UNDELETE DATA FROM TABLE;

CHRISTOPH BERG

PGconf.EU 2024, cybertec-postgresql.com

CYBERTEC
POSTGRESQL SERVICES & SUPPORT

Christoph Berg

- Senior PostgreSQL Engineer at CYBERTEC
- Debian Developer (all things PostgreSQL)
- PostgreSQL Major Contributor (all things Debian)
- ham radio operator (DF7CB)

Things happen

```
=> DELETE FROM addressbook WHERE name ~ 'Heinz';
DELETE 2
PANIC:  WHY WERE THERE 2 ROWS AND NOT JUST ONE
```

Now what?

- ROLLBACK;
  - begin; delete from table; oops; rollback;
- restore from backup
  - have backups
  - test backups
  - know how PITR works

Now what?

- MVCC
- DELETE just *marks* the data as deleted, it doesn't actually *delete* it
- so it must be still there
- how do we get at it?

UNDELETE data FROM table;

- how does PostgreSQL store data
- reading deleted data from tables
- autovacuum
- reading deleted data from the WAL

How does PostgreSQL store data - MVCC in a nutshell

- secret extra system columns in each table
- new rows have `xmin` set
- deleted rows have `xmax` set

```
=# select xmin, xmax, * from addressbook;
 xmin | xmax |      name       | address
------+------+----------------+---------
 2834 |    0 | Heinz Erhardt | Hamburg
 2838 |    0 | Heinz Rühmann | München
(2 rows)
```

How does PostgreSQL store data - MVCC in a nutshell

xmax visible for rows that have not been deleted yet

Session 1:

```
=# begin;
=*# delete from addressbook where name = 'Heinz Erhardt';
```

Session 2:

```
=# select xmin, xmax, * from addressbook;
 xmin | xmax |     name      | address
------+------+---------------+---------
 2834 | 2840 | Heinz Erhardt | Hamburg
 2838 |    0 | Heinz Rühmann | München
(2 rows)
```

Let’s read all the rows with xmax > 0

```
=# select * from addressbook where xmax <> 0;
     name      | address
---------------+----------
 Heinz Erhardt | Hamburg
(1 row)
```

... COMMIT ...

```
=# select * from addressbook where xmax <> 0;
 name | address
------+---------
(0 rows)
```

Getting at deleted rows

- the row is there
- PostgreSQL just doesn’t show it
- pg_dirtyread
  - https://github.com/df7cb/pg_dirtyread

pg_dirtyread

```
$ sudo apt install postgresql-17-dirtyread

=# create extension pg_dirtyread;
CREATE EXTENSION

=# \dx+ pg_dirtyread
Objects in extension "pg_dirtyread"
       Object description
---------------------------------
 function pg_dirtyread(regclass)
(1 row)
```

pg_dirtyread('addressbook')

- pg_dirtyread takes a table name and returns all rows

```
=# select * from pg_dirtyread('addressbook');
ERROR:  a column definition list is required for functions returning "record"
```

pg_dirtyread(‘addressbook’) as tbl(. . .)

- pg_dirtyread takes a table name and returns all rows
- returns a record, need to provide column names and types

```
=# select * from pg_dirtyread('addressbook') as ab(name text, address text);
     name      | address
---------------+----------
 Heinz Erhardt | Hamburg
 Heinz Rühmann | München
(2 rows)
```

pg_dirtyread(‘addressbook’)

- can choose which columns to show

```
=# select * from pg_dirtyread('addressbook') as ab(name text);
     name
---------------
 Heinz Erhardt
 Heinz Rühmann
(2 rows)
```

pg_dirtyread('addressbook')

- can read system columns
- special column `dead` *

```
=# select * from pg_dirtyread('addressbook')
     as ab(xmin xid, xmax xid, dead boolean, name text);
 xmin | xmax | dead |      name
------+------+------+----------------
 2834 | 2840 | t    | Heinz Erhardt
 2840 |    0 | f    | Heinz Rühmann
(2 rows)
```

* Relies on hint bits

Salvaging our data

Finally:

```
=# select name, address from pg_dirtyread('addressbook')
     as ab(dead boolean, name text, address text) where dead;
      name     | address
---------------+-----------
 Heinz Erhardt | Hamburg
(1 row)
```

Putting the data back

```
=# insert into addressbook
     select name, address from pg_dirtyread('addressbook')
       as ab(dead boolean, name text, address text) where dead;
INSERT 0 1

=# select * from addressbook;
      name     | address
---------------+----------
 Heinz Rühmann | München
 Heinz Erhardt | Hamburg
(2 rows)
```

But.

- if several rows were deleted, identify them by `xmax`
- beware of the https://en.wikipedia.org/wiki/Halloween_Problem
  - use a separate table to restore data
- TOAST tables make this more interesting
- alternative approach: pg_filedump
  - https://github.com/df7cb/pg_filedump

VACUUM;

- vacuum is PostgreSQL's garbage collection
- actually deletes rows that are marked to be deleted
- autovacuum every minute
- triggers after 50 rows + 20% of table size
- when deleting too many things, you have less than a minute to avoid autovacuum

Now what?

- WAL
- records all changes done in a database
- “delete item 4 on page 0”

WAL

```
=# select ctid, * from addressbook;
 ctid  |      name      | address
-------+----------------+---------
 (0,3) | Heinz Rühmann | München
 (0,4) | Heinz Erhardt | Hamburg
(2 rows)
=# delete from addressbook where name = 'Heinz Erhardt';
DELETE 1
=# vacuum addressbook;
VACUUM
=# select pg_walfile_name(pg_current_wal_lsn());
     pg_walfile_name
--------------------------
 000000010000000000000030
(1 row)
```

pg_waldump pg_wal/000000010000000000000030

```
rmgr: Heap        len (rec/tot):     59/   195, tx:       2844,
  lsn: 0/309C0AD0, prev 0/309C0A58,
  desc: DELETE xmax: 2844, off: 4, infobits: [KEYS_UPDATED],
  flags: 0x01, blkref #0: rel 1663/5/64076 blk 0 FPW
```

- item marked as deleted

```
rmgr: Heap2       len (rec/tot):     56/    56, tx:          0,
  lsn: 0/309C0BF8, prev 0/309C0BC0,
  desc: PRUNE_VACUUM_SCAN snapshotConflictHorizon: 2844, isCatalogRel: F,
  nplans: 0, nredirected: 0, ndead: 0, nunused: 1, unused: [4],
  blkref #0: rel 1663/5/64076 blk 0
```

- vacuum zeroed the free space
- ???

Full page write (FPW)

full_page_writes (boolean)

When this parameter is on, the PostgreSQL server writes the *entire content of each disk page* to WAL during the first modification of that page after a checkpoint.

The default is on.

- WAL record contains full page
- including our deleted row!
- marked for deletion, but whatever

The plan

- put safety goggles on
- extract FPW from WAL
- create a table with the correct structure
- put these pages into that table
- use pg_dirtyread

Extract FPW from WAL

- pg_waldump 16+ has a `--save-fullpage` switch (or backpatch)

```
$ mkdir fpw
$ /usr/lib/postgresql/17/bin/pg_waldump --save-fullpage=fpw \
    pg_wal/000000010000000000000030
$ ls -al fpw
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309A6C78.1663.5.2678.1_main
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309A7920.1663.5.2679.1_main
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309A88C0.1663.5.64080.0_main
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309A9598.1663.5.64076.0_main
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309A9848.1663.5.3079.0_main
-rw-rw-r-- 1 postgres 8192 00000001-00000000-309AA568.1663.5.3080.1_main
```

TIMELINE-LSN.RELTABLESPACE.DATOID.**RELNODE**.BLKNO_FORK

Create a table with the correct structure

```
=# create table ab_restore (like addressbook);
CREATE TABLE

=# select relname, relfilenode from pg_class
      where relname in ('addressbook', 'ab_restore');
   relname    | relfilenode
-------------+-------------
 addressbook |       64076
 ab_restore  |       64083
(2 rows)

$ ls -al base/5/64083
-rw------- 1 postgres postgres 0  1. Okt 22:12 base/5/64083
```

Put these pages into that table

```
$ cat fpw/00000001-00000000-309A9598.1663.5.64076.0_main > base/5/64083
```

- could be several files, just `cat` them together
- possibly better stop PostgreSQL since we change files on disk

Use pg_dirtyread

```
=# select * from ab_restore;
     name      | address
---------------+---------
 Heinz Rühmann | München
(1 row)

=# select * from pg_dirtyread('ab_restore') as ab(name text, address text);
     name      | address
---------------+---------
 Heinz Rühmann | München
 Heinz Erhardt | Hamburg
(2 rows)
```

But.

- works best when DELETE was the first change to this page (since the last checkpoint)
  - interference from hint bits, HOT pruning, . . .
- if several rows are deleted, only the first one has the `xmax` mark
- hard to tell deleted and live tuples apart
- TOAST tables make everything even more interesting

Summary

- pg_dirtyread can help in some cases
- pg_waldump can help in harder cases
- this was a toy example, real-world cases might be less clean
- did we mention you should have backups?
- thanks!