Structures internes des données sur disque

Sébastien Lardière

Loxodata

31 mai 2016

Structures internes des données sur disque

1. Répertoires et Fichiers
   - Définitions et OIDs
2. Outils
   - pg_filedump
   - Extensions
     - pageinspect
     - FSM
     - VM
   - pg_xlogdump

LOXODATA

Fichiers

PostgreSQL stocke les données dans des répertoires et des fichiers :

- Un répertoire par base de données
- Un répertoire par TABLESPACE, contenant des bases de données
- Les données d'une DB peuvent être dans plusieurs TABLESPACE

Définitions

Création d'objets :

Example

```
test=# create table mytable
( id serial primary key, d timestamptz, t text, b bool );
insert into mytable ( d, t, b)
   values ( now(), 'test 1', true ),
          ( now(), 'test 2', false );
INSERT 0 2

test=# select oid from pg_database where datname = 'test';
oid | 25101
test=# select relfilenode from pg_class where relname='mytable';
relfilenode | 25137
```

LOXODATA

Définitions

Création d'objets :

Example

```
$ ls -sh $PGDATA/base/25101/25137
8.0K /var/lib/postgresql/9.5/main/base/25101/25137
```

Fichiers

Plusieurs fichiers par tables :

- Les données dans des segments de 1GB (.1, .2, etc) : tableaux de pages de 8K.
- Fichier `<id>_vm` : contient les pages de données visibles
- Fichier `<id>_fsm` : liste les espaces disponibles dans les pages
- Un fichier par index, similaire aux tables
- Une table TOAST par table

Définitions

Création d'un tablespace :

Example

```
test=# create tablespace test1
       location '/var/lib/postgresql/9.5/tblspctest1' ;
CREATE TABLESPACE

test=# select oid from pg_tablespace where spcname='test1';
oid | 25142
test=# select pg_tablespace_location(25142);
pg_tablespace_location | /var/lib/postgresql/9.5/tblspctest1

test=# alter table mytable set tablespace test1 ;
ALTER TABLE
test=#  select reltablespace from pg_class where relname='mytable';
reltablespace | 25142
```

LOXODATA

Définitions

Création d'un tablespace :

Example

```
ls -sh /var/lib/postgresql/9.5/tblspctest1/PG_9.5_201510051/25101/25143*
8.0K /var/lib/postgresql/9.5/tblspctest1/PG_9.5_201510051/25101/25143
 24K /var/lib/postgresql/9.5/tblspctest1/PG_9.5_201510051/25101/25143_fsm
8.0K /var/lib/postgresql/9.5/tblspctest1/PG_9.5_201510051/25101/25143_vm
```

LOXODATA

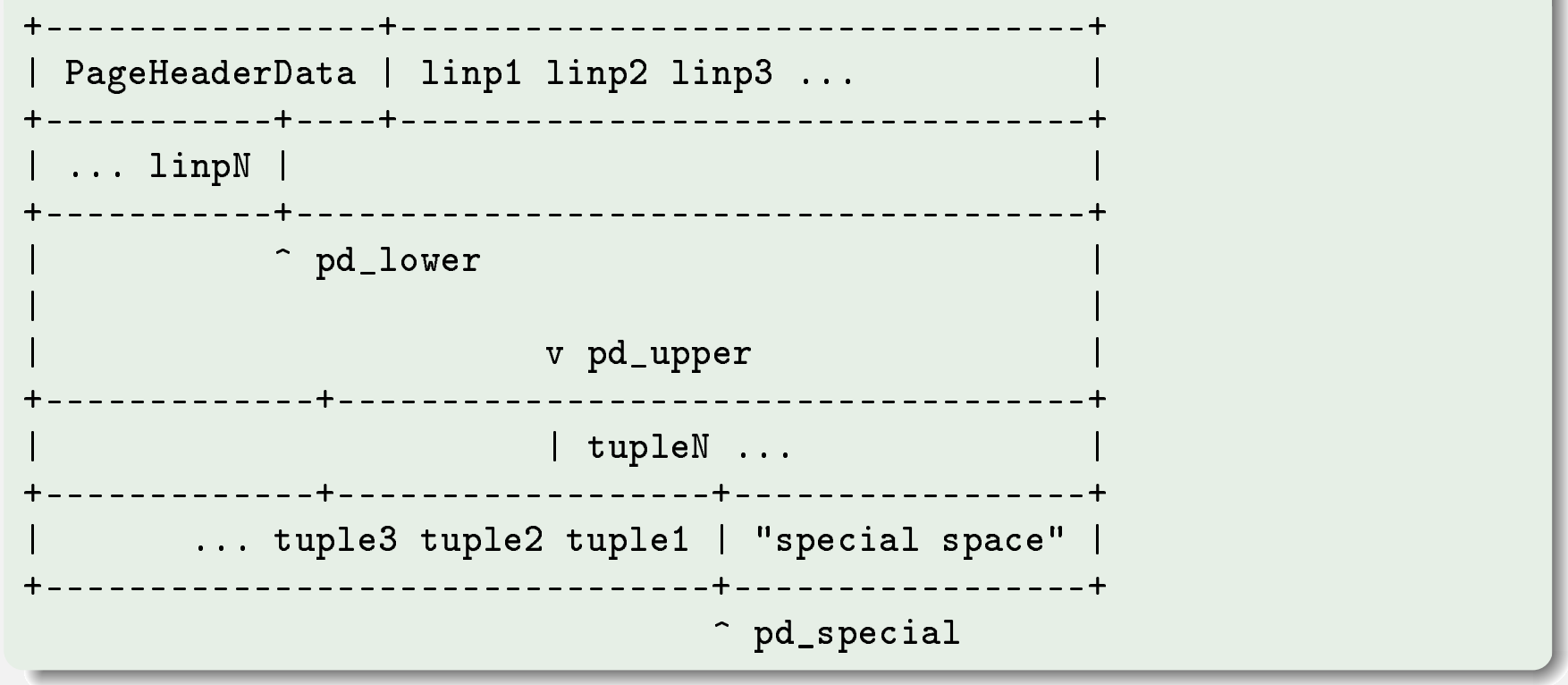
Pages

Pages de 8K, composées de :

- `PageHeaderData` : Structure de 24 octets, informations à propos de la page
- `ItemIdData` : Paire (offset,length) pointant sur les données
- `Free space` : Espace libre
- `Items` : Données
- `Special space` : Espace spécifique aux index

src/include/storage/bufpage.h

Example



Pages

`PageHeaderData`, 24 octets composé de :

- `pd_lsn` : Octet du XLOG correspondant à la dernière modification de la page
- `pd_checksum` : Somme de contrôle
- `pd_flags` : Bits d'états
- `pd_lower` : décalage du début de l'espace libre
- `pd_upper` : décalage de la fin de l'espace libre
- `pd_special` : décalage de l'espace "spécial"
- `pd_pagesize_version` : taille et version de la page
- `pd_prune_xid` : plus vieux Xmax non-traité.

LOXODATA

Items

`HeapTupleHeaderData` composé de :

- `t_xmin` : XID d'insertion
- `t_xmax` : XID de suppression
- `t_cid` : CID d'insertion et de suppression (surcharge avec `t_xvac`)
- `t_xvac` : XID pour l'opération VACUUM déplaçant une version de ligne
- `t_ctid` : TID en cours pour cette version de ligne ou pour une version plus récente
- `t_infomask2` nombre d'attributs
- `t_infomask` : flag bits
- `t_hoff` : data offset

LOXODATA

pg_filedump

Outil en ligne de commande :

- permet de comprendre de contenu d'un fichier (table ou index)
  - `-a` : Display absolute addresses when formatting (Block header information is always block relative)
  - `-i` : Interprète les détails des lignes
  - `-b` : Affichage des données binaires
  - `-d`, `-f` : Affiche les données formatées
  - `-k` : Vérifie les sommes de contrôle
  - `-R` : Permet d'indiquer une plage de blocs

pg_filedump

Example

```
pg_filedump -i 25137

Block    0 ********************************************************
<Header> -----
 Block Offset: 0x00000000         Offsets: Lower      40 (0x0028)
 Block: Size 8192  Version    4            Upper    8000 (0x1f40)
 LSN:  logid      2 recoff 0x34c86198      Special  8192 (0x2000)
 Items:    4                      Free Space: 7960
 Checksum: 0x0000  Prune XID: 0x00000000  Flags: 0x0000 ()
 Length (including item array): 40
```

LOXODATA

pg_filedump

Example

```
pg_filedump -i 25137


<Data> ------
 Item   1 -- Length:   48  Offset: 8144 (0x1fd0)  Flags: NORMAL
  XMIN: 1149  XMAX: 0  CID|XVAC: 0
  Block Id: 0  linp Index: 1   Attributes: 4   Size: 24
  infomask: 0x0802 (HASVARWIDTH|XMAX_INVALID)
```

pageinspect

pageinspect est une extension fournie avec PostgreSQL :

Example

```
test=# create extension pageinspect;
CREATE EXTENSION
test=# \dx+ pageinspect
    Objects in extension "pageinspect"
            Object Description
-----------------------------------------
 function brin_metapage_info(bytea)
 function brin_page_items(bytea,regclass)
 ...
```

pageinspect

pageinspect est une extension fournie avec PostgreSQL, disposant de fonctions :

- `get_raw_page()`
- `heap_page_item_attrs()`, `heap_page_items()`
- `page_header()`
- `fsm_page_contents()`
- `tuple_data_split()`
- `brin_metapage_info()`, `brin_page_items()`
- `brin_page_type()`, `brin_revmap_data()`
- `bt_metap()`, `bt_page_items()`, `bt_page_stats()`
- `gin_leafpage_items()`, `gin_metapage_info()`, `gin_page_opaque_info()`

LOXODATA

pageinspect

Example

```
test=# insert into mytable values( 1, now(), 'texte 1', true ) ;
INSERT 0 1
test=# select lp, lp_off, lp_flags, lp_len, t_xmin, t_xmax, t_ctid,
t_infomask, t_data from heap_page_items(get_raw_page('mytable', 0));
-[ RECORD 1 ]+
lp          | 1
lp_off      | 8136
lp_flags    | 1
lp_len      | 49
t_xmin      | 560
t_xmax      | 0
t_ctid      | (0,1)
t_infomask | 2818
t_data      | \x0100000000000000e28d5502a9d6010011746578746520310 1
```

pageinspect

Example

```
test=# insert into mytable values( 2, now(), 'texte 2', false );
INSERT 0 1
test=# select lp, lp_off, lp_flags, lp_len, t_xmin, t_ctid, t_data from heap_pa
-[ RECORD 1 ]+
lp          | 1
lp_off      | 8136
lp_flags    | 1
lp_len      | 49
t_xmin      | 560
t_ctid      | (0,1)
t_data      | \x0100000000000000e28d5502a9d6010011746578746520310 1
-[ RECORD 2 ]-
lp          | 2
lp_off      | 8080
lp_flags    | 1
lp_len      | 49
t_xmin      | 561
t_ctid      | (0,2)
t_data      | \x0200000000000000600e403a9d6010011746578746520320 0
```

pageinspect

Example

```
test=# select txid_current(), t_xmin, t_xmax, t_ctid,
t_infomask, t_attrs from
heap_page_item_attrs(get_raw_page('mytable', 0), 'mytable', true);

-[ RECORD 6 ]+
txid_current | 659
t_xmin       | 575
t_xmax       | 0
t_ctid       | (0,6)
t_infomask   | 2818
t_attrs      | {"\\x06000000","\\x4082faa9c21c0000","\\x74657874652033"...
```

LOXODATA

pageinspect

Example

```
test=# insert into mytable values
( 6, '2001-01-01 00:00:01 +00'::timestamptz, 'texte 3', false ) ;
ERROR:  duplicate key value violates unique constraint "mytable_pkey"
DETAIL:  Key (id)=(6) already exists.
test=# select txid_current(), t_xmin, t_xmax, t_ctid, t_infomask, t_attrs
from heap_page_item_attrs(get_raw_page('mytable', 0), 'mytable', true);
-[ RECORD 6 ]+
txid_current | 663
t_xmin       | 575
t_xmax       | 0
t_ctid       | (0,6)
t_infomask   | 2818
t_attrs      | {"\\x06000000","\\x4082faa9c21c0000","\\x74657874652033"...
-[ RECORD 7 ]+
txid_current | 663
t_xmin       | 662
t_xmax       | 0
t_ctid       | (0,7)
t_infomask   | 2050
t_attrs      | {"\\x06000000","\\x4082faa9c21c0000","\\x74657874652033"...
```

pageinspect

Inspection d'un B-tree :

Example

```
test=# SELECT * FROM bt_metap('mytable_pkey');
-[ RECORD 1 ]-----
magic     | 340322
version   | 2
root      | 1
level     | 0
fastroot  | 1
fastlevel | 0
```

pageinspect

Inspection d'un B-tree :

Example

```
test=# SELECT * FROM bt_page_stats('mytable_pkey', 1);
-[ RECORD 1 ]-+-----
blkno         | 1
type          | l
live_items    | 5
dead_items    | 0
avg_item_size | 16
page_size     | 8192
free_size     | 8048
btpo_prev     | 0
btpo_next     | 0
btpo          | 0
btpo_flags    | 3
```

pageinspect

Inspection d'un B-tree :

Example

```
SELECT * FROM bt_page_items('mytable_pkey', 1);
-[ RECORD 1 ]----------------------
itemoffset | 1
ctid       | (0,1)
itemlen    | 16
nulls      | f
vars       | f
data       | 01 00 00 00 00 00 00 00
-[ RECORD 2 ]----------------------
itemoffset | 2
ctid       | (0,2)
itemlen    | 16
nulls      | f
vars       | f
data       | 02 00 00 00 00 00 00 00
```

pageinspect

Example

```
test=# delete from mytable where id=6;
DELETE 1
test=# select txid_current(), t_xmin, t_xmax, t_ctid, t_infomask, t_attrs from
-[ RECORD 5 ]+
txid_current | 665
t_xmin       | 566
t_xmax       | 0
t_ctid       | (0,5)
t_infomask   | 2818
t_attrs      | {"\\x05000000","\\x402227afbd7f0ffd","\\x74657874652033"...
-[ RECORD 6 ]+
txid_current | 665
t_xmin       | 575
t_xmax       | 664
t_ctid       | (0,6)
t_infomask   | 770
t_attrs      | {"\\x06000000","\\x4082faa9c21c0000","\\x74657874652033"...
```

pageinspect

Example

```
test=# vacuum ;
VACUUM
test=# select txid_current(), t_xmin, t_xmax, t_ctid, t_infomask, t_attrs
from heap_page_item_attrs(get_raw_page('mytable', 0), 'mytable', true);
-[ RECORD 5 ]+
txid_current | 666
t_xmin       | 566
t_xmax       | 0
t_ctid       | (0,5)
t_infomask   | 2818
t_attrs      | {"\\x05000000","\\x402227afbd7f0ffd","\\x74657874652033"...
-[ RECORD 6 ]+
txid_current | 666
t_xmin       |
t_xmax       |
t_ctid       |
t_infomask   |
t_attrs      |
```

pg_freespacemap

Extension permettant d'inspecter le contenu du FSM d'une table :

Example

```
test=# create extension pg_freespacemap ;
CREATE EXTENSION
test=# \dx+ pg_freespacemap
 Objects in extension "pg_freespacemap"
           Object Description
--------------------------------------
 function pg_freespace(regclass)
 function pg_freespace(regclass,bigint)
(2 rows)
test=# select * from pg_freespace('mytable');
-[ RECORD 1 ]
blkno | 0
avail | 7840
```

LOXODATA

pg_visibility

Extension permettant d'inspecter la `Visibility Map` d'une table :

Example

```
test=# create extension pg_visibility ;
CREATE EXTENSION

test=# select * from pg_visibility('mytable', 0);
-[ RECORD 1 ]--+--
all_visible    | t
all_frozen     | t
pd_all_visible | t
```

pg_xlogdump

Example

```
~$ pg_xlogdump pg_xlog/000000010000000000000001

rmgr: Heap        len (rec/tot):     14/ 15130, tx:          3,
lsn: 0/01000148, prev 0/01000128, desc: UPDATE off 33 xmax 3 ;
new off 40 xmax 0, blkref #0: rel 1663/1/1259 blk 3 FPW,
blkref #1: rel 1663/1/1259 blk 0 FPW

rmgr: Btree       len (rec/tot):      2/  3713, tx:          3,
lsn: 0/01003C80, prev 0/01000148, desc: INSERT_LEAF off 20,
blkref #0: rel 1663/1/2662 blk 1 FPW
```

Des Questions ?

Merci à vous ! Des Questions ?

- Sébastien Lardière - Loxodata
- s.lardiere@loxodata.com
- Twitter @slardiere

LOXODATA