

# PostgreSQL 17 beta: Incremental Backups



**data egret**

Your remote PostgreSQL DBA team

Stefan FERCOT

[stefan.fercot@dataegret.com](mailto:stefan.fercot@dataegret.com)

# SECURING YOUR DATABASE AVAILABILITY, SO THAT YOUR TEAM CAN FOCUS ON NEW FEATURE DEVELOPMENT.

- Migrations
- DB audit
- Performance optimisation
- Backup & restore
- Architectural review
- Advising Data Science teams
- Developer training

**on premise & cloud**



## EXPERTISE

Senior DBA with **10+ years**  
of PostgreSQL  
administration **experience**



## DEVELOPMENT

Involved in  
**new feature and**  
**extension development**



## TAILORED APPROACH

Felxible approach and  
**dedicated team** focused  
on success of your project



## COMMUNITY

**Contributing Sponsor.**  
Deeply involved in the  
PostgreSQL community

# Stefan Fercot

- Senior PostgreSQL Expert @Data Egret
- pgBackRest fan & contributor
- aka. pgstef
- <https://pgstef.github.io>

*Need a Disaster and Recovery Plan? ;-)  
Contact **Data Egret** to talk to me about backups and  
high-availability!*

# Agenda

- PostgreSQL 17
- incremental backups
  - WAL summaries
  - `pg_basebackup` incremental copy
  - `pg_combinebackup`
- Limitations and further considerations
  - `pg_verifybackup`, *evergreen* backup

# PostgreSQL 17

*PostgreSQL 17 Beta 1 is planned to be release on  
May 23, 2024.*

<https://www.postgresql.org/message-id/flat/0c8c951a-4432-4f15-b0a7-d3290d56f139%40postgresql.org>

# Early testing?

- <https://yum.postgresql.org/packages/#pg17>
  - YUM alpha packages
  - `pgdg17-updates-testing`
- [https://wiki.postgresql.org/wiki/Apt/FAQ#Development\\_snapshots](https://wiki.postgresql.org/wiki/Apt/FAQ#Development_snapshots)
  - APT development snapshots
  - `*-pgdg-snapshot`

# Example

```
sudo dnf install -y pgdg-redhat-repo-latest.noarch.rpm
sudo dnf -qy module disable postgresql
sudo dnf config-manager --set-enabled pgdg17-updates-testing
sudo dnf updateinfo
sudo dnf install -y postgresql17-server

PGSETUP_INITDB_OPTIONS="--data-checksums"
sudo -E /usr/pgsql-17/bin/postgresql-17-setup initdb
sudo systemctl enable --now postgresql-17
```



# Incremental backups

```
$ git show dc212340058b4e7ecfc5a7a81ec50e7a207bf288
```

```
Author: Robert Haas <rhaas@postgresql.org>
```

```
Date:   Wed Dec 20 09:49:12 2023 -0500
```

```
    Add support for incremental backup.
```

An incremental backup is like a regular full backup except that some relation files are replaced with files with names like `INCREMENTAL.${ORIGINAL_NAME}`, and the `backup_label` file contains additional lines identifying it as an incremental backup.

The new `pg_combinebackup` tool can be used to reconstruct a data directory from a full backup and a series of incremental backups.

Patch by me. Reviewed by Matthias van de Meent, Dilip Kumar, Jakub Wartak, Peter Eisentraut, and Álvaro Herrera. Thanks especially to Jakub for incredibly helpful and extensive testing.

even more work done after that, with lots of hackers involved!



# Documentation is great!

<https://www.postgresql.org/docs/devel/continuous-archiving.html#BACKUP-INCREMENTAL-BACKUP>

## 25.3.3. Making an Incremental Backup

You can use `pg_basebackup` to take an incremental backup by specifying the `--incremental` option. You must supply, as an argument to `--incremental`, the backup manifest to an earlier backup from the same server. In the resulting backup, non-relation files will be included in their entirety, but some relation files may be replaced by smaller incremental files which contain only the blocks which have been changed since the earlier backup and enough metadata to reconstruct the current version of the file.




To figure out which blocks need to be backed up, the server uses WAL summaries, which are stored in the data directory, inside the directory `pg_wal/summaries`. If the required summary files are not present, an attempt to take an incremental backup will fail. The summaries present in this directory must cover all LSNs from the start LSN of the prior backup to the start LSN of the current backup. Since the server looks for WAL summaries just after establishing the start LSN of the current backup, the necessary summary files probably won't be instantly present on disk, but the server will wait for any missing files to show up. This also helps if the WAL summarization process has fallen behind. However, if the necessary files have already been removed, or if the WAL summarizer doesn't catch up quickly enough, the incremental backup will fail.

When restoring an incremental backup, it will be necessary to have not only the incremental backup itself but also all earlier backups that are required to supply the blocks omitted from the incremental backup. See `pg_combinebackup` for further information about this requirement. Note that there are restrictions on the use of `pg_combinebackup` when the checksum status of the cluster has been changed; see `pg_combinebackup limitations`.

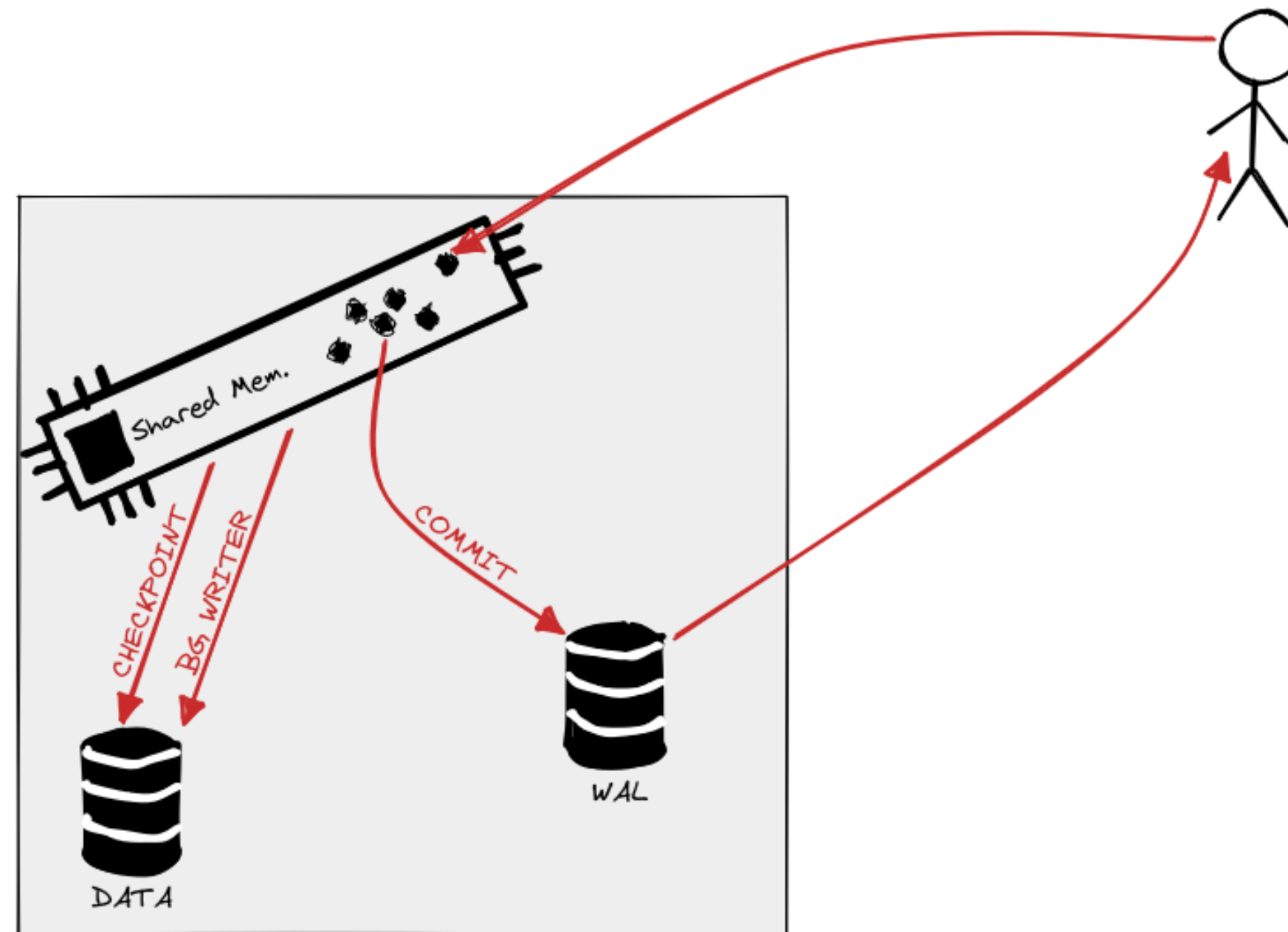
Note that all of the requirements for making use of a full backup also apply to an incremental backup. For instance, you still need all of the WAL segment files generated during and after the file system backup, and any relevant WAL history files. And you still need to create a `recovery.signal` (or `standby.signal`) and perform recovery, as described in [Section 25.3.5](#). The requirement to have earlier backups available at restore time and to use `pg_combinebackup` is an additional requirement on top of everything else. Keep in mind that PostgreSQL has no built-in mechanism to figure out which backups are still needed as a basis for restoring later incremental backups. You must keep track of the relationships between your full and incremental backups on your own, and be certain not to remove earlier backups if they might be needed when restoring later incremental backups.

Incremental backups typically only make sense for relatively large databases where a significant portion of the data does not change, or only changes slowly. For a small database, it's simpler to ignore the existence of incremental backups and simply take full backups, which are simpler to manage. For a large database all of which is heavily modified, incremental backups won't be much smaller than full backups.

# What is an “incremental” backup?

- instead of copying the whole database instance...
  - just copy the data that has changed
-  faster, and less disk space needed
-  more work to restore it!
-  how to quickly and reliably identify which data has changed?

# Write Ahead Log



# WAL summarizer

```
$ git show 174c480508ac25568561443e6d4a82d5c1103487
Author: Robert Haas <rhaas@postgresql.org>
Date:   Wed Dec 20 08:41:09 2023 -0500
    Add a new WAL summarizer process.
```

- `summarize_wal` enables or disables this new background process
- writes WAL summary files into `pg_wal/summaries`
  - one file per checkpoint cycle
- automatically deletes summary files
  - older than `wal_summarize_keep_time` (default is 10 days)

# Example

```
postgres=# ALTER SYSTEM SET summarize_wal TO on;
```

```
ALTER SYSTEM
```

```
postgres=# SELECT pg_reload_conf();
```

```
pg_reload_conf
```

```
-----
```

```
t
```

```
(1 row)
```

```
postgres=# SELECT * FROM pg_settings WHERE name='summarize_wal';
```

```
name          | summarize_wal
```

```
setting       | on
```

```
category      | Write-Ahead Log / Summarization
```

```
short_desc    | Starts the WAL summarizer process to enable incremental backup.
```

```
context       | sighup
```

## Example (2)

```
$ pgbench -i -s 60 pgbench
$ du -hs 17/data/pg_wal/summaries/*
20K 17/data/pg_wal/summaries/000000010000000022402E68000000002F9DEC80.summary
20K 17/data/pg_wal/summaries/00000001000000002F9DEC80000000005041A5D0.summary

$ psql -c "select * from pg_available_wal_summaries();"
 tli | start_lsn | end_lsn
-----+-----+-----
    1 | 0/22402E68 | 0/2F9DEC80
    1 | 0/2F9DEC80 | 0/5041A5D0
```

- for deeper analysis or insights:
  - `pg_available_wal_summaries()`,
  - `pg_get_wal_summarizer_state()`
  - `pg_wal_summary_contents()`, `pg_walsummary` client

## pg\_basebackup

```
$ pg_basebackup --help|grep incremental
-i, --incremental=OLDMANIFEST
    take incremental backup
```

- only works with plain format
- need to provide just backup manifest
  - from the previous backup (aka. reference backup)



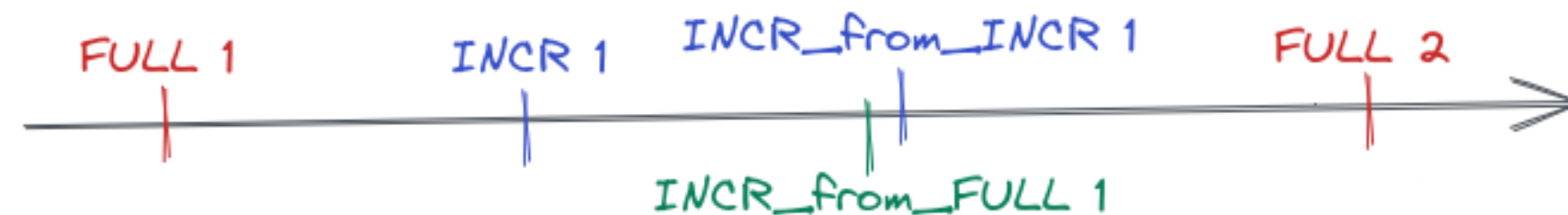
# What's in the manifest?

```
$ pg_basebackup -D full_1 --checkpoint=fast
$ cat full_1/backup_manifest
{"System-Identifier": 7362166132444199273,
  "Files": [
    ...
    { "Path": "base/1/1255", "Size": 811008,
      "Last-Modified": "2024-04-26 13:24:39 GMT",
      "Checksum-Algorithm": "CRC32C", "Checksum": "02b18ea4" },
  ],
  "WAL-Ranges": [
    { "Timeline": 1, "Start-LSN": "0/60000028", "End-LSN": "0/60000120" }
  ],
```

# Example

```
$ pgbench -i -s 10 pgbench
$ pg_basebackup -D incr_1 --incremental=full_1/backup_manifest -c fast

$ pgbench -i -s 10 pgbench
$ pg_basebackup -D incr_from_full1 --incremental=full_1/backup_manifest -c fast
$ pg_basebackup -D incr_from_incr1 --incremental=incr_1/backup_manifest -c fast
$ pg_basebackup -D full_2 --checkpoint=fast
```



## pg\_combinebackup

- reconstruct a data directory from an incremental backup and dependent backups

```
$ pg_combinebackup --output=restore_1 full_1 incr_1 incr_from_incr1  
  
$ pg_combinebackup --output=restore_2 full_1 incr_from_full1  
  
$ pg_combinebackup --output=restore_3 full_1 incr_from_incr1/  
error: backup at "full_1" starts at LSN 0/60000028, but expected 0/69000028
```

- not designed to help you keep track of which backups depend on which other

# Limitations

- does not recompute page checksums
  - take a new full backup after changing the checksum state of the cluster
- only attempts to verify that the backups have the correct relationship to each other...
  - not that each individual backup is intact
  - for that, use `pg_verifybackup`

## pg\_verifybackup

*orthogonality: a tool should do one job and do it as well as possible*

```
$ cp -rfP incr_from_incr1 incr_from_incr1_corrupted
$ rm incr_from_incr1_corrupted/base/1/INCREMENTAL.2675
$ pg_verifybackup incr_from_incr1_corrupted
error: "base/1/INCREMENTAL.2675" is present in the manifest but not on disk

$ pg_combinebackup --output=restore_3 full_1 incr_1 incr_from_incr1_corrupted
$ pg_verifybackup restore_3
backup successfully verified
```

# The “evergreen” backup

- `pg_combinebackup` produces a synthetic full backup...
  - that can be used as source for new backups
  - or for other `pg_combinebackup` invocation

```
# initial idea from Robert Haas
$ pg_basebackup -D sunday_full -c fast
$ pg_basebackup -D monday_incr --incremental=sunday_full/backup_manifest -c fast
$ pg_combinebackup -o monday_full sunday_full monday_incr
$ rm -rf sunday_full monday_incr
```

# Summary

- new feature that needs a lot of orchestration around it
- consider carefully your retention policy
  - run `pg_verifybackup`
  - keep track of the backup references
  - keep `wal_summarize_keep_time` in mind
  - don't over-complicate things!
- regularly try to restore your backups



# About PostgreSQL recovery...

*Schrödinger's Law of Backups*

*The condition/state of any backup is unknown until  
a restore is attempted.*

WEBINAR

## THE PATH TO A SUCCESSFUL POSTGRES SQL RECOVERY

20 JUNE, 13:00-14:00CEST

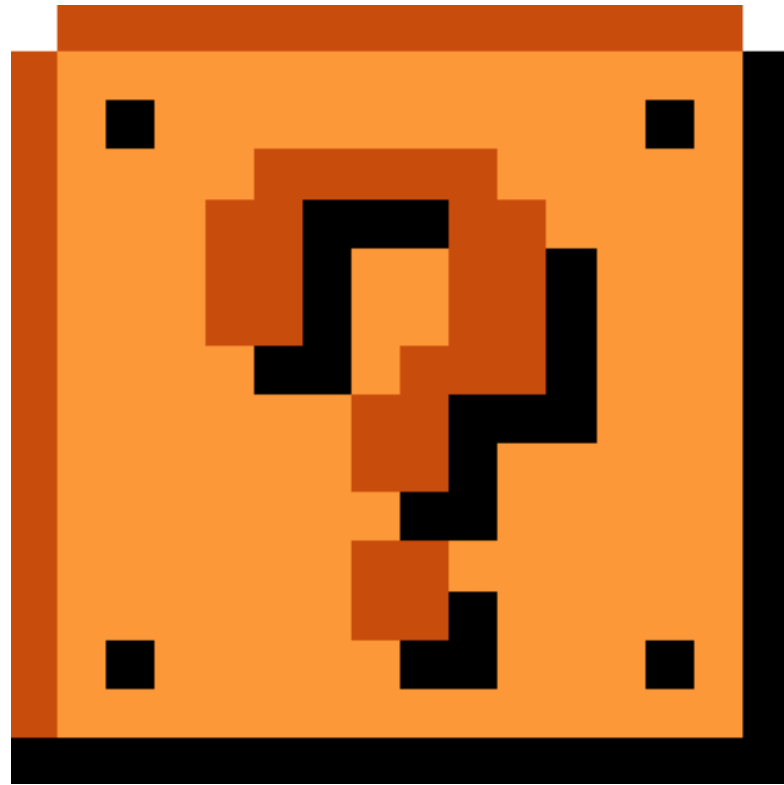
- Step-by-step restore procedure
- Various settings influencing the PostgreSQL recovery
- Practical scenarios using a quick demo setup



**Register!**

To boost your  
confidence in facing  
PostgreSQL recovery  
challenges!

# Questions?



Thank you for your attention!