Hint Bits, die Visibility Map und der ganze Rest

Christoph Berg <christoph.berg@credativ.de>

Leipzig, 2019-05-10

Transaktionen in PostgreSQL

credativ

- Jede Transaktion hat eine ID
- 64 Bit
- gespeichert werden aber nur 32 Bit
  - Wraparound. Später.
- jedes Tupel hat Transaktionsinformationen
  - Sichtbarkeit von . . . bis
- wie funktioniert das?

CREATE TABLE

credativ

```
create table foo (id int, t text);

insert into foo values (1, 'Hallo');
insert into foo values (2, 'Ich bin weg');
delete from foo where id = 2;
insert into foo values (3, 'Ich bin alt');
update foo set t = 'Ich bin neu' where id = 3;
insert into foo values (4, 'Ich bin doch nicht weg');

begin;
delete from foo where id = 4;
insert into foo values (5, 'Ich bin doch nicht da');
rollback;
```

SELECT *

```
select * from foo;

 id |           t
----+-----------------------
  1 | Hallo
  3 | Ich bin neu
  4 | Ich bin doch nicht weg
```

Transaktions-Nummern

credativ

```
select xmin, xmax, * from foo;

  xmin   |  xmax   | id |           t
---------+---------+----+-----------------------
 219322 |       0 |  1 | Hallo
 219326 |       0 |  3 | Ich bin neu
 219327 | 219329 |  4 | Ich bin doch nicht weg

select txid_current();

 txid_current
--------------
       219331
```

pg_dirtyread

credativ

```
create extension pg_dirtyread;

select * from pg_dirtyread('foo')
              t(xmin xid, xmax xid, id int, t text);

  xmin   |  xmax   | id |           t
---------+---------+----+-----------------------
 219322 |       0 |  1 | Hallo
 219323 | 219324 |  2 | Ich bin weg
 219325 | 219326 |  3 | Ich bin alt
 219326 |       0 |  3 | Ich bin neu
 219327 | 219329 |  4 | Ich bin doch nicht weg
 219330 |       0 |  5 | Ich bin doch nicht da
```

https://github.com/df7cb/pg_dirtyread

txid_status

```
select txid_status(219324);

 txid_status
-------------
 committed
```

pg_dirtyread

credativ

```
select xmin, txid_status(xmin::text::bigint),
       xmax, txid_status(xmax::text::bigint), id, t
  from pg_dirtyread('foo')
       t(xmin xid, xmax xid, id int, t text);
```

xmin	txid_status	xmax	txid_status	id	
219322	committed	0		1	Hallo
219323	committed	219324	committed	2	Ich bin
219325	committed	219326	committed	3	Ich bin
219326	committed	0		3	Ich bin
219327	committed	219329	aborted	4	Ich bin
219330	aborted	0		5	Ich bin

pg_xact

```
-rw-------  1 postgres postgres 57344 Mai  6 14:16 0000

0000d620  55 55 55 55 55 95 5a 95  5a 55 a9 55 95 5a 55 55
          |UUUUU.Z.ZU.U.ZUU|
```

- Bit-Array, 2 Bits pro Transaktion
- 1 Million (2^20) Transaktionen pro Datei = max. 256 KiB
- 00 = in progress
  01 = committed
  10 = aborted
  11 = it's complicated
- include/access/clog.h

SELECT *

credativ

- Datei öffnen
- Seite lesen
- Tupel lesen
- xmin, xmax lesen
- pg_xact lesen
- Werte vergleichen
- Tupel ausgeben oder überspringen
- GOTO 30

SET speed = on;

credativ

- Caching!
- Seiten (8 KiB): shared_buffers (Default: 128 MiB)
- pg_xact (auch 8 KiB): SLRU (Default: 4..128)

SET speed = more;

credativ

- Hint bits
  - Sichtbarkeit direkt im Tupel/in der Seite speichern
- Visibility map
  - Sichtbarkeit pro Seite in einem Bit-Array speichern

pageinspect

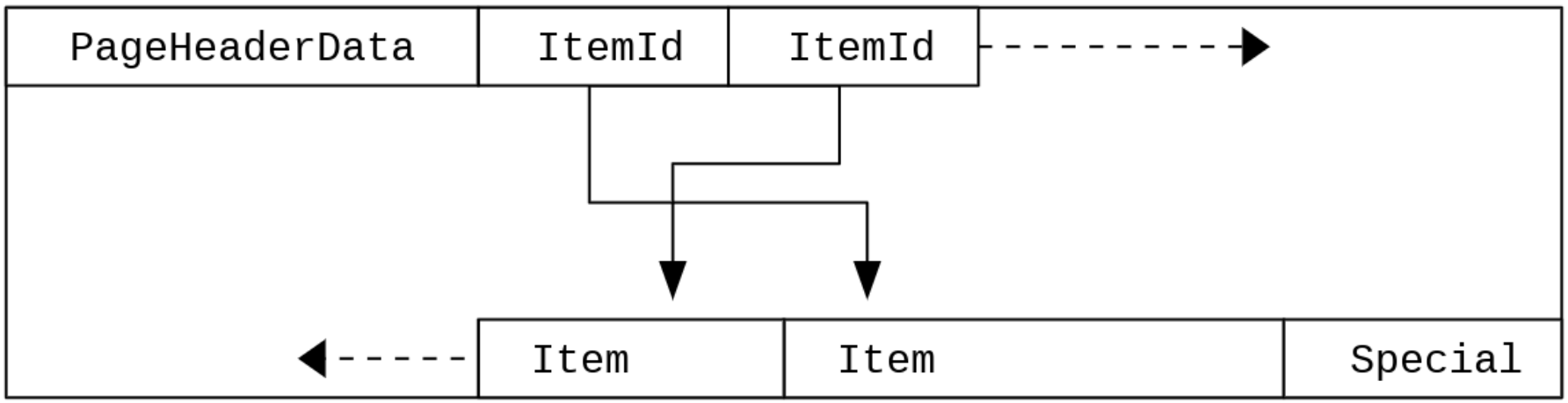
credativ

```
create extension pageinspect;

\dx+ pageinspect
                Objekte in Erweiterung "pageinspect"
                              Objektbeschreibung
-------------------------------------------------------------------
 function fsm_page_contents(bytea)
 function get_raw_page(text,integer)
 function get_raw_page(text,text,integer)
 function heap_page_item_attrs(bytea,regclass)
 function heap_page_item_attrs(bytea,regclass,boolean)
 function heap_page_items(bytea)
 function page_checksum(bytea,integer)
 function page_header(bytea)
```

get_raw_page

```
select get_raw_page('foo', 0) \gx

\x01000000180dfb900000000003000f01e00200420bc580300d89f44
  00b09f5000889f5000609f5000289f6600f09e640000000000000000
  ....
```

Page-Layout



- Gleiches Layout für Tabellen und die meisten Indexe
- Header: 24 Bytes
- ItemId: Pointer auf Item
  - Pointer von vorne, Items von hinten
- Special: leer bei Tabellen

page_header


```
select * from page_header(get_raw_page('foo', 0)) \gx

-[ RECORD 1 ]---------
lsn      | 1/90FB0D18
checksum | 0
flags    | 0
lower    | 48
upper    | 7920
special  | 8192
pagesize | 8192
version  | 4
prune_xid | 219324
```

- lsn: neuester WAL-Record für diese Seite

Tupel-Layout

- Header (23 Bytes)
- Null-Bitmaske (optional)
- OID (optional, deprecated)
- Daten

heap_page_items


```
select * from heap_page_items(get_raw_page('foo', 0)) \gx
lp           | 1          <---- ctid (0,1)
lp_off       | 8152
lp_flags     | 1
lp_len       | 34
t_xmin       | 219322
t_xmax       | 0
t_field3     | 0
t_ctid       | (0,1)
t_infomask2  | 2          <----
t_infomask   | 2306       <----
t_hoff       | 24
t_bits       |
t_oid        |
t_data       | \x010000000d48616c6c6f
```

Hint Bits in der Infomask

credativ

```
select t_xmin, xmin_committed, t_xmax, xmax_committed
  from heap_page_items(get_raw_page('foo', 0)),
       t_infomask(t_infomask, t_infomask2)
 where lp = 1 \gx

t_xmin          | 219322
xmin_committed | t
t_xmax          | 0
xmax_committed | f
```

https://github.com/df7cb/pg_dirtyread/blob/master/contrib/infomask.sql

Slow motion

credativ

```
truncate foo;
insert into foo values (1, 'Hallo');
checkpoint;
select t_xmin, xmin_committed, t_xmax, xmax_committed
  from heap_page_items(get_raw_page('foo', 0)),
       t_infomask(t_infomask, t_infomask2)
 where lp = 1 \gx

t_xmin          | 219342
xmin_committed | f
t_xmax          | 0
xmax_committed | f
```

Slow motion

```
explain (analyze, buffers) select * from foo where id = 1;

Seq Scan on foo  (cost=0.00..25.88 rows=6 width=36)
                 (actual time=0.026..0.027 rows=1 loops=1)
  Filter: (id = 1)
  Buffers: shared hit=1 dirtied=1   <----

t_xmin          | 219342
xmin_committed  | t       <----
t_xmax          | 0
xmax_committed  | f
```

- SELECT *schreibt* in die Tabelle

SELECT *

credativ

- Datei öffnen
- Seite lesen
- Tupel lesen
- xmin, xmax und Hint Bits lesen
- Dann:
  - Hint Bits auswerten oder
  - pg_xact lesen
- Werte vergleichen
- Tupel ausgeben oder überspringen
- GOTO 30

WAL

- Hint Bits normalerweise nicht im WAL
  - Standby Server schreiben eigene Hint Bits
- . . . aber mit Data Checksums, oder `wal_log_hints`
  - dann muss Primary selbst VACUUM (oder SELECT) machen

Visibility Map

- Nochmal das Gleiche, aber außerhalb der Tabelle in eigener (kleinerer!) Datei

```
-rw------- 1 postgres postgres  8192 Mai 6 17:04 50163
-rw------- 1 postgres postgres 24576 Mai 6 17:04 50163_fsm
-rw------- 1 postgres postgres  8192 Mai 6 17:04 50163_vm
```

Visibility Map

credativ

```
create index on foo(id);
set enable_seqscan = off;
set enable_bitmapscan = off;

explain (analyze, buffers) select id from foo where id = 1;

Index Only Scan using foo_id_idx on foo
   Index Cond: (id = 1)
   Heap Fetches: 1          <----
   Buffers: shared hit=2
```

Visibility Map

credativ

```
vacuum foo;

explain (analyze, buffers) select id from foo where id = 1;

 Index Only Scan using foo_id_idx on foo
   Index Cond: (id = 1)
   Heap Fetches: 0          <----
   Buffers: shared hit=2
```

pg_visibility


```
create extension pg_visibility;

select * from pg_visibility('foo');

 blkno | all_visible | all_frozen | pd_all_visible
-------+-------------+------------+----------------
     0 | t           | f          | t
```

Freezing

credativ

- Transaktionsnummern sind 64 Bit
- gespeichert werden nur 32 Bit
- nach 2^{31} Transaktionen müssen alte Tupel (genauer: alte Transaktionsnummern) “eingefroren” werden
- `autovacuum: VACUUM foo (to prevent wraparound)`
- früher: FrozenTransactionId (2)
- jetzt: Hint-Bits
- außerdem in der Visibility Map

Metainformationen

- PostgreSQL speichert redundante Zusatzinformationen im Page- und Tupel-Header und Relation Forks
- Hint Bits beschleunigen Zugriff, sorgen aber für extra Schreiblast
- Visibility Map wird nur durch VACUUM geschrieben
  - INSERT-only-Tabellen!
- Autovacuum tunen