>>

PostgreSQL Page Internals

- PostgreSQL Page Representation
- TOAST'ing

❖ PostgreSQL Page Representation

Functions: **`src/backend/storage/page/*.c`**

Definitions: **`src/include/storage/bufpage.h`**

Each page is 8KB (default **`BLCKSZ`**) and contains:

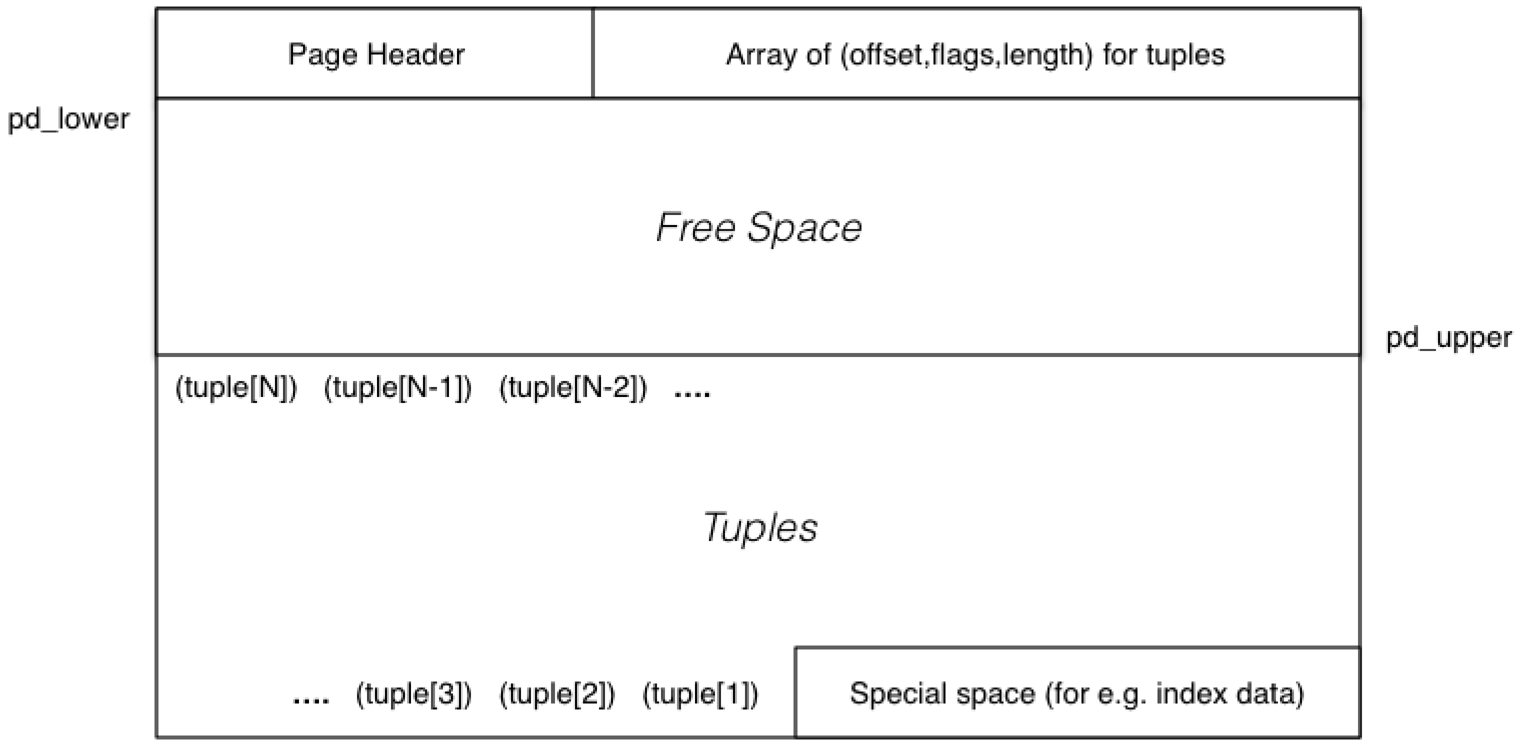
- header (free space pointers, flags, xact data)
- array of (offset,length) pairs for tuples in page
- free space region (between array and tuple data)
- actual tuples themselves (inserted from end towards start)
- (optionally) region for special data (e.g. index data)

Large data items are stored in separate (TOAST) files (implicit)

Also supports ~SQL-standard BLOBs (explicit large data items)

❖ PostgreSQL Page Representation (cont)

PostgreSQL page layout:


Page Header
Array of (offset,flags,length) for tuples
pd_lower
Free Space
pd_upper
(tuple[N]) (tuple[N-1]) (tuple[N-2]) ....
Tuples
.... (tuple[3]) (tuple[2]) (tuple[1])
Special space (for e.g. index data)

<< ^ >>

❖ PostgreSQL Page Representation (cont)

Page-related data types:

```
// a Page is simply a pointer to start of buffer
typedef Pointer Page;

// indexes into the tuple directory
typedef uint16  LocationIndex;

// entries in tuple directory (line pointer array)
typedef struct ItemIdData
{
   unsigned   lp_off:15,    // tuple offset from start of page
              lp_flags:2,   // unused,normal,redirect,dead
              lp_len:15;    // length of tuple (bytes)
} ItemIdData;
```

❖ PostgreSQL Page Representation (cont)

Page-related data types: (cont)

```
typedef struct PageHeaderData
{
   XLogRecPtr    pd_lsn;       // xact log record for last change
   uint16        pd_tli;       // xact log reference information
   uint16        pd_flags;     // flag bits (e.g. free, full, ...
   LocationIndex pd_lower;     // offset to start of free space
   LocationIndex pd_upper;     // offset to end of free space
   LocationIndex pd_special;   // offset to start of special space
   uint16        pd_pagesize_version;
   TransactionId pd_prune_xid;// is pruning useful in data page?
   ItemIdData    pd_linp[1];   // beginning of line pointer array
} PageHeaderData;

typedef PageHeaderData *PageHeader;
```

❖ PostgreSQL Page Representation (cont)

Operations on **`Page`**s:

```
void PageInit(Page page, Size pageSize, ...)
```

- initialize a **`Page`** buffer to empty page
- in particular, sets **`pd_lower`** and **`pd_upper`**

```
OffsetNumber PageAddItem(Page page,
                         Item item, Size size, ...)
```

- insert one tuple (or index entry) into a **`Page`**
- fails if: not enough free space, too many tuples

```
void PageRepairFragmentation(Page page)
```

- compact tuple storage to give one large free space region

❖ PostgreSQL Page Representation (cont)

PostgreSQL has two kinds of pages:

- heap pages which contain tuples
- index pages which contain index entries

Both kinds of page have the same page layout.

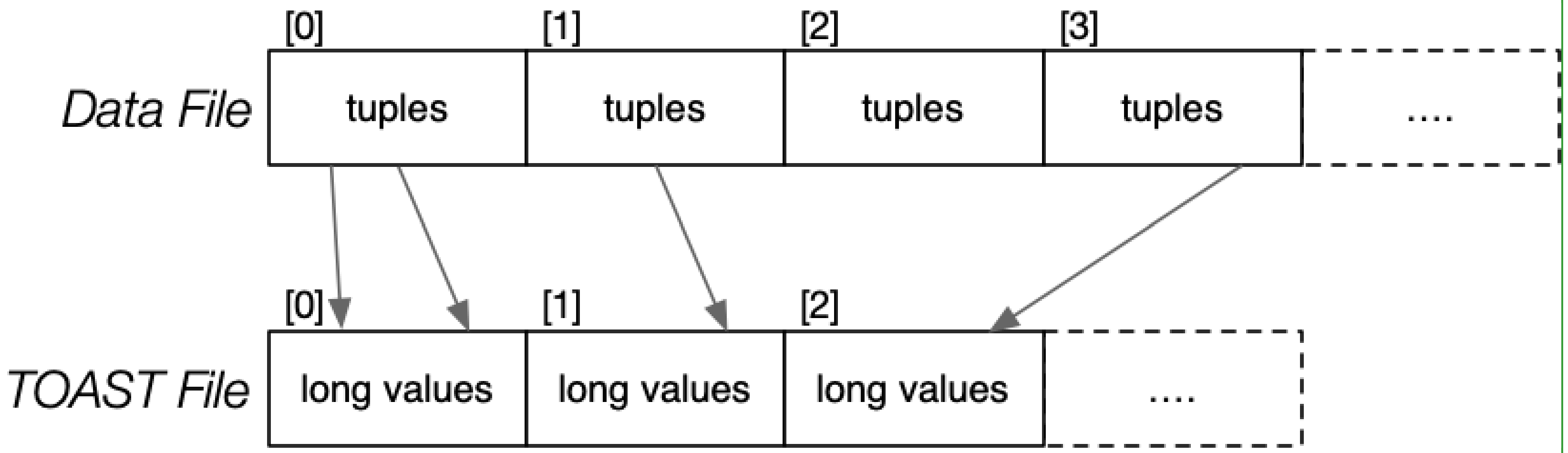
One important difference:

- index entries tend be a smaller than tuples
- can typically fit more index entries per page

❖ TOAST'ing

TOAST = The Oversized-Attribute Storage Technique

- handles storage of large attribute values (> 2KB) (e.g. long **`text`**)



<< ^

❖ TOAST'ing (cont)

Large attribute values are stored out-of-line (i.e. in separate file)

- "value" of attribute in tuple is a reference to TOAST data
- TOAST'd values may be compressed
- TOAST'd values are stored in 2K chunks

Strategies for storing TOAST-able columns ...

- **`PLAIN`** ... allows no compression or out-of-line storage
- **`EXTENDED`** ... allows both compression and out-of-line storage
- **`EXTERNAL`** ... allows out-of-line storage but not compression
- **`MAIN`** ... allows compression but not out-of-line storage

Produced: 23 Feb 2021