>>

# PostgreSQL Tuples

^ >>

## ❖ PostgreSQL Tuples

Definitions: **`include/postgres.h`**, **`include/access/*tup*.h`**

Functions: **`backend/access/common/*tup*.c`** e.g.

- **`HeapTuple heap_form_tuple(desc,values[],isnull[])`**
- **`heap_deform_tuple(tuple,desc,values[],isnull[])`**

PostgreSQL implements tuples via:

- a contiguous chunk of memory
- starting with a header giving e.g. #fields, nulls
- followed by data values (as a sequence of **`Datum`**)

## ❖ PostgreSQL Tuples (cont)

**`HeapTupleData`** contains information about a stored tuple

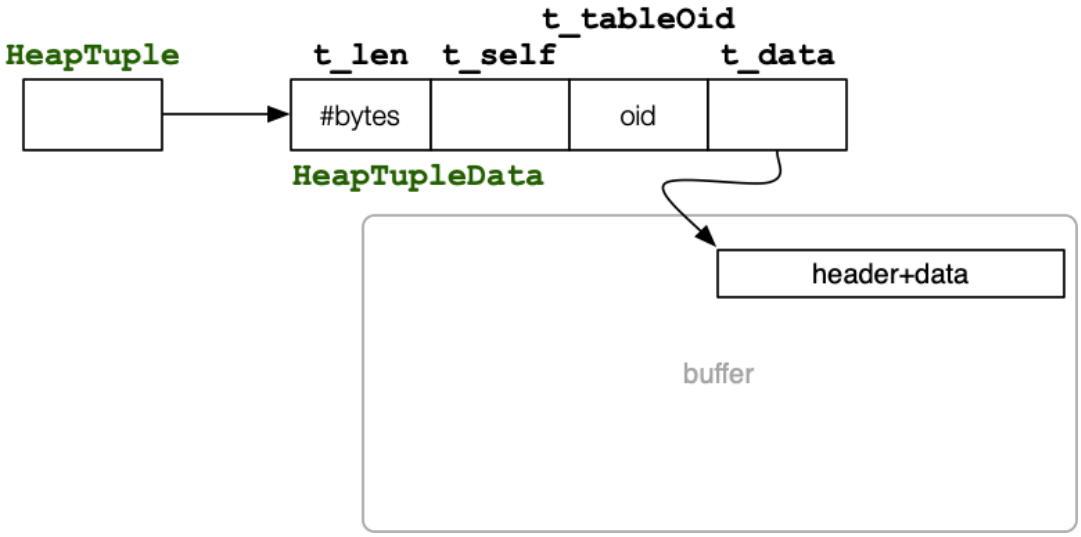
```
typedef HeapTupleData *HeapTuple;

typedef struct HeapTupleData
{
  uint32            t_len;  // length of *t_data
  ItemPointerData t_self;   // SelfItemPointer
  Oid          t_tableOid;  // table the tuple came from
  HeapTupleHeader t_data;   // -> tuple header and data
} HeapTupleData;
```

**`HeapTupleHeader`** is a pointer to a location in a buffer

<< ^ >>

## ❖ PostgreSQL Tuples (cont)

Structure of **`HeapTuple`**:



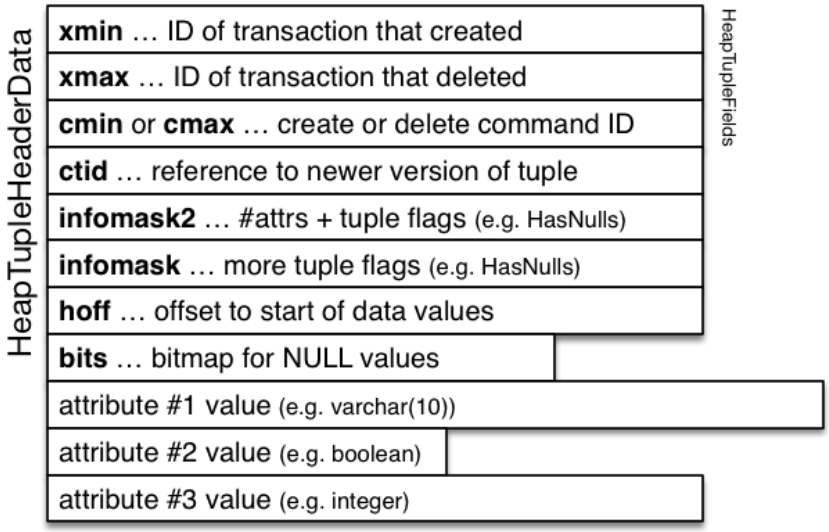
## ❖ PostgreSQL Tuples (cont)

PostgreSQL stores each record as tuple header, followed by data:

```
typedef HeapTupleHeaderData *HeapTupleHeader;

typedef struct HeapTupleHeaderData // simplified
{
  HeapTupleFields t_heap;
  ItemPointerData t_ctid;       // TID of newer version
  uint16          t_infomask2; // #attributes + flags
  uint16          t_infomask;  // flags e.g. has_null
  uint8           t_hoff;      // sizeof header incl. t_bits
  // above is fixed size (23 bytes) for all heap tuples
  bits8           t_bits[1];   // bitmap of NULLs, var.len.
  // OID goes here if HEAP_HASOID is set in t_infomask
  // actual data follows at end of struct
} HeapTupleHeaderData;
```

## ❖ PostgreSQL Tuples (cont)

Tuple structure:



## ❖ PostgreSQL Tuples (cont)

Some of the bits in **`t_infomask`** ..

```
#define HEAP_HASNULL        0x0001
          /* has null attribute(s) */
#define HEAP_HASVARWIDTH    0x0002
          /* has variable-width attribute(s) */
#define HEAP_HASEXTERNAL    0x0004
          /* has external stored attribute(s) */
#define HEAP_HASOID_OLD     0x0008
          /* has an object-id field */
```

Location of **`NULL`**s is stored in **`t_bits[]`** array

## ❖ PostgreSQL Tuples (cont)

Tuple-related data types: (cont)

```
typedef struct HeapTupleFields  // simplified
{
  TransactionId t_xmin;  // inserting xact ID
  TransactionId t_xmax;  // deleting or locking xact ID
  union {
    CommandId   t_cid;   // inserting or deleting command ID
    TransactionId t_xvac;// old-style VACUUM FULL xact ID
  } t_field3;
} HeapTupleFields;
```

Note that not all system fields from stored tuple appear

- **oid** is stored after the tuple header, if used
- both **xmin**/**xmax** are stored, but only one of **cmin**/**cmax**

## ❖ PostgreSQL Tuples (cont)

Tuple-related data types: (cont)

```
// TupleDesc: schema-related information for HeapTuples

typedef struct tupleDesc
{
  int           natts;        // # attributes in tuple
  Oid           tdtypeid;     // composite type ID for tuple type
  int32         tdtypmod;     // typmod for tuple type
  bool          tdhasoid;     // does tuple have oid attribute?
  int           tdrefcount;   // reference count (-1 if not counting)
  TupleConstr *constr;        // constraints, or NULL if none
  FormData_pg_attribute attrs[];
  // attrs[N] is a pointer to description of attribute N+1
} *TupleDesc;
```

<< ^ >>

## ❖ PostgreSQL Tuples (cont)

Tuple-related data types: (cont)

```
// FormData_pg_attribute:
// schema-related information for one attribute

typedef struct FormData_pg_attribute
{
  Oid      attrelid;     // OID of reln containing attr
  NameData attname;      // name of attribute
  Oid      atttypid;     // OID of attribute's data type
  int16    attlen;       // attribute length
  int32    attndims;     // # dimensions if array type
  bool     attnotnull;   // can attribute have NULL value
  .....                  // and many other fields
} FormData_pg_attribute;
```

For details, see **include/catalog/pg_attribute.h**

<< ^ >>

## ❖ PostgreSQL Attribute Values

Attribute values in PostgreSQL tuples are packaged as **`Datum`**s

```
// representation of a data value
typedef uintptr_t Datum;
```

The actual data value:

- may be stored in the **`Datum`** (e.g. **`int`**)
- may have a header with length (for varlen attributes)
- may be stored in a TOAST file (if large value)

<< ^ >>

## ❖ PostgreSQL Attribute Values (cont)

Attribute values can be extracted as **`Datum`** from **`HeapTuple`**s

```
Datum heap_getattr(
        HeapTuple tup,      // tuple (in memory)
        int attnum,         // which attribute
        TupleDesc tupDesc, // field descriptors
        bool *isnull        // flag to record NULL
)
```

**`isnull`** is set to true if value of field is **`NULL`**

**`attnum`** can be negative ... to access system attributes (e.g. OID)

For details, see **`include/access/htup_details.h`**

<< ^

## ❖ PostgreSQL Attribute Values (cont)

Values of **`Datum`** objects can be manipulated via e.g.

```
// DatumGetBool:
//   Returns boolean value of a Datum.

#define DatumGetBool(X) ((bool) ((X) != 0))

// BoolGetDatum:
//   Returns Datum representation for a boolean.

#define BoolGetDatum(X) ((Datum) ((X) ? 1 : 0))
```

For details, see **`include/postgres.h`**

Produced: 28 Feb 2021