdba: 0x018019f7
    block_row_dump:
        tl: 54 fb: --H-F--- lb: 0x1  cc: 45
            col  0: *NULL*
            col  1: *NULL*
            col  2: *NULL*
               ...
            col 43: *NULL*
            col 44: *NULL*
```

As you can see, there is no intra-block chaining – second row piece was created in another block.

Example 3

- 1. Create table TEST with 355 columns (c_1, c_2, …, c_355)
- 2. insert into test(c_1) values(1)
- 3. update test set c_300=2
- 4. update test set c_301=3
- 5. update test set c_302=4
- 6. dump data blocks

```
bdba: 0x018019f3
    block_row_dump:
        tl: 10 fb: -------- lb: 0x1  cc: 1
            col  0: *NULL*

bdba: 0x018019f4
    block_row_dump:
        tl: 264 fb: -----L-- lb: 0x1  cc: 255
            col  0: *NULL*
            col  1: *NULL*
               ...
            col 249: *NULL*
            col 250: *NULL*
            col 251: *NULL*
            col 252: [ 2]  c1 03
            col 253: [ 2]  c1 04
            col 254: [ 2]  c1 05
```

```
bdba: 0x018019f6
    block_row_dump:
        tl: 10 fb: -------- lb: 0x1  cc: 1
            col  0: *NULL*

bdba: 0x018019f7
    block_row_dump:
        tl: 56 fb: --H-F--- lb: 0x1  cc: 45
            col  0: [ 2]  c1 02
            col  1: *NULL*
            col  2: *NULL*
            col  3: *NULL*
               ...
            col 42: *NULL*
            col 43: *NULL*
            col 44: *NULL*
```

This dump shows us 4 row pieces:
First row piece contains 255 columns, second – 45, and 2 row pieces – just by one row.

So we can analyze it step-by-step:

2. insert into test(c_1) values(1)

After insert we have just one row piece with 1 field.

3. update test set c_300=2

After this update, we have 2 row pieces:

1) c_1-c_45
2) c_46-c_300

4. update test set c_301=3

This update split row piece c_46-c_300 into 2 row pieces:

1) c_46
2) c_47-c_301

So we have 3 row pieces now: c_1-c_45, c_46, c_47-c_301

5. update test set c_302=4

This update split row piece c_47-c_301 into 2 row pieces:

1) c_47
2) c_48-c_302

And we've got 4 row pieces: c_1-c_45, c_46, c_47, c_48-c_302

Example 4

- 1. Create table TEST4 with 355 columns (c_1, c_2, …, c_355)
- 2. insert into test(c_1) values(null)
- 3. update columns C_256..C_355:

```
for i in 256..355 loop
  execute immediate 'update test set c_'||i||'='||i;
end loop;
```

- 4. dump data blocks

```
grep -c "^bdba"
Count of bdba: 103

grep -c "^tl: "
Count of row pieces: 101
```

Summary

- One row piece can store up to 255 columns
- Oracle splits fields by row pieces in reverse order
- Oracle doesn’t store trailing null fields in a row (not in row piece)
- Before 12.2: Next row piece can be stored in the same block only with inserts. When you run update, oracle will place new row piece into another block.