Hewlett Packard Enterprise

May 25, 2024

PostgreSQL 17 New Features

With Examples (Beta 1)

Hewlett Packard Enterprise Japan G.K.

Noriyoshi Shinoda

Hewlett Packard
Enterprise

Index

Hewlett Packard
Enterprise

1. About This Document

1.1. Purpose

The purpose of this document is to provide information about the major new features of open-source RDBMS PostgreSQL 17 (17.0) Beta 1.

1.2. Audience

This document is written for engineers who already know PostgreSQL, such as installation, basic management, etc.

1.3. Scope

This document describes the major difference between PostgreSQL 16 (16.3) and PostgreSQL 17 (17.0) Beta 1. As a general rule, this document examines the features of behavior change. This document does not describe and verify all new features. In particular, the following new features are not included.

- Bugfix
- Performance improvement by changing internal behavior
- Improvement of regression test
- Operability improvement by the psql command tab input
- Improvement of the pgbench command
- Improvement of documentation, modify typos in the source code
- Refactoring without a change in behavior

1.4. Software Version

The contents of this document have been verified for the following versions and platforms.

Table 1 Version

Software	Version
PostgreSQL	PostgreSQL 16.3 (for comparison) PostgreSQL 17 (17.0) Beta 1 (May 21, 2024 6:55 p.m.)
Operating System	Red Hat Enterprise Linux 8 Update 5 (x86-64)
Configure options	--with-ssl=openssl --with-lz4 --with-zstd --with-llvm --with-libxml --with-icu --enable-injection-points

1.5. The Question, comment, and Responsibility

The contents of this document are not an official opinion of Hewlett Packard Enterprise Japan, G.K. The author and affiliation company do not take any responsibility for the problem caused by the mistake of contents. If you have any comments for this document, please contact Noriyoshi Shinoda (noriyoshi.shinoda@hpe.com), Hewlett Packard Enterprise Japan, G.K.

1.6. Notation

This document contains examples of the execution of the command or SQL statement. Execution examples are described according to the following rules:

Table 2 Examples of notation

Notation	Description
#	Shell prompt for Linux root user.
$	Shell prompt for Linux general user.
Bold	The user input string.
postgres=#	The psql command prompt for PostgreSQL administrator.
postgres=>	The psql command prompt for PostgreSQL general user.
Underline	Important output items.
<<PASSWORD>>	Replaced by password string.
…	Indicates that it is omitted.

The syntax is described in the following rules:

Table 3 Syntax rules

Notation	Description
Italic	Replaced by the name of the object which users use, or the other syntax.
[ ]	Indicate that it can be omitted.
{ A | B }	Indicate that it is possible to select A or B.
…	General syntax. It is the same as the previous version.

2. New Features Summary

More than 200 new features have been added to PostgreSQL 17. This chapter provides an overview of typical new features and benefits. Details of the new features will be explained in "3. New Feature Detail".

2.1. Improvements to adapt to large scale environments

The following features have been added that can be applied to large scale environments:

□ Incremental Backup

Incremental backups are now supported as a standard feature of PostgreSQL. pg_basebackup command now has an option to take incremental backups. To perform an incremental backup, the manifest of the base-backup is used to determine the incremental backup. New command pg_combinebackup merges the base and incremental backups.

□ Improved Partition Table

The features to split an existing partition into multiple partitions and to merge multiple partitions into a single partition have been added.

□ Supports large-scale memory

The SLRU cache is divided into multiple banks, which is expected to reduce the lock range and improve cache search speed. Additionally, several parameters have been added to determine the size of each SLRU memory region.

2.2. Improved reliability

PostgreSQL 17 implements the following enhancements to improve reliability.

□ Logical Replication Slot Synchronization

The replication slot information used for logical replication can now be synchronized to the standby server for streaming replication.

□ Wait for Streaming Replication

The feature has been added to wait for logical replication updates until change data has been sent to the standby server for streaming replication.

2.3. Improved maintainability

The following features that can improve operability have been added.

□ Checkpointer Statistics

The pg_stat_checkpointer view has been added to retrieve checkpointer process statistics. some columns in the pg_stat_checkpointer view have been moved from the pg_stat_bgwriter view.

□ Enhancement of the parameter files

The maximum length of token names that can be used in the pg_hba.conf and pg_ident.conf files has been changed from 256 bytes to unlimited.

□ MAINTAIN privilege

The MAINTAIN privilege has been added to perform maintenance operations such as VACUUM, ANALYZE, and REINDEX statements. The pg_maintain predefined role has been added to allow maintenance operations on objects for all users.

□ Login event trigger

The login event trigger is now available that executes upon successful authentication.

2.4. Improvements in Programming

The following functions have been added to SQL statements.

□ JSON-related

Multiple JSON constructors and many JSONPATH methods have been added. Also added JSON_EXISTS, JSON_QUERY, JSON_VALUE, and JSON_TABLE functions.

□ COPY statement

An option has been added to continue processing in the occurrence of data type conversion errors.

□ MERGE statement

The MERGE statement now supports updatable views. Also, the RETURNING clause and BY SOURCE clause can now be specified.

□ PL/pgSQL

The %TYPE and %ROWTYPE attributes have been added to variable declarations to indicate the data type of table columns and entire tuples.

2.5. Preparing for future new features

PostgreSQL 17 is now ready for features that will be provided in future versions.

□ Wait Event view

The pg_wait_events view has been added to retrieve the names of wait events. Currently, only the name and description of the wait event is available. In the future, it is expected to acquire the cumulative time of wait events, etc.

□ Providing new I/O methods

The foundation for using asynchronous I/O and multiple block I/O is now provided.

2.6. Incompatibility

In PostgreSQL 17, the following specifications have been changed from PostgreSQL 16.

2.6.1. End of support

PostgreSQL 17 changed the supported versions for the following platforms and tools. [1301c80, 8e278b6, 820b5af, 0b16bb8, cc09e65]

The platforms and tools that are no longer supported are listed below.

- Microsoft Visual Studio
- LLVM 9 or earlier
- IBM AIX
- adminpack contrib module

Changes in supported versions of components required to build PostgreSQL 17 are as follows.

- OpenSSL 1.0.2 or later
- LLVM 10 or later

2.6.2. Configure command

The following configure command options have been removed. [68a4b58, 1c1eec0]

Table 4 Removed options

Option	Description	Note
--disable-thread-safety	Disable thread safety for client libraries	
--with-CC	Compiler specification (deprecated since July 2000)	

2.6.3. MERGE statement

The SELECT privilege on the target table is now required even when specifying the DO NOTHING clause in a MERGE statement. This specification will be backported to PostgreSQL 15 and later. [4989ce7]

Example 1 Behavior of PostgreSQL 16.2

```
postgres=# CREATE TABLE merge1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> MERGE INTO merge1 USING (SELECT 1) ON true
               WHEN MATCHED THEN DO NOTHING ;
MERGE 0
```

Example 2 Behavior of PostgreSQL 17

```
postgres=# CREATE TABLE merge1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> MERGE INTO merge1 USING (SELECT 1) ON true
               WHEN MATCHED THEN DO NOTHING ;
ERROR:  permission denied for table merge1
```

2.6.4. Explain statement

The output format of subqueries with the EXPLAIN statement has been changed. [fd0398f]

Example 3 PostgreSQL 16 output

```
postgres=> EXPLAIN (COSTS OFF) SELECT * FROM data1 WHERE
               c1=(SELECT MAX(c1) FROM data1) ;
                                  QUERY PLAN
------------------------------------------------------------------------------
Index Scan using data1_pkey on data1
   Index Cond: (c1 = $1)
   InitPlan 2 (returns $1)
     ->  Result
           InitPlan 1 (returns $0)
             ->  Limit
                   ->   Index  Only  Scan  Backward  using  data1_pkey  on  data1
data1_1
                         Index Cond: (c1 IS NOT NULL)
(8 rows)
```

Example 4 PostgreSQL 17 output

```
postgres=> EXPLAIN (COSTS OFF) SELECT * FROM data1 WHERE
               c1=(SELECT MAX(c1) FROM data1) ;
                                  QUERY PLAN
------------------------------------------------------------------------------
Index Scan using data1_pkey on data1
   Index Cond: (c1 = (InitPlan 2).col1)
   InitPlan 2
     ->  Result
           InitPlan 1
             ->  Limit
                   ->   Index  Only  Scan  Backward  using  data1_pkey  on  data1
data1_1
(7 rows)
```

2.6.5. Pgrowlocks function

The output of the modes column of the results of executing the pgrowlocks function in the pgrowlocks extension has been changed. This change will be backported to the previous version. [15d5d74]

Table 5 Modes column output

Previous version	New version	Note
Share	For Share	
Key Share	For Key Share	

Example 5 Execution result of pgrowlocks function

```
postgres=> SELECT * FROM pgrowlocks('data1') ;
 locked_row | locker | multi | xids  |       modes       |  pids
------------+--------+-------+-------+-------------------+--------
 (0,1)      |    747 | f     | {747} | {"For Share"}     | {9555}
 (0,2)      |    747 | f     | {747} | {"For Key Share"} | {9555}
(2 rows)
```

2.6.6. Other functions

There is an incompatibility with the pg_walfile_name / pg_walfile_name_offset functions. In previous versions, these functions returned the previous segment number when the LSN was on a segment boundary. It has been changed to always return the current segment number of the LSN. This fix is also reflected in previous versions. [344afc7]

3. New Feature Detail

3.1. Architecture

3.1.1. Modified System catalogs

The following catalogs and views have been changed. [1e68e43, 007693f, 46ebdfe, 78806a9, 13aeaf0, 3ee2f25, b0e96f3, e64c733 ,e83d1b0, 96f0526, bc3c8db, 12915a5, 4f62250, 46a0cd4, c393308, 776621a, ddd5f4f, 030e10f, f696c0c, 012460e, 6ae701b, a11f330, 6d49c8d, 7294396, 667e65a, 74604a3, 74604a3]

Table 6 Add system catalogs and views

Catalog/View name	Description
pg_wait_events	The name and description of the waiting event will be output
pg_stat_checkpointer	The execution statistics of checkpointer process is output

Table 7 System catalogs/views with additional columns

Catalog/View name	Add column	Data type	Description
pg_database	dathasloginevt	boolean	Defined login event triggers
pg_replication_slots	failover	boolean	Logical slot enabled to be synced to the standbys
	synced	boolean	Logical slot that was synced from a primary server
	invalidation_reason	text	The reason for the slot's invalidation
	inactive_since	timestamp with time zone	The time since the slot has become inactive
pg_stat_progress_copy	tuples_skipped	bigint	Number of tuples skipped
pg_stat_progress_vacuum	indexes_total	bigint	Total number of indexes that will be vacuumed
	indexes_processed	bigint	Number of indexes processed
	dead_tuple_bytes	bigint	Amount of dead tuple data
	max_dead_tuple_bytes	bigint	Amount of dead tuple data based on

Catalog/View name	Add column	Data type	Description
			maintenance_work_mem
pg_stat_subscription	worker_type	text	Type of the subscription worker process
pg_stats	range_length_histogram	anyarray	A histogram of the lengths of non-empty values
	range_empty_frac	real	Fraction of column entries whose values are empty ranges
	range_bounds_histogram	anyarray	A histogram of lower and upper bounds of non-empty values
pg_subscription	subfailover	boolean	The associated replication slots are enabled to be synchronized to the standbys

Table 8 System catalogs/views with removed columns

Catalog/View name	Remove column	Description
pg_stat_bgwriter	buffers_backend	Move to pg_stat_checkpointer
	buffers_backend_fsync	Move to pg_stat_checkpointer
	checkpoints_timed	Move to pg_stat_checkpointer
	checkpoint_req	Move to pg_stat_checkpointer
	checkpoint_write_time	Move to pg_stat_checkpointer
	checkpoint_sync_time	Move to pg_stat_checkpointer
	buffers_checkpoint	Move to pg_stat_checkpointer
pg_stat_progress_vacuum	num_dead_tuples	Renamed to dead_tuple_bytes
	max_dead_tuples	Renamed to max_dead_tuple_bytes

Table 9 System catalogs/views with changed columns name

Catalog/View name	Before	After	Note
pg_database	daticulocale	datlocale	
pg_collation	coliculocale	collocate	

Table 10 View which in information_schema schema with removed column name

Catalog/View name	Removed column	Description
element_types	domain_default	Due to defects in standard specifications

Table 11 System catalogs/views with modified output

Catalog / View name	Description
pg_attribute	The NOT NULL constraint on the attstattarget column was removed, and the value indicating the default statistics was changed from -1 to NULL.
pg_backend_memory_contexts	The "search_path processing cache" is now output.
pg_statistic_ext	The data type of the stxstattarget column was changed from integer to smallint. stxstattarget column NOT NULL constraints were removed.
pg_stat_wal	WAL information output by the bgwriter process is added. This change is backported to PostgreSQL 14 or later.

□ Pg_wait_events view

The following is the detailed description of the pg_wait_events view added to PostgreSQL 17. The pg_wait_events view provides the names and descriptions of registered wait events. [1e68e43]

Table 12 The pg_wait_events view

Column name	Data type	Description	Note
type	text	Type of Wait Event	Activity, Lock, IO, etc.
name	text	Name of Wait Event	
description	text	Wait Event description	

Example 6 Reference to the pg_wait_events view

```
postgres=> SELECT name, description FROM pg_wait_events ;
      name             |      description
-----------------------+-------------------------------------------------
 ArchiverMain          | Waiting in main loop of archiver process
 AutoVacuumMain        | Waiting in main loop of autovacuum launcher process
 BgWriterHibernate     | Waiting in background writer process, hibernating
 BgWriterMain          | Waiting in main loop of background writer process
...
```

□ Pg_stat_checkpointer view

The following is the detail of the pg_stat_checkpointer view, which is new to PostgreSQL 17. [96f0526, 12915a5]

Table 13 The pg_stat_checkpointer view

Column name	Data type	Description
num_timed	bigint	Number of scheduled checkpoints
num_requested	bigint	Number of checkpoints requested
restart_points_timed	bigint	Number of restart points timed out or rescheduled
restart_points_req	bigint	Number of restart points requested
restart_points_done	bigint	Number of restart points executed
write_time	double precision	Total write time
sync_time	double precision	Total synchronization time
buffers_written	bigint	Number of buffers written
stats_reset	timestamp with time zone	Reset timestamps

Example 7 Reference to the pg_stat_checkpointer view

```
postgres=> SELECT * FROM pg_stat_checkpointer ;
-[ RECORD 1 ]-------+------------------------------
num_timed           | 84
num_requested       | 1
restartpoints_timed | 0
restartpoints_req   | 0
restartpoints_done  | 0
write_time          | 93982
sync_time           | 5
buffers_written     | 935
stats_reset         | 2024-05-23 23:44:02.608669+09
```

3.1.2. Logical Replication Enhancements

The following features have been added to Logical Replication.

□ Integration with Streaming Replication

The WAL Sender (walsender) process of logical replication can be configured to send any update data to the plugin after confirming that the replication slot in streaming replication has received the WAL. The primary server parameter standby_slot_names specifies the name of the streaming replication slot that should confirm the transfer. To use this feature, the 'failover' attribute of the logical

replication slot must be set as described below. If the replication slot specified in standby_slot_names becomes inactive Logical replication transfers are also stopped and the following message is periodically output to the primary server log. [bf279dd]

Example 8 Log when streaming replication standby instance downs

```
WARNING:  replication slot "slot1" specified in parameter standby_slot_names
does not have active_pid
DETAIL:  Logical replication is waiting on the standby associated with "slot1".
HINT:  Consider starting standby associated with "slot1" or amend parameter
standby_slot_names.
```

□ Replication slot new attribute

The parameter 'failover' has been added to the pg_create_logical_replication_slot function. The default value of this parameter is 'false'. Replication slots with this parameter set to 'true' can synchronize LSN information to the standby instance of streaming replication. The 'failover' column has been added to the pg_replication_slots catalog to store the added attribute values. [c393308]

Example 9 Creating the Replication Slot with Failover Attribute

```
postgres=# \df pg_create_logical_replication_slot
List of functions
-[ RECORD 1 ]-------+------------------------------------------------------------
Schema              | pg_catalog
Name                | pg_create_logical_replication_slot
Result data type    | record
Argument data types | slot_name name, plugin name, temporary boolean DEFAULT
false, twophase boolean DEFAULT false, failover boolean DEFAULT false, OUT
slot_name name, OUT lsn pg_lsn
Type                | func

postgres=# SELECT pg_create_logical_replication_slot
               ('logislot1', 'test_decoding', false, false, true) ;
-[ RECORD 1 ]----------------------+----------------------
pg_create_logical_replication_slot | (logislot1,0/70004C0)
```

□ Subscription new attribute

The new attribute FAILOVER can now be specified for subscriptions. To store this attribute, a subfailover column has been added to the pg_subscription catalog. If this attribute is set to TRUE for the subscription, replication slots created on the primary server will also have the FAILOVER attribute at the same time.

Example 10 Create SUBSCRIPTION with FAILOVER attribute

```
postgres=# CREATE SUBSCRIPTION sub1 CONNECTION 'host=dbsvr1 port=5432
        dbname=postgres' PUBLICATION pub1 WITH (FAILOVER = TRUE) ;
NOTICE:  created replication slot "sub1" on publisher
CREATE SUBSCRIPTION
postgres=# SELECT subname, subfailover FROM pg_subscription ;
 subname | subfailover
---------+-------------
 sub1    | t
(1 row)
```

□ Logical Replication Slot Synchronization

The information of logical replication slots created on the primary server can be periodically synchronized to the standby server in streaming replication. [93db6cb]

The following conditions are required on the standby instance to use this feature.

- Set the parameter sync_replication_slots to 'on' (default 'off')
- Set the parameter wal_level to 'logical' (default 'replica')
- Use the replication slot with the 'failover' option specified
- Setting of the parameter primary_slot_name
- Parameter hot_standby_feedback set to 'on' (default 'off')
- Include 'dbname' setting in parameter primary_conninfo

If the above conditions are met, the "slotsync worker" process is launched. If the conditions are not met, the following log is output.

Example 11 Synchronization error log for replication slots

```
LOG:  slot synchronization requires hot_standby_feedback to be enabled
LOG:  slot synchronization requires primary_slot_name to be defined
ERROR:  slot synchronization requires dbname to be specified in primary_conninfo
```

The replication slot synchronization status is output to the log of the standby server. Below is a log of the slot synchronization worker being started and the new replication slot being automatically created.

Example 12 Replication slot synchronization success log

```
LOG:  slot sync worker started
LOG:  newly created slot "sub1" is sync-ready now
```

□ Pg_sync_replication_slots function

PostgreSQL 17 adds the function pg_sync_replication_slots to synchronize logical replication slots on streaming replication standby servers. This function forces synchronization of replication slot information even in environments where the parameter sync_replication_slots is set to 'off'. [ddd5f4f]

Example 13 Forced synchronization of replication slots

```
postgres=# SELECT pg_sync_replication_slots() ;
 pg_sync_replication_slots
---------------------------


(1 row)
```

□ Use hash the index on subscribers

The subscribers can now use not only Btree indexes but also Hash indexes for update processing. [edca342]

□ Pg_logical_emit_message function

Added parameter 'flush' to control WAL flush. The default value is 'false', which means no messages will be flushed. [173b56f]

Syntax 1 Pg_logical_emit_message function

```
pg_lsn pg_logical_emit_message(transactional boolean, prefix text, message
        bytea|text, flush boolean DEFAULT false)
```

3.1.3. Streaming Replication Enhancements

Data to be transferred to the standby instance is now retrieved from the WAL buffer if possible. Previously, the data was reloaded from the WAL file after the writing was completed. [91f2cae]

3.1.4. Incremental Backup

The incremental backups are now available as a standard feature of PostgreSQL. [174c480, dc21234, ee1bfd1, d9ef650, f8ce4ed]

□ WAL summary

To use incremental backup, the WAL summarize feature must be enabled. The WAL summarize feature can be enabled by setting the parameter wal_level to 'replica' or 'logical' and changing the parameter summarize_wal (default value 'off') to 'on'. When this feature is enabled, a "walsummarizer" process is invoked in the instance and summary data of the WAL file is stored in the ${PGDATA}/pg_wal/summaries directory.

Example 14 WAL summary files

```
$ ls data/pg_wal/summaries/
000000010000000001000028000000000010B2D50.summary
000000010000000010B2D5000000000014E8508.summary
000000010000000014E850800000000014E8608.summary
000000010000000014E860800000000014EEF98.summary
000000010000000014EEF9800000000014EF098.summary
000000010000000014EF09800000000014EF3D0.summary
000000010000000014EF3D00000000002000028.summary
000000010000000002000028000000008000028.summary
…
```

The WAL summary file is automatically removed when the parameter wal_summary_keep_time (default value '10d') is exceeded. The following functions are provided to retrieve WAL summary information.

Table 14 WAL summary retrieval functions

Function name	Description
pg_available_wal_summaries	Returns the starting LSN and ending LSN for each WAL summary file
pg_wal_summary_contents	Return update information between specified LSNs
pg_get_wal_summarizer_state	Returns status of WAL summary creation

Syntax 2 WAL summary retrieval functions

```
record pg_available_wal_summaries()
record pg_wal_summary_contents(tli bigint, start_lsn pg_lsn, end_lsn pg_lsn)
record pg_get_wal_summarizer_state()
```

Example 15 Functions for WAL summary information

```
postgres=> SELECT * FROM pg_available_wal_summaries() ;
-[ RECORD 1 ]--------
tli       | 1
start_lsn | 0/5A77BB0
end_lsn   | 0/BF29980
...
postgres=> SELECT * FROM pg_wal_summary_contents(1, '0/5A77BB0', '0/BF29980') ;
-[ RECORD 1 ]--+-----
relfilenode    | 1259
reltablespace  | 1663
reldatabase    | 1
relforknumber  | 0
relblocknumber | 3
is_limit_block | f
...
postgres=> SELECT * FROM pg_get_wal_summarizer_state() ;
-[ RECORD 1 ]--+----------
summarized_tli | 1
summarized_lsn | 0/BF11AE8
pending_lsn    | 0/BF11BF0
summarizer_pid | 9876
```

□ Incremental Backup

The incremental backup is a feature that retrieves the difference between the base backup and the incremental backup.

To obtain incremental backups, use the manifest file output during base backup. The incremental backup is performed by specifying the manifest file of the base backup as the reference for the --incremental option (shortened option -i) of the pg_basebackup command.

Example 16 Execute the incremental backup

```
$ pg_basebackup -D back.1
$ psql postgres demo
psql (17beta1)
Type "help" for help.
postgres=> INSERT INTO data1 VALUES (generate_series(1, 1000000), 'data1') ;
INSERT 0 1000000
postgres=> \q
$ pg_basebackup -D back.inc1 --incremental=back.1/backup_manifest
```

The pg_combinebackup command has been added to merge base and incremental backups. To merge incremental backups, the pg_combinebackup command specifies the directories where base and incremental backup are stored, and the --output option (shortened -o) specifies the directory to be merged. The directory to be merged must not exist or must be an empty directory.

Example 17 Merge backup data

```
$ pg_combinebackup back.1 back.inc1 --output=data.2
$ ls data.2
backup_label      pg_dynshmem    pg_serial     PG_VERSION
backup_manifest   pg_hba.conf    pg_snapshots  pg_wal
base              pg_ident.conf  pg_stat       pg_xact
current_logfiles  pg_logical     pg_stat_tmp   postgresql.auto.conf
global            pg_multixact   pg_subtrans   postgresql.conf
log               pg_notify      pg_tblspc
pg_commit_ts      pg_replslot    pg_twophase
```

□ Manifest file enhancements

The System Identifier is now output in the manifest file. The version of the manifest file has been changed to version 2. [2041bc4]

Example 18 Manifest file enhancements

```
$ pg_basebackup -D back.1
$ grep System-Identifier back.1/backup_manifest
"System-Identifier": 7345983777657046843,
$ grep Manifest-Version back.1/backup_manifest
{ "PostgreSQL-Backup-Manifest-Version": 2,
```

The pg_combinebackup command must specify a backup with the same System Identifier. In the following example, an error occurs when specifying a difference retrieved from a different database.

Example 19 Check the system ID

```
$ pg_combinebackup back.a1 inc.b2 --output=data.1
pg_combinebackup: error: back.a1/global/pg_control: expected system identifier
7347556981648665444, but found 7347557495025884081
```

3.1.5. Partition Table Enhancements

The following new features have been implemented for partition tables.

□ Merge partitions

The ALTER TABLE statement for the RANGE / LIST partition table is now able to merge multiple partitions into a single partition. Specify existing multiple partitions in the MERGE PARTITIONS clause and the name of the newly created partition to be merged in the INTO clause. [1adf16b]

Syntax 3 MERGE PARTITIONS clause

```
ALTER TABLE table_name MERGE PARTITIONS (partition#1, partition#2, …)
        INTO new_partition
```

In the example below, partitions part1v1 and part1v2 are merged into part1v12 partition.

Example 20 Merge partitions

```
postgres=> CREATE TABLE part1(c1 INT PRIMARY KEY, c2 VARCHAR(10))
                PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES
                FROM (0) TO (1000000) ;
CREATE TABLE
postgres=> CREATE TABLE part1v2 PARTITION OF part1 FOR VALUES
                FROM (1000000) TO (2000000) ;
CREATE TABLE
postgres=> INSERT INTO part1 VALUES (generate_series(1,1999999), 'data1') ;
INSERT 0 1999999
postgres=> ALTER TABLE part1 MERGE PARTITIONS (part1v1, part1v2) INTO part1v12 ;
ALTER TABLE
```

The RANGE partitions with non-adjacent data ranges cannot be merged.

Example 21 Partition merge failure

```
postgres=> CREATE TABLE part2(c1 INT PRIMARY KEY, c2 VARCHAR(10))
                PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part2v1 PARTITION OF part2 FOR VALUES
                FROM (0) TO (1000000) ;
CREATE TABLE
postgres=> CREATE TABLE part2v2 PARTITION OF part2 FOR VALUES
                FROM (1000000) TO (2000000) ;
CREATE TABLE
postgres=> CREATE TABLE part2v3 PARTITION OF part2
                FOR VALUES FROM (2000000) TO (3000000) ;
CREATE TABLE
postgres=> ALTER TABLE part2 MERGE PARTITIONS (part2v1, part2v3) INTO part2v13 ;
ERROR:  lower bound of partition "part2v3" conflicts with upper bound of
previous partition "part2v1"
```

□ Allow exclude constraints

Previously, only unique constraints using Btree were allowed for partitions. PostgreSQL 17 now also allows EXCLUDE CONSTRAINT with the same restrictions. However, column comparisons must be performed for equality. In the second example below, an error occurs because the partition key does not include the constraint target column. [8c852ba]

Example 22 Allow exclude constraints

```
postgres=> CREATE TABLE partc1 (col1 INT4RANGE,
        EXCLUDE USING GIST (col1 WITH = )) PARTITION BY RANGE (col1) ;
CREATE TABLE
postgres=> CREATE TABLE partc2 (col1 INT4RANGE, col2 INT4RANGE,
        EXCLUDE USING GIST (col2 WITH =)) PARTITION BY RANGE (col1) ;
ERROR:  unique constraint on partitioned table must include all partitioning
columns
DETAIL:  EXCLUDE constraint on table "partc3" lacks column "col1" which is part
of the partition key.
```

□ Split partition

The SPLIT clause has been added to the ALTER TABLE statement for the RANGE / LIST partition table, which allows partitions to be split into multiple partitions. Specify the source partition in the SPLIT PARTITION clause of the ALTER TABLE statement and the destination partition and partitioning value in the INTO clause. [87c21bb]

Syntax 4 SPLIT PARTITION clause

```
ALTER TABLE table_name SPLIT PARTITION old_partition INTO
        (PARTITION partition#1 VALUES …,
         PARTITION parttiotn#2 VALUES …)
```

Example 23 SPLIT PARTITION clause

```
postgres=> CREATE TABLE part3(c1 INT PRIMARY KEY, c2 VARCHAR(10))
                PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part3v1 PARTITION OF part3 FOR VALUES
                FROM (0) TO (2000000) ;
CREATE TABLE
postgres=> INSERT INTO part3 VALUES (generate_series(1,1999999), 'data1') ;
INSERT 0 1999999
postgres=> ALTER TABLE part3 SPLIT PARTITION part3v1 INTO
             (PARTITION part3v2 FOR VALUES FROM (0) TO (1000000),
              PARTITION part3v3 FOR VALUES FROM (1000000) TO (2000000)) ;
ALTER TABLE
```

□ Support for IDENTITY column

The INSERT statements are now supported on partitions of partitioned tables with IDENTITY columns. The second INSERT statement in the example below will cause an error in PostgreSQL 16. [6995863]

Example 24 Supports for IDENTITY column

```
postgres=> CREATE TABLE part1(c1 INT, c2 INT GENERATED ALWAYS AS IDENTITY)
                PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1
                FOR VALUES FROM (0) TO (1000000) ;
CREATE TABLE
postgres=> INSERT INTO part1(c1) VALUES (100) ;
INSERT 0 1
postgres=> INSERT INTO part1v1(c1) VALUES (200) ; ← PostgreSQL 16 でエラー
INSERT 0 1
postgres=> SELECT * FROM part1 ;
 c1  | c2
-----+----
 100 |  1
 200 |  2
(2 rows)
```

Example 25 Behavior of PostgreSQL 16

```
postgres=> INSERT INTO part1v1(c1) VALUES (200) ;
ERROR:  null value in column "c2" of relation "part1v1" violates not-null
constraint
```

□ Enhancements to Partition Pruning

The IS UNKNOWN, IS NOT UNKNOWN clause for boolean columns now supports partition pruning. [07c36c1]

3.1.6. Roles and Privileges

Predefined roles and privileges that can be granted have been added.

□ MAINTAIN privilege

The MAINTAIN privilege can now be granted on tables and materialized views. A role granted this privilege can execute VACUUM, ANALYZE, REINDEX, REFRESH MATERIALIZED VIEW, CLUSTER, and LOCK TABLE statements on the target relation. The result of the \dp meta-command in the psql command will output 'm'. [ecb0fd3]

Example 26 Grant MAINTAIN privilege

```
postgres=# CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=# GRANT MAINTAIN ON data1 TO demo ;
GRANT
postgres=# \dp data1
                                Access privileges
 Schema | Name  | Type  |     Access privileges      | Column privileges | …
--------+-------+-------+----------------------------+-------------------+-…
 public | data1 | table | postgres=arwdDxtm/postgres+|                   |
        |       |       | demo=m/postgres            |                   |
(1 row)

postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> VACUUM data1 ;
VACUUM
```

□ Addition of pg_maintain role

The pg_maintain predefined role has been added. This role is able to grant MAINTAIN privilege on all relations.

□ Modification of pg_monitor role

The predefined role pg_monitor has been granted permission to execute the pg_current_logfile function. [8d8afd4]

Example 27 Execution privilege of pg_current_logfile function

```
postgres=# CREATE USER demo PASSWORD '<<password>>' ;
CREATE ROLE
postgres=# GRANT pg_monitor TO demo ;
GRANT ROLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> SELECT pg_current_logfile() ;
          pg_current_logfile
---------------------------------------
 log/postgresql-2024-05-24_085557.log
(1 row)

postgres=> SELECT pg_current_logfile('csvlog') ;
 pg_current_logfile
--------------------


(1 row)
```

3.1.7. Built-in Locale Provider

PostgreSQL 17 has added a 'built-in' locale provider that is independent of external environments such as Libc or icu. The BUILTIN can now be specified in commands and SQL statements that specify a locale provider. [2d819a0, f69319f]

Example 28 Specifying a builtin locale provider

```
postgres=# CREATE DATABASE builtdb LOCALE_PROVIDER BUILTIN LOCALE 'C.UTF-8'
               TEMPLATE template0 ;
CREATE DATABASE
postgres=# SELECT datcollate, datctype, datlocale FROM pg_database
               WHERE datname='builtdb' ;
-[ RECORD 1 ]-------
datcollate | C.UTF-8
datctype   | C.UTF-8
datlocale  | C.UTF-8
```

Currently, the only locales offered by this locale provider are C and C.UTF-8. The built-in locale C.UTF-8 provides advantages versus the libc locale, such as faster sorting, faster case conversion, and is platform-independent.

3.1.8. Event Trigger

The following event triggers have been added.

□ Login event trigger

The login event trigger, which is executed on successful authentication from the client, has been added. The following is an example of the event trigger that appends the connection time to the table. [e83d1b0]

Example 29 Publishing login event trigger

```
postgres=# CREATE OR REPLACE FUNCTION save_login()
             RETURNS event_trigger SECURITY DEFINER
             LANGUAGE plpgsql AS $$
           BEGIN
             INSERT INTO public.login_history VALUES(current_timestamp);
           END; $$ ;
CREATE FUNCTION
postgres=# CREATE EVENT TRIGGER login_trigger ON login
             EXECUTE FUNCTION save_login() ;
CREATE EVENT TRIGGER
postgres=# ALTER EVENT TRIGGER login_trigger ENABLE ALWAYS ;
ALTER EVENT TRIGGER
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> SELECT * FROM login_history ;
           login
-------------------------------
 2024-05-24 15:44:21.889142
(1 row)
```

□ Reindex DDL trigger

The DDL event triggers can now be generated at the start and end of the execution of REINDEX statements. [f21848d]

Example 30 Generate REINDEX event trigger (REINDEX start)

```
postgres=# CREATE OR REPLACE FUNCTION reindex_start_func()
           RETURNS event_trigger AS $$
            BEGIN
              RAISE NOTICE 'START REINDEX: % %', tg_event, tg_tag ;
            END; $$ LANGUAGE plpgsql ;
CREATE FUNCTION
postgres=# CREATE EVENT TRIGGER trg_reindex_start ON ddl_command_start
             WHEN TAG IN ('REINDEX')
             EXECUTE PROCEDURE reindex_start_func() ;
CREATE EVENT TRIGGER
postgres=# REINDEX INDEX idx1_data1 ;
NOTICE:  START REINDEX: ddl_command_start REINDEX
REINDEX
```

Example 31 Generate REINDEX event trigger (REINDEX finish)

```
postgres=# CREATE FUNCTION reindex_end_func()
           RETURNS event_trigger AS $$
             DECLARE
               obj record ;
             BEGIN
               FOR obj IN SELECT * FROM pg_event_trigger_ddl_commands()
               LOOP
                 RAISE NOTICE 'REINDEX END: command_tag=% type=% identity=%',
                   obj.command_tag, obj.object_type, obj.object_identity ;
               END LOOP ;
            END; $$ LANGUAGE plpgsql ;
CREATE FUNCTION
postgres=# CREATE EVENT TRIGGER trg_reindex_end ON ddl_command_end
             WHEN TAG IN ('REINDEX')
             EXECUTE PROCEDURE reindex_end_func() ;
CREATE EVENT TRIGGER
postgres=# REINDEX TABLE data1 ;
NOTICE:  START REINDEX: ddl_command_start REINDEX
NOTICE:  REINDEX END: command_tag=REINDEX type=index identity=public.idx1_data1
NOTICE:  REINDEX END: command_tag=REINDEX type=index identity=public.idx2_data1
REINDEX
```

3.1.9. Configuration File

The maximum length of tokens in the pg_hba.conf and pg_ident.conf files has been extended from 256 bytes to unlimited. In complex environments configured with LDAP, there were problems with exceeding the previous maximum length. [38df84c]

3.1.10. Log files

The following new changes have been implemented in the output of the server logs.

□ Logging of trust connections

When the parameter log_connections is set to "on", logs are output for trust connections as follows. [38df84c]

Example 32 Logging for trust connection

```
LOG:  connection received: host=[local]
LOG:  connection authenticated: user="postgres" method=trust
        (/postgres/data/pg_hba.conf:117)
LOG:  connection authorized: user=postgres database=postgres
        application_name=psql
```

□ Replication Slot logging

The logs about replication slots are output at the LOG level (usually at the DEBUG1 level) when the parameter log_replication_commands is set to 'on'. The following example shows the logs at the beginning and end of replication. [7c3fb50]

Example 33 Replication slot logging

```
LOG:  received replication command: START_REPLICATION SLOT "slot1" 0/3000000
TIMELINE 1
LOG:  acquired physical replication slot "slot1"
STATEMENT:  START_REPLICATION SLOT "slot1" 0/3000000 TIMELINE 1
LOG:  released physical replication slot "slot1"
```

□ Recovery related logging

The logs at the start and end of the recovery process have been added. The following example shows the log at instance startup for streaming replication. [1d35f70]

Example 34 Recovery logging

```
LOG:  starting backup recovery with redo LSN 0/2000028, checkpoint LSN
0/2000080, on timeline ID 1
LOG:  entering standby mode
LOG:  redo starts at 0/2000028
LOG:  completed backup recovery with redo LSN 0/2000028 and end LSN 0/2000120
```

3.1.11. Hook

The following Object Access Type (OAT) hooks have been added to the ALTER TABLE statement. [352ea3a]

- { ENABLE | DISABLE | [NO] FORCE } ROW LEVEL SECURITY
- { ENABLE | DISABLE } TRIGGER
- { ENABLE | DISABLE } RULE

3.1.12. Libpq

The following APIs have been added and enhanced.

□ Connect string sslnegotiation

The connection option sslnegotiation has been added. This option determines how the encryption method is negotiated during the client connection. This option can also be specified by the environment variable PGSSLNEGOTIATION. [d39a49c, fb5718f]

Table 15 Possible values for the option

Value	Description	Note
postgres	Perform PostgreSQL protocol negotiation.	Default
direct	First attempt to establish a standard SSL connection and if that fails reconnect and perform the negotiation.	
requiredirect	Attempt to establish a standard SSL connection and if that fails return a connection failure immediately.	

□ Close Portals and Statements

Functions to close portals and statements have been added. Functions with "send" in the name are non-blocking versions. [28b5726]

Table 16 Added functions

Function	Description
PQclosePrepared	Close the prepaid statement.
PQclosePortal	Close the portal.
PQsendClosePrepared	Close the Prepared statement. Do not wait for completion.
PQsendClosePortal	Closes the portal. Do not wait for completion.

□ Password change

The function PQchangePassword has been added to change the password of a connected user. [a7be2a6]

□ Changes to chunk mode

The PGresults now retrieves multiple tuples at once. The function PQsetChunkedRowsMode has been added to change the mode accordingly. [4643a2b]

□ Pipeline Synchronization

The PQsendPipelineSync function has been added. This function is similar to the PQpipelineSync function, but it does not flush a synchronization message to the server unless the output buffer reaches a threshold value. [4794c2]

□ Sequence operation

The functions have been added to manipulate sequences, such as the relation_open function. [449e798]

Table 17 Added functions

Function	Description	Note
sequence_open	Open the sequence	
sequence_close	Close the sequence	

□ Injection Points

The following functions have been added to support injection points. [d86d20f]

Table 18 Added functions

Function	Description	Note
InjectionPointAttach	Attaching Callbacks	
InjectionPointDetach	Detaching a callback	

□ Non-Blocking cancel functions

Previously, the PQcancel API used blocking requests. It is now possible to send a cancel request without blocking. [61461a3]

Table 19 Added functions

Function	Description
PQcancelCreate	Prepare to send a cancel request.
PQcancelBlocking	Requesting discard of the current command in blocking state.
PQcancelPoll	Request discard of current command in non-blocking state.
PQcancelStart	Request discard of current command in non-blocking state.
PQcancelStatus	Retrieve the status of a canceled connection.
PQcancelSocket	Retrieve the identifier of the canceled connection socket.
PQcancelErrorMessage	Get the last error generated by the cancel operation.
PQcancelFinish	Close the connection.
PQcancelReset	Reset PGcancelConn and reuse it for a new connection.

□ Background Worker

The flag BGWORKER_BYPASS_ROLELOGINCHECK can now be specified in the functions BackgroundWorkerInitializeConnection and BackgroundWorkerInitializeConnectionBy-Oid. This flag allows bypassing the role login check. [e768919]

□ DSM Registry

The function GetNamedDSMSegment has been added to allow shared libraries to easily use shared memory. Previously, shared libraries specified in the parameter shared_preload_libraries had to request shared memory via the shmem_request_hook hook. [8b2bcf3]

□ DSM Initial Size

Previously, DSMs were created at 1 MB and the maximum is still extended to DSA_MAX_SEGMENT_SIZE. The dsa_create_ext and dsa_create_in_place_ext functions have been added to allow the initial and maximum sizes to be specified. [bb952c8]

□ Streaming I/O

Abstracts access to relations and allows relation data to be accessed as a stream of buffers. This modification will make it easier to make changes in the future when implementations such as asynchronous I/O are added. It is used to execute the ANALYZE statement and perform a sequential scans on the table. [b7b0f3f, 041b968, b5a9b18]

Table 20 Added functions

Function	Description	Note
read_stream_begin_relation	Create a new read streaming object.	
read_stream_next_buffer	Retrieve a single fixed buffer from the stream.	
read_stream_reset	Release the queued buffer and reset the stream.	
read_stream_end	Release the stream.	

□ Multi-block read

The multi-block version of the ReadBuffer API is now provided. The maximum number of blocks is determined by the new parameter io_combine_limit. In the future, asynchronous I/O may be used internally. [210622c]

Table 21 Added functions

Function	Description	Note
StartReadBuffer	Start reading a single block.	
StartReadBuffers	Start reading multi blocks.	
WaitReadBuffers	Wait for the completion of block read.	

□ Bump Memory Allocation

The memory allocator BumpContext has been added, which is suitable for memory allocations with short life times. BumpContext memory is now used for sorting tuples and for VACUUM processing. [29f6a95, 6ed83d5, 8a1b31e]

Table 22 Added functions

Function	Description
BumpContextCreate	Create Bump MemoryContext.
BumpAlloc	Allocate Bump memory.
BumpFree	Free Bump memory.
BumpRealloc	Re-allocate Bump memory.
BumpReset	Frees all memory which is allocated in the given set.
BumpDelete	Frees all memory which is allocated in the given context.
BumpGetChunkContext	Un-supported.
BumpGetChunkSpace	Un-supported.
BumpIsEmpty	Returns true if 'block' contains no chunk.
BumpStats	Compute stats about memory consumption of the context.
BumpCheck	Walk through chunks and check consistency of memory.

□ Polling file descriptor

The PQsocketPoll function has been provided to monitor file descriptors. [f5e4ded]

3.1.13. Access Method

The following enhancements have been added to the access methods.

□ Parallel build of BRIN index

The multiple worker processes can now be used to build BRIN indexes. The amcanbuildparallel flag has been added to the index access method structure IndexAmRoutine. [b437571]

□ heap_page_prune function

The parameters of the HEAP_PAGE_PRUNE function have been changed to allow multiple options to be specified in a bitmap. Currently, only HEAP_PAGE_PRUNE_MARK_UNUSED_NOW and HEAP_PAGE_PRUNE_FREEZE options are available. [3d0f730, 6dbb490]

3.1.14. Wait Events

The following features have been added for wait events.

□ Modified Wait Events

The following wait events have been added or changed. [fa88928, c9af054, c8e318b, 0013ba2, 5c430f9, d86d20f]

Table 23 Added Wait Events

Wait Event Name	Class	Description
CheckpointDelayComplete	IPC	Waiting for the backend to complete checkpoint
CheckpointDelayStart	IPC	Waiting for backend to start checkpoint
MultixactCreation	IPC	Waiting for multi-transaction creation
ReplicationSlotsyncMain	Activity	Replication slot synchronization wait
ReplicationSlotsyncShutdown	Activity	Waiting for replication slot to stop
WaitForStandbyConfirmation	Client	Waiting for standby confirmation
WalSummarizerError	Timeout	WAL summary error
WALSummarizerWal	Activity	Waiting for WAL summary I/O
WalSummaryRead	IO	Waiting for WAL summary to load
WalSummaryReady	IPC	Waiting for WAL summary generation
WalSummaryWrite	IO	Waiting for WAL summary writing
InjectionPoint	LWLock	Waiting for injection point loading

Table 24 Renamed wait event

Name before change	Name after change	Note
AutoVacuumMain	AutovacuumMain	
BaseBackupRead	BasebackupRead	
BaseBackupSync	BasebackupSync	
BaseBackupWrite	BasebackupWrite	
BgWorkerShutdown	BgworkerShutdown	
BgWorkerStartup	BgworkerStartup	
BgWriterHibernate	BgwriterHibernate	
BgWriterMain	BgwriterMain	
BufferIO	BufferIo	
BufFileRead	BuffileRead	
BufFileTruncate	BuffileTruncate	
BufFileWrite	BuffileWrite	
DSMAllocate	DsmAllocate	
DSMFillZeroWrite	DsmFillZeroWrite	

Name before change	Name after change	Note
GSSOpenServer	GssOpenServer	
LibPQWalReceiverConnect	LibpqwalreceiverConnect	
LibPQWalReceiverReceive	LibpqwalreceiverReceive	
LockFileAddToDataDirRead	LockFileAddtodatadirRead	
LockFileAddToDataDirSync	LockFileAddtodatadirSync	
LockFileAddToDataDirWrite	LockFileAddtodatadirWrite	
LockFileReCheckDataDirRead	LockFileRecheckdatadirRead	
ProcArrayGroupUpdate	ProcarrayGroupUpdate	
SLRUFlushSync	SlruFlushSync	
SLRURead	SlruRead	
SLRUSync	SlruSync	
SLRUWrite	SlruWrite	
SSLOpenServer	SslOpenServer	
SysLoggerMain	SysloggerMain	
WALBootstrapSync	WalBootstrapSync	
WALBootstrapWrite	WalBootstrapWrite	
WALCopyRead	WalCopyRead	
WALCopySync	WalCopySync	
WALCopyWrite	WalCopyWrite	
WALInitSync	WalInitSync	
WALInitWrite	WalInitWrite	
WALRead	WalRead	
WALSenderTimelineHistoryRead	WalsenderTimelineHistoryRead	
WalSenderWaitForWAL	WalSenderWaitForWal	
WALSync	WalSync	
WALSyncMethodAssign	WALSyncMethodAssign	
WALWrite	WALWrite	

□ List of Wait Events

The list of wait events is now written in the file src/backend/utils/activity/wait_event_names.txt and the code and documentation are generated automatically. [59cbf60]

Example 35 The part of wait_event_names.txt

```
...
Section: ClassName - WaitEventClient

CLIENT_READ     "Waiting to read data from the client."
CLIENT_WRITE    "Waiting to write data to the client."
...
```

□ Custom Type

The method of registering custom wait events has been changed. New API "uint32 WaitEventExtensionNew(const char *wait_event_name)" is provided. With this, the shared memory hook is no longer required. [af720b4]

3.1.15. FILLFACTOR

The FILLFACTOR is now considered in the estimated number of tuples when table statistics are not gathered. Previously, FILLFACTOR was not considered, which could result in an overestimation of the number of tuples. [29cf61a]

Example 36 Estimated number of tuples in PostgreSQL 17

```
postgres=> CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) WITH (FILLFACTOR=50) ;
CREATE TABLE
postgres=> INSERT INTO data1 VALUES (generate_series(1, 100000), 'data1') ;
INSERT 0 100000
postgres=> EXPLAIN SELECT COUNT(*) FROM data1 ;
                            QUERY PLAN
-------------------------------------------------------------------
 Aggregate  (cost=1875.08..1875.09 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..1717.46 rows=63046 width=0)
(2 rows)
```

Example 37 Estimated number of tuples in PostgreSQL 16

```
postgres=> CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) WITH (FILLFACTOR=50) ;
CREATE TABLE
postgres=> INSERT INTO data1 VALUES (generate_series(1, 100000), 'data1') ;
INSERT 0 100000
postgres=> EXPLAIN SELECT COUNT(*) FROM data1 ;
                              QUERY PLAN
-----------------------------------------------------------------------
 Aggregate  (cost=2663.15..2663.16 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..2347.92 rows=126092 width=0)
(2 rows)
```

3.1.16. VACUUM

The TidStore is now used to store tuple IDs for VACUUM instead of a simple array. This change reduces the amount of memory used, which used to be larger, and eliminates the 1 GB limit. In addition, some column names in the pg_stat_progress_vacuum view have been changed. [667e65a]

3.1.17. LLVM

PostgreSQL 17 now supports LLVM 17, 18. Backported to supported older versions of PostgreSQL. [76200e5, d282e88]

3.1.18. Acceleration

The CRC calculation function COMP_CRC32C can now be executed natively in environments that support LoongArch. [4d14ccd]

In environments where the Intel AVX-512 instruction set can be used, visibilitymap_count and pg_popcount functions can now be executed faster. [41c51f0, 792752a]

3.1.19. UNICODE

PostgreSQL 17 has the following changes in Unicode support. [9d17e5f, 9e2e4f0, ad49994]

- Unicode version changed from 15.0.0 to 15.1.0 (https://unicode.org/versions/Unicode15.1.0/)

- Unicode CLDR version changed from 43 to 45 (https://cldr.unicode.org/index/downloads/cldr-45)
- Add update-unicode build target to update UNICODE table

3.2. SQL Statement

This section explains new features related to SQL statements.

3.2.1. ALTER OPERATOR

The following attributes can now be altered by the ALTER OPEARTOR statement. Previously, these could only be specified in the CREATE OPERATOR statement. However, once a flag is set, it cannot be changed. [2b5154b]

- COMMUTATOR
- NEGATOR
- HASHES
- MERGES

Example 38 Attribute modification by ALTER OPERATOR statement

```
postgres=# CREATE OR REPLACE FUNCTION alter_op_test_fn_bool_real(boolean, real)
        RETURNS boolean AS $$ SELECT NULL::BOOLEAN; $$ LANGUAGE SQL IMMUTABLE ;
CREATE FUNCTION
postgres=# CREATE OPERATOR === (
             LEFTARG = boolean,
             RIGHTARG = real,
             PROCEDURE = alter_op_test_fn_bool_real) ;
CREATE OPERATOR
postgres=# ALTER OPERATOR === (boolean, real) SET (MERGES = false) ;
ALTER OPERATOR
postgres=# ALTER OPERATOR === (boolean, real) SET (HASHES = false) ;
ALTER OPERATOR
```

3.2.2. ALTER SYSTEM

A parameter containing module names (e.g., foo.bar) used by extensions, etc., can now be specified in the ALTER SYSTEM statement. In the previous version, only parameters for which a SET statement had been executed in the past could be set with the ALTER SYSTEM statement. [2d870b4]

Example 39 ALTER SYSTEM statement to change parameters for extensions

```
postgres=# ALTER SYSTEM SET foo.bar = 100 ;
ALTER SYSTEM
postgres=# SELECT pg_reload_conf() ;
 pg_reload_conf
----------------
 t
(1 row)

postgres=# SHOW foo.bar ;
 foo.bar
---------
 100
(1 row)
```

3.2.3. ALTER TABLE

The following enhancements have been implemented to the ALTER TABLE statement.

□ Change to default access method

To change to the access method specified in the parameter default_table_access_method, it is now possible to specify DEFAULT for the access method name. [d61a6ca]

Syntax 5 ALTER TABLE SET ACCESS METHOD statement

```
ALTER TABLE table_name SET ACCESS METHOD DEFAULT
```

Example 40 Change access method

```
postgres=> ALTER TABLE data1 SET ACCESS METHOD DEFAULT ;
ALTER TABLE
```

□ Change Generate Column

The formulas for generated columns (GENERATE clause) can now be changed. If the formula for the column is changed, the existing data in the table will be recalculated. [5d06e99]

Syntax 6 ALTER TABLE SET EXPRESSION statement

```
ALTER TABLE table_name ALTER COLUMN column_name SET EXPRESSION AS (expression)
```

Example 41 Specifying the SET EXPRESSION clause

```
postgres=> CREATE TABLE data1(c1 INT, c2 INT GENERATED ALWAYS AS
               (c1 * 2) STORED) ;
CREATE TABLE
postgres=> INSERT INTO data1(c1) VALUES (100) ;
INSERT 0 1
postgres=> SELECT * FROM data1 ;
 c1  | c2
-----+-----
 100 | 200
(1 row)

postgres=> ALTER TABLE data1 ALTER COLUMN c2 SET EXPRESSION AS (c1 * 200) ;
ALTER TABLE
postgres=> SELECT * FROM data1 ;
 c1  |  c2
-----+-------
 100 | 20000
(1 row)
```

□ Statistics Settings

It is now possible to use DEFAULT when setting the volume of statistics stored in a single column to the parameter default_statistics_target. In previous versions, -1 was specified. When using the default value, the attstattarget column in the pg_attribute catalog will be set to the NULL value. Previously, the NOT NULL constraint was specified for this column. [4f62250, 5567996]

Syntax 7 ALTER TABLE SET STATISTICS statement

```
ALTER TABLE table_name ALTER COLUMN column_name SET STATISTICS DEFAULT
```

Example 42 Specifying the SET STATISTICS clause

```
postgres=> ALTER TABLE data1 ALTER COLUMN c1 SET STATISTICS 110 ;
ALTER TABLE
postgres=> SELECT attname, attstattarget FROM pg_attribute WHERE attrelid=
  (SELECT oid FROM pg_class WHERE relname='data1') AND attname='c1' ;
 attname | attstattarget
---------+---------------
 c1      |           110
(1 row)

postgres=> ALTER TABLE data1 ALTER COLUMN c1 SET STATISTICS DEFAULT ;
ALTER TABLE
postgres=> SELECT attname,attstattarget FROM pg_attribute WHERE attrelid=
   (SELECT oid FROM pg_class WHERE relname='data1') AND attname='c1' ;
 attname | attstattarget
---------+---------------
 c1      |          null
(1 row)
```

3.2.4. CLUSTER

The syntax for enclosing options in parentheses (()) is now permitted when the table name is omitted. This enhancement supports the same syntax as the REINDEX and VACUUM statements. [cdaedfc]

Example 43 CLUSTER with VERBOSE clause

```
postgres=> CLUSTER (VERBOSE) ;
INFO:  clustering "public.data1" using index scan on "data1_pkey1"
INFO:  "public.data1": found 0 removable, 1000000 nonremovable row versions in
5406 pages
DETAIL:  0 dead row versions cannot be removed yet.
CPU: user: 0.22 s, system: 0.00 s, elapsed: 0.25 s.
CLUSTER
```

3.2.5. COPY

The following new features have been implemented in the COPY statement.

□ FORCE_NULL option

An asterisk (*) can now be specified in the FORCE_NULL and FORCE_NOT_NULL options of the COPY statement. [f6d4c9c]

Example 44 Specifying the FORCE_NULL * option

```
postgres=> COPY data1 FROM stdin WITH (FORMAT csv, FORCE_NULL *) ;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself, or an EOF signal.
>> 100,data1
>> 200,""
>> 300,data3
>> \.
COPY 3
postgres=> SELECT * FROM data1 ;
 c1  |  c2
-----+-------
 100 | data1
 200 | null
 300 | data3
(3 rows)
```

□ ON_ERROR option

The ON_ERROR option has been added to the COPY FROM statement. This option changes the behavior when an error occurs. The following values can be specified for this option. [9e2d870, b725b7e, a6d0fa5]

Table 25 Available value for the ON_ERROR option

Value	Description
stop	Stop processing when an error occurs.
ignore	Ignore the occurrence of data conversion errors and continue the processing.

Even if this option is set to 'ignore', the COPY statement will terminate abnormally if a constraint violation (primary key constraint or CHECK constraint) occurs.

Example 45 Specifying the ON_ERROR option

```
postgres=> CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> COPY data1 FROM STDIN WITH (FORMAT csv, ON_ERROR ignore) ;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself, or an EOF signal.
>> 100,data1
>> 200,data2
>> ABC,data3
>> 400,data4
>> \.
NOTICE:  1 row were skipped due to data type incompatibility
COPY 3
postgres=> SELECT * FROM data1 ;
 c1  |    c2
-----+--------
 100 |  data1
 200 |  data2
 400 |  data4
(3 rows)
```

□ LOG_VERBOSITY option

The LOG_VERBOSITY option has been added to the COPY statement to change the level of log output. The LOG_VERBOSITY option controls the output of errors generated by a COPY FROM statement with the ON_ERROR option set to 'ignore'. The possible values are as follows. [f5a2278]

Table 26 Available value for the LOG_VERBOSITY option

Value	Description	Note
default	Output standard log level.	Default
verbose	Output verbose log level.	

Example 46 Specifying the LOG_VERBOSITY option

```
postgres=> COPY data1 FROM STDIN WITH (FORMAT csv, ON_ERROR ignore,
                LOG_VERBOSITY default) ;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself, or an EOF signal.
>> 100,data1
>> 200,data2
>> ABC,data3
>> 400,data4
>> \.
NOTICE:  1 row was skipped due to data type incompatibility
COPY 3
postgres=> TRUNCATE TABLE data1 ;
TRUNCATE TABLE
postgres=> COPY data1 FROM STDIN WITH (FORMAT csv, ON_ERROR ignore,
                LOG_VERBOSITY verbose) ;
Enter data to be copied followed by a newline.
End with a backslash and a period on a line by itself, or an EOF signal.
>> 100,data1
>> 200,data2
>> ABC,data3
>> 400,data4
>> \.
NOTICE:  skipping row due to data type incompatibility at line 3 for column c1:
"ABC"
NOTICE:  1 row was skipped due to data type incompatibility
COPY 3
```

3.2.6. CREATE TABLE

The access method can now be specified for the partitioned table, using the CREATE TABLE USING or ALTER TABLE SET ACCESS METHOD statements. The specified value will be used as the default value for the access method for the partitions created under the partitioned table. The existing partitions will not be modified. [374c7a2]

Syntax 8 CREATE TABLE PARTITION BY USING statement

```
CREATE TABLE table_name(column,…) PARTITION BY partition_method (column)
        USING table_access_method
```

Example 47 Specifying the access method

```
postgres=> CREATE TABLE part1 (c1 INT PRIMARY KEY, c2 VARCHAR(10))
                PARTITION BY RANGE(c1) USING heap ;
CREATE TABLE
postgres=> ALTER TABLE part1 SET ACCESS METHOD DEFAULT ;
ALTER TABLE
```

3.2.7. EXPLAIN

The following new features have been added to the EXPLAIN statement.

□ Enhancements to JSON Format

Local I/O Read Time" and "Local I/O Write Time" are now output in JSON format. To enable this output, the parameter track_io_timing must be set to 'on'. [295c36c]

Example 48 EXPLAIN statement in JSON format

```
postgres=> EXPLAIN (ANALYZE, BUFFERS, FORMAT json) SELECT * FROM data1 ;
              QUERY PLAN
--------------------------------------
 [                                   +
   {                                 +
     "Plan": {                       +
       "Node Type": "Seq Scan",      +
       "Parallel Aware": false,      +
…
       "Shared I/O Write Time": 0.000,+
       "Local I/O Read Time": 0.000,  +
       "Local I/O Write Time": 0.000, +
       "Temp I/O Read Time": 0.000,   +
…
```

□ MEMORY option

The MEMORY option has been added to the EXPLAIN statement to output the memory settings when creating an execution plan. The EXPLAIN statement with the MEMORY clause outputs information about the amount of memory allocated (allocated) and the amount of memory actually used (used). [5de890e]

Example 49 EXPLAIN statement with MEMORY clause

```
postgres=> EXPLAIN (MEMORY) SELECT COUNT(*) FROM data1 ;
                              QUERY PLAN
-----------------------------------------------------------------------
 Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
 Planning:
   Memory: used=9kB allocated=16kB
(4 rows)
```

□ SERIALIZE option

The SERIALIZE clause has been added that outputs information on the amount of data and time output to the client by the query execution. This option is used in conjunction with the ANALYZE clause. The following values can be added to the option. [0628670]

Table 27 Available value for the SERIALIZE option

Value	Description	Note
none	Do not output serialization information	Default
text	Output in text format	Default for SERIALIZE only
binary	Output in binary format	

Example 50 EXPLAIN statement with SERIALIZE clause

```
postgres=> EXPLAIN (ANALYZE, SERIALIZE BINARY) SELECT * FROM data1 ;
                                QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=10) (actual
 time=0.005..36.797 rows=1000000 loops=1)
 Planning Time: 0.056 ms
 Serialization: time=84.192 ms  output=18555kB  format=binary
 Execution Time: 154.041 ms
(4 rows)
```

3.2.8. MERGE

The following new features have been added to the MERGE statement.

□ BY SOURCE clause

The condition specification ``WHEN NOT MATCHED BY SOURCE'' has been added to the MERGE statement. This condition operates on tuples that do not exist in the source table but exist in the target table. The UPDATE, DELETE, and DO NOTHING clauses can be specified as the action for the target tuple. [0294df2]

Example 51 Specifying the BY SOURCE clause

```
postgres=> SELECT * FROM src1 ;
 c1  |  c2
-----+---------
 100 | source1
 200 | source2
(2 rows)

postgres=> SELECT * FROM tgt1 ;
 c1  |  c2
-----+---------
 100 | target1
 300 | target3
(2 rows)

postgres=> MERGE INTO tgt1 AS t USING src1 AS s ON s.c1 = t.c1
    WHEN NOT MATCHED THEN INSERT VALUES (s.c1, s.c2)
    WHEN NOT MATCHED BY SOURCE THEN UPDATE SET c2='not matched' ;
MERGE 2
postgres=> SELECT * FROM tgt1 ;
 c1  |    c2
-----+------------
 100 | target1
 200 | source2
 300 | not matched
(3 rows)
```

The traditional WHEN NOT MATCHED clause is treated as a statement with the BY TARGET clause omitted.

□ RETURNING clause

The MERGE statements can now specify the RETURNING clause to return the data of updated tuples, and a special function merge_action has been added to determine the type (INSERT/UPDATE/DELETE) of the tuple returned by the RETURNING clause. The output of the RETURNING clause is available as a source relation like any other DML. [c649fa2]

Example 52 MERGE statement with the RETURNING clause

```
postgres=> CREATE TABLE src1(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> INSERT INTO src1 VALUES (100, 'data1'),(300, 'data3') ;
INSERT 0 2
postgres=> CREATE TABLE tgt1(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> INSERT INTO tgt1 VALUES (100, 'data1'),(200, 'data2') ;
INSERT 0 2
postgres=> MERGE INTO tgt1 t
  USING src1 s ON s.c1 = t.c1
  WHEN MATCHED THEN
     UPDATE SET c2 = 'updated'
  WHEN NOT MATCHED THEN
     INSERT (c1, c2) VALUES (s.c1, s.c2)
  RETURNING merge_action(), t.* ;
 merge_action | c1  |   c2
--------------+-----+---------
 UPDATE       | 100 | updated
 INSERT       | 300 | data3
(2 rows)

MERGE 2
```

□ MERGE statement for updatable views

The updatable views can now be specified as targets in the MERGE statement. [5f2e179]

Example 53 MERGE statement for updatable views

```
postgres=> CREATE TABLE src1(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> INSERT INTO src1 VALUES (generate_series(1, 10), 'data1') ;
INSERT 0 10
postgres=> CREATE TABLE tgt1(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE VIEW view1 AS SELECT * FROM tgt1 ;
CREATE VIEW
postgres=> MERGE INTO view1 AS v1 USING src1 AS s1
  ON v1.c1 = s1.c1
  WHEN NOT MATCHED THEN
    INSERT (c1, c2) VALUES (s1.c1, s1.c2) ;
MERGE 10
```

3.2.9. PL/pgSQL

The following enhancements have been implemented in PL/pgSQL.

□ Addition of data type attributes

The %TYPE attribute, which indicates the data type of a column, and the %ROWTYPE attribute, which indicates the data type of an entire tuple, are now available. This eliminates the need to directly describe the data type of a column in a PL/pgSQL program. [5e8674d]

Example 54 %TYPE attribute and %ROWTYPE attribute

```
postgres=> CREATE OR REPLACE FUNCTION func1() RETURNS TEXT AS $$
  DECLARE
    curs1 CURSOR FOR SELECT * FROM data1;
    data1var public.data1%ROWTYPE;
    col2var public.data1.c2%TYPE;
  BEGIN
    OPEN curs1;
    LOOP
      FETCH curs1 INTO data1var;
      IF NOT FOUND THEN
        EXIT;
      END IF;
      col2var := data1var.c2;
      RAISE NOTICE 'Column2: %', col2var;
    END LOOP;
    CLOSE curs1;
    RETURN null;
  END; $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

□ INTERNAL Sub-Transaction

The BeginInternalSubTransaction API can now be run in parallel mode. In the example below, PostgreSQL 16 executes a PL/pgSQL function with PARALLEL SAFE specified in parallel mode, and an error occurs. [0075d78]

Example 55 Execution error occurs in PostgreSQL 16

```
postgres=> CREATE FUNCTION zero_divide() RETURNS INT AS $$
             DECLARE v INT := 0;
           BEGIN
             RETURN 10 / v;
           END;
           $$ LANGUAGE plpgsql PARALLEL SAFE ;
CREATE FUNCTION
postgres=> CREATE FUNCTION error_trap_test() RETURNS TEXT AS $$
           BEGIN
             PERFORM zero_divide();
             RETURN 'no error detected!';
           EXCEPTION WHEN division_by_zero THEN
             RETURN 'division_by_zero detected';
           END;
           $$ LANGUAGE plpgsql PARALLEL SAFE ;
CREATE FUNCTION
postgres=> SET debug_parallel_query = on ;
SET
postgres=> SELECT error_trap_test() ;
ERROR:  cannot start subtransactions during a parallel operation
CONTEXT:  PL/pgSQL function error_trap_test() line 2 during statement block
entry
```

3.2.10. Data Types

The following enhancements have been implemented for PostgreSQL 17 data types.

□ INTERVAL

The +/-Infinity is now supported for the INTERVAL type. [519fc1b]

Example 56 INTERVAL type and Infinity

```
postgres=> SELECT timestamp 'infinity' - timestamp '-infinity' ;
 ?column?
----------
 infinity
(1 row)

postgres=> SELECT age(timestamp '-infinity', timestamp 'infinity') ;
    age
-----------
 -infinity
(1 row)
```

□ AT LOCAL clause

The AT LOCAL clause can now be specified to convert timestamps with time zones to local time zones. [97957fd]

Example 57 Specify the AT LOCAL clause

```
postgres=> SHOW timezone ;
  TimeZone
------------
 Asia/Tokyo
(1 row)

postgres=> SELECT CAST('1978-07-07 19:38 America/New_York'
        AS TIMESTAMP WITH TIME ZONE) AT LOCAL ;
      timezone
---------------------
 1978-07-08 08:38:00
(1 row)
```

□ RANGE

The expressions containing constant range values can now be directly compared for the range operators '<@' and '@>'. [075df6b]

Example 58 Behavior of PostgreSQL 17

```
postgres=> EXPLAIN (VERBOSE, COSTS OFF)
                SELECT CURRENT_DATE <@ DATERANGE 'empty' ;
   QUERY PLAN
------------------
 Result
   Output: false
(2 rows)
```

Example 59 Behavior of PostgreSQL 16

```
postgres=> EXPLAIN (VERBOSE, COSTS OFF)
                SELECT CURRENT_DATE <@ DATERANGE 'empty' ;
                   QUERY PLAN
-------------------------------------------------
 Result
   Output: (CURRENT_DATE <@ 'empty'::daterange)
 Query Identifier: 4806765806694262066
(3 rows)
```

3.2.11. JSON

The JSON-related syntax has been enhanced.

□ JSON Constructor

The following JSON constructor functions have been added. [03734a7]

Syntax 9 JSON constructor

```
JSON( expression [ FORMAT JSON [ ENCODING UTF8 ] ] [ { WITH | WITHOUT } UNIQUE
[ KEYS ]])

text JSON_SCALAR( expression )

text | bytea JSON_SERIALIZE( expression [ FORMAT JSON [ ENCODING UTF8 ] ]
[ RETURNING data_type [ FORMAT JSON [ ENCODING UTF8 ] ] ])
```

Example 60 Added JSON constructor

```
postgres=> SELECT JSON('{ "val": 1}'::bytea FORMAT JSON ENCODING UTF8) ;
    json
-------------
 { "val": 1}
(1 row)

postgres=> SELECT JSON_SCALAR(123.45::numeric) ;
 json_scalar
-------------
 123.45
(1 row)

postgres=> SELECT JSON_SERIALIZE(JSON('{ "val" : 1 }')) ;
 json_serialize
----------------
 { "val" : 1 }
(1 row)
```

□ JSONPATH method

The following JSONPATH methods have been added. [66ea94e, 4697454]

Table 28 Added JSONPATH methods

Method	Description	Note
.bigint()	Returns a large integer value	
.boolean()	Return True or False	
.date()	Returns a date	
.decimal([p [,s]])	Returns a number with a specified precision	
.integer()	Returns an integer	
.number()	Returns a numeric value	
.string()	Returns a character string	
.time()	Returns a time	
.time_tz()	Returns the time with timezone	
.timestamp()	Returns a timestamp	
.timestamp_tz()	Returns a timestamp with timezone	

Example 61 JSONPATH methods

```
postgres=> SELECT
    jsonb_path_query_array('[1, "yes", false]', '$[*].boolean()'),
    jsonb_path_query('"2023-08-15"', '$.datetime().string()'),
    jsonb_path_query('{"len": "9876543219"}', '$.len.bigint()'),
    jsonb_path_query('1234.5678', '$.decimal(6, 2)'),
    jsonb_path_query('{"len": "12345"}', '$.len.integer()'),
    jsonb_path_query('{"len": "123.45"}', '$.len.number()'),
    jsonb_path_query('"2023-08-15"', '$.date()'),
    jsonb_path_query('"12:34:56"', '$.time()'),
    jsonb_path_query('"12:34:56 +05:30"', '$.time_tz()'),
    jsonb_path_query('"2023-08-15 12:34:56"', '$.timestamp()') ;
-[ RECORD 1 ]----------+---------------------
jsonb_path_query_array | [true, true, false]
jsonb_path_query       | "2023-08-15"
jsonb_path_query       | 9876543219
jsonb_path_query       | 1234.57
jsonb_path_query       | 12345
jsonb_path_query       | 123.45
jsonb_path_query       | "2023-08-15"
jsonb_path_query       | "12:34:56"
jsonb_path_query       | "12:34:56+05:30"
jsonb_path_query       | "2023-08-15T12:34:56"
```

The timestamp_tz method cannot be executed on the jsonb_path_query function because the time zone is not IMMUTABLE.

Example 62 Timestamp with Time zone

```
postgres=> SELECT
        jsonb_path_query('"2024-05-24 12:34:56"', '$.timestamp_tz()') ;
ERROR:  cannot convert value from timestamp to timestamptz without time zone
usage
HINT:  Use *_tz() function for time zone support.
postgres=> SELECT
        jsonb_path_query_tz('"2024-05-24 12:34:56"', '$.timestamp_tz()') ;
     jsonb_path_query_tz
---------------------------
 "2024-05-24T03:34:56+00:00"
```

□ JSON Query

The functions have been added to query within JSON data. The JSON_EXISTS function returns true if the item was generated using the PASSING clause. The JSON_QUERY and JSON_VALUE functions return the result of applying the PASSING clause from the specified JSON data. [6185c97]

Syntax 10 JSON Query

```
JSON_EXISTS (context_item, path_expression [ PASSING { value AS varname }
[, ...]] [ { TRUE | FALSE | UNKNOWN | ERROR } ON ERROR ])

JSON_QUERY (context_item, path_expression [ PASSING { value AS varname } [, ...]]
[ RETURNING data_type [ FORMAT JSON [ ENCODING UTF8 ] ] ] [ { WITHOUT | WITH
{ CONDITIONAL | [UNCONDITIONAL] } } [ ARRAY ] WRAPPER ] [ { KEEP | OMIT } QUOTES
[ ON SCALAR STRING ] ] [ { ERROR | NULL | EMPTY { [ ARRAY ] | OBJECT } | DEFAULT
expression } ON EMPTY ] [ { ERROR | NULL | EMPTY { [ ARRAY ] | OBJECT } | DEFAULT
expression } ON ERROR ])

JSON_VALUE (context_item, path_expression [ PASSING { value AS varname } [, ...]]
[ RETURNING data_type ] [ { ERROR | NULL | DEFAULT expression } ON EMPTY ]
[ { ERROR | NULL | DEFAULT expression } ON ERROR ])
```

Example 63 JSON query

```
postgres=> SELECT JSON_EXISTS(jsonb '{"a": 1, "b": 2}', '$.* ? (@ > $x)'
                PASSING 1 AS x) ;
 json_exists
-------------
 t
(1 row)

postgres=> SELECT JSON_QUERY(jsonb '"aaa"', '$' RETURNING text KEEP QUOTES) ;
 json_query
------------
 "aaa"
(1 row)

postgres=> SELECT JSON_VALUE(jsonb '[1,2]', '$[*]' DEFAULT '0' ON ERROR) ;
 json_value
------------
 0
(1 row)
```

□ Relational conversion

The JSON_TABLE function has been added to convert JSON data to the relational view. [de36004, bb766cd]

Syntax 11 JSON_TABLE function

```
JSON_TABLE (context_item, path_expression [ AS json_path_name ] [ PASSING
{ value AS varname } [, ...] ] COLUMNS ( json_table_column [, ...] )
    [ { ERROR | EMPTY } ON ERROR ]
)
```

Example 64 Execute JSON_TABLE function

```
postgres=> SELECT * FROM JSON_TABLE(jsonb '123', '$'
        COLUMNS (item int PATH '$', foo int)) bar ;
 item | foo
------+-----
  123 |
(1 row)

postgres=>  SELECT * FROM JSON_TABLE(jsonb '"a"', '$'
  COLUMNS (a int PATH 'strict $.a' DEFAULT 1 ON EMPTY DEFAULT 2 ON ERROR)) jt ;
 a
---
 2
(1 row)
```

□ Jsonb_populate_record_valid function

The jsonb_populate_record_valid function has been added. This function returns true if the jsonb_populate_record function does not generate an error for the specified JSON object, or false if it does. [1edb3b4]

Syntax 12 Jsonb_populate_record_valid function

```
boolean jsonb_populate_record_valid(anyelement, jsonb)
```

Example 65 Execute jsonb_populate_record_valid function

```
postgres=> CREATE TYPE jsb_char2 AS (a char(2)) ;
CREATE TYPE
postgres=> SELECT jsonb_populate_record_valid(NULL::jsb_char2,
                '{"a": "aaa"}') ;
jsonb_populate_record_valid
----------------------------
 f
(1 row)
```

3.2.12. Functions

The following functions have been added/enhanced.

□ Pg_promote

The error message is now output if sending a signal to the postmaster process fails. [f593c55]

Syntax 13 Error messages that may be output

```
ERROR: failed to send signal to postmaster: {PID}
ERROR: terminating connection due to unexpected postmaster exit
```

□ Pg_basetype

The function pg_basetype has been added. This function returns the base type for a given domain. [b154d8a]

Syntax 14 Pg_basetype function

```
regtype pg_basetype(regtype)
```

Example 66 Execute pg_basetype function

```
postgres=> CREATE DOMAIN mytext AS text ;
CREATE DOMAIN
postgres=> CREATE DOMAIN mytext_child AS mytext ;
CREATE DOMAIN
postgres=> SELECT pg_basetype('mytext'::regtype) ;
 pg_basetype
-------------
 text
(1 row)

postgres=> SELECT pg_basetype('mytext_child'::regtype) ;
 pg_basetype
-------------
 text
(1 row)
```

□ Random

The functions have been added to generate random numbers that have "minimum value ≤ random ≤ maximum value" given a minimum and maximum value. [e634132]

Syntax 15 Random function

```
integer random(min integer, max integer)
bigint random(min bigint, max bigint)
numeric random(min numeric, max numeric)
```

Example 67 Execute random function

```
postgres=> SELECT random(100, 200), random(10.25, 20.34) ;
 random | random
--------+--------
    153 |  19.21
(1 row)
```

□ To_bin / to_oct

The function to convert a number to a binary string or octal string has been added. [260a1f1]

Syntax 16 To_bin / to_oct function

```
text to_bin(integer | bigint)
text to_oct(integer | bigint)
```

Example 68 Execute to_bin / to_oct function

```
postgres=> SELECT TO_BIN(100) bin, TO_OCT(100) oct ;
   bin   | oct
---------+-----
 1100100 | 144
(1 row)
```

□ To_regtypemod

The to_regtypemod function has been added. This function retrieves the typemod information of a datatype from a datatype string. This information can be used with the format_type function, etc. [1218ca9]

Syntax 17 To_regtypemod function

```
integer to_regtypemod(text)
```

Example 69 Execute to_regtypemod function

```
postgres=> SELECT to_regtypemod('varchar(10)'),
        format_type(to_regtype('varchar(10)'), to_regtypemod('varchar(10)')) ;
 to_regtypemod |      format_type
---------------+-----------------------
            14 | character varying(10)
(1 row)
```

□ To_timestamp

The to_timestamp function can now specify TZ and OF in the format string. In previous versions, it was only available in the to_char function. [8ba6fdf]

Example 70 Execute to_timestamp function with TZ/OF

```
postgres=> SELECT
        TO_TIMESTAMP('2024-05-26 09:12 -05', 'YYYY-MM-DD HH12:MI OF') ;
      to_timestamp
------------------------
 2024-05-26 23:12:00+09
(1 row)

postgres=> SELECT
        TO_TIMESTAMP('2024-05-26 09:12 EST', 'YYYY-MM-DD HH12:MI TZ') ;
      to_timestamp
------------------------
 2024-05-26 23:12:00+09
(1 row)
```

□ UNICODE related

The basic functions related to UNICODE versions have been added. The unicode_version function outputs the UNICODE version used by the database cluster, and the icu_unicode_version function outputs the UNICODE version used by the ICU library. The unicode_assigned function verifies whether the specified string is assigned to UNICODE. [a02b37f]

Syntax 18 UNICODE related functions

```
text unicode_version()
text icu_unicode_version()
boolean unicode_assigned(text)
```

Example 71 Execute UNICODE related functions

```
postgres=> SELECT unicode_version() ;
-[ RECORD 1 ]---+-----
unicode_version | 15.1

postgres=> SELECT icu_unicode_version() ;
-[ RECORD 1 ]-------+-----
icu_unicode_version | 10.0

postgres=> SELECT unicode_assigned('ABC') ;
-[ RECORD 1 ]----+--
unicode_assigned | t
```

The unicode_assigned function can be executed only when the database encoding is UTF-8. The following error will occur if it is executed in a database with encoding set to EUC_JP.

Example 72 Execute on EUC_JP encoded database

```
postgres=> SELECT unicode_assigned('ABC') ;
ERROR:  Unicode categorization can only be performed if server encoding is UTF8
```

□ XMLText

The XMLText function of the XML standard (SQL/XML X038) has been added. This function converts text information into XML text nodes. The --with-libxml option is required at PostgreSQL build time to execute this function. [526fe0d]

Syntax 19 XMLText function

```
xml xmltext(text)
```

Example 73 Execute xmltext function

```
postgres=> SELECT xmltext('foo & <"bar">') ;
              xmltext
-----------------------------------
 foo &amp; &lt;&quot;bar&quot;&gt;
(1 row)
```

□ Uuid_extract_version / uuid_extract_timestamp

The functions to extract version numbers and timestamps from UUIDs have been added. [794f10f]

Syntax 20 UUID related functions

```
smallint uuid_extract_version(uuid)
timestamp with time zone uuid_extract_timestamp(uuid)
```

Example 74 Execute UUID related functions

```
postgres=> SELECT uuid_extract_version(gen_random_uuid()) ;
 uuid_extract_version
----------------------
                    4
(1 row)

postgres=> SELECT
        uuid_extract_timestamp('C232AB00-9414-11EC-B3C8-9F6BDECED846') ;
 uuid_extract_timestamp
------------------------
 2022-02-23 04:22:22+09
(1 row)
```

□ Pg_column_toast_chunk_id

The newly added pg_column_toast_chunk_id function returns the chunk ID of TOAST data. [d1162cf]

Syntax 21 Pg_column_toast_chunk_id functions

```
oid pg_column_toast_chunk_id(any)
```

Example 75 Execute pg_column_toast_chunk_id function

```
postgres=> CREATE TABLE data1 (col1 TEXT, col2 TEXT STORAGE EXTERNAL) ;
CREATE TABLE
postgres=> INSERT INTO data1 VALUES ('A', repeat('B', 8192)) ;
INSERT 0 1
postgres=> SELECT pg_column_toast_chunk_id(col2) FROM data1 ;
-[ RECORD 1 ]-------------+------
pg_column_toast_chunk_id | 16412
```

□ Pg_stat_reset_shared

The behavior of the pg_stat_reset_shared function has changed. If the parameter to the pg_stat_reset_shared function is omitted or specified as NULL, all statistics counters are reset. Also, the pg_stat_reset_shared function parameter can now contain the string 'slru' to reset the pg_stat_slru view. [23c8c0c, 2e8a0ed]

Syntax 22 Pg_stat_reset_shared functions

```
void pg_stat_reset_shared(target text DEFAULT NULL::text)
```

□ Pg_stat_reset_slru

If the parameter of the pg_stat_reset_slru function is omitted, all SLRU statistics counters are reset. [e5cca62]

Syntax 23 Pg_stat_reset_slru functions

```
void pg_stat_reset_slru(target text DEFAULT NULL)
```

The element name specified as a parameter to this function has been changed. Entering a string other than the changed name will reset the "other" item. [bcdfa5f]

Table 29 Changed Element Name

Before changed Name	After changed Name	Note
CommitTs	commit_timestamp	
MultiXactMember	multixact_member	
MultiXactOffset	multixact_offset	
Notify	notify	
Serial	serializable	
Subtrans	subtransaction	
Xact	transaction	

3.2.13. Optimizer

Faster execution plans are now chosen in the following cases.

□ GROUP BY optimization

If the GROUP BY clause with multiple columns is not related to the sort order, the sorting process can now be reduced by swapping the columns. This behavior is enabled by setting the parameter enable_group_by_reordering to 'on' (default 'on'). [0452b46]

Example 76 Execution Plan on PostgreSQL 17

```
postgres=> EXPLAIN (COSTS OFF) SELECT COUNT(*) FROM data1 GROUP BY col2, col1 ;
                    QUERY PLAN
-----------------------------------------------
 GroupAggregate
   Group Key: col1, col2
   ->  Index Only Scan using idx1_data1 on data1
(3 rows)
```

Example 77 Execution Plan on PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT COUNT(*) FROM data1 GROUP BY col2, col1 ;
          QUERY PLAN
------------------------------
 GroupAggregate
   Group Key: col2, col1
   ->  Sort
         Sort Key: col2, col1
         ->  Seq Scan on data1
(5 rows)
```

□ UNION optimization

The Merge Append plan can now be used for lookups that contain a UNION clause in the subquery. Previously, Sort plan was required. [66c0185]

Example 78 Execution Plan on PostgreSQL 17

```
postgres=> EXPLAIN (COSTS OFF) SELECT COUNT(*) FROM
  (SELECT c1 FROM merge1 UNION SELECT c1 FROM merge2) ;
                          QUERY PLAN
-------------------------------------------------------------------
 Aggregate
   ->  Unique
         ->  Merge Append
               Sort Key: merge1.c1
               ->  Index Only Scan using merge1_pkey on merge1
               ->  Index Only Scan using merge2_pkey on merge2
(6 rows)
```

Example 79 Execution Plan on PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT COUNT(*) FROM
                (SELECT c1 FROM merge1 UNION SELECT c1 FROM merge2) ;
                 QUERY PLAN
-------------------------------------------
 Aggregate
   ->  Unique
         ->  Sort
               Sort Key: merge1.c1
               ->  Append
                     ->  Seq Scan on merge1
                     ->  Seq Scan on merge2
(7 rows)
```

□ Better IS [NOT] NULL Handling

The optimizations have been implemented to reduce unnecessary evaluation of IS NULL and IS NOT NULL clauses. [b262ad4]

Example 80 Execution Plan on PostgreSQL 17

```
postgres=> \d data1
                    Table "public.data1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | integer               |           | not null |
 c2     | character varying(10) |           |          |
Indexes:
    "data1_pkey" PRIMARY KEY, btree (c1)

postgres=> EXPLAIN SELECT COUNT(*) FROM data1 WHERE c1 IS NOT NULL ;
                            QUERY PLAN
-------------------------------------------------------------------
 Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
(2 rows)
```

Example 81 Execution Plan on PostgreSQL 16

```
postgres=> \d data1
                    Table "public.data1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | integer               |           | not null |
 c2     | character varying(10) |           |          |
Indexes:
    "data1_pkey" PRIMARY KEY, btree (c1)

postgres=> EXPLAIN SELECT COUNT(*) FROM data1 WHERE c1 IS NOT NULL ;
                            QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
   ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
         Filter: (c1 IS NOT NULL)
(3 rows)
```

□ Better Parallel DISTINCT Processing

The Gather Merge plan is now available for parallel execution of SELECT statements with DISTINCT clauses. [7e0ade0]

Example 82 Execution Plan on PostgreSQL 17

```
postgres=> EXPLAIN (COSTS OFF) SELECT DISTINCT c2 FROM data1 ;
                    QUERY PLAN
--------------------------------------------------
 Unique
   ->  Gather Merge
         Workers Planned: 2
         ->  Sort
               Sort Key: c2
               ->  HashAggregate
                     Group Key: c2
                     ->  Parallel Seq Scan on data1
(8 rows)
```

Example 83 Execution Plan on PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT DISTINCT c2 FROM data1 ;
                     QUERY PLAN
-----------------------------------------------------
 Unique
   ->  Sort
         Sort Key: c2
         ->  Gather
               Workers Planned: 2
               ->  HashAggregate
                     Group Key: c2
                     ->  Parallel Seq Scan on data1
(8 rows)
```

□ GiST Index optimization

The Incremental Sort plan is now available for GiST and SP-GiST indexes. [625d5b3]

Example 84 Execution Plan on PostgreSQL 17

```
postgres=> CREATE INDEX idx1_gist1 ON gist1 USING GIST(c1) ;
CREATE INDEX
postgres=> EXPLAIN (COSTS OFF) SELECT c1, c2, c1<->POINT(5,5) dist FROM gist1
                ORDER BY dist, c2 LIMIT 1 ;
                    QUERY PLAN
-------------------------------------------------
 Limit
   ->  Incremental Sort
         Sort Key: ((c1 <-> '(5,5)'::point)), c2
         Presorted Key: ((c1 <-> '(5,5)'::point))
         ->  Index Scan using idx1_data1 on data1
               Order By: (c1 <-> '(5,5)'::point)
(6 rows)
```

Example 85 Execution Plan on PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT c1, c2, c1<->POINT(5,5) dist FROM data1
                ORDER BY dist, c2 LIMIT 1 ;
                   QUERY PLAN
-------------------------------------------------
 Limit
   ->  Sort
         Sort Key: ((c1 <-> '(5,5)'::point)), c2
         ->  Seq Scan on data1
(4 rows)
```

3.3. Configuration parameters

In PostgreSQL 17, the following parameters have been changed.

3.3.1. Added parameters

The following parameters have been added. [7750fef, a14354c ,2cdf131, 174c480, 0452b46, 51efe38, 53c2a97, bf279dd, d3ae2a2, 210622c, 705843d]

Table 30 Added parameters

Parameter name	Description (context)	Default value
allow_alter_system	Allow execution of ALTER SYSTEM statement (sighup)	on
commit_timestamp_buffers	Commit timestamp cache size (postmaster)	0 (auto)
enable_group_by_reordering	GROUP BY replacement optimization (user)	on
event_triggers	Enable event triggers (superuser)	on
huge_pages_status	Indicate whether Huge Pages are being used (internal)	-
io_combine_limit	Maximum number of Raad/Write blocks (user)	128kB
max_notify_queue_pages	Maximum number of pages allocated to NOTIFY/LISTEN queue (postmaster)	1048576
multixact_member_buffers	Member cache size for multi-transaction (postmaster)	32
multixact_offset_buffers	Multi-transaction offset cache size (postmaster)	16
notify_buffers	LISTEN/NOTIFY message cache size (postmaster)	16
serializable_buffers	Serializable transaction cache size (postmaster)	32
standby_slot_names	Replication slot name for standby database (sighup)	''
subtransaction_buffers	Size of sub-transaction cache (postmaster)	0 (auto)

Parameter name	Description (context)	Default value
summarize_wal	Output WAL summary (sighup)	off
sync_replication_slots	Whether replication slots are synchronized (sighup)	off
trace_connection_negotiation	Obtain SSL negotiation trace (postmaster)	off
transaction_buffers	Transaction status cache size (postmaster)	0 (auto)
transaction_timeout	Transaction execution timeout (user)	0
wal_summary_keep_time	WAL summary retention duration (sighup)	10d

□ Allow_alter_system

This parameter determines whether the ALTER SYSTEM statement is allowed to be executed. The default value is 'on', which allows the ALTER SYSTEM statement to be executed. This parameter itself cannot be changed by the ALTER SYSTEM statement.

Example 86 Permission to execute ALTER SYSTEM statement

```
postgres=# SHOW allow_alter_system ;
-[ RECORD 1 ]------+----
allow_alter_system | off

postgres=# ALTER SYSTEM SET work_mem='16MB' ;
ERROR:  ALTER SYSTEM is not allowed in this environment
postgres=# SELECT pg_reload_conf() ;
-[ RECORD 1 ]--+--
pg_reload_conf | t

postgres=# SHOW allow_alter_system ;
-[ RECORD 1 ]------+---
allow_alter_system | on

postgres=# ALTER SYSTEM SET allow_alter_system = off ;
ERROR:  parameter "allow_alter_system" cannot be changed
```

□ Huge_pages_status

If the parameter huge_pages has the default value (try), no log output will be generated even if the acquisition of Huge Pages fails. The parameter huge_pages_status can be checked to see if the cluster is using Huge Pages.

Table 31 Possible values

Value	Description	Note
on	Using Huge Pages	
off	Not using Huge Pages	
unknown	Unknown	Only shown in special cases

□ Transaction_timeout

Specify the maximum transaction execution time in milliseconds. If a timeout occurs, the session is forcibly disconnected. The default value is 0, which means no timeout will occur.

Example 87 Transaction timeout

```
postgres=> SET transaction_timeout = '10s' ;
SET
postgres=> BEGIN ;
BEGIN
postgres=*> SELECT pg_sleep(10) ;
FATAL:  terminating connection due to transaction timeout
server closed the connection unexpectedly
        This probably means the server terminated abnormally
        before or while processing the request.
The connection to the server was lost. Attempting reset: Succeeded.
```

3.3.2. Modified Parameters

The following parameters have been changed in terms of setting range and choices. [d0c2860, e48b19c, 7c3fb50]

Table 32 Modified Parameters

Parameter name	Changes
wal_sync_method	The configuration value fsync_writethrough is no longer available in Microsoft Windows environments.
log_connections	Information on trust connections is now output.
log_replication_commands	Replication slot addition information is now output.

3.3.3. Parameters with default values changed

The following parameters have changed default values. [98f320e]

Table 33 Parameters with default values changed

Parameter name	PostgreSQL 15	PostgreSQL 16	Note
server_version	16.3	17beta1	
server_version_num	160003	170000	
vacuum_buffer_usage_limit	256kB	2MB	

3.3.4. Removed parameters

The following parameters have been removed. [884eee5, f691f5b, c7a3e6b]

Table 34 Removed parameters

Parameter name	Reason
db_user_namespace	It was removed because it was determined that there were few users.
old_snapshot_threshold	Removed due to accuracy and performance issues. It is a desirable feature, and improved implementations may be available in the future.
trace_recovery_messages	Removed because it can be replaced by pg_waldump command, etc.

3.4. Utilities

Describes the major enhancements of utility commands.

3.4.1. Clusterdb

The --all and other options can now be used at the same time. [1b49d56]

Example 88 Specifying the --all and --table options

```
$ clusterdb --all --table=data
clusterdb: clustering database "demodb"
clusterdb: clustering database "postgres"
clusterdb: clustering database "template1"
```

3.4.2. Configure

The following features have been implemented in the configure command.

□ Output version

The LLVM and OpenSSL versions are now output when executing the configure command. [5e4dacb, 55a428a]

Example 89 Execution log of configure command

```
$ ./configure --with-llvm --with-ssl=openssl
...
checking for llvm-config... /usr/bin/llvm-config
configure: using llvm 12.0.1
...
checking for openssl... /usr/bin/openssl
configure: using openssl: OpenSSL 1.1.1k  FIPS 25 Mar 2021
...
```

□ Injection Points

The Injection points are features that allow developers to execute custom code, assist in testing for complex conflicts, etc. To use this feature, it is necessary to specify --enable-injection-points when executing the configure command. [d86d20f]

3.4.3. Initdb

The following enhancements have been implemented to the initdb command. [2d819a0, f69319f]

□ --locale-provider option

The "builtin" can now be specified as the locale provider.

□ --builtin-locale option

Added --builtin-locale option to specify the locale for built-in locale providers.

Example 90 Specifying the builtin locale provider

```
$ initdb --locale-provider=builtin --builtin-locale=C.UTF8 data
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with this locale configuration:
  default collation provider:  builtin
  default collation: C.UTF-8
  LC_COLLATE:  C.UTF-8
...
```

3.4.4. Pg_archivecleanup

The following new features have been added to the pg_archivecleanup command. [dd7c60f, 3f8c98d]

□ Long name options

The long name options have been added to the single-character options.

Table 35 Add/change options

Short option	Long option	Description
-d	--debug	Outputs debug logs to standard error.
-n	--dry-run	Outputs the name of the file to be deleted to standard output.
-x	--strip-extension=*EXT*	Specify extensions to exclude from file names.

□ Remove history files

The --clean-backup-history option (shortened option -b) has been added to remove the history files.

3.4.5. Pg_combinebackup

The new command pg_combinebackup has been added. This command merges base and incremental backups. [dc21234]

Table 36 Major options

Option name	Short option	Description
--debug	-d	Output debug information
--dry-run	-n	Run test only
--no-sync	-N	Don't wait for storage sync
--output	-o	Output directory
--tablespace-mapping=*OLD=NEW*	-T	Tablespace mapping
--manifest-checksums=*METHOD*	-	Manifest checksum specification
--no-manifest	-	Don't use manifest
--sync-method=*METHOD*	-	Specifying the storage synchronization method
--clone	-	Using the clone system call
--copy-file-range	-	Using the copy_file_range system call
--help	-?	Output usage information
--version	-V	Output version information

Example 91 Execute pg_combinebackup command

```
$ pg_combinebackup back.1 back.inc1 --output=data.2
$ ls data.2
PG_VERSION          current_logfiles  pg_dynshmem     pg_multixact  pg_snapshots
pg_tblspc     postgresql.auto.conf
backup_label        global            pg_hba.conf     pg_notify     pg_stat
pg_twophase   postgresql.conf
backup_manifest     log               pg_ident.conf   pg_replslot   pg_stat_tmp
pg_wal
base                pg_commit_ts      pg_logical      pg_serial     pg_subtrans
pg_xact
```

3.4.6. Pg_createsubscriber

The new command pg_createsubscriber has been added. This command converts a streaming replication standby server to a logical replication standby server. The major advantage of using this command is to reduce the initial data copy overhead that occurs when setting up a logical replication environment. [d44032d]

Table 37 Major options

Option name	Short option	Description
--database=*DBNAME*	-d	Database name for which to create SUBSCRIPTION
--pgdata=*DATADIR*	-D	Database cluster for streaming replication to be converted
--dry-run	-n	Conversion test
--subscriber-port=*PORT*	-p	Port number of the subscriber
--publisher-server=*CONN*	-P	Connection string to publisher
--socket-directory=*DIR*	-s	Socket directory to use
--recovery-timeout=*SECS*	-t	Seconds to wait for recovery to end
--subscriber-username=*USER*	-U	SUBSCRIPTION owner
--publication=*NAME*	-	Name of the PUBLICATION
--replication-slot=*NAME*	-	Replication slot name
--subscription=*NAME*	-	SUBSCRIPTION name
--verbose	-v	
--config-file=*FILE*	-	Configuration file's path
--version	-V	Output version information
--help	-?	Output usage information

Example 92 Execute pg_createsubscriber command

```
$ pg_createsubscriber -D data.stby --publisher-server='host=dbsvr1 port=5432
dbname=postgres'
LOG:  redirecting log output to logging collector process
HINT:  Future log output will appear in directory "log".
LOG:  redirecting log output to logging collector process
HINT:  Future log output will appear in directory "log".
```

The following requirements are necessary to execute the pg_createsubscriber command.

- The primary server parameter wal_level must be set to 'logical'.
- The standby instance must be shut down.

When the pg_createsubscriber command is executed, PUBLICATION and logical replication slots are created on the primary server with all tables in the specified database (FOR ALL TABLES). The standby server is no longer a standby for streaming replication and the SUBSCRIPTION is created. The default PUBLICATION and replication slot name created on the primary server and the SUBSCRIPTION name created on the standby server is "pg_createsubscriber_{DB OID}_{random integer}".

3.4.7. Pg_basebackup

The following new features have been implemented in the pg_basebackup command.

□ --dbname option

If a database name is specified in the --dbname option (-d) at the same time as the --write-recovery-conf option (-R), the database name will be output to the primary_conninfo parameter setting specified in the postgresql.auto.conf file. [a145f42]

Example 93 Add dbname to primary_conninfo parameter

```
$ pg_basebackup -D back --dbname="dbname=demodb" -R
$ cat back/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo    =    'user=postgres    passfile=''/home/postgres/.pgpass''
channel_binding=disable     port=5432     sslmode=disable     sslcompression=0
sslcertmode=disable          sslsni=1          ssl_min_protocol_version=TLSv1.2
gssencmode=disable krbsrvname=postgres gssdelegation=0 target_session_attrs=any
load_balance_hosts=disable dbname=demodb'
```

□ --help option

The default value of the --checkpoint option is now output in the help message. [cd02b35]

Example 94 --help option outputs

```
$ pg_basebackup --help
pg_basebackup takes a base backup of a running PostgreSQL server.
...
General options:
  -c, --checkpoint=fast|spread
                         set fast or spread (default) checkpointing
...
```

□ --incremental option

The --incremental option (shortened option -i) to perform incremental backups has been added. This option specifies the path to the manifest file on which the backup is based.

3.4.8. Pg_dump

The following options have been added to the pg_dump command.

□ --filter option

The --filter option has been added to the pg_dump / pg_dumpall / pg_restore commands. This option specifies the file that describes the list of objects to be included (or excluded) from the dump file. The format of the file is as follows. [a5cf808]

Syntax 24 Filter file format

```
include | exclude object_type_name object_name_pattern
```

Table 38 Object type

Object type name	Object	Equivalent command options
extension	Extension	--extension
foreign_data	Foreign Table	--include-foreign-data
table	Table	--table
table_and_children	Table and Child Table	--table-and-children
table_data	Table data only	--exclude-table-data
table_data_and_children	Table and Child Table data	--exclude-table-data-and-children
schema	Schema	--schema

Example 95 --filter option

```
$ cat filter.txt
include table data*
$ pg_dump -d postgres --filter=filter.txt -f postgres.dmp
```

□ --exclude-extension option

The --exclude-extension option has been added to exclude extensions with specified pattern names. [522ed12]

3.4.9. Pg_restore

The pg_restore command now has a --transaction-size option to perform a commit once the specified number of objects have been processed. The default behavior is to commit on a per-SQL basis. This option provides an intermediate behavior between per-SQL commit and single-transaction processing with the --single-transaction option. [959b38d]

Example 96 Specifying the --transaction-size option

```
$ pg_restore --help | grep transaction
  -1, --single-transaction     restore as a single transaction
  --transaction-size=N         commit after every N objects
```

3.4.10. Pg_resetwal

The order in which options are displayed in the --help option has been changed. [b5da1b3]

Example 97 --help option output

```
$ pg_resetwal --help
pg_resetwal resets the PostgreSQL write-ahead log.

Usage:
  pg_resetwal [OPTION]... DATADIR

Options:
 [-D, --pgdata=]DATADIR  data directory
  -f, --force             force update to be done even after unclean shutdown
or
                         if pg_control values had to be guessed
  -n, --dry-run          no update, just show what would be done
  -V, --version          output version information, then exit
  -?, --help             show this help, then exit

Options to override control file values:
  -c, --commit-timestamp-ids=XID,XID
                                  set oldest and newest transactions bearing
                                  commit timestamp (zero means no change)
  -e, --epoch=XIDEPOCH            set next transaction ID epoch
  -l, --next-wal-file=WALFILE     set minimum starting location for new WAL
  -m, --multixact-ids=MXID,MXID   set next and oldest multitransaction ID
  -o, --next-oid=OID              set next OID
  -O, --multixact-offset=OFFSET   set next multitransaction offset
  -u, --oldest-transaction-id=XID  set oldest transaction ID
  -x, --next-transaction-id=XID   set next transaction ID
      --wal-segsize=SIZE          size of WAL segments, in megabytes

Report bugs to <pgsql-bugs@lists.postgresql.org>.
PostgreSQL home page: <https://www.postgresql.org/>
```

3.4.11. Pg_upgrade

The following features have been implemented to the pg_upgrade command.

□ Preserve subscriber state

The subscriber (SUBSCRIPTION) state can now be preserved. In previous versions, only meta-data was preserved. [9a17be1]

□ Migrate logical replication slot

The logical replication slots are now recreated on the destination cluster during upgrade. [29d0a77]

□ --copy-file-range option

The option --copy-file-range has been added to copy files using the copy_file_range system call in Linux and FreeBSD environments. [d93627b]

3.4.12. Pg_walsummary

The command pg_walsummary has been added. This command parses the contents of the WAL summary file. [ee1bfd1]

Table 39 Available options

Option name	Short option	Description	Note
--indivisual	-i	Outputs detailed block information	
--quiet	-q	Perform parsing of files only	
--help	-?	Output usage	

Example 98 Execute the pg_walsummary command

```
$ pg_walsummary
        pg_wal/summaries/000000010000000005000030000000005A829E8.summary
TS 1663, DB 1, REL 1259, FORK main: block 0
TS 1663, DB 1, REL 1259, FORK main: block 3
TS 1663, DB 1, REL 1259, FORK main: blocks 7..8
…
```

3.4.13. Pgindent

The option name of the pgindent command has been changed. [387aecc]

Table 40 Renamed options

Previous Option Name	New Option Name	Note
--show-diff	--diff	
--silent-diff	--check	

3.4.14. Psql

The following enhancements have been implemented to the psql command.

□ View of Aggregate Functions

The argument names have been added to the view of aggregate functions with multiple arguments. [b575a26]

Example 99 View function information

```
postgres=> \df json_object_agg
List of functions
-[ RECORD 1 ]-------+------------------------
Schema              | pg_catalog
Name                | json_object_agg
Result data type    | json
Argument data types | key "any", value "any"
Type                | agg
```

□ \sf, \ef, \sv, \ev meta-command

The semicolons (;) at the end of lines are now ignored. In previous versions, an error occurred. [390298f]

Example 100 Ignore the semicolon at the end of the line

```
postgres=> \sf func1() ;
CREATE OR REPLACE FUNCTION public.func1()
 RETURNS integer
 LANGUAGE sql
 RETURN 10
```

□ \watch meta-command

The min_rows (abbreviated m), which indicates the minimum number of output tuples, has been added to the watch meta-command. The iterative operation terminates when the specified number of tuples are no longer output. In the example below, the \watch command terminates because all tuples have been deleted from the data1 table. [f347ec7]

Example 101 Specify the minimum number of tuples

```
postgres=> \watch 10 min_rows=1
Fri 24 May 2024 09:28:17 PM JST (every 10s)

 c1  |  c2
-----+-------
 100 | data1
 200 | data2
(2 rows)

Fri 24 May 2024 09:28:27 PM JST (every 10s)

 c1 | c2
----+----
(0 rows)

postgres=>
```

3.4.15. Reindexdb

The following features have been added to the reindexdb command.

□ Combining options

The --all and other options can now be used at the same time. [24c928a]

Example 102 Specifying --all and --schema options

```
$ reindexdb --all --schema=public
reindexdb: reindexing database "demodb"
reindexdb: reindexing database "postgres"
reindexdb: reindexing database "template1"
```

□ Parallelization

The --jobs and --index options can now be used at the same time. Multiple indexes on different tables can be processed in parallel. [47f99a4]

Example 103 Specify --index and --jobs options

```
$ reindexdb --jobs=2 --index=idx_data1 --index=idx_data2
```

3.4.16. Vacuumdb

The --all and other options can now be used at the same time. [648928c]

Example 104 Specifying --all and --exclude-schema options

```
$ vacuumdb --all --exclude-schema=public
vacuumdb: vacuuming database "demodb"
vacuumdb: vacuuming database "postgres"
vacuumdb: vacuuming database "template1"
```

3.4.17. More than one command

The following commands now have an additional option "--sync-method" to determine the storage synchronization method for files. This option is disabled if the --no-sync option is specified. The utilities for which the --sync-method option has been added are as follows. [8c16ad3, cccc6cd]

- initdb
- pg_basebackup
- pg_checksums
- pg_dump
- pg_rewind
- pg_upgrade

Table 41 Possible values for option

Value for option	Description	Note
fsync	Recursively synchronize file-by-file using the fsync system call.	Default
syncfs	Synchronize the entire file system using the syncfs system call.	

Example 105 Specify option --sync-method

```
$ pg_basebackup -D back.1 --sync-method=fsync
$ pg_dump -d postgres -f dump.dat --sync-method=fsync
```

3.5. Contrib modules

Describes new features related to the Contrib modules.

3.5.1. Amcheck

The bt_index_check function now has a checkunique parameter to check the integrity of unique constraints. The following example shows the error when corruption is found in the index data1_pkey for the primary key. [5ae2087]

Example 106 Specify checkunique parameter

```
postgres=# SELECT bt_index_check('data1_pkey', true, true) ;
ERROR:  index uniqueness is violated for index "data1_pkey"
DETAIL:   Index tid=(1,84) and tid=(1,85) (point to heap tid=(0,84) and
tid=(0,85)) page lsn=0/1573AB8.
```

The --checkunique option has been added to the pg_amcheck command to do the same.

Example 107 Specify --checkunique option

```
$ pg_amcheck --checkunique -d postgres
btree index "postgres.public.data1_pkey":
    ERROR:  index uniqueness is violated for index "data1_pkey"
    DETAIL:   Index tid=(1,84) and tid=(1,85) (point to heap tid=(0,84) and
tid=(0,85)) page lsn=0/1573AB8.
```

3.5.2. Pg_buffercache

The following functions have been added to the pg_buffercache module. [13453ee]

Table 42 Added function

Function name	Description	Note
pg_buffercache_evict	Remove any block from the buffer pool.	SUPERUSER only

3.5.3. Pg_stat_statements

The following features have been added to the pg_stat_statements module.

□ Enhancements of the pg_stat_statements view

The pg_stat_statements view has been modified with I/O-related columns and JIT-related and timestamp-related columns added. [13d0072, 5147ab1, dc9f8a7, 5a3423a]

Table 43 Added columns

Column Name	Data Type	Description
local_blk_read_time	double precision	Local block read time
local_blk_write_time	double precision	Local block write time
shared_blk_read_time	double precision	Shared block read time
shared_blk_write_time	double precision	Shared block write time
jit_deform_count	bigint	JIT deform processing count
jit_deform_time	double precision	JIT deform processing time
stats_since	timestamp with time zone	Statement statistics collection time
minmax_stats_since	timestamp with time zone	Statement min/max statistics collection time

Table 44 Removed column

Column Name	Description
blk_read_time	Split into shared_blk_read_time and local_blk_read_time
blk_write_time	Split into shared_blk_write_time and local_blk_write_time

□ Constants for Execution Statements

The SAVEPOINT, ROLLBACK TO, RELEASE, COMMIT PREPARED, PREPARE TRANSACTION, ROLLBACK PREPARED, DEALLOCATE, and overloaded CALL statements are saved in the pg_stat_statements table as constants with parameter symbols. Previously, statements with different specified names were saved separately. [31de7e6, 638d42a, bb45156, 11c34b3]

Example 108 Constant SAVEPOINT statement

```
postgres=> BEGIN ;
BEGIN
postgres=*> SAVEPOINT sp1 ;
SAVEPOINT
postgres=*> SAVEPOINT sp2 ;
SAVEPOINT
postgres=*> SAVEPOINT sp3 ;
SAVEPOINT
postgres=*> COMMIT ;
COMMIT
postgres=> SELECT calls, rows, query FROM pg_stat_statements WHERE
               query LIKE 'SAVEPOINT%' ;
 calls | rows |    query
-------+------+-------------
     3 |    0 | SAVEPOINT $1
(1 row)
```

Example 109 Constant CALL statement

```
postgres=> CALL overload(1) ;
CALL
postgres=> CALL overload('A') ;
CALL
postgres=> CALL in_out(1, NULL, 1) ;
CALL
postgres=> CALL in_out(2, 1, 2) ;
CALL
postgres=> SELECT calls, rows, query FROM pg_stat_statements ;
 calls | rows |          query
-------+------+--------------------------
     1 |    1 | <insufficient privilege>
     1 |    0 | CALL overload($1)
     1 |    0 | CALL overload($1)
     2 |    0 | CALL in_out($1, $2, $3)
(4 rows)
```

Example 110 Constant PREPARE statement and COMMIT statement

```
postgres=> BEGIN ;
BEGIN
postgres=*> PREPARE TRANSACTION 'Transaction#1' ;
PREPARE TRANSACTION
postgres=> COMMIT PREPARED 'Transaction#1' ;
COMMIT PREPARED
postgres=> BEGIN ;
BEGIN
postgres=*> PREPARE TRANSACTION 'Transaction#2' ;
PREPARE TRANSACTION
postgres=> COMMIT PREPARED 'Transaction#2' ;
COMMIT PREPARED
postgres=> SELECT calls, query FROM pg_stat_statements ORDER BY query ;
 calls |          query
-------+--------------------------
     1 | <insufficient privilege>
     2 | BEGIN
     2 | COMMIT PREPARED $1
     2 | PREPARE TRANSACTION $1
(4 rows)
```

□ Pg_stat_statements_reset function

The parameter minmax_only has been added to the pg_stat_statements_reset function. Setting this parameter to true will reset the maximum/minimum value (min_plan_time column, min_exec_time column, etc.) information. [43cbeda]

Example 111 Reset min/max values

```
postgres=# SELECT query, calls, min_exec_time FROM pg_stat_statements
                WHERE queryid=228406429875636958 ;
           query            | calls | min_exec_time
----------------------------+-------+---------------
 SELECT COUNT(*) FROM data1 |     2 |        20.938
(1 row)

postgres=# SELECT pg_stat_statements_reset(0, 0, 228406429875636958, true) ;
   pg_stat_statements_reset
-------------------------------
 2024-05-24 15:35:14.552288+09
(1 row)

postgres=# SELECT query, calls, min_exec_time FROM pg_stat_statements
                WHERE queryid=228406429875636958 ;
           query            | calls | min_exec_time
----------------------------+-------+---------------
 SELECT COUNT(*) FROM data1 |     2 |             0
(1 row)
```

3.5.4. Postgres_fdw

The following features have been added to the postgres_fdw module.

□ Increase tuple cost

The default cost per tuple to access a remote table has been changed from 0.01 to 0.2. [cac169d]

□ De-parsing Semi-Join

The EXISTS clause used between tables in the same FOREIGN SERVER can now be executed remotely. [824dbea]

Example 112 Execution Plan on PostgreSQL 17

```
postgres=> EXPLAIN (VERBOSE, COSTS OFF) SELECT COUNT(*) FROM data1 d1
        WHERE EXISTS (SELECT * FROM data2 d2 WHERE d1.c1 = d2.c1) ;
                                QUERY PLAN
-------------------------------------------------------------------------------

 Foreign Scan
   Output: (count(*))
   Relations: Aggregate on ((public.data1 d1) SEMI JOIN (public.data2 d2))
   Remote SQL: SELECT count(*) FROM public.data1 r1 WHERE EXISTS (SELECT NULL
FROM public.data2 r2 WHERE ((r1.c1 = r2.c1)))
(4 rows)
```

Example 113 Execution Plan on PostgreSQL 16

```
postgres=> EXPLAIN (VERBOSE, COSTS OFF) SELECT COUNT(*) FROM data1 d1
        WHERE EXISTS (SELECT * FROM data2 d2 WHERE d1.c1 = d2.c1) ;
                        QUERY PLAN
-----------------------------------------------------------

 Aggregate
   Output: count(*)
   ->  Hash Semi Join
         Hash Cond: (d1.c1 = d2.c1)
         ->  Foreign Scan on public.data1 d1
               Output: d1.c1
               Remote SQL: SELECT c1 FROM public.data1
         ->  Hash
               Output: d2.c1
               ->  Foreign Scan on public.data2 d2
                     Output: d2.c1
                     Remote SQL: SELECT c1 FROM public.data2
(12 rows)
```

3.5.5. Ltree

The hash indexes are now accessible for match searches. [485f0aa]

Example 114 Using hash indexes for ltree types

```
postgres=> \d ltree1
              Table "public.ltree1"
 Column | Type  | Collation | Nullable | Default
--------+-------+-----------+----------+---------
 path   | ltree |           |          |
Indexes:
    "path_hash_idx" hash (path)

postgres=> EXPLAIN (COSTS OFF)
        SELECT * FROM ltree1 WHERE path='Top.Hobbies.Amateurs_Astronomy' ;
                            QUERY PLAN
-------------------------------------------------------------
 Index Scan using path_hash_idx on ltree1
   Index Cond: (path = 'Top.Hobbies.Amateurs_Astronomy'::ltree)
(2 rows)
```

3.5.6. Injection_points

The extension module injection_points has been added to the src/test/modules/injection_points directory. This module provides the infrastructure to utilize basic injection points. The injection_points module provides the following functions. [49cd2b9, f587338]

Table 45 Added functions

Function Name	Description	Note
injection_points_attach	Create Injection Point	
injection_points_detach	Delete Injection Point	
injection_points_run	Execute Injection Point	
injection_points_set_local	Limit injection point execution to the current process	
injection_points_wakeup	Wakeup Injection Point	

3.5.7. Test_radixtree

The extension test_radixtree has been added to the directory src/test/modules/test_radixtree. This extension is a test implementation of the paper "The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases" published by Viktor Leis, Alfons Kemper and Thomas Neumann in 2013 (https://ieeexplore.ieee.org/document/6544812). [ee1b30f]

Table 46 Added function

Function name	Description	Note
test_radixtree	Execute the Radix Tree test.	

Example 115 Using the test_radixtree extension

```
postgres=# CREATE EXTENSION test_radixtree ;
CREATE EXTENSION
postgres=# SELECT test_radixtree() ;
NOTICE:  testing node node-4 with shift 0 and ascending keys
NOTICE:  testing node node-4 with shift 0 and descending keys
NOTICE:  testing node node-4 with shift 8 and ascending keys
NOTICE:  testing node node-4 with shift 8 and descending keys
NOTICE:  testing node node-4 with shift 56 and ascending keys
NOTICE:  testing node node-4 with shift 56 and descending keys
NOTICE:  testing node node-16-lo with shift 0 and ascending keys
...
NOTICE:  testing node node-256 with shift 56 and ascending keys
NOTICE:  testing node node-256 with shift 56 and descending keys
 test_radixtree
----------------


(1 row)
```

3.5.8. Test_tidstore

The extension module test_tidstore has been added to the src/test/modules/test_tidstore directory. The test_tidstore module is a test module for efficiently storing large sets of TIDs. [30e1442]

Table 47 Added functions

Function Name	Description	Note
test_create	Creating a test	
check_set_block_offsets	Validate TID in store against array	
test_is_full	Check for memory overflow	
test_destroy	Discarding the test	

3.5.9. Xid_wraparound

The extension module xid_wraparound has been added to the src/test/modules/xid_wraparound directory. This module provides functions to quickly burn through lots of transaction IDs. [e255b64]

Table 48 Added functions

Function Name	Description	Note
consume_xids	To proceed to the specified value for the transaction ID.	
consume_xids_until	To Proceed to the specified value of Transaction ID.	

Example 116 Burn through lots of transaction ID

```
postgres=# CREATE EXTENSION xid_wraparound ;
CREATE EXTENSION
postgres=# SELECT txid_current() ;
 txid_current
--------------
          753
(1 row)

postgres=# SELECT consume_xids('100000000') ;
NOTICE:  consumed 10000055 / 100000000 XIDs, latest 0:10000809
NOTICE:  consumed 20000864 / 100000000 XIDs, latest 0:20001618
...
NOTICE:  consumed 90004750 / 100000000 XIDs, latest 0:90005504
 consume_xids
--------------
    100000754
(1 row)
```

3.5.10. Miscellaneous

In addition to those described above, the following test extension modules have been added under the src/test/modules directory.

Table 49 Added modules

Module Name	Description
test_dsa	Test module for Dynamic Shared Area (DSA). [325f540]
test_dsm_registory	Test module for Dynamic Shared Memory (DSM) registration. [8b2bcf3]
test_json_parser	Test module for JSON parsing. [3311ea8]
test_resowner	Test module for resource owner feature. [b8bff07]

Hewlett Packard
Enterprise

URL list

The following websites are references to create this material.

□ Release Notes

https://www.postgresql.org/docs/17/release-17.html

□ Commitfests

https://commitfest.postgresql.org/

□ PostgreSQL 17 Manual

https://www.postgresql.org/docs/17/index.html

□ PostgreSQL 17 Open Items

https://wiki.postgresql.org/wiki/PostgreSQL_17_Open_Items

□ Git

git://git.postgresql.org/git/postgresql.git

□ GitHub

https://github.com/postgres/postgres

□ Announce of the PostgreSQL 17 Beta 1

https://www.postgresql.org/about/news/postgresql-17-beta-1-released-2865/

□ Michael Paquier - PostgreSQL committer

https://paquier.xyz/

□ Qiita (Nuko@Yokohama) (Japanese)

http://qiita.com/nuko_yokohama

□ pgsql-hackers Mailing list

https://www.postgresql.org/list/pgsql-hackers/

□ PostgreSQL Developer Information

https://wiki.postgresql.org/wiki/Development_information

□ pgPedia

https://pgpedia.info/postgresql-versions/postgresql-17.html

□ SQL Notes

https://sql-info.de/postgresql/postgresql-17/articles-about-new-features-in-postgresql-17.html

□ Slack - postgresql-jp (Japanese)

https://postgresql-jp.slack.com/

Hewlett Packard Enterprise

Change history

Change history

Version	Date	Author	Description
0.1	Apr 11, 2024	Noriyoshi Shinoda	Create an internal review version. Reviewers: Tomoo Takahashi Akiko Takeshima (Hewlett Packard Enterprise Japan)
1.0	May 25, 2024	Noriyoshi Shinoda	Verification completed in PostgreSQL 17 Beta 1 environment.