Tuple internals:
exposing, exploring and explaining

Nikolay Shaplov

About the lecturer

Nikolay Shaplov

- Works in Postgres Professional (Russia)
- Open Source Software developer
- Based in Moscow

Contacts

- E-mail: n.shaplov@postgrespro.ru
- Jabber: n.shaplov@postgrespro.ru

Postgres PROFESSIONAL

Agenda

- Introduction
- Attributes in tuple
  - Fixed-length attributes
  - Variable-length attributes
  - NULL values
- Alignment
- Long variable-length attributes
  - Compression
  - TOAST
- Optimization tricks
  - Size
  - Access performance

Postgres data structures for dummies

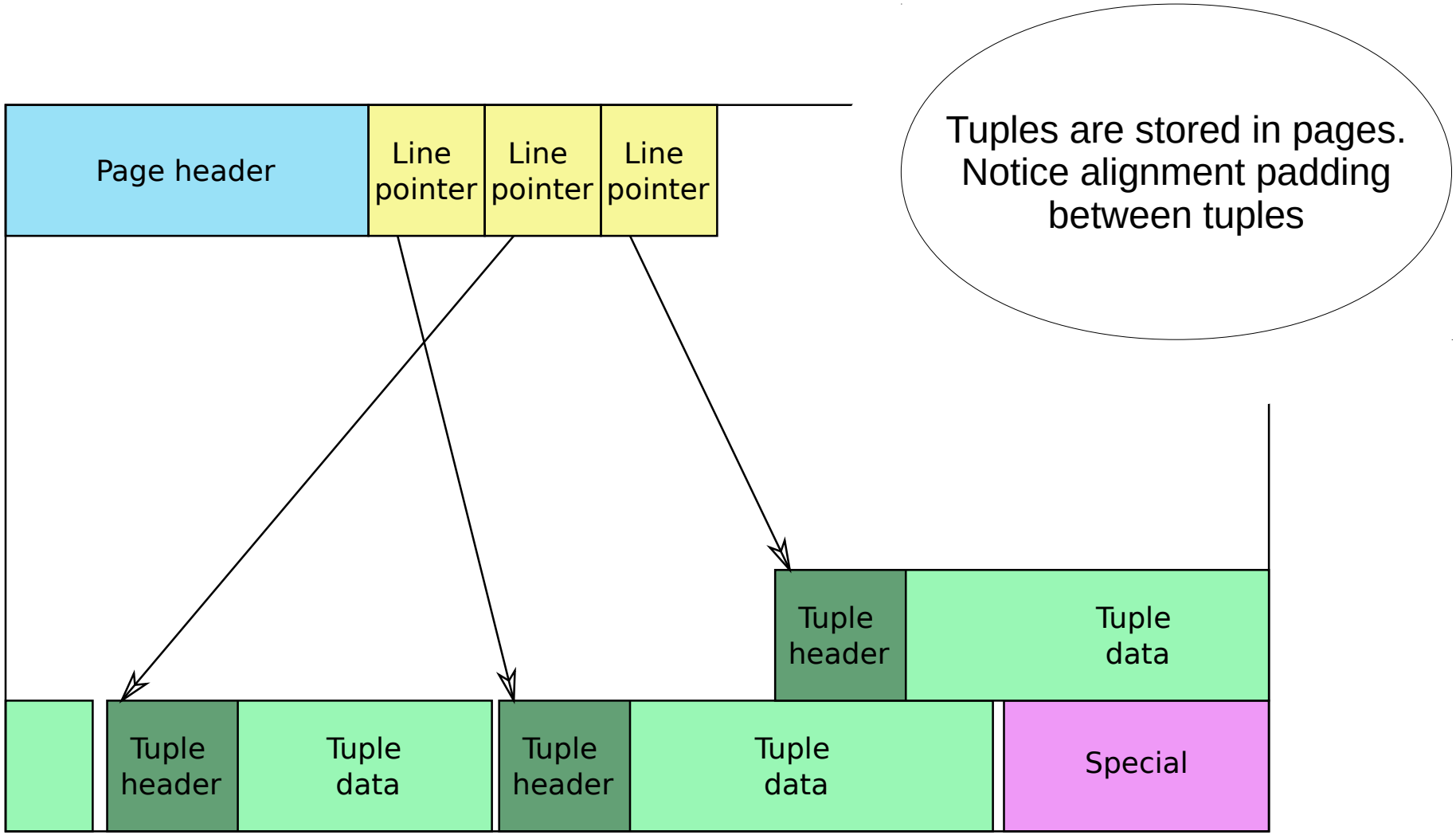
Bruce Momjian, PostgreSQL Internals Through Pictures:

- File structure
- Page structure
- Tuple header structure

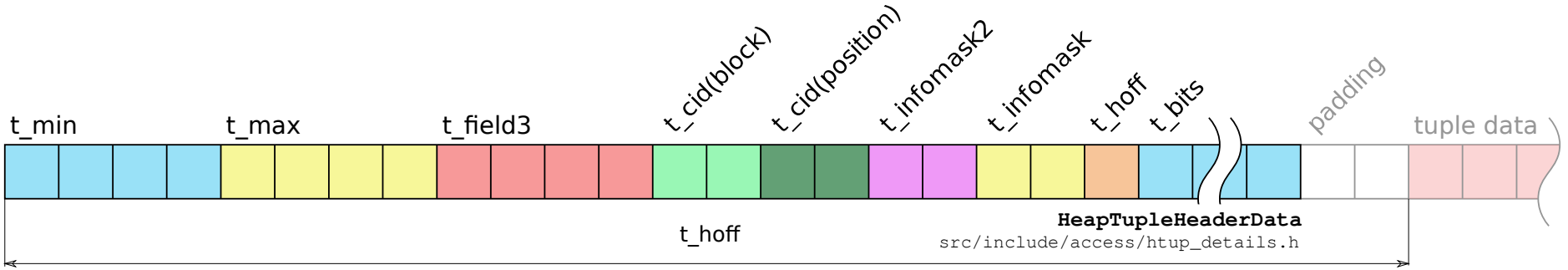
Nikolay Shaplov, Tuple Internals: exposing, exploring & explaining:

- Tuple attributes layout

Page layout


Page header
Line pointer
Line pointer
Line pointer
Tuples are stored in pages. Notice alignment padding between tuples
Tuple header
Tuple data
Tuple header
Tuple data
Tuple header
Tuple data
Special

Tuple header



t_min
t_max
t_field3
} tuple visibility

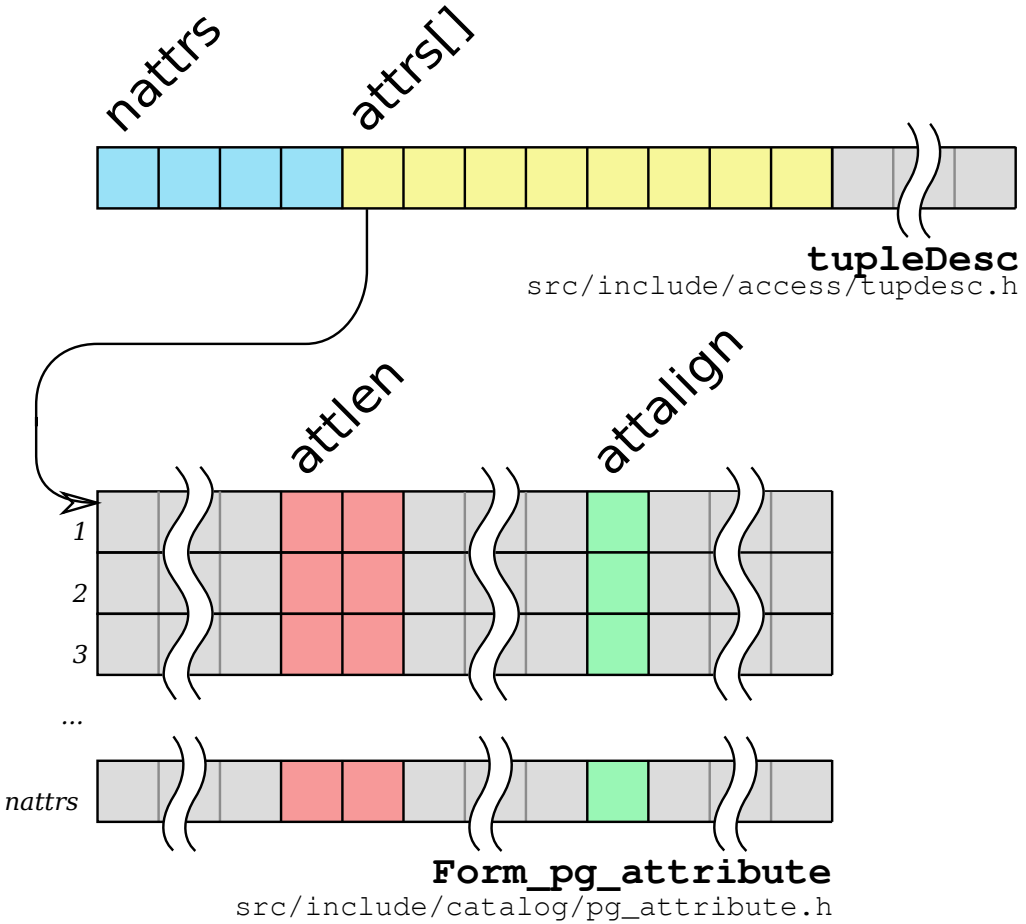
t_cid – tuple location in storage

t_infomask2
t_infomask
} flags + nattrs

t_bits – NULL attributes bitmap
t_hoff – offset of tuple data

There is no information about attributes layout in tuple header

Tuple descriptor



Attributes layout
are stored in tupleDesc,
available from
CreateTupleDescCopyConstr
function for each relation

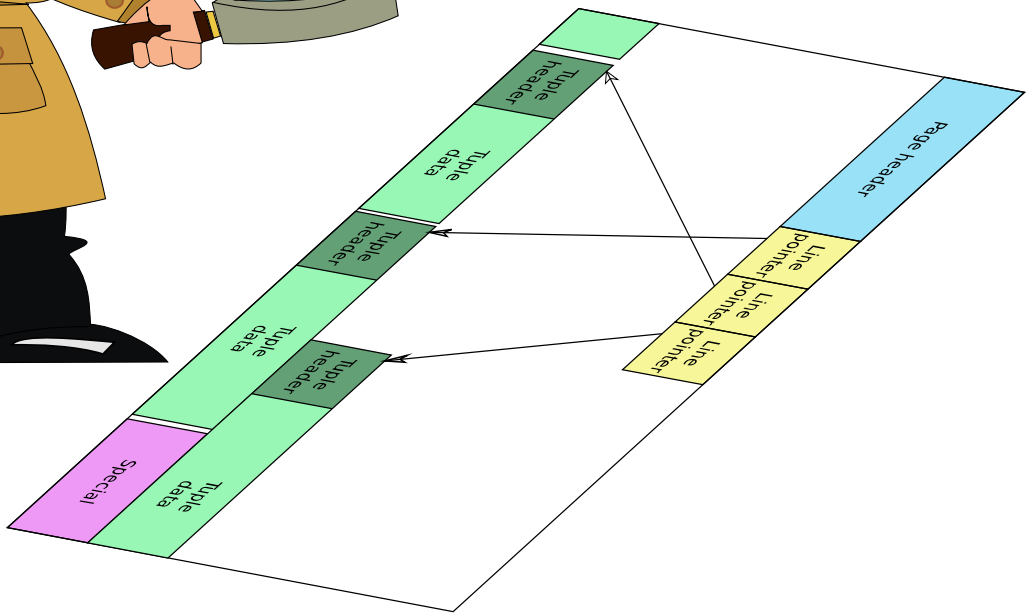
attlen>0 – fixed-length attribute
attlen=-1 – variable-length attribute

attalign:

c – no alignment
s – 2-bytes alignment
i – 4 bytes alignment
d – 8 or 4 byte alignment

Pageinspect extension

Patches d6061f8 + 0271e27 allows to look into tuple raw data. Will be available in postgres 9.6. Available in Postgres Pro 9.5 now!



Fixed-length attribute types

```
test=# create table test (a int, b int, c int);
CREATE TABLE
test=# insert into test VALUES (1,2,3);
INSERT 0 1
test=# select lp, t_data from heap_page_items(get_raw_page('test', 0));
```

```
 lp  | t_data
-----+--------------------------------
 1   | \x010000000200000003000000
```

Numbers are stored as numbers :-)

```
test=# select * from heap_page_item_attrs(get_raw_page('test',0),'test'::regclass);
```

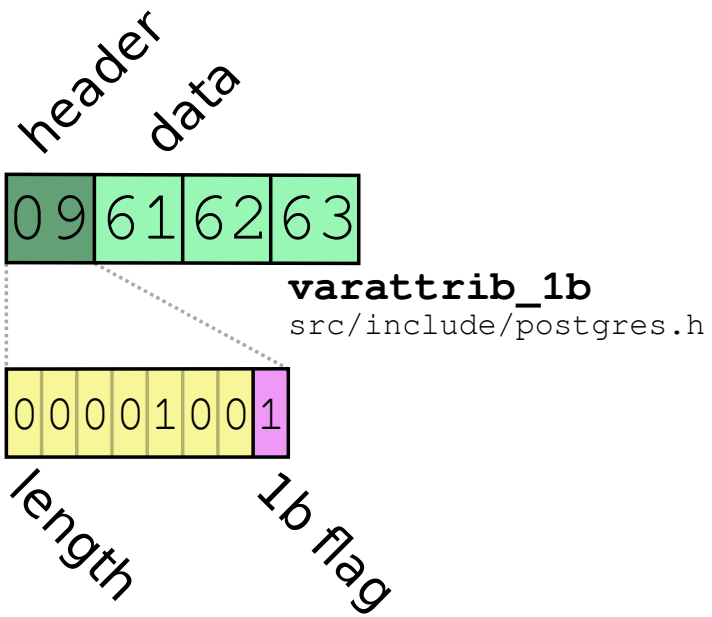
```
 lp  | t_attrs
-----+-----------------------------------------
 1   | {"\\x01000000","\\x02000000","\\x03000000"}
```

Variable-length attribute types 1-byte header

```
test=# create table test (s varchar);
CREATE TABLE
test=# insert into test VALUES ('abcd'),('abc');
INSERT 0 1
test=# select lp, t_data from
                    heap_page_items(get_raw_page('test', 0));
```

One byte header for strings up to 126 bytes length

```
 lp  | t_data
-----+---------------
 1   | \x0b61626364
 2   | \x09616263
```

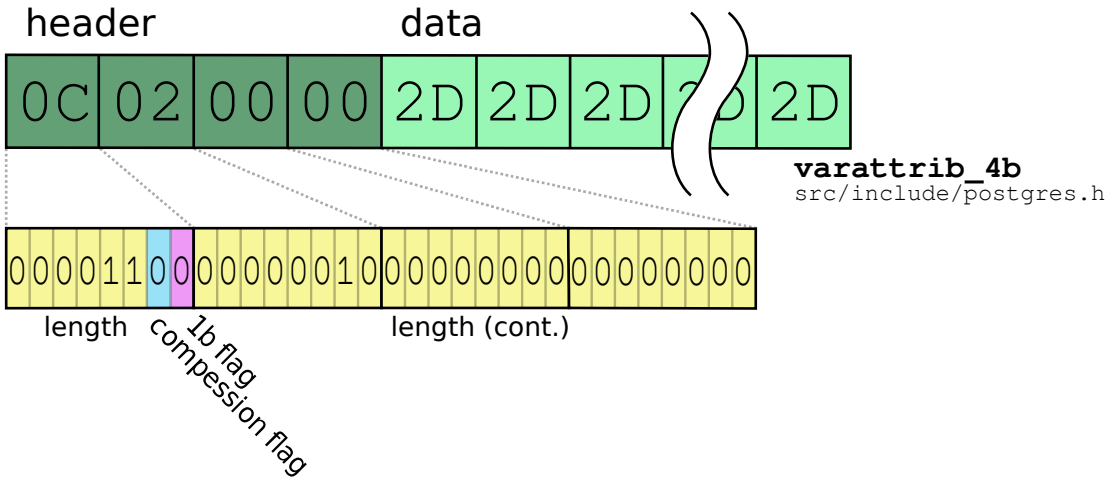


Variable-length attribute types 4-byte header

```
test=# create table test (a varchar);
CREATE TABLE
test=# insert into test VALUES (repeat('+',126)),(repeat('-',127));
INSERT 0 1
test=# select lp, t_data from
                heap_page_items(get_raw_page('test', 0));
```

Four byte header is for strings longer than 126 bytes, or for compressed strings

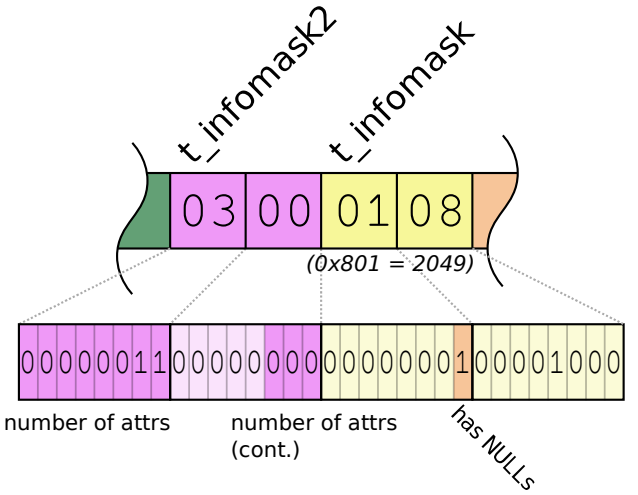
```
 lp |  t_data
----+----------------------
  1 | \xff2b2b2b2b2b2b2b2b2b2b2b2b2b2b2b...
  2 | \x0c0200002d2d2d2d2d2d2d2d2d2d2d2d2d...
```



Infomask and Infomask2

```
test=# create table test (a int, b int, c int);
CREATE TABLE
test=# insert into test VALUES (1,2,3),(1,null,3);
INSERT 0 2
test=# select lp, t_infomask2, t_infomask, t_bits, t_data from heap_page_items(get_raw_page('test', 0));
```

lp	t_infomask2	t_infomask	t_bits	t_data
1	3	2048		\x01000000020000003000000
2	3	2049	10100000	\x010000003000000



src/include/access/htup_details.h

t_infomask stores various flags, including has NULLs flag; t_infomask2 stores number of attributes and some flags

NULLs

```
test=# create table test (a int, b int, c int);
CREATE TABLE
test=# insert into test VALUES (1,2,3),(1,null,3);
INSERT 0 2
test=# select lp, t_infomask, t_bits, t_data from
                                heap_page_items(get_raw_page('test', 0));
```

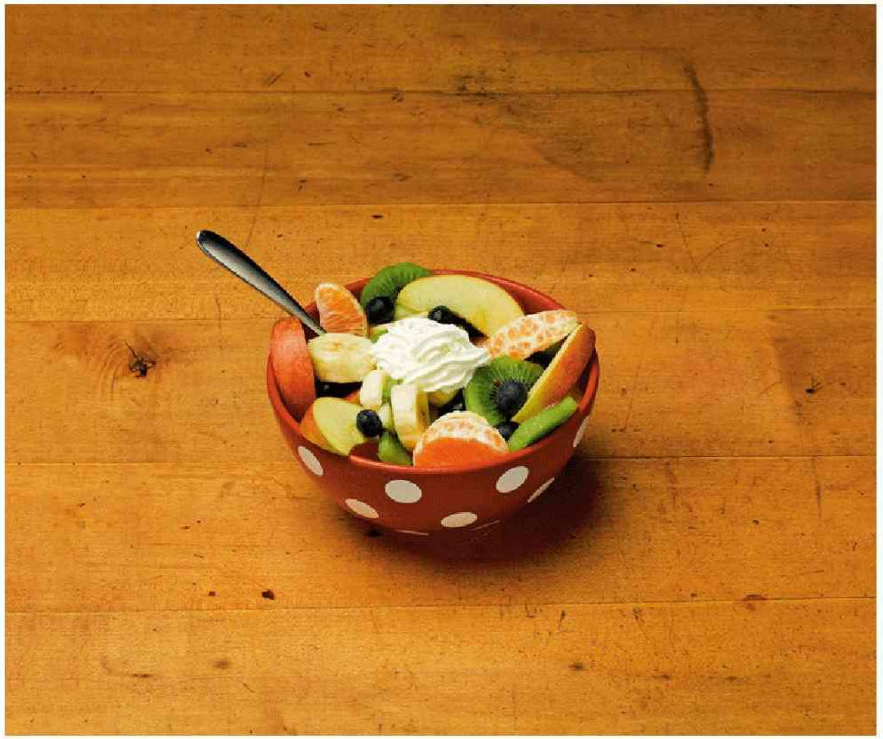
NULL values are not stored in tuple data

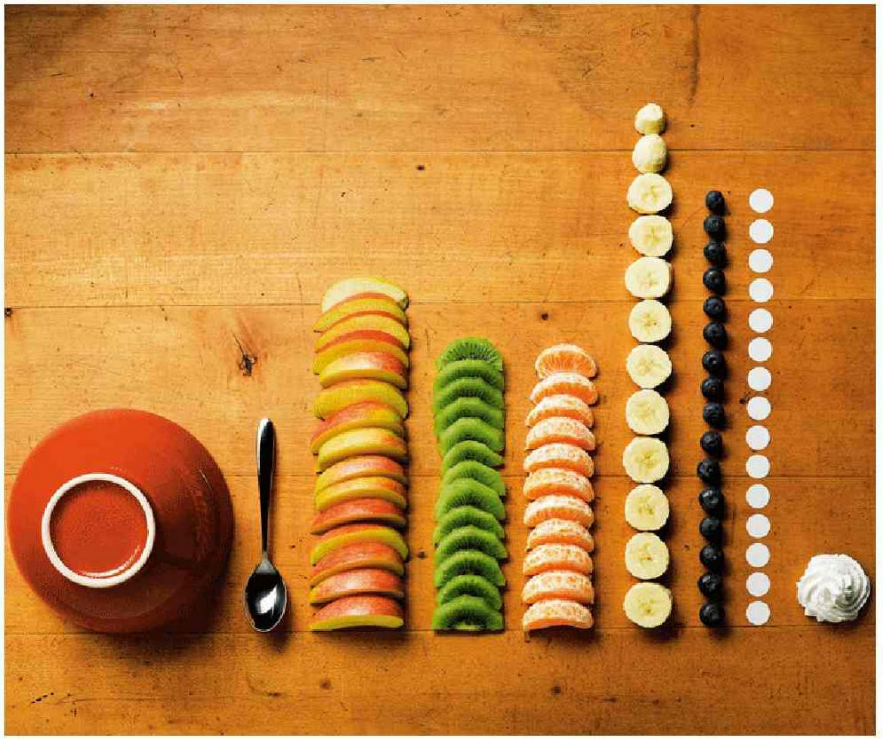
```
 lp  | t_infomask | t_bits   | t_data
-----+------------+----------+----------------------------
 1   |       2048 |          | \x010000000200000003000000
 2   |       2049 | 10100000 | \x0100000003000000
```

```
test=# select lp, t_infomask, t_bits, t_attrs from
                                heap_page_item_attrs(get_raw_page('test',0),'test'::regclass)
```

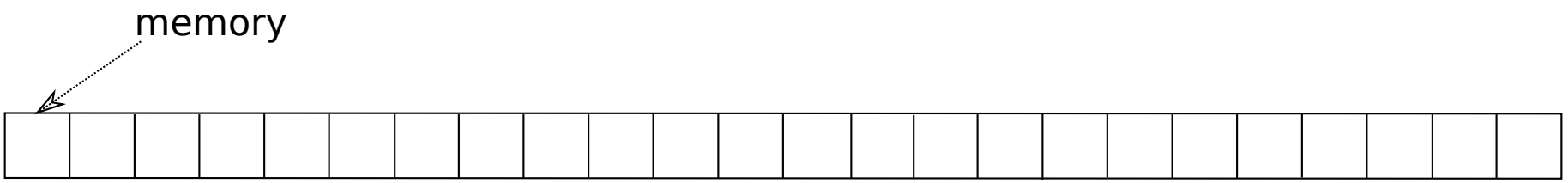
```
 lp | t_infomask | t_bits   | t_attrs
----+------------+----------+-----------------------------------------
 1  |       2048 |          |{"\\x01000000","\\x02000000","\\x03000000"}
 2  |       2049 |10100000  |{"\\x01000000",NULL,"\\x03000000"}
```

Alignment

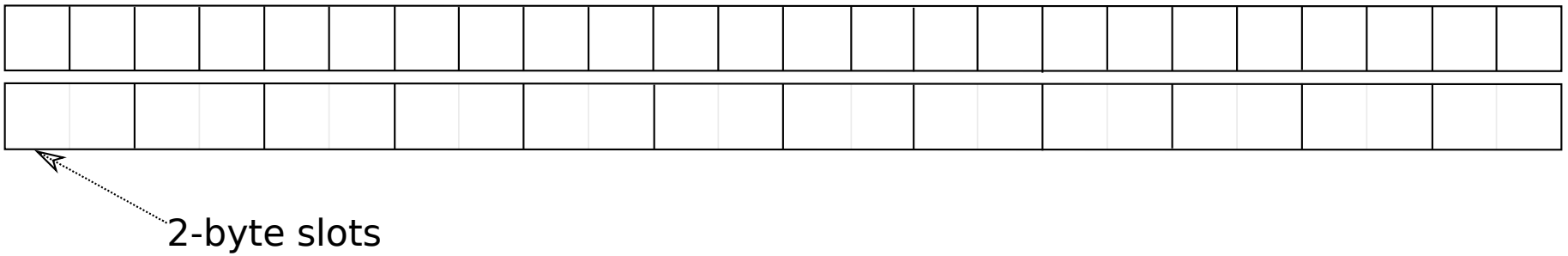




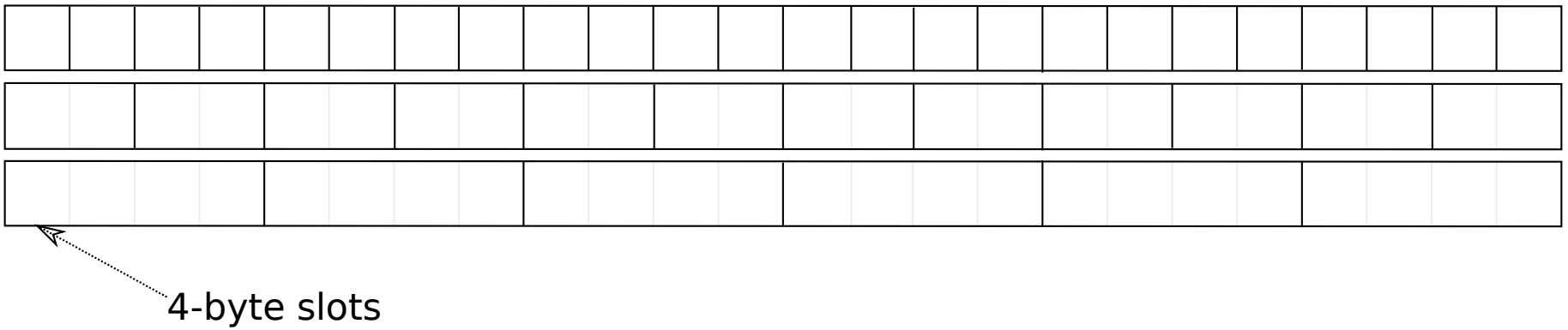
Alignment



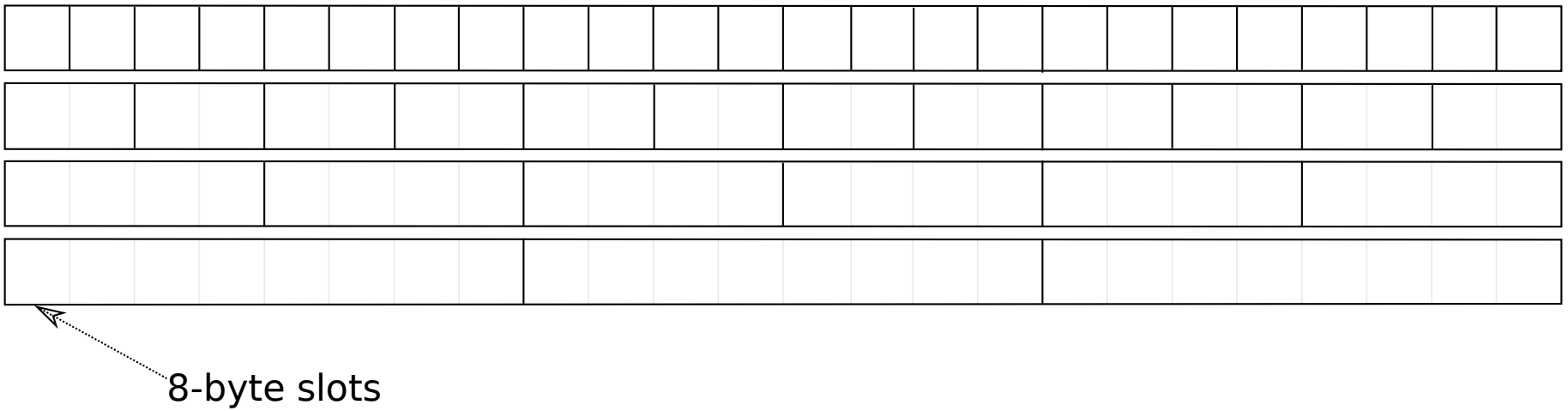
Alignment



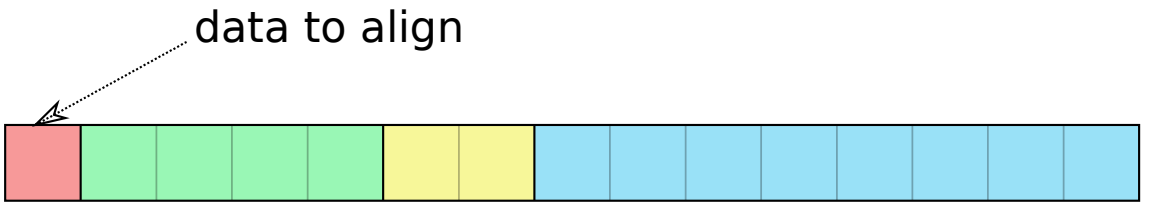
Alignment



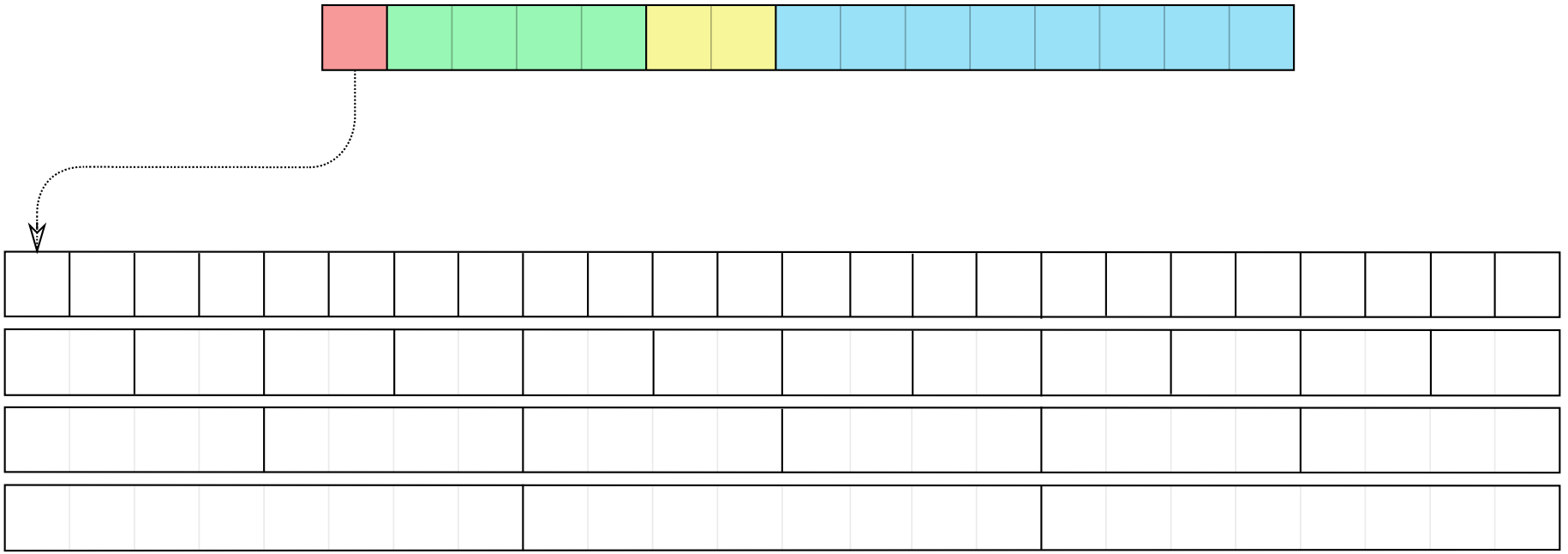
Alignment



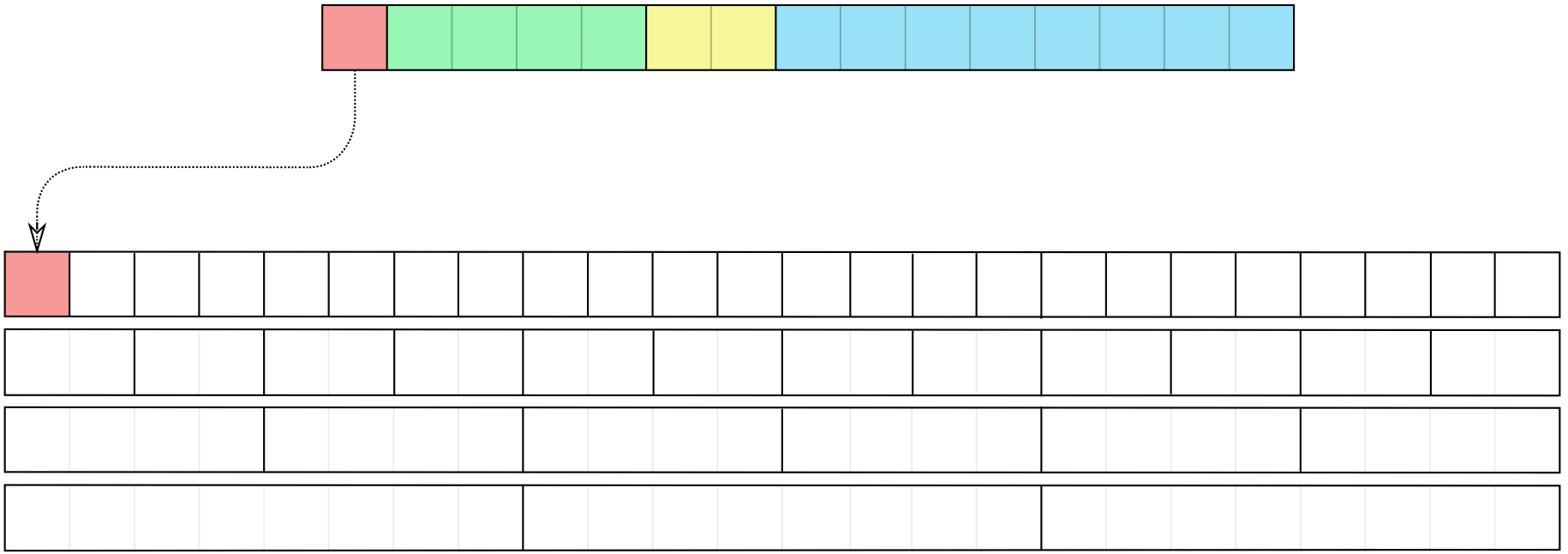
Alignment



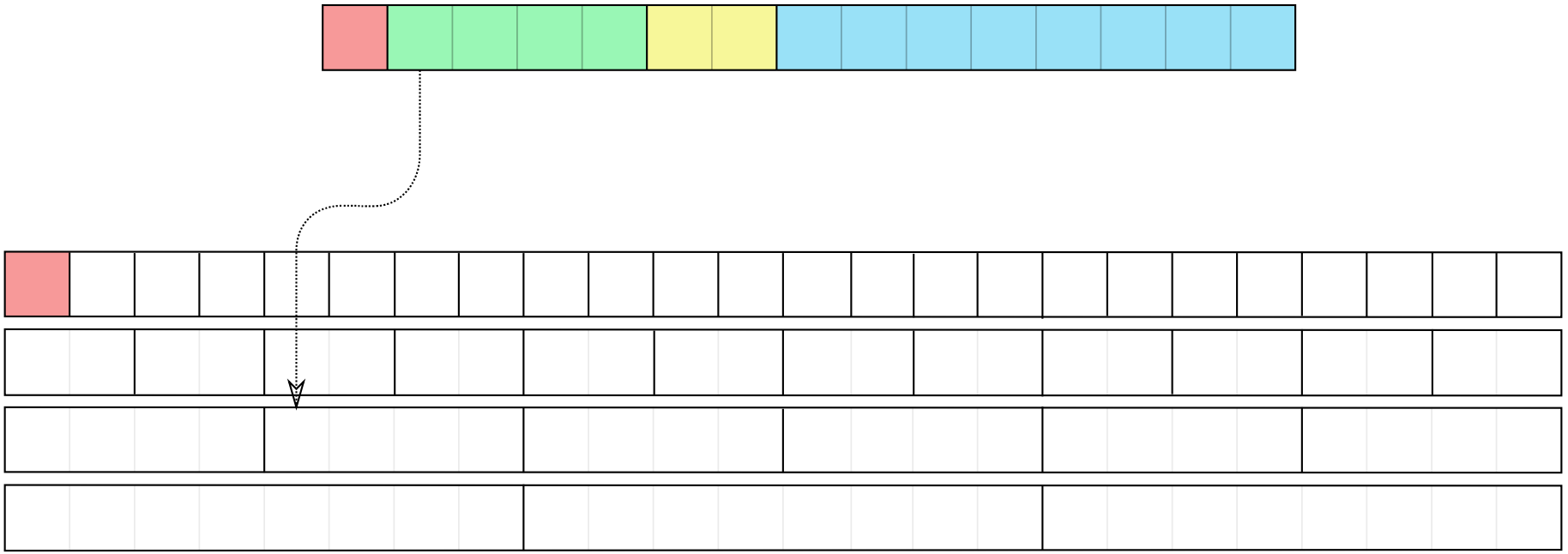
Alignment



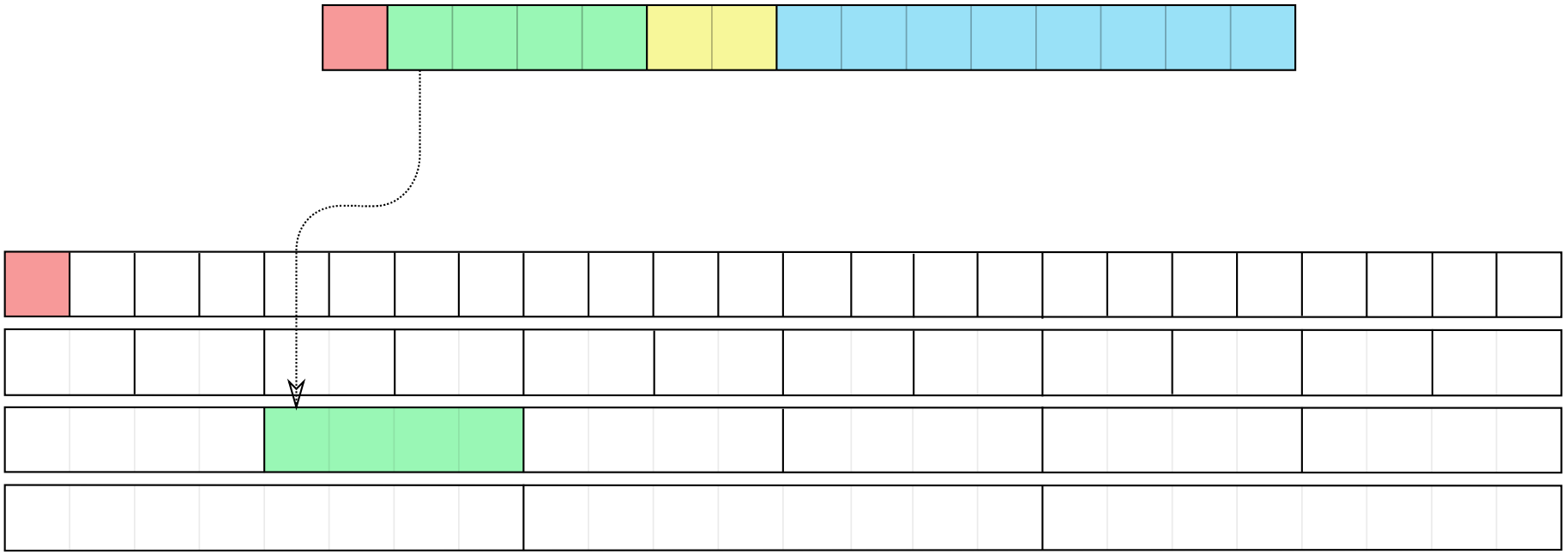
Alignment



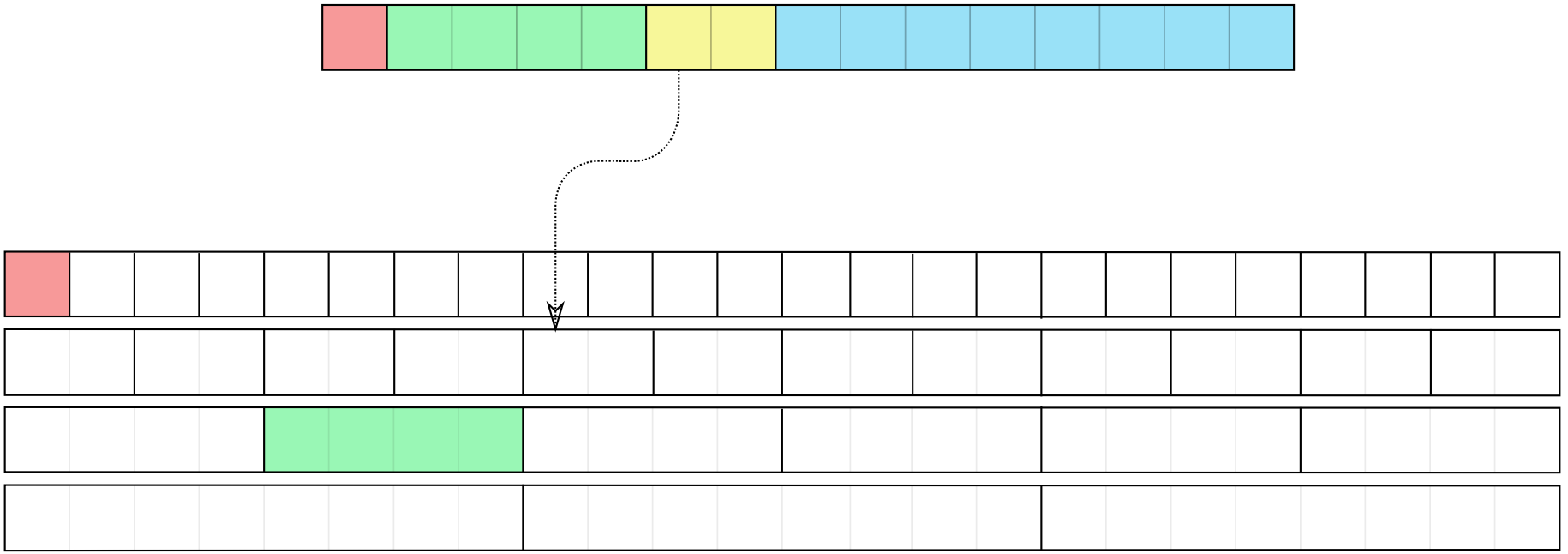
Alignment



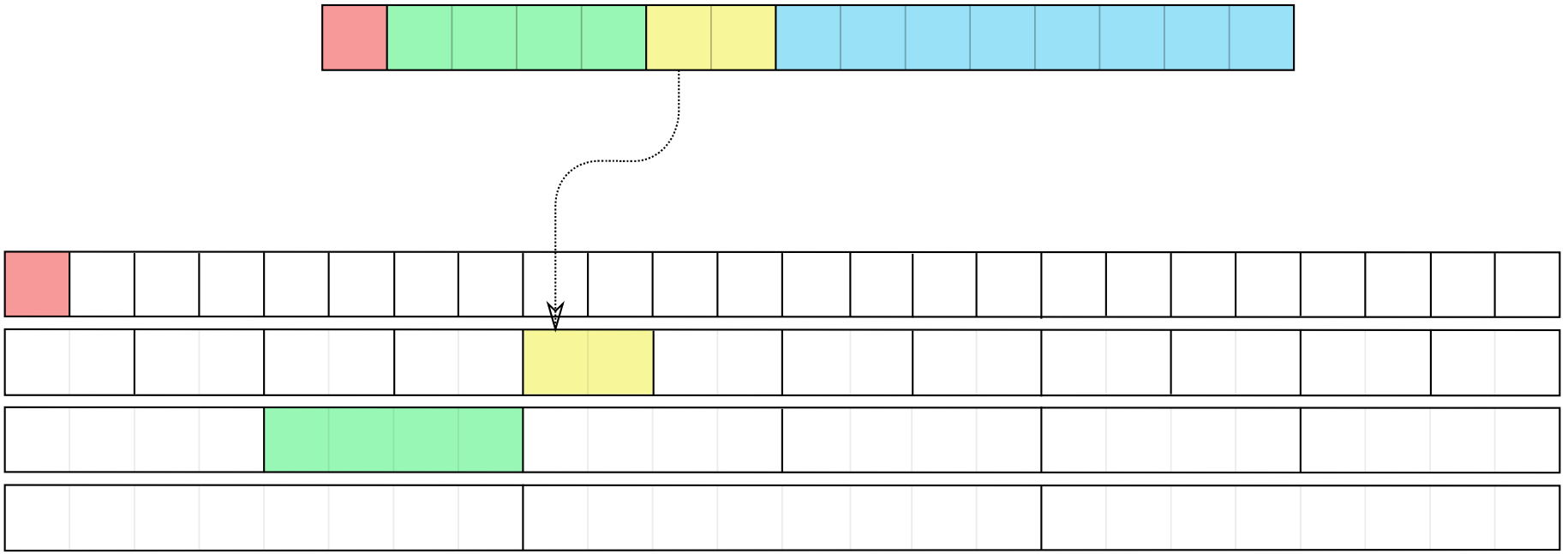
Alignment



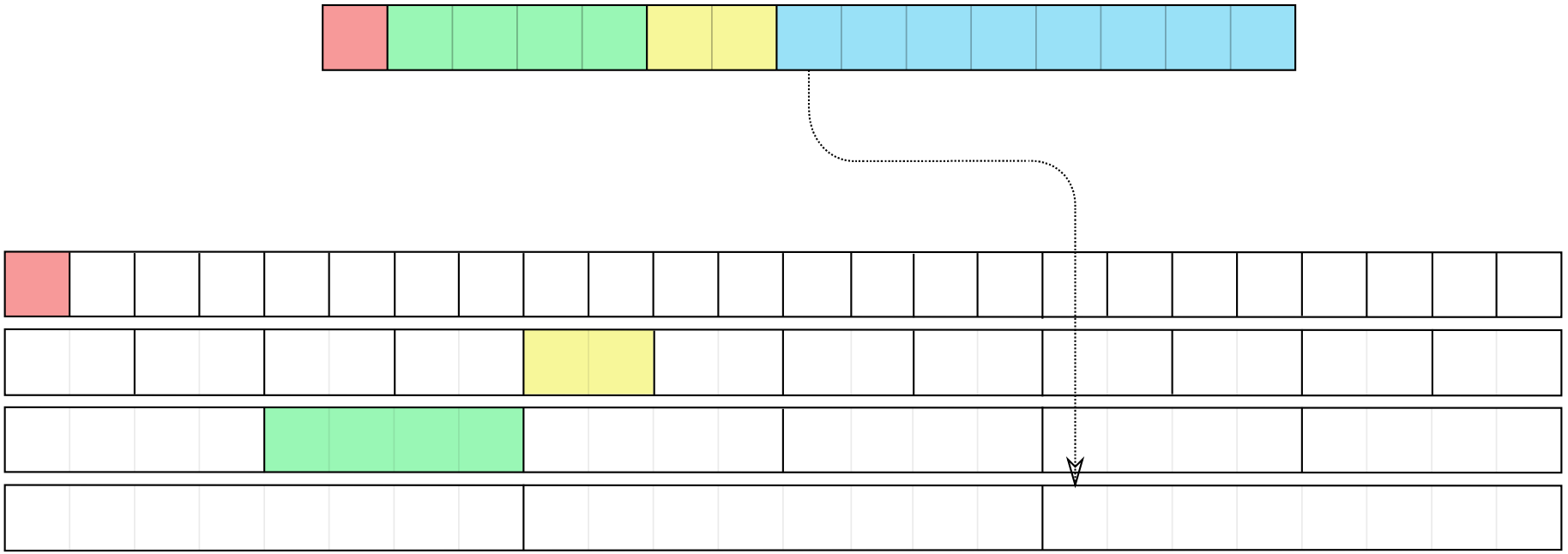
Alignment



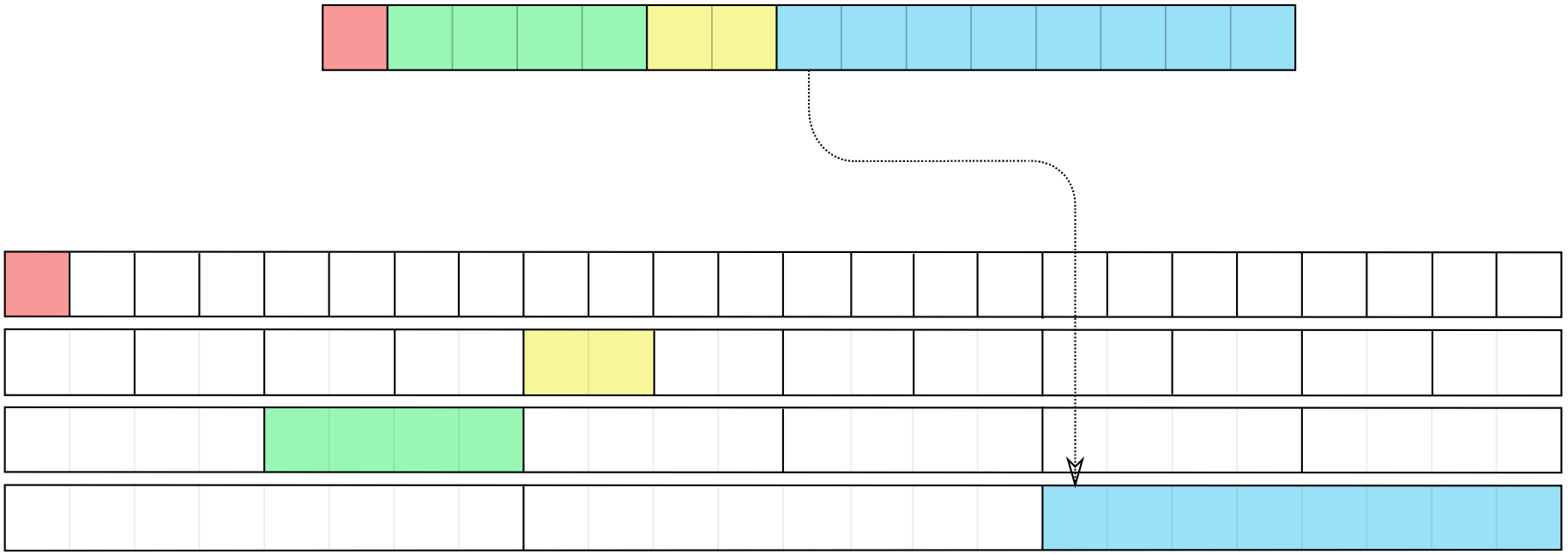
Alignment



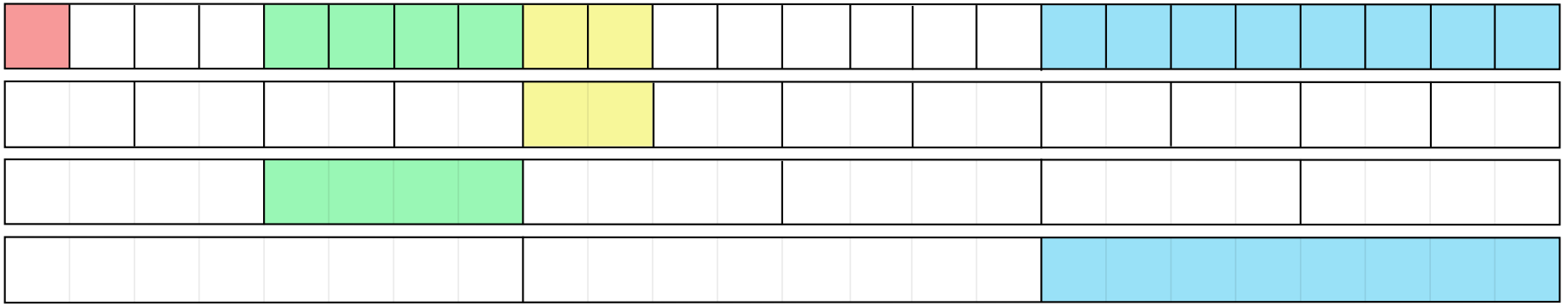
Alignment



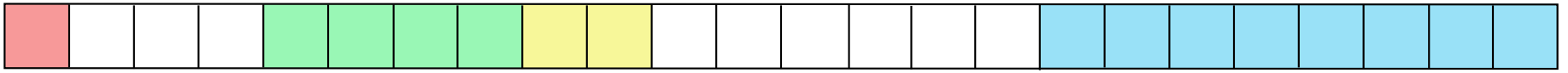
Alignment



Alignment



Alignment

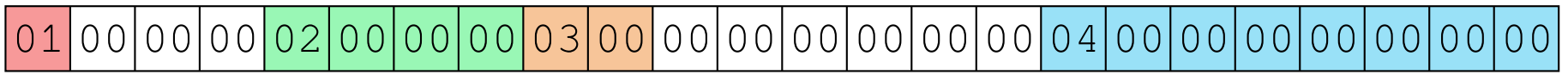


Alignment of fixed-length attributes

```
test=# create table test (a boolean, b int, c smallint, d bigint);
CREATE TABLE
test=# insert into test VALUES ('t',2,3,4);
INSERT 0 1
test=# select lp, t_data from
                heap_page_items(get_raw_page('test', 0));
```

Fixed-length attributes are stored properly aligned

```
 lp |                          t_data
----+-----------------------------------------------------
  1 | \x0100000002000000030000000000000400000000000000
```

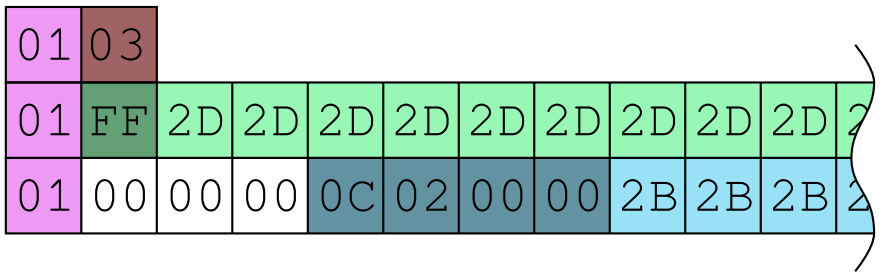


Alignment of variable-length attributes

```
test=# create table test (a boolean, b varchar);
CREATE TABLE
test=# insert into test VALUES ('t',''),
test-#                         ('t',repeat('-',126)),
test-#                         ('t',repeat('+',127));
INSERT 0 3
test=# select lp, t_data from heap_page_items(get_raw_page('test', 0));
```

```
lp | t_data
---+------------------------------
1  | \x0103
2  | \x01ff2d2d2d2d2d2d2d2d...
3  | \x010000000c0200002b2b2b2b2b...
```

For variable-length attributes header is aligned: 4-byte header aligned as four byte value, 1-byte header is not aligned at all



Alignment on 64- and 32-bit CPU

```
# create table test (b int, c bigint);
CREATE TABLE
# insert into test VALUES (1,2);
INSERT 0 1
```

Alignment works in different ways on different CPUs

64bit

```
lp | t_data
---+-----------------------------------------
1  | \x0100000000000000020000000000000000
```

32bit

```
lp | t_data
---+-----------------------------
1  | \x0100000002000000000000000
```

"Missing" attributes in tuple

```
test=# create table test (a int, b int);
CREATE TABLE
test=# insert into test VALUES (1,10);
INSERT 0 2
test=# ALTER TABLE test ADD COLUMN c int;
ALTER TABLE
test=# insert into test VALUES (3,30,300);
INSERT 0 1
```

Postgres does not update tuples when you add column with NULL default value, instead it just returns NULL when user asks for that "missing" attribute

```
test=# select lp, t_infomask2, t_data from heap_page_items(get_raw_page('test', 0));

lp   |   t_infomask2   | t_data
----+-------------+----------------------------
1    | 2             | \x010000000a000000
2    | 3             | \x030000001e0000002c010000
```

```
test=# select lp, t_infomask2, t_attrs from
                   heap_page_item_attrs(get_raw_page('test',0),'test'::regclass);

lp  | t_infomask2 | t_attrs
---+----------+---------------------------------------
1   | 2         | {"\\x01000000","\\x0a000000",NULL}
2   | 3         | {"\\x03000000","\\x1e000000","\\x2c010000"}
```

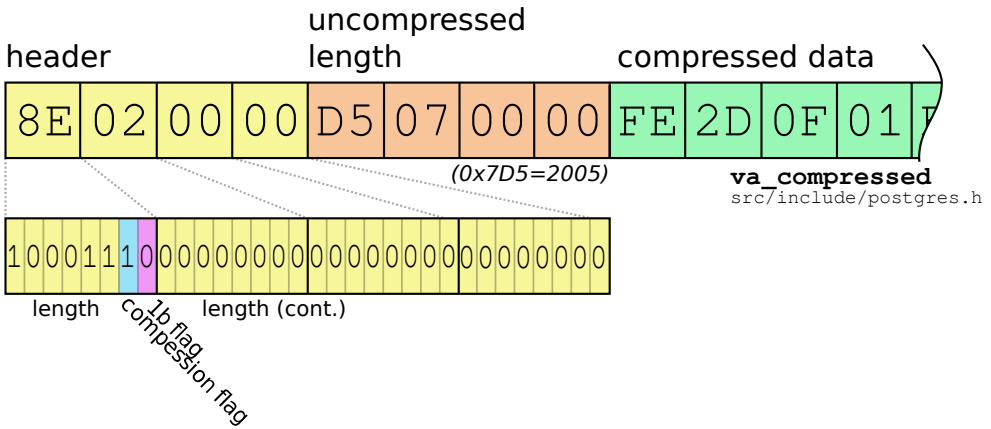
Big data storage

- Compression
- TOAST

Compressed attributes

Postgres tries to compress variable-length attributes with LZ algorithm if tuple size is ≳ 2K

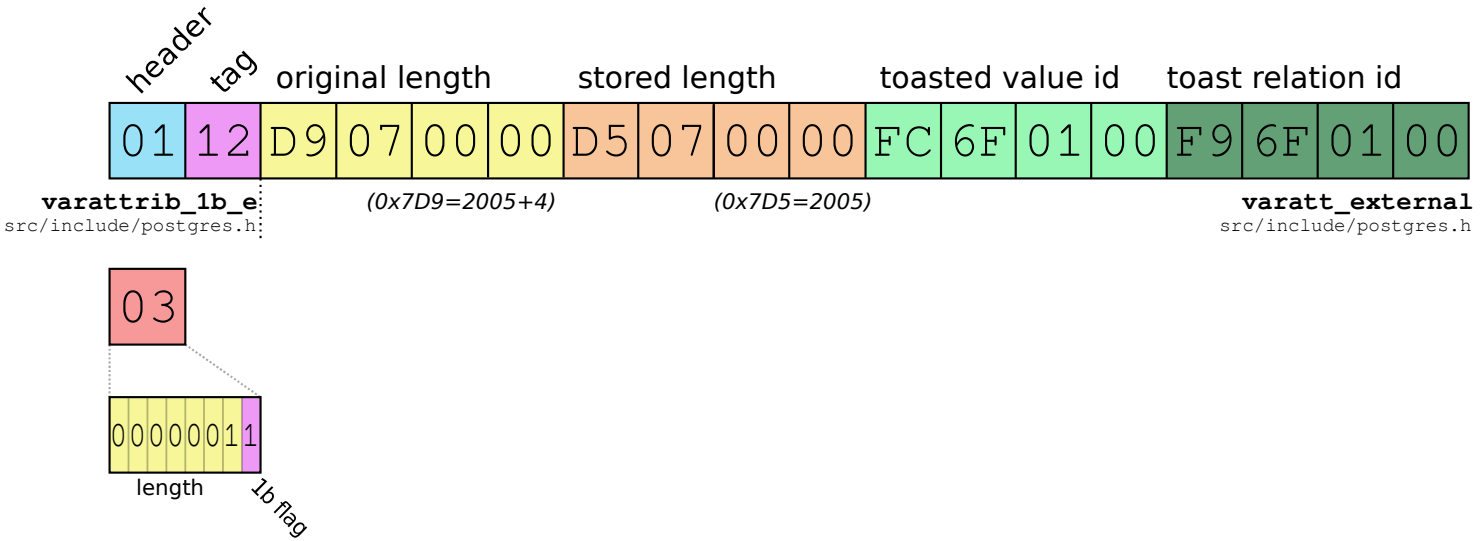
```
test=# create table test (a varchar);
CREATE TABLE
test=# insert into test VALUES (repeat('-',2004)),
test-#                         (repeat('-',2005));
INSERT 0 2
test=# select lp, t_data from heap_page_items(get_raw_page('test', 0));

lp |    t_data
---+-----------------------------------------------------------------------------
1  | \x601f00002d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d2d.....
2  | \x8e000000d5070000fe2d0f01ff0f01ff0f01ff0f01ff0f01ff0f01ff0f01ff010f014b
```



TOASTed attributes

```
test=# create table test (a varchar);
CREATE TABLE
test=# ALTER TABLE test ALTER a SET STORAGE EXTERNAL;
ALTER TABLE
test=# insert into test VALUES (repeat('-',2005),
test-#                          (''));
INSERT 0 2
test=# select lp, t_attrs from
                    heap_page_items(get_raw_page('test', 0));
```

```
lp  |  t_attrs
----+-----------------------------------------
1   | \x0112d9070000d5070000fc6f0100f96f0100
2   | \x03
```



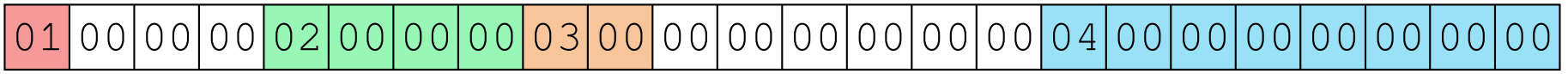
Empty string has 0x03 one byte header. 0x01 is unreachable, so it is used as TOASTed string marker

Postgres PROFESSIONAL

Optimization

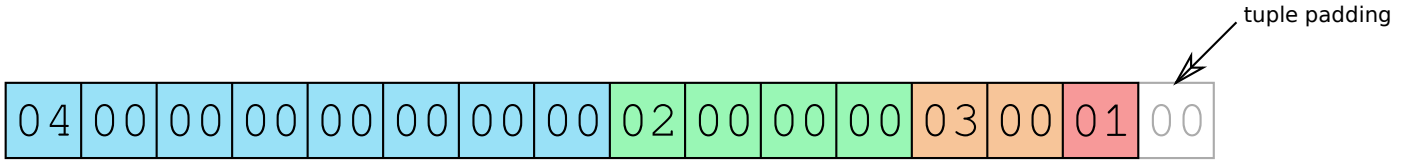
Size optimization

```
test=# create table test (a boolean, b int, c smallint, d bigint);
```



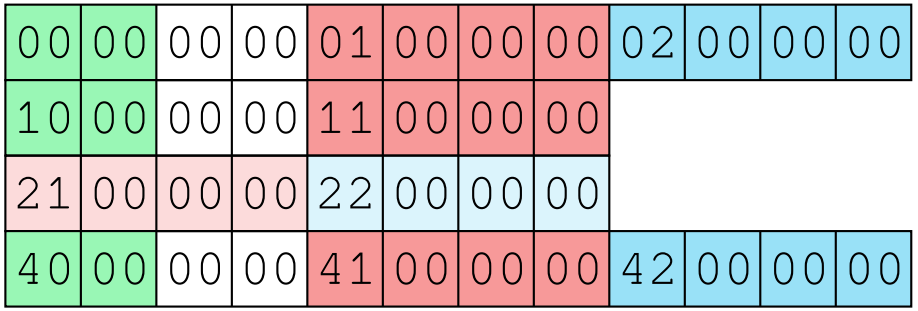
Reorder attributes in to get rid of unnecessary padding

```
test=# create table test (d bigint, b int, c smallint, a boolean);
```



Access performance optimization

```
create table test (a smallint, b int, c int);
insert into test VALUES (0,   1,  2),
                        (16,  17, NULL),
                        (NULL,33, 34),
                        (64,  65, 66);
```



If there are no NULLs and variable-length attributes, position of an attribute is predictable and can be cached

optimization

Access performance

- Not NULL fixed-length attributes go first
- Then put fixed-length attributes with less NULLs
- Move all variable-length attributes to the right

optimization

- 1 000 bigint attributes: N1..N1000
- One string in the head or at the tail of a row
- 1 000 000 rows
- Benchmark query: SELECT SUM(N1000)

Lets benchmark access performance with varchar at the beginning and in the end of tuple

Test Environment

CPU: 4 socket Xeon V7, 15 Core per socket

Shared Memory: 100Gb

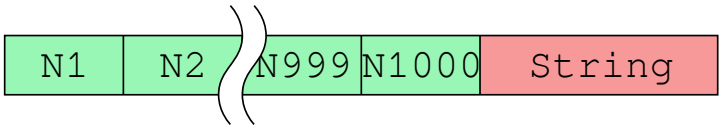
pgbench

clients: 100

jobs: 100

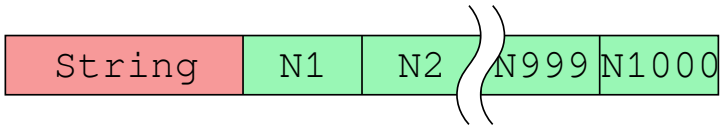
time: 360

String is last



tps = 64.373621

String is first



tps = 50.387503

64.373621 / 50.387503 = **1.27**

Postgres PROFESSIONAL

Postgres PROFESSIONAL

Look for a newer versions at

https://github.com/dhyannataraj/tuple-internals-presentation