

SCaLE 21x

Pasadena, 2024-03-15

Recovering from Data Loss Despite Not Having a Backup: A Postgres True Story

Jimmy Angelakos



About me

- Systems & Database Architect
- Based in Edinburgh, Scotland
- Open Source user & contributor (25+ years)
- PostgreSQL exclusively (16+ years)
- Author, PostgreSQL Mistakes and How to Avoid Them
- Co-author, PostgreSQL 16 Administration Cookbook
- **pg_statviz** PostgreSQL extension



Excuse me, what?

- You heard right, “no backup”
- Actual company
- With customers
- These things happen $_ _ (_ _) _ _ / _ _$

The scene

- Phone rings at 5:00 pm
 - OK maybe it wasn't the phone, it was Zoom™
 - Who uses phones to talk shop?
- Tired voice of the CTO
- Company lost entire database
- No backup
- Asking if recovery is possible

THE END

- Thank you
- Buy my books

What's happening?

- DB is critical to company operations
- The website IS the database
- Website down for > 1 week
- Users are starting to grumble
- Stakeholders are starting to worry

(i)



What's happening?

- Disk crash wiped out production DB server
- No redundancy, PostgreSQL database is gone
- Most recent backup is months old
- Website data needs to be up-to-date
- Database recovery company quote: 2 weeks
 - “no guarantees”

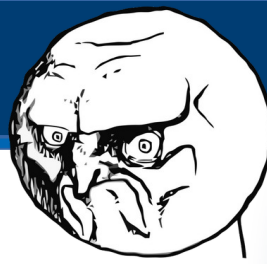
(ii)

How did this happen?

- Start-up that grew over time
- Transitioned to PostgreSQL decades ago
- **pg_dump** for backup
- Script silently failed for months
- Is it really “bad luck”? We know hardware fails.

A glimmer of hope...

- Data recovery company has recovered some files
- Looks like it may have been just a controller failure
- Company gives me dump to see what's salvageable



A glimmer of hope... but **NO.**

- Data recovery company has recovered some files
- Looks like it may have been just a controller failure
- Company gives me dump to see what's salvageable
- Files randomly distributed in recovery directories

0001 /

0002 /

0003 /

(...)

INTERMISSION

- Let's talk about how Postgres puts stuff on disk

Physical DB structure on disk

(i)

```
$ cd /var/lib/postgresql/16/
```

```
$ ls -la data/
```

```
total 136
```

```
drwx----- 20 postgres postgres 4096 Mar 12 12:51 .
drwx-----  3 postgres postgres 4096 Mar 12 12:51 ..
-rw-----  1 postgres postgres   3 Mar 12 12:51 PG_VERSION
drwx-----  8 postgres postgres 4096 Mar 12 12:56 base
drwx-----  2 postgres postgres 4096 Mar 12 12:51 conf.d
drwx-----  2 postgres postgres 4096 Mar 12 12:56 global
drwx-----  2 postgres postgres 4096 Mar 12 12:51 pg_commit_ts
drwx-----  2 postgres postgres 4096 Mar 12 12:51 pg_dynshmem
-rw-r--r--  1 postgres postgres  856 Mar 12 12:51 pg_hba.conf
-rw-----  1 postgres postgres 2640 Mar 12 12:51 pg_ident.conf
```

```
(...)
```

Physical DB structure on disk

(ii)

- `base/`
- `/var/lib/pgsql/16/data/base/` (get it?)
- Contains directories for individual DBs (name is DB **oid**)

```
$ ls -la base/
```

```
total 40
```

```
drwx-----  8 postgres postgres 4096 Mar 12 12:56 .
drwx----- 20 postgres postgres 4096 Mar 12 12:51 ..
drwx-----  2 postgres postgres 4096 Mar 12 12:55 1
drwx-----  2 postgres postgres 4096 Mar 12 12:52 16582
drwx-----  2 postgres postgres 12288 Mar 12 19:30 16587
drwx-----  2 postgres postgres 4096 Mar 12 12:51 4
drwx-----  2 postgres postgres 4096 Mar 12 12:54 5
drwx-----  2 postgres postgres 4096 Mar 12 12:56 postgres_tmp
```

Physical DB structure on disk

(iii)

```
SELECT * FROM pg_database WHERE oid=16587 \gx
```

```
-[ RECORD 1 ]-----
```

oid		16587
datname		pgbench
datdba		10
encoding		6
datlocprovider		c
datistemplate		f
datallowconn		t
datconnlimit		-1
datfrozenxid		722
datminmxid		1
dattablespace		1663
datcollate		en_US.UTF-8
(...)		

Physical DB structure on disk

(iv)

- Each DB directory contains table files, indexes etc.

```
SELECT relname, oid, relfilenode FROM pg_class  
WHERE relname = 'pgbench_accounts' \gx
```

```
-[ RECORD 1 ]-----  
relname      | pgbench_accounts  
oid          | 16594  
relfilenode  | 16600
```

Physical DB structure on disk

(v)

- Table **oid** may not match **relfilenode**

```
$ cd base
```

```
$ ls -la 16587/16600*
```

```
-rw----- 1 postgres postgres 13434880 Mar 12 19:30 16587/16600  
-rw----- 1 postgres postgres    24576 Mar 12 19:30 16587/16600_fsm  
-rw----- 1 postgres postgres     8192 Mar 12 19:30 16587/16600_vm
```

- Tables > 1GB are split into multiple files

```
$ ls -lah 16587/16608*
```

```
-rw----- 1 postgres postgres 1.0G Mar 13 22:54 16587/16608  
-rw----- 1 postgres postgres 142M Mar 13 22:54 16587/16608.1
```


So what's the plan?

- Recreate data directory structure in **/opt/recovery**
- Copy files inside database directory
- Attempt to start Postgres from **/opt/recovery**
- **pg_dump** production database
 - Will ensure everything can be read correctly
- Restore dump to a fresh instance
- ????
- PROFIT!!!

Copy files inside data directory

- Looks daunting:

```
$ ls -la base/1
```

```
total 7676
```

```
drwx----- 2 postgres postgres 4096 Mar 13 18:42 .
drwx----- 7 postgres postgres 4096 Mar 13 18:42 ..
-rw----- 1 postgres postgres 8192 Mar 12 12:51 112
-rw----- 1 postgres postgres 8192 Mar 12 12:51 113
-rw----- 1 postgres postgres 122880 Mar 12 19:24 1247
-rw----- 1 postgres postgres 24576 Mar 12 12:51 1247_fsm
-rw----- 1 postgres postgres 8192 Mar 12 19:24 1247_vm
-rw----- 1 postgres postgres 491520 Mar 12 19:24 1249
-rw----- 1 postgres postgres 24576 Mar 12 19:24 1249_fsm
(...)
```

- BUT: **oids** below **16384** are reserved for system use

After some clickety clack...

(i)

- Does PostgreSQL start up?

```
$ pg_ctl -D /opt/recovery start
```

```
waiting for server to start....
```

```
2024-03-13 23:00:38 UTC [:@//:20543]: [1] LOG: ending
```

```
log output to stderr
```

```
2024-03-13 23:00:38 UTC [:@//:20543]: [2] HINT:
```

```
Future log output will go to log destination
```

```
"syslog".
```

```
stopped waiting
```

```
pg_ctl: could not start server
```

```
Examine the log output.
```

After some clickety clack...

(ii)

- The log says:

```
[2] FATAL: could not access status of transaction 803
```

```
[3] DETAIL: Could not open file "pg_xact/0000": No  
such file or directory.
```

```
[8] LOG: startup process (PID 20424) exited with exit  
code 1
```

```
[9] LOG: aborting startup due to startup process  
failure
```

```
[10] LOG: database system is shut down
```

More typing ensues... (i)

- What's `pg_xact/0000` ?
- Unfortunately, these files appear to have been lost
- `pg_xact/` holds transaction commit state data: 256 KB files, 4 transactions per byte
- We want status `01 (COMMITTED)`
so let's fill it with `01010101` (octal `125`)

```
$ dd if=/dev/zero bs=256k count=1 | \  
tr '\000' '\125' > pg_xact/0000  
1+0 records in  
1+0 records out  
262144 bytes (262 kB, 256 KiB) copied,  
0.00208236 s, 126 MB/s
```

More typing ensues...

(ii)

- How about now?

```
$ pg_ctl -D /opt/recovery start
waiting for server to start....
2024-03-13 23:50:02 UTC [:@//:21417]: [1] LOG:  ending log output to stderr
2024-03-13 23:50:02 UTC [:@//:21417]: [2] HINT:  Future log output will go
to log destination "syslog".
done
server started
```



- Can we connect?

```
$ psql
psql: error: connection to server on socket
"/var/run/postgresql/.s.PGSQL.5432" failed: FATAL:  database "postgres"
does not exist
DETAIL:  The database subdirectory "base/5" is missing.
```

After some more clickety clack... (i)

- Now?

```
$ psql
psql (16.2)
Type "help" for help.
```

```
postgres=#
```



After some more clickety clack... (ii)

- Let's say we want to recover database **pgbench**

```
postgres=# \c pgbench
```

```
connection to server on socket
```

```
"/var/run/postgresql/.s.PGSQL.5432" failed: FATAL:
```

```
database "pgbench" does not exist
```

```
DETAIL: The database subdirectory "base/16587" is  
missing.
```

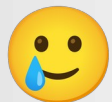
```
Previous connection kept
```


After some more clickety clack... (iii)

- Let's see...

```
$ psql pgbench  
psql (16.2)  
Type "help" for help.
```

```
pgbench=#
```



Let's do it, people are waiting

```
$ pg_dump pgbench > pgbench.dump
```

```
$
```

- OMG!
- Are we done?
- No, not yet.
- Need to restore into a fresh DB

Restoring the dump

```
$ psql pgbench -f pgbench.dump  
(...)
```

```
ERROR:  could not create unique index  
"pgbench_accounts_pkey"  
DETAIL:  Key (aid)=(123) is duplicated.
```

- To be expected, since many transactions were switched to **COMMITTED**, resurrecting dead rows
- This happened 100 or so more times...
- Customer confirmed which row version to keep

Dramatic save

- Restore completed on Sunday afternoon (< 48h)
- Stakeholders were holding a conference call to decide what to do next
- News of the recovery, applause broke out on the call
- Relief all around

BUT

- I couldn't relax until we created a streaming standby (with RepMgr) and set up backups (with Barman)
- Then, it was finally time for a beer 🍺


Extremely lucky

- Most files were recoverable
- Even if you've lost files you can use this methodology
 - Save whatever can be saved
 - Reconstruct/fake files such as **pg_filenode.map** or **pg_control**
- If the file for the table is gone, then the table is gone
 - But you can just remove it from the catalog
- **pg_wal**
 - PITR is going to replay up to the last feasible point, so, whatever you have is whatever you have...

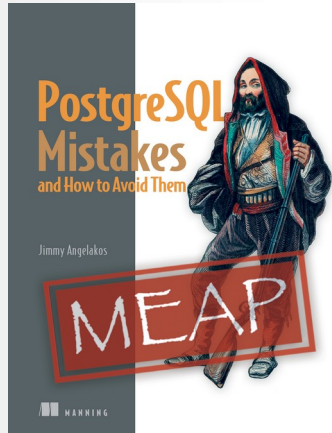
What not to do

- **pg_dump** as backup
 - No PITR
 - You need to set up automation, monitoring, alerting, testing manually
 - Did you miss the first half of the talk?
- Maintain radio silence – people will start talking
 - Security breach?

What to do

- **DON'T PANIC** 
- Have redundancy & automated, tested backups
- Always operate on a copy of the recovered data
- Keep the team informed at every step
- Degree of recovery matters, but speed matters too
- Don't decide what to recover yourself, ask

45% off for 3 months
Code: **ctwscalex45**



20% off until March 22nd
Code: **20JIMMY**

