

PostgreSQL 15 New Features

With Examples (GA)

Hewlett Packard Enterprise Japan LLC.
Noriyoshi Shinoda

Index

Index.....	2
1. About This Document.....	5
1.1. Purpose.....	5
1.2. Audience.....	5
1.3. Scope.....	5
1.4. Software Version.....	5
1.5. The Question, comment, and Responsibility.....	6
1.6. Notation.....	6
2. New Features Summary.....	7
2.1. Improvements to adapt to large scale environments.....	7
2.2. Improve reliability.....	7
2.3. Improved maintainability.....	8
2.4. Improved in Programming.....	8
2.5. Preparing for future new features.....	8
2.6. Incompatibility.....	9
2.6.1. Access privileges to the PUBLIC schema.....	9
2.6.2. Exclusive backup mode.....	9
2.6.3. Psql.....	10
2.6.4. Literal.....	12
2.6.5. ANALYZE.....	13
2.6.6. Python 2.....	13
2.6.7. Date_bin.....	13
2.6.8. Pg_amcheck.....	14
2.6.9. Pgcrypto.....	14
2.6.10. Pg_dump / pg_dumpall.....	14
2.6.11. Pg_upgrade.....	14
2.6.12. Postmaster.....	14
2.6.13. Array_to_tsvector.....	14
2.6.14. Xml2.....	15
2.6.15. Pipeline mode.....	15
2.6.16. Extension.....	15
3. New Feature Detail.....	16
3.1. Architecture.....	16
3.1.1. Modified System catalogs.....	16

3.1.2. Logical Replication Enhancements.....	21
3.1.3. Parallel Query Enhancements	25
3.1.4. WAL compression.....	26
3.1.5. Archive Library	26
3.1.6. Global Locale Provider	27
3.1.7. Statistics Information	29
3.1.8. GiST Index.....	30
3.1.9. Wait Events	30
3.1.10. Role.....	31
3.1.11. Libpq	31
3.1.12. Custom WAL resource manager.....	32
3.1.13. Hook.....	32
3.1.14. Custom scan providers	33
3.1.15 Build target	33
3.1.16. Dynamic Shared Memory.....	34
3.2. SQL Statement.....	35
3.2.1. NUMERIC data type.....	35
3.2.2. ALTER DATABASE.....	36
3.2.3. ALTER TABLE	36
3.2.4. COPY.....	37
3.2.5. CLUSTER	38
3.2.6. CREATE DATABASE	38
3.2.7. CREATE TABLE	39
3.2.8. CREATE UNIQUE INDEX.....	40
3.2.9. CREATE SEQUENCE.....	41
3.2.10. CREATE VIEW.....	42
3.2.11. EXPLAIN	43
3.2.12. GRANT	43
3.2.13. MERGE	44
3.2.14. VACUUM.....	45
3.2.15. Functions	46
3.3. Configuration parameters.....	52
3.3.1. Added parameters	52
3.3.2. Modified Parameters	53
3.3.3. Parameters with default values changed.....	54
3.3.4. Removed parameter	54



3.3.5. Error when changing parameters	55
3.4. Utilities	56
3.4.1. Configure	56
3.4.2. Psql	56
3.4.3. Pg_amcheck	59
3.4.4. Pg_basebackup	59
3.4.5. Pg_dump	61
3.4.6. Pg_recvlogical	61
3.4.7. Pg_receivewal	61
3.4.8. Pg_resetwal	62
3.4.9. Pg_rewind	62
3.4.10. Pg_upgrade	62
3.4.11. Pg_waldump	63
3.5. Contrib modules	65
3.5.1. Amcheck	65
3.5.2. Basebackup_to_shell	65
3.5.3. File_fdw	65
3.5.4. Pg_stat_statements	66
3.5.5. Pg_walinspect	67
3.5.6. Postgres_fdw	68
3.5.7. Sepgsql	70
URL list	71
Change history	72

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 15 (15.0).

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 14 (14.5) and PostgreSQL 15 (15.0). 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 psql command tab input
- Improvement of pgbench command
- Improvement of documentation, modify typo 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 14.5 (for comparison) PostgreSQL 15 (15.0) (October 10, 2022 21:01:24)
Operating System	Red Hat Enterprise Linux 8 Update 6 (x86-64)
Configure option	--with-ssl=openssl --with-python --with-lz4 --with-zstd --with-llvm --with-icu

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, LLC.

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 notation

Notation	Description
#	Shell prompt for Linux root user.
\$	Shell prompt for Linux general user.
Bold	The user input string.
postgres=#	psql command prompt for PostgreSQL administrator.
postgres=>	psql command prompt for PostgreSQL general user.
<u>Underline</u>	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
<i>Italic</i>	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 15. 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:

□ Parallel Query Improvements

Parallel queries now work for SELECT DISTINCT statements.

□ Improved Compression Method

LZ4 and Zstandard are now available for compressing WALs and base backups. `pg_receivewal` command compression method has also been added.

□ Improvements to utilization statistics

The stats collector process has been deprecated, and performance statistics have been changed from communicating using UDP to using a shared memory area.

2.2. Improve reliability

PostgreSQL 15 implements the following enhancements to improve reliability.

□ Addition of archive library

Archive logging can now be performed by a shared library. The Contrib module `basic_archive` has been added as a reference implementation.

□ Checkpoint log

The default value for the `log_checkpoints` parameter has changed to "on". The checkpoint execution log will be output by default.

2.3. Improved maintainability

The following features that can improve operability have been added.

□ Improvements in logical Replication

Logical Replication has been greatly enhanced in PostgreSQL 15. Logical Replication can now only replicate tuples that meet certain conditions. The conditions for which tuples can be replicated are specified in the WHERE clause of the CREATE PUBLICATION or ALTER PUBLICATION statement. Replication columns can now be selected by specifying a list of columns in addition to the table in the CREATE PUBLICATION statement. Also, updates to specified LSNs can now be skipped.

□ Monitoring Enhancements

The pg_stat_subscription_stats view has been added to check the status of errors that occur on the subscription worker in Logical Replication environment. Pg_stat_statements module provides information on temporary file I/O times and JITs. information can now be monitored.

□ Log File Format

Log files can now be output in JSON format.

2.4. Improved in Programming

Added MERGE statement to SQL syntax. The MERGE statement can execute INSERT / UPDATE / DELETE processing at once depending on whether the join conditions of the tables match or not match.

2.5. Preparing for future new features

PostgreSQL 15 is now ready for features that will be provided in future versions.

□ Change table access method

Added syntax to modify table access methods in the ALTER TABLE and ALTER MATERIALIZED VIEW statements.

2.6. Incompatibility

In PostgreSQL 15, the following specifications have been changed from PostgreSQL 14.

2.6.1. Access privileges to the PUBLIC schema

Previously, the public schema had CREATE and USAGE privileges for all users (PUBLIC). In PostgreSQL 15, access to the public schema is limited to the database owner (pg_database_owner role). The owner of the public schema has also changed to pg_database_owner. With this change, it will be in the recommended state after CVE-2018-1058.

Example 1 Access privileges to the PUBLIC schema

```
postgres=# CREATE USER demo PASSWORD 'demo' ;
CREATE ROLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> CREATE TABLE public.data1(c1 INT, c2 VARCHAR(10)) ;
ERROR: permission denied for schema public
LINE 1: CREATE TABLE public.data1(c1 INT, c2 VARCHAR(10)) ;
          ^

postgres=> CREATE VIEW public.view1 AS SELECT 1 ;
ERROR: permission denied for schema public
postgres=> CREATE SEQUENCE public.seq1 ;
ERROR: permission denied for schema public
```

2.6.2. Exclusive backup mode

The deprecated exclusive backup mode has been removed. Exclusive backup prohibits multiple simultaneous backups. To start an exclusive backup, set the exclusive parameter of the pg_start_backup function to true (default). The function name used to start/end online backup has changed due to the deletion of exclusive backup mode.

Table 4 Function name change

Operation	PostgreSQL 14	PostgreSQL 15	Note
Start backup	pg_start_backup	pg_backup_start	
Stop backup	pg_stop_backup	pg_backup_stop	
Backup confirmation	pg_is_in_backup	-	Removed
Backup start time	pg_backup_start_time	-	Removed

Example 2 Execute online backup

```

postgres=# SELECT pg_backup_start('start online backup#1', false) ;
 pg_backup_start
-----
 0/4000028
(1 row)

postgres=# \! cp -r data backup
postgres=# SELECT pg_backup_stop(true) ;
NOTICE:  all required WAL segments have been archived
          pg_backup_stop
-----
(0/4000138,"START WAL LOCATION: 0/4000028 (file 000000010000000000000004)+
CHECKPOINT LOCATION: 0/4000060                                     +
BACKUP METHOD: streamed                                           +
BACKUP FROM: primary                                             +
START TIME: 2022-10-13 23:03:20 JST                               +
LABEL: start online backup#1                                     +
START TIMELINE: 1                                               +
", "")
(1 row)

```

2.6.3. Psql

The psql command has the following changes related to compatibility.

- Supported version of the connected instance

Connections to servers prior to PostgreSQL 9.2 are no longer supported. A warning message will be output when connecting to a server with PostgreSQL 9.2 or earlier.

□ Password change

The behavior of the `\password` command has been changed. Previously, the password of the user specified at login was changed, but the user is determined based on the result of the execution of the `SELECT CURRENT_USER` statement. This changes the behavior of the `SET SESSION AUTHORIZATION` statement. In addition, the name of the user to be changed is now displayed at the prompt. This specification change will be backported to the previous version.

Example 3 Behavior of PostgreSQL 14

```
postgres=# SET SESSION AUTHORIZATION demo ;
SET
postgres=> \password
Enter new password: <<PASSWORD>> ← Password of postgres user
Enter it again: <<PASSWORD>> ← Password of postgres user
ERROR: must be superuser to alter superuser roles or change superuser attribute
postgres=>
```

Example 4 Behavior of PostgreSQL 15

```
postgres=# SET SESSION AUTHORIZATION demo ;
SET
postgres=> \password
Enter new password for user "demo": <<PASSWORD>> ← Password of demo user
Enter it again: <<PASSWORD>> ← Password of demo user
postgres=>
```

□ Displaying Multiple Results

In previous versions, the `psql` command would print only the last result if multiple result sets were returned; PostgreSQL 15 prints all results. To change to the same behavior as in the previous version, execute the command `\set SHOW_ALL_RESULTS off`.

Example 5 Behavior of PostgreSQL 14

```
postgres=> SELECT 1 \; SELECT 2 ;
?column?
-----
         2
(1 row)
```

Example 6 Behavior of PostgreSQL 15

```
postgres=> SELECT 1 \; SELECT 2 ;
?column?
-----
      1
(1 row)

?column?
-----
      2
(1 row)

postgres=> \set SHOW_ALL_RESULTS off
postgres=> SELECT 1 \; SELECT 2 ;
?column?
-----
      2
(1 row)
```

□ --Single-transaction option behavior

If the --single-transaction option is specified and an error occurs with the ON_ERROR_STOP variable defined, a ROLLBACK statement is automatically executed.

2.6.4. Literal

Literal strings that begin with a number were previously split into a number part and a string part, but PostgreSQL 15 causes an error.

Example 7 Behavior of PostgreSQL 14

```
postgres=> SELECT 123abc ;
abc
-----
 123
(1 row)
```

Example 8 Behavior of PostgreSQL 15

```
postgres=> SELECT 123abc ;
ERROR:  trailing junk after numeric literal at or near "123a"
LINE 1: SELECT 123abc;
           ^
```

2.6.5. ANALYZE

The ANALYZE statement has been changed to use `maintenance_io_concurrency` instead of `effective_io_concurrency` for execution. This specification will be backported to PostgreSQL 14.

2.6.6. Python 2

Python 2 support in PL/Python has been removed due to the expiration of Python 2.7 support; Python 2 is rejected when running the 'configure' command. Python 3.2 and above is supported.

Example 9 Python 2 only environment

```
$ ./configure --with-python
checking build system type... x86_64-pc-linux-gnu
...
configure: using perl 5.26.3
checking for python3... no
checking for python... /usr/bin/python
configure: using python 2.7.18 (default, Feb 10 2022, 14:26:12)
configure: error: Python version 2.7 is too old (version 3 or later is required)
```

The extensions such as `hstore_plpython2u`, `jsonb_plpython2u`, and `ltree_plpython2u` have also been removed.

2.6.7. Date_bin

Negative values can no longer be specified for the interval of the `date_bin` function. This specification change will be backported to the previous version.

Example 10 Execute date_bin function

```
postgres=> SELECT date_bin('-2'::interval, timestamp '1970-01-01 01:00:00',  
timestamp '1970-01-01 00:00:00') ;  
ERROR:  stride must be greater than zero
```

2.6.8. Pg_amcheck

The --quiet option (short -q) of the pg_amcheck command has been removed. This fix will also be backported to PostgreSQL 14.

2.6.9. Pgcrypto

The proprietary built-in encryption algorithm has been removed and OpenSSL is now required.

2.6.10. Pg_dump / pg_dumpall

Removed the upgrade feature from versions prior to PostgreSQL 9.2. Also, the --no-synchronized-snapshots option has been removed.

2.6.11. Pg_upgrade

Removed the upgrade feature from versions prior to PostgreSQL 9.2.

2.6.12. Postmaster

The postmaster startup option --forkboot has been changed to --forkaux.

2.6.13. Array_to_tsvector

Disallow making an empty lexeme via array_to_tsvector function.

Example 11 Execute the array_to_tsvector function

```
postgres=> SELECT array_to_tsvector (ARRAY['base', 'hidden', 'rebel', '']) ;  
ERROR:  lexeme array may not contain empty strings
```



2.6.14. Xml2

The `xml_is_well_formed` function has been removed from the Contrib module `xml2`.

2.6.15. Pipeline mode

Removed support for `PQsendQuery` function in pipeline mode. Use the `PQsendQueryParams` function instead.

2.6.16. Extension

`CREATE OR REPLACE` statements for existing objects are disallowed unless they already belong to the extension.

3. New Feature Detail

3.1. Architecture

3.1.1. Modified System catalogs

The following catalogs have been changed.

Table 5 Added system catalogs and views

Catalog/View name	Description
pg_ident_file_mappings	Provides a summary of the contents of the client user name mapping configuration file.
pg_parameter_acl	Provides ACL information that allows the execution of the ALTER SYSTEM SET statement.
pg_publication_namespace	Stores the mapping between the schema and the PUBLICATION.
pg_stat_recovery_prefetch	Provides prefetch status during recovery.
pg_stat_subscription_stats	Stores initial data synchronization failure information that is performed when the subscription is created.

Table 6 System catalogs/views with additional columns

Catalog/View name	Added column	Data type	Description
pg_collation	colliculocale	text	ICU Collation name
pg_constraint	confdelsetcols	smallint[]	Column with ON DELETE clause for foreign key
pg_database	daticulocale	text	ICU Locale name
	datlocprovider	char	ICU Collation Provider name
	datcollversion	boolean	Collation version
pg_index	indnullsnotdistinct	boolean	Indicates the nullable setting of the unique index
pg_publication_rel	Prqual	pg_node_tree	Tuple conditions for replication
	prattrs	int2vector	Column number to replicate
pg_statistic_ext_data	stxdinherit	boolean	If true, the stats include inheritance child columns

Catalog/View name	Added column	Data type	Description
pg_stats_ext	inherited	boolean	If true, the stats include inheritance child columns
pg_stats_ext_exprs	inherited	boolean	If true, the stats include inheritance child columns
pg_subscription	subskiplsn	pg_lsn	Skipped LSN
	subtwophasestate	char	Indicates the state of Two-Phase mode
	subdisableonerr	boolean	Disable subscription on error occurs
information_schema.tables_constraints	nulls_distinct	information_schema.yes_or_no	Indicates the nullable setting of the unique index

Table 7 System catalogs/views with columns removed

Catalog name	Column name	Description
pg_database	datlastsysoid	Removed because not used

Table 8 System catalogs/views with modified output

Catalog / View name	Description
pg_type	Added 'Z' to the typcategory column to indicate internal use.
pg_collation	The data type of collcollate and collctype columns has changed from 'name' to 'text'.
pg_database	The data type of the datcollate and datctype columns has changed from 'name' to 'text'.
pg_backend_memory_counters	Information on the memory area for statistical data (PgStat *) is now output.
pg_shmem_allocations	Added Shared Memory Stats, XLOG Recovery Ctl, and XLogPrefetchStats tuples.

Among the modified system catalogs, the details of the major catalogs are described below.

□ Pg_ident_file_mappings catalog

This is a view for searching the contents of the pg_ident.conf file from SQL statements.

Table 9 Pg_ident_file_mappings catalog

Column name	Data type	Description
line_number	integer	Line number in the pg_ident.conf file
map_name	text	Map name
sys_name	text	System name
pg_username	text	PostgreSQL user name
error	text	Error message

Example 12 Reference to pg_ident_file_mapping view.

```
postgres=# SELECT * FROM pg_ident_file_mapping ORDER BY 1 ;
```

line_number	map_name	sys_name	pg_username	error
43	map1	sysname1	pgname1	
44	map2	sysname2	pgname2	

(2 rows)

□ Pg_parameter_acl catalog

The catalog records configuration parameters for which privileges have been granted to one or more roles.

Table 10 Pg_parameter_acl catalog

Column name	Data type	Description
oid	oid	Object ID
parname	text	The parameter name to allow change
paracl	aclitem[]	Role information to allow changes

□ Pg_publication_namespace

Information is stored when the FOR TABLE IN SCHEMA clause is specified in the CREATE PUBLICATION statement. Provides a mapping between PUBLICATION and SCHEMA.

Table 11 Pg_publication_namespace catalog

Column name	Data type	Description
oid	oid	Row identifier
pnpubid	oid	Reference to PUBLICATION
pnnspid	oid	Reference to SCHEMA

Example 13 Reference to pg_publication_namespace catalog

```
postgres=> SELECT * FROM pg_publication_namespace ;
 oid | pnpubid | pnnspid |
-----+-----+-----+
 16411 |   16410 |   2200 |
(1 row)
```

□ Pg_stat_recovery_prefetch view

Provides prefetched block statistics during recovery. It is also possible to check the status of replication processing in the standby instance of the streaming replication environment.

Table 12 Pg_stat_recovery_prefetch view

Column name	Data type	Description
stats_reset	timestamp with time zone	Date time when the view was reset
prefetch	bigint	Number of blocks prefetched because they were not in the buffer pool
hit	bigint	Number of blocks not prefetched because they were already in the buffer pool
skip_init	bigint	Number of blocks not prefetched because they would be zero-initialized
skip_new	bigint	Number of blocks not prefetched because they didn't exist yet
skip_fpw	bigint	Number of blocks not prefetched because a full-page image was included in the WAL
skip_rep	bigint	Number of blocks not prefetched because they were already recently prefetched
wal_distance	integer	How many bytes ahead the prefetcher is looking
block_distance	integer	How many blocks ahead the prefetcher is looking
io_depth	integer	How many prefetches have been initiated but are not yet known to have been completed

□ Pg_stat_subscription_stats view

The pg_stat_subscription_stats view contains error information about logical replication subscription workers.

Table 13 Pg_stat_subscription_stats view

Column name	Data type	Description
subid	oid	OID of the subscription
subname	name	Name of the subscription
apply_error_count	bigint	Number of times an error occurred while applying changes
sync_error_count	bigint	Number of times an error occurred during the initial table synchronization
stats_reset	timestamp with time zone	Time at which these statistics were last reset

Example 14 Reference to pg_stat_subscription_stats view

```
postgres=# SELECT * FROM pg_stat_subscription_stats ;
-[ RECORD 1 ]-----+-----
subid          | 16395
subname        | sub1
apply_error_count | 0
sync_error_count | 23
stats_reset    |
```

□ Reference to memory related view

The `pg_backend_memory_contexts` view and `pg_shmem_allocations` view can now be referenced not only by users with the SUPERUSER attribute but also by users belonging to the `pg_read_all_stats` role.

Example 15 Grant pg_read_all_stats role

```
postgres=# GRANT pg_read_all_stats TO demo ;
GRANT ROLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> SELECT COUNT(*) FROM pg_backend_memory_contexts ;
 count
-----
    122
(1 row)
```

3.1.2. Logical Replication Enhancements

The following features have been added to Logical Replication.

□ Column-specific replication

It is now possible to replicate only specific columns in a table. Specify a column list for the table name in the CREATE PUBLICATION or ALTER PUBLICATION statement. The target column number is stored in the prattrs column of the pg_publication_rel catalog.

Example 16 Column specific replication

```
postgres=> CREATE TABLE repl1(c1 INT PRIMARY KEY, c2 INT, c3 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE PUBLICATION pub1 FOR TABLE repl1(c1, c2) ;
CREATE PUBLICATION
postgres=> SELECT oid, prpubid, prrelid, prattrs FROM pg_publication_rel ;
-[ RECORD 1 ]--
oid      | 16394
prpubid  | 16393
prrelid  | 16388
prattrs  | 1_2
```

When executing an UPDATE or DELETE statement, the column specification must include all replica identities (primary key, etc.). In the example below, the UPDATE statement is in error by specifying a column that does not include the primary key in the publication.

Example 17 UPDATE statement failure

```
postgres=> CREATE TABLE repl2(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE PUBLICATION pub2 FOR TABLE repl2(c2) ;
CREATE PUBLICATION
postgres=> UPDATE repl2 SET c2='update' WHERE c1=100 ;
ERROR:  cannot update table "repl2"
DETAIL:  Column list used by the publication does not cover the replica
identity.
```

□ Row filtered replication

By specifying the WHERE clause in the CREATE PUBLICATION statement or the ALTER

PUBLICATION statement, the tuples in the table can be limited and replicated. Replication conditions are specified for each table. The WHERE clause description must be enclosed in parentheses.

Example 18 Specify WHERE clause

```
postgres=> ALTER PUBLICATION pub1 SET TABLE rep11 WHERE (c1 < 1000) ;  
ALTER PUBLICATION
```

In order to execute UPDATE or DELETE statements on the table to be replicated, the WHERE clause must include a part of the replica identity (primary key, etc.).

Example 19 WHERE clause without replica identity

```
postgres=> CREATE TABLE rep13(c1 INT PRIMARY KEY, c2 VARCHAR(10)) ;  
CREATE TABLE  
postgres=> CREATE PUBLICATION pub3 FOR TABLE rep13 WHERE (c2='update') ;  
CREATE PUBLICATION  
postgres=> UPDATE rep13 SET c2='modify' WHERE c1=1000 ;  
ERROR: cannot update table "rep13"  
DETAIL: Column used in the publication WHERE expression is not part of the  
replica identity.  
postgres=> DELETE FROM rep13 WHERE c1=1000 ;  
ERROR: cannot delete from table "rep13"  
DETAIL: Column used in the publication WHERE expression is not part of the  
replica identity.
```

The conditional statement that can be specified in the WHERE clause must be statically determined. Therefore, user-defined functions and MUTABLE-specified functions cannot be specified.

Example 20 Specify the MUTABLE function

```
postgres=> CREATE PUBLICATION pub4 FOR TABLE rep11 WHERE (c1 = random()) ;  
ERROR: invalid publication WHERE expression  
LINE 1: ...EATE PUBLICATION pub4 FOR TABLE rep11 WHERE (c1 = random());  
^  
DETAIL: User-defined or built-in mutable functions are not allowed.
```

The conditions specified in the WHERE clause are converted and stored in the prqual column of the pg_publication_rel catalog. The psql command can be checked with the \d command or the \dRp+ command.

Example 21 \d command

```
postgres=> \d repl1
                Table "public.repl1"
  Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
  c1     | integer                |           | not null |
  c2     | character varying(10) |           |         |
Indexes:
    "repl1_pkey" PRIMARY KEY, btree (c1)
Publications:
    "pub1" WHERE (c1 < 1000)
```

- Specify all tables in the schema

The syntax is now available to register all tables in a particular schema with a PUBLICATION object. Specify the FOR TABLES IN SCHEMA clause and schema name in the CREATE PUBLICATION statement (or ALTER PUBLICATION statement). This syntax can only be executed by users with the SUPERUSER attribute. In the case of the FOR ALL TABLE clause that was available in the past, tables of all schemas were registered.

Syntax

```
CREATE PUBLICATION publication_name FOR TABLES IN SCHEMA schema_name
ALTER PUBLICATION publication_name ADD TABLES IN SCHEMA schema_name
```

Example 22 Specify IN SCHEMA clause

```
postgres=# CREATE PUBLICATION pub5 FOR TABLES IN SCHEMA schema1 ;
CREATE PUBLICATION
postgres=# \dRp
```

List of publications							
Name	Owner	All tables	Inserts	Updates	Deletes	Truncates	...
pub5	postgres	f	t	t	t	t	f

(1 row)

The `pg_publication_namespace` catalog has been added with this implementation.

□ Skip LSN

It is now possible to specify the `SKIP` clause in the `ALTER SUBSCRIPTION` statement. Specify LSN in the `SKIP` clause. Execution of this statement allows the `SUBSCRIPTION` side of logical replication to skip transactions that caused synchronization errors.

Syntax

```
ALTER SUBSCRIPTION subscription_name SKIP (LSN=' lsn_value' )
```

The skipped LSN can be obtained in the `subskiplsn` column of the `pg_subscription` catalog. The `psql` command added a Skip LSN entry to the output of the `\dRs +` command.

Example 23 Specify the SKIP clause

```
postgres=# ALTER SUBSCRIPTION sub1 SKIP (LSN = '0/30B51E0') ;
ALTER SUBSCRIPTION
postgres=# SELECT subskiplsn FROM pg_subscription ;
subskiplsn
-----
0/30B51E0
(1 row)
```


- Disable SUBSCRIPTION when an error occurs

The option `disable_on_error` can now be specified to disable SUBSCRIPTION if a replication error occurs. The default is false and no disabling is performed. The column `subdisableonerr` has been added to the `pg_subscription` catalog to indicate this option value.

Example 24 Execute CREATE SUBSCRIPTION statement

```
postgres=# CREATE SUBSCRIPTION sub1 CONNECTION 'host=remsvr1, dbname=postgres'  
PUBLICATION pub1 WITH (disable_on_error=true, two_phase=true) ;  
NOTICE: created replication slot "sub1" on publisher  
CREATE SUBSCRIPTION
```

- TWO_PHASE option

The replication protocol command `CREATE_REPLICATION_SLOT` has been expanded and the `TWO_PHASE` option has been added. This makes it possible to decode `PREPARE TRANSACTION`, `COMMIT PREPARED`, `ROLLBACK PREPARED`, etc.

3.1.3. Parallel Query Enhancements

The `SELECT DISTINCT` statement can now work with parallel queries.

Example 25 Parallel query for SELECT DISTINCT

```
postgres=> EXPLAIN SELECT DISTINCT c2 FROM data1 ;  
QUERY PLAN  
-----  
Unique (cost=11614.55..11614.56 rows=1 width=6)  
-> Sort (cost=11614.55..11614.56 rows=2 width=6)  
Sort Key: c2  
-> Gather (cost=11614.33..11614.54 rows=2 width=6)  
Workers Planned: 2  
-> HashAggregate (cost=10614.33..10614.34 rows=1 width=6)  
Group Key: c2  
-> Parallel Seq Scan on data1 (cost=0.00..9572.67 ro...  
(8 rows)
```

3.1.4. WAL compression

It is now possible to specify LZ4 and Zstandard as the compression method for the full-page image output to WAL. Change the parameter `wal_compression` to perform WAL compression. The setting value `on`, which is the conventional compression method, is an alias for the setting value `pglz`. The default value remains `off` without any change.

Example 26 WAL compression settings

```
postgres=# SELECT name, setting, vartype, enumvals FROM pg_settings
           WHERE name='wal_compression' ;
-[ RECORD 1 ]-----
name       | wal_compression
setting    | off
vartype    | enum
enumvals   | {pglz, lz4, zstd, on, off}
```

To use value `lz4` or `zstd`, it must be compiled with `--with-lz4` and `--with-zstd` specified when executing 'configure' command.

3.1.5. Archive Library

It is now possible to use a shared library in addition to the traditional command (`archive_command` parameter) as a way to create an archive of WAL files. The `archive_library` parameter has been added to specify the archive library name. If `archive_library` is not set, the `archive_command` parameter can still be used. The Contrib module `basic_archive` has been added as a standard archive library. This Contrib module does not require the execution of the `CREATE EXTENSION` statement. The `basic_archive.archive_directory` parameter can be specified as the archive destination directory.

Example 27 Specify the Archive Library

```

postgres=# SELECT * FROM pg_settings WHERE name LIKE 'archive_library' ;
-[ RECORD 1 ]-----+-----
name          | archive_library
setting       | basic_archive
unit          |
category      | Write-Ahead Log / Archiving
short_desc    | Sets the library that will be called to archive a WAL file.
extra_desc    | An empty string indicates that "archive_command" should be
used.
context       | sighup
vartype       | string
source        | configuration file
min_val       |
max_val       |
enumvals      |
boot_val      |
reset_val     | basic_archive
sourcefile    | /home/postgres/data/postgresql.conf
sourceline    | 255
pending_restart | f

postgres=# SHOW basic_archive.archive_directory ;
-[ RECORD 1 ]-----+-----
basic_archive.archive_directory | /home/postgres/arch

```

If both `archive_library` and `archive_command` are specified, `archive_library` is used.

3.1.6. Global Locale Provider

It is now possible to specify an ICU as the global locale provider. Allows ICU to be used as the default locale provider for the entire database cluster or database. You must use a binary compiled with the `--with-icu` option to use this feature. The `--locale-provider` option that specifies the locale provider name can be specified for the `initdb` command and `createdb` command. Specify `libc` or `icu` for this option. The `--icu-locale` option allows you to specify the ICU locale name.

Example 28 Execute initdb command

```
$ initdb --locale-provider=icu --icu-locale=en --encoding=utf8 -D 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:

```
provider:    icu
ICU locale:  en
LC_COLLATE:  en_US.utf8
LC_CTYPE:    en_US.utf8
LC_MESSAGES: en_US.utf8
LC_MONETARY: en_US.utf8
LC_NUMERIC:  en_US.utf8
LC_TIME:     en_US.utf8
```

The default text search configuration will be set to "english".

...

Added LOCALE_PROVIDER to specify the locale provider and ICU_LOCALE to specify the locale name to the options of the CREATE DATABASE statement.

Example 29 Execute CREATE DATABASE statement

```
postgres=# CREATE DATABASE icudb1 LOCALE_PROVIDER=icu ICU_LOCALE=en
          TEMPLATE=template0 ;
CREATE DATABASE
postgres=# SELECT datname, daticulocale FROM pg_database
          WHERE datname=' icudb1' ;
 datname | daticulocale
-----+-----
 icudb1  | en
(1 row)
```

The psql command outputs ICU Locale and Locale Provider in the output of the \l command.

Example 30 List databases

```
postgres=> \l
```

List of databases						
Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider...
postgres	postgres	UTF8	en_US.utf8	en_US.utf8	en	icu
template0	postgres	UTF8	en_US.utf8	en_US.utf8	en	icu
template1	postgres	UTF8	en_US.utf8	en_US.utf8	en	icu

(3 rows)

3.1.7. Statistics Information

In previous versions, the stats collector process received UDP protocol packets and periodically wrote them to temporary files. PostgreSQL 15 now stores statistics in shared memory. Information on the shared memory area for statistics can be obtained by searching the `pg_shmem_allocations` view and `pg_backend_memory_contexts` view. The stats collector process has been deprecated as a result of this specification change.

Example 31 Shared memory for Statistics information

```
postgres=# SELECT * FROM pg_shmem_allocations WHERE name LIKE 'Shared%Stats' ;
```

name	off	size	allocated_size
Shared Memory Stats	147331584	263312	263424

(1 row)

```
postgres=# SELECT name, total_bytes, free_bytes, used_bytes FROM
pg_backend_memory_contexts WHERE name LIKE 'PgStat%' ;
```

name	total_bytes	free_bytes	used_bytes
PgStat Shared Ref Hash	7224	680	6544
PgStat Shared Ref	4096	1432	2664
PgStat Pending	4096	2168	1928

(3 rows)

3.1.8. GiST Index

The following enhancements have been implemented to the GiST index:

□ Index statistics

Access to the SP-GiST index is now counted in the `pg_stat_*` views.

□ BOOL type

It is now possible to create an index for a Boolean type.

Example 32 GiST index creation for BOOK type

```
postgres=> CREATE EXTENSION btree_gist ;
CREATE EXTENSION
postgres=> CREATE TABLE gist1(c1 INT, c2 BOOL, c3 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE INDEX idx1_gist1 ON gist1 USING gist (c2) ;
CREATE INDEX
```

3.1.9. Wait Events

The following wait events have been added.

Table 14 Added Wait Events

Wait Event name	Type	Description
ArchiveCleanupCommand	IPC	Waiting for the <code>archive_cleanup_command</code> command to complete
ArchiveCommand	IPC	Waiting for the <code>archive_command</code> command to complete
BaseBackupSync	IO	Waiting for storage synchronization of base backup
BaseBackupWrite	IO	Waiting for base backup write
RecoveryEndCommand	IPC	Waiting for the <code>recovery_end_command</code> command to complete
RestoreCommand	IPC	Waiting for the <code>restore_command</code> command to complete
VersionFileWrite	IO	Waiting for the version file to be written while creating a database.
VacuumTruncate	Timeout	Waiting to acquire an exclusive lock to truncate off any empty pages

The wait events PgStatMain, LogicalChangesRead, LogicalChangesWrite, LogicalSubxactRead, and LogicalSubxactWrite have been removed.

3.1.10. Role

The predefined role pg_checkpoint has been added. This role holder can execute the CHECKPOINT statement.

Example 33 Grant pg_checkpoint role

```
postgres=# CREATE USER check1 PASSWORD 'check1' ;
CREATE ROLE
postgres=# \connect postgres check1
You are now connected to database "postgres" as user "check1".
postgres=> CHECKPOINT ;
ERROR: must be superuser or have privileges of pg_checkpoint to do CHECKPOINT
postgres=> \connect postgres postgres
You are now connected to database "postgres" as user "postgres".
postgres=# GRANT pg_checkpoint TO check1 ;
GRANT ROLE
postgres=# \connect postgres check1
You are now connected to database "postgres" as user "check1".
postgres=> CHECKPOINT ;
CHECKPOINT
```

3.1.11. Libpq

The following enhancements have been implemented in the Libpq API:

- Environment variable PG_COLORS

It is now possible to specify 'note' in the color specification item of the terminal output.

- PQsslAttribute

The PQsslAttribute(NULL, "library") function can now be executed to obtain the name of the library in use.

□ PQcancel

The connection string `tcp_user_timeout` and `keepalives` are now available for executing the PQcancel API.

□ Allow root-owned SSL private keys in libpq

Libpq applies the same private key file ownership and permission checks as used on the backend. The private key can be owned by either the current user or the root

3.1.12. Custom WAL resource manager

The custom WAL resource managers can now be registered for the `shared_preload_libraries` parameter. The `pg_get_wal_resource_managers` function has been added to check the current WAL resource managers.

Example 34 Check WAL resource manager

```
postgres=# SELECT * FROM pg_get_wal_resource_managers() ;
 rm_id |      rm_name      | rm_builtin
-----+-----+-----
    0 | XLOG              | t
    1 | Transaction       | t
    2 | Storage           | t
    3 | CLOG              | t
...
```

3.1.13. Hook

The following hooks have been added.

□ String object access

Added an access hook for string objects. Executed when accessing the GUC string.

Example 35 Hook definition (src/include/catalog/objectaccess.h)

```
typedef void (*object_access_hook_type_str) (ObjectAccessType access,  
                                             Oid classId,  
                                             const char *objectStr,  
                                             int subId,  
                                             void *arg);  
  
extern PGDLLIMPORT object_access_hook_type_str object_access_hook_str;
```

□ Shared memory request

It must be specified from within the `_PG_init` function when the extension module gets the shared memory.

Example 36 Hook definition (src/include/miscadmin.h)

```
typedef void (*shmem_request_hook_type) (void);  
extern PGDLLIMPORT shmem_request_hook_type shmem_request_hook;  
extern PGDLLIMPORT bool process_shmem_requests_in_progress;
```

□ Object access hook

Execution of the `ALTER TABLE SET {LOGGED, UNLOGGED, ACCESS METHOD}` statement now executes an object access hook.

3.1.14. Custom scan providers

The custom scan providers can now choose to support projection. In previous versions, all custom scan providers needed projection support. Control is performed by the `CUSTOMPATH_SUPPORT_PROJECTION` macro.

3.1.15 Build target

The target name of Makefile to be specified when building from source code has been added. The following targets do not build or install documentation.

- world-bin
- install-world-bin



This specification will also be backported to older versions.

3.1.16. Dynamic Shared Memory

Dynamic Shared Memory (DSM) is now also available in single user mode.

3.2. SQL Statement

This section explains new features related to SQL statements.

3.2.1. NUMERIC data type

Negative numbers can now be specified for NUMERIC type scales. If a negative number is specified, it will be rounded to the specified digit. Also, the range of values that can be specified for the scale has been extended from -1,000 to 1,000.

Example 37 Negative number for NUMERIC

```
postgres=> CREATE TABLE type1( c1 NUMERIC(5, -2) ) ;
CREATE TABLE
postgres=> INSERT INTO type1 VALUES (12345.678) ;
INSERT 0 1
postgres=> INSERT INTO type1 VALUES (1234567.89) ;
INSERT 0 1
postgres=> INSERT INTO type1 VALUES (12345678.901) ;
ERROR:  numeric field overflow
DETAIL:  A field with precision 5, scale -2 must round to an absolute value
less than 10^7.
postgres=> INSERT INTO type1 VALUES (-1234567.89) ;
INSERT 0 1
postgres=> SELECT * FROM type1 ;
  c1
-----
 12300
1234600
-1234600
(3 rows)
```

3.2.2. ALTER DATABASE

The syntax for tracking the version of a collation has been added to the ALTER DATABASE statement.

Syntax

```
ALTER DATABASE database_name REFRESH COLLATION VERSION
```

Added a new function `pg_database_collation_actual_version` to get the collation version and a `datcollversion` column for the version in the `pg_database` view. If it detects that the collation version of the database is different from the operating system, a warning will be output when connecting to the corresponding database. Below is an example of migration from Red Hat Enterprise Linux 7.8 to Red Hat Enterprise Linux 8.6.

Example 38 Refresh collation version

```
$ psql demodb
WARNING: database "demodb" has a collation version mismatch
DETAIL:  The database was created using collation version 58.0.6.50, but the
operating system provides version 153.80.
HINT:  Rebuild all objects in this database that use the default collation and
run ALTER DATABASE demodb REFRESH COLLATION VERSION, or build PostgreSQL with
the right library version.
psql (15.0)
Type "help" for help.

demodb=# ALTER DATABASE demodb REFRESH COLLATION VERSION ;
NOTICE:  changing version from 58.0.6.50 to 153.80
ALTER DATABASE
```

3.2.3. ALTER TABLE

The SET ACCESS METHOD clause can now be specified in ALTER TABLE statements that change the access method for a table; the SET ACCESS METHOD clause can also be specified in ALTER MATERIALIZED VIEW statements.

Syntax

- ALTER TABLE *table_name* SET ACCESS METHOD *method_name*
- ALTER MATERIALIZED VIEW *view_name* SET ACCESS METHOD *method_name*

Example 39 Execute ALTER TABLE statement

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

A `table_rewrite` trigger is fired when you execute an ALTER TABLE SET ACCESS METHOD statement on a materialized view.

3.2.4. COPY

The following enhancements have been added to the COPY statement.

□ COPY TO statement

The format 'text' and header 'true' can now be used simultaneously in the COPY TO statement options. At the same time, the `file_fdw` extension can now enable headers in text format.

Example 40 Text Format and Header

```
postgres=# CREATE EXTENSION file_fdw ;  
CREATE EXTENSION  
postgres=# CREATE SERVER textsvr FOREIGN DATA WRAPPER file_fdw ;  
CREATE SERVER  
postgres=# CREATE FOREIGN TABLE text1(c1 INT, c2 VARCHAR(10)) SERVER textsvr  
OPTIONS (filename '/tmp/data.txt', format 'text', header 'true') ;  
CREATE FOREIGN TABLE  
postgres=# COPY data1 TO '/home/postgres/data1.txt' (FORMAT text, HEADER true) ;  
COPY 100  
postgres=# \! cat data1.txt  
c1      c2  
1       data1  
...
```

□ COPY FROM statement

The HEADER option can now be specified as 'match'. When this option is specified, COPY FROM checks that the header row and column names match when the COPY FROM statement is executed. In the example below, the first row of the CSV file and the column names in the table are different, resulting in an error.

Example 41 Header check

```
postgres=# \! cat /tmp/csvlog1.csv
c1, c3
1, data1
2, data2
3, data3
postgres=# COPY csvlog1 FROM '/tmp/csvlog1.csv' WITH
           (format 'csv', header 'match') ;
ERROR:  column name mismatch in header line field 2: got "c3", expected "c2"
CONTEXT: COPY csvlog1, line 1: "c1,c3"
```

3.2.5. CLUSTER

The CLUSTER statements can now be executed on partitioned tables.

Example 42 Clustering of partition tables

```
postgres=> CREATE TABLE part1 (c1 INT PRIMARY KEY, c2 VARCHAR(10))
           PARTITION BY LIST(c1) ;
CREATE TABLE
postgres=> CLUSTER part1 USING part1_pkey ;
CLUSTER
```

3.2.6. CREATE DATABASE

The following enhancements have been implemented for the CREATE DATABASE statement

□ ICU Locale

The CREATE DATABASE statement now includes a LOCALE_PROVIDER clause to specify the locale provider, an ICU_LOCALE clause to specify the ICU locale, and a COLLATION_VERSION clause to specify the collation version.

Example 43 COLLATION_VERSION の指定

```
postgres=# CREATE DATABASE icudb1 LOCALE_PROVIDER=icu ICU_LOCALE=en  
          COLLATION_VERSION=' 58.0.6.50' TEMPLATE=template0 ;  
CREATE DATABASE
```

□ STRATEGY clause

The STRATEGY clause has been added to the CREATE DATABASE statement to specify how the database should be created; specifying WAL_LOG in the STRATEGY clause will cause the new database to be copied from the template block by block, with information for each block output to WAL. This is an effective method when the template is small and is the default value. Specify FILE_COPY to perform the behavior of the previous version.

Example 44 Specify STRATEGY clause

```
postgres=# CREATE DATABASE strategydb1 STRATEGY WAL_LOG ;  
CREATE DATABASE
```

The option --strategy, which corresponds to the STRATEGY clause, has also been added to the createdb command.

□ OID clause

The object ID (OID) of the database to be created can now be specified; the OID can be any free number greater than 16384.

Example 45 Specify OID clause

```
postgres=# CREATE DATABASE oiddb1 OID=17000 ;  
CREATE DATABASE
```

3.2.7. CREATE TABLE

Column lists can now be specified in the ON DELETE SET NULL and SET DEFAULT clauses of foreign key definitions in CREATE TABLE and ALTER TABLE statements. In the previous version, it was not possible to specify columns. The confdelsetcols column has been added to the pg_constraint catalog.

Example 46 ON Specify DELETE SET NULL clause

```
postgres=> CREATE TABLE phtable1 (  
    tid INT,  
    id INT,  
    name1 VARCHAR(10), PRIMARY KEY (tid, id)) ;  
CREATE TABLE  
postgres=> CREATE TABLE fhtable1 (  
    tid INT,  
    id INT,  
    fk_id_del_set_null INT,  
    fk_id_del_set_default INT DEFAULT 0,  
    FOREIGN KEY (tid, fk_id_del_set_null) REFERENCES phtable1  
        ON DELETE SET NULL (fk_id_del_set_null),  
    FOREIGN KEY (tid, fk_id_del_set_default) REFERENCES phtable1  
        ON DELETE SET DEFAULT (fk_id_del_set_default)  
    ) ;  
CREATE TABLE
```

3.2.8. CREATE UNIQUE INDEX

It is now possible to specify NULLS DISTINCT and NULLS NOT DISTINCT for the unique index attribute. Multiple null values can be stored in a unique index with the NULLS DISTINCT clause. If nothing is specified, the NULLS DISTINCT specification is the default. An index with the NULLS NOT DISTINCT clause cannot contain multiple null values. An `indnullsnotdistinct` column has been added to the `pg_index` catalog to indicate this attribute.

Syntax

- CREATE UNIQUE INDEX *index_name* ON ... NULLS [NOT] DISTINCT
- CONSTRAINT *constraint_name* NULLS [NOT] DISTINCT

Example 47 Create UNIQUE INDEX with NULLS DISTINCT clause

```
postgres=> CREATE TABLE data1(c1 INT, c2 INT, c3 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE UNIQUE INDEX idx1_data1 ON data1(c1) NULLS DISTINCT ;
CREATE INDEX
postgres=> CREATE UNIQUE INDEX idx2_data1 ON data1(c2) NULLS NOT DISTINCT ;
CREATE INDEX
postgres=> INSERT INTO data1 VALUES (1, 1, 'data1') ;
INSERT 0 1
postgres=> INSERT INTO data1 VALUES (2, NULL, 'data2') ;
INSERT 0 1
postgres=> INSERT INTO data1 VALUES (NULL, 3, 'data3') ;
INSERT 0 1
postgres=> INSERT INTO data1 VALUES (NULL, NULL, 'data4') ;
ERROR:  duplicate key value violates unique constraint "idx2_data1"
DETAIL:  Key (c2)=(null) already exists.
postgres=> INSERT INTO data1 VALUES (NULL, 5, 'data5') ;
INSERT 0 1
```

Similar specifications can be made for the unique constraints of the CREATE TABLE and ALTER TABLE statements.

Example 48 UNIQUE constraint

```
postgres=> CREATE TABLE const1(c1 INT UNIQUE NULLS NOT DISTINCT, c2 INT) ;
CREATE TABLE
postgres=> ALTER TABLE const1 ADD CONSTRAINT uniq1 UNIQUE NULLS DISTINCT (c2) ;
ALTER TABLE
```

3.2.9. CREATE SEQUENCE

The UNLOGGED clause can now be specified in the CREATE SEQUENCE statement to suppress WAL output. The sequence created by the CREATE UNLOGGED SEQUENCE statement has the repersistence column in the pg_class catalog as 'u'. The attributes of the sequence can be changed with the ALTER SEQUENCE SET LOGGED statement or the ALTER SEQUENCE SET UNLOGGED statement.

Example 49 Create UNLOGGED SEQUENCE

```

postgres=> CREATE UNLOGGED SEQUENCE seq1 ;
CREATE SEQUENCE
postgres=> \d seq1
                Unlogged sequence "public.seq1"
  Type | Start | Minimum |      Maximum      | Increment | Cycles? | Cache
-----+-----+-----+-----+-----+-----+-----
 bigint |      1 |      1 | 9223372036854775807 |          1 | no      |      1

postgres=> SELECT relname, relpersistence FROM pg_class WHERE relname = 'seq1' ;
 relname | relpersistence
-----+-----
 seq1    | u
(1 row)

postgres=> ALTER SEQUENCE seq1 SET LOGGED ;
ALTER SEQUENCE

```

If the instance crashes, UNLOGGED SEQUENCE will be reset to its default value.

3.2.10. CREATE VIEW

The `security_invoker` option has been added to the `CREATE VIEW` statement. By default, access to the table from which the view is based is enforced with the privileges of the view owner. When the `security_invoker` option is set to `True` for a view, access to the table is subject to the privileges of the user accessing the view. This option can also be specified in the `ALTER VIEW` statement.

Syntax

- `CREATE VIEW view_name WITH (security_invoker = true|false) AS sql_clause`
- `ALTER VIEW view_name SET (security_invoker = true|false)`
- `ALTER VIEW view_name RESET (security_invoker)`

Example 50 Security_invoker option

```

postgres=> CREATE VIEW view1 WITH(security_invoker) AS SELECT * FROM data1 ;
CREATE VIEW

```

3.2.11. EXPLAIN

I/O time for temporary files is now output when BUFFERS and VERBOSE are specified in JSON format.

Example 51 Output of temporary file I/O time

```
postgres=# SET track_io_timing = on ;
SET
postgres=# EXPLAIN (ANALYZE, BUFFERS, VERBOSE, FORMAT JSON)
           SELECT * FROM data1 ;
           QUERY PLAN
-----
 [
 ...
           "I/O Write Time": 0.000,      +
           "Temp I/O Read Time": 0.000, +
           "Temp I/O Write Time": 0.000 +
 }
 ...
```

3.2.12. GRANT

Roles can now be granted permission to change specific parameters through the ALTER SYSTEM statement.

Syntax

```
GRANT {ALTER SYSTEM | SET} ON PARAMETER parameter_name TO role_spec
REVOKE {ALTER SYSTEM | SET} ON PARAMETER parameter_name FROM role_spec
```

Example 52 Execute GRANT ALTER SYSTEM statement

```
postgres=# GRANT ALTER SYSTEM ON PARAMETER log_statement TO demo ;
GRANT
postgres=# SELECT * FROM pg_parameter_acl ;
-[ RECORD 1 ]-----
oid      | 16385
parname  | log_statement
paracl   | {postgres=sA/postgres,demo=A/postgres}

postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> ALTER SYSTEM SET log_statement = 'all' ;
ALTER SYSTEM
```

3.2.13. MERGE

The MERGE statement that simultaneously executes INSERT / DELETE / UPDATE statements on a table from a join condition is now supported. Tuples matching the join condition can be specified in the WHEN MATCHED clause to update (UPDATE) or delete (DELETE) or do nothing (DO NOTHING). Additional conditions may be specified. The behavior of tuples that do not match the condition is specified in the WHEN NOT MATCHED clause.

Syntax

```
MERGE INTO target_table [AS alias]
USING source_table [AS alias]
ON join_clause
WHEN MATCHED [AND condition] THEN
    UPDATE SET ... | DELETE | DO NOTHING
WHEN NOT MATCHED
    INSERT VALUES ... | DO NOTHING
```

Example 53 Execute MERGE statement

```
postgres=> MERGE INTO dst1 AS d USING src1 AS s ON d.c1 = s.c1
           WHEN MATCHED THEN
             UPDATE SET c2 = s.c2
           WHEN NOT MATCHED THEN
             INSERT VALUES (s.c1, s.c2) ;
MERGE 2
postgres=> MERGE INTO dst1 AS d USING src1 AS s ON d.c1 = s.c1
           WHEN MATCHED AND s.c1 < 1000 THEN
             DELETE
           WHEN NOT MATCHED THEN
             DO NOTHING ;
MERGE 2
```

The MERGE statement cannot be used in the WITH clause or at the same time as the COPY statement.

Example 54 Execute MERGE statement and WITH clause

```
postgres=> WITH data1 AS (
           MERGE INTO dst1 USING src1 ON (true)
           WHEN MATCHED THEN DELETE
         ) SELECT * FROM data1 ;
ERROR:  MERGE not supported in WITH query
LINE 1: WITH data1 AS (
           ^
```

3.2.14. VACUUM

The execution log of the VACUUM VERBOSE statement has changed significantly. Average read rate, buffer usage, WAL output information, etc. have been added. Details on how `pg_class.relfrozensid` and `pg_class.relminmxid` have progressed are also now reported.

Example 55 Execute VACUUM VERBOSE statement

```
postgres=> VACUUM VERBOSE data1 ;
INFO:  vacuuming "postgres.public.data1"
INFO:  finished vacuuming "postgres.public.data1": index scans: 1
pages: 0 removed, 5406 remain, 5406 scanned (100.00% of total)
tuples: 500000 removed, 500000 remain, 0 are dead but not yet removable
removable cutoff: 757, which was 0 XIDs old when operation ended
...
avg read rate: 0.000 MB/s, avg write rate: 0.068 MB/s
buffer usage: 18975 hits, 0 misses, 1 dirtied
WAL usage: 18953 records, 0 full page images, 4007392 bytes
system usage: CPU: user: 0.11 s, system: 0.00 s, elapsed: 0.11 s
INFO:  vacuuming "postgres.pg_toast.pg_toast_32787"
...
```

3.2.15. Functions

The following functions have been added/extended.

□ MAX / MIN

MAX function and MIN function for `xid8` type have been implemented.

□ Replication related functions

The following function to get replication information has been added. These functions can be executed by a user with the `SUPERUSER` attribute or the `pg_monitor` role.

Table 15 Added functions

Function name	Description
<code>pg_ls_logicalmapdir</code>	Returns information in the <code>pg_logical/mappings</code> directory
<code>pg_ls_logicalsnapdir</code>	Returns information in the <code>pg_logical/snapshots</code> directory
<code>pg_ls_replslotdir</code>	Returns information in the <code>pg_replslot/SLOTNAME</code> directory

Example 56 Get Logical Replication information

```
postgres=# SELECT pg_create_physical_replication_slot('slot1') ;
pg_create_physical_replication_slot
-----
(slot1,)
(1 row)

postgres=# SELECT * FROM pg_ls_replslotdir('slot1') ;
 name | size |      modification
-----+-----+-----
state |  200 | 2022-10-14 09:52:04+09
(1 row)

postgres=# SELECT * FROM pg_ls_logicalsnapdir() ;
 name      | size |      modification
-----+-----+-----
0-3005558.snap |  128 | 2022-10-14 00:01:29+09
(1 row)

postgres=# SELECT * FROM pg_ls_logicalmapdir() ;
 name | size | modification
-----+-----+-----
(0 rows)
```

Internally, it uses the `READ_REPLICATION_SLOT` command to read the replication slot information.

□ `Pg_log_backend_memory_contexts`

The check for `SUPERUSER` permission has been removed, and it can now be used by general users by granting permission using the `GRANT` statement.

□ Regular expression functions

The `regexp_count`, `regexp_instr`, and `regexp_substr` have been added as functions that handle regular expressions. Parameters that can be specified have been added to the `regexp_replace` function.

Table 16 Added/changed functions

Function name	Description
regexp_count	Returns the number of times the pattern is matched.
regexp_instr	Returns the location that matches the pattern.
regexp_like	Returns whether it matches the pattern.
regexp_replace	Replace the part that matches the pattern. Added start and flags options.
regexp_substr	Returns the part that matches the pattern.

Syntax

- integer REGEXP_COUNT (string text, pattern text [, start integer [, flags text]])
- integer REGEXP_INSTR (string text, pattern text [, start integer [, N integer [, endpos integer [, flags text [, subexpr integer]]]]])
- boolean REGEXP_LIKE (string text, pattern text [, flags text])
- text REGEXP_REPLACE (string text, pattern text, replace text start integer, N integer [, flags text])
- text REGEXP_SUBSTR (string text, pattern text [, start integer [, N integer [, flags text [, subexpr integer]]]])

Example 57 Execution of regular expression functions

```
postgres=> SELECT regexp_count(' ABCABC', ' Abc', 1, 'i') ;
-[ RECORD 1 ]+--
regexp_count | 2

postgres=> SELECT regexp_like(' PostgreSQL', ' Postgre(s|S)QL') ;
-[ RECORD 1 ]--
regexp_like | t

postgres=> SELECT regexp_instr(' ABCDEFGHT', ' D.F') ;
-[ RECORD 1 ]+--
regexp_instr | 4

postgres=> SELECT regexp_replace(' PostgreSQL', 'a|e|i|o|u', 'X', 1, 2) ;
-[ RECORD 1 ]--+-----
regexp_replace | PostgrXSQL

postgres=> SELECT regexp_substr(' ABCDEFGHT', ' D.F', 2) ;
-[ RECORD 1 ]-+-----
regexp_substr | DEF
```

□ Unnest

The MULTIRANGE type can now be specified for UNNEST functions.

Syntax

```
setof anyrange UNNEST(anymultirange)
```

Example 58 Execution of the unnest function

```
postgres=> SELECT unnest(int4multirange(int4range('5', '6'), int4range('1',
'2')));
unnest
-----
[1, 2)
[5, 6)
(2 rows)
```

□ Pg_size_bytes / pg_size_pretty

It now supports up to petabytes.

Example 59 Execution of the pg_size_pretty function

```
postgres=> SELECT pg_size_pretty(pg_size_bytes(' 10.5 PB')) ;
 pg_size_pretty
-----
 11 PB
(1 row)
```

□ Starts_with

The starts_with function can now use BTREE indexes. At the same time, the ^@ operator will do the same.

Example 60 Execution of the starts_with function

```
postgres=> EXPLAIN SELECT * FROM data1 WHERE c2 ^@ '999' ;
                QUERY PLAN
-----
Index Scan using idx2_data1 on data1 (cost=0.42..1337.98 rows=100 width=12)
  Index Cond: ((c2 >= '999'::text) AND (c2 < '99:'::text))
  Filter: (c2 ^@ '999'::text)
(3 rows)

postgres=> EXPLAIN SELECT * FROM data1 WHERE starts_with(c2, '999') ;
                QUERY PLAN
-----
Index Scan using idx2_data1 on data1 (cost=0.42..1337.98 rows=100 width=12)
  Index Cond: ((c2 >= '999'::text) AND (c2 < '99:'::text))
  Filter: starts_with(c2, '999'::text)
(3 rows)
```

□ Pg_settings_get_flags

Attribute information of GUC is returned as an array of text. If a non-existent GUC name is specified, NULL is returned.

Example 61 Execution of the pg_settings_get_flags function

```
postgres=> SELECT * FROM pg_settings_get_flags('session_authorization') ;
           pg_settings_get_flags
```

```
-----
{NO_RESET_ALL, NO_SHOW_ALL, NOT_IN_SAMPLE}
(1 row)
```

```
postgres=> SELECT * FROM pg_settings_get_flags('bad_guc_name') ;
           pg_settings_get_flags
```

```
-----
null
(1 row)
```

- Time specified character string

OF, TZH and TZM can now be used in lowercase.

Example 62 Specify 'of', 'tzh', 'tzm'

```
postgres=> SELECT to_char(current_date, 'of / tzh / tzm') ;
           to_char
```

```
-----
+09 / +09 / 00
(1 row)
```

- Poly_distance

The poly_distance function has been implemented. This function existed in previous versions, but when executed it returned an ERRCODE_FEATURE_NOT_SUPPORTED error.

Example 63 Execute the poly_distance function

```
postgres=> SELECT
poly_distance('((10, 10), (20, 20))'::polygon, '(30, 30), (40, 40)'::polygon) ;
           poly_distance
```

```
-----
14.142135623730951
(1 row)
```

3.3. Configuration parameters

In PostgreSQL 15 the following parameters have been changed.

3.3.1. Added parameters

The following parameters have been added.

Table 17 Added parameters

Parameter name	Description (context)	Default value
allow_in_place_tablespaces	Developer parameters to create a tablespace in the pg_tblspc directory (superuser)	off
archive_library	WAL archive library name (sighup)	-
log_startup_progress_interval	Time before startup process outputs log if the long-running process occurs (sighup)	10s
recovery_prefetch	Whether WAL prefetching is performed during recovery (sighup)	try
recursive_worktable_factor	Multiplier to determine work table size for recursive queries (user)	10
shared_memory_size	The calculated size of main shared memory (internal)	-
shared_memory_size_in_huge_pages	Number of Huge Pages pages used for shared memory (internal)	-
stats_fetch_consistency	Determines the behavior when cumulative statistics are accessed multiple times (user)	cache
wal_decode_buffer_size	Buffer size for WAL decoding (postmaster)	512kB

□ allow_in_place_tablespaces

Tablespaces can be created in a database cluster by setting this parameter to on and specifying an empty string (") in the LOCATION clause of the CREATE TABLESPACE statement.

Example 64 Create In place tablespace

```
postgres=# SET allow_in_place_tablespaces = on ;
SET
postgres=# CREATE TABLESPACE ts1 LOCATION '' ;
CREATE TABLESPACE
postgres=# \! ls -l data/pg_tblspc/
total 0
drwx-----. 3 postgres 29 Oct 14 14:16 17153
```

3.3.2. Modified Parameters

The following parameters have been changed in terms of setting range and choices.

Table 18 Modified Parameters

Parameter name	Changes
compute_query_id	Regress can now be specified as a configuration value. This is used to facilitate regression testing.
dynamic_shared_memory_type	The default value on Solaris has changed to "sysv".
log_destination	Jsonlog can now be specified as a configuration value to output logs in JSON format. This change will also be backported to PostgreSQL 14.3 and later.
wal_compression	Pglz, lz4, and zstd compression methods can now be specified. The configuration value on is assumed to be pglz.
shared_preload_libraries	Now also processed in single user mode.

□ JSON format log file

If jsonlog is specified for the parameter log_destination, the log file will be output in JSON format. The file extension is .json. A file with a .log extension is also created that outputs only some information.

Example 65 JSON format log file

```

postgres=> SHOW log_destination ;
log_destination
-----
jsonlog
(1 row)
postgres=> \! cat postgresql-2022-10-14_141810.json
{"timestamp":"2022-10-14 14:18:10.657
JST","pid":14876,"session_id":"6348f112.3a1c","line_num":1,"session_start":"20
22-10-14 14:18:10 JST","txid":0,"error_severity":"LOG","message":"ending log
output to stderr","hint":"Future log output will go to log destination
¥"jsonlog¥".","backend_type":"postmaster","query_id":0}
{"timestamp":"2022-10-14 14:18:10.657
JST","pid":14876,"session_id":"6348f112.3a1c","line_num":2,"session_start":"20
22-10-14 14:18:10 JST","txid":0,"error_severity":"LOG","message":"starting
PostgreSQL 15.0 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 8.5.0 20210514
(Red Hat 8.5.0-10), 64-bit","backend_type":"postmaster","query_id":0}

```

3.3.3. Parameters with default values changed

The following parameters have changed default values.

Table 19 Parameters with default values changed

Parameter name	PostgreSQL 14	PostgreSQL 15	Note
hash_mem_multiplier	1.0	2.0	
log_autovacuum_min_duration	-1	10min	
log_checkpoints	off	on	
server_version	14.5	15.0	
server_version_num	140005	150000	

3.3.4. Removed parameter

The following parameter have been removed.

Table 20 Removed parameter

Parameter name	Reason
stats_temp_directory	It was removed due to the shared memory of statistics.

3.3.5. Error when changing parameters

Some extension modules (auto_explain, pg_prewarm, pg_stat_statements, plperl, plpgsql) now generate errors when attempting to set parameters that do not exist after loading the module.

Example 66 Output error

```
postgres=> CREATE FUNCTION getwork() RETURNS TEXT AS $$  
BEGIN  
    RETURN current_setting('work_mem') ;  
END ;  
$$ LANGUAGE plpgsql ;  
CREATE FUNCTION  
postgres=> SET plpgsql.bad_parameter_name TO false ;  
ERROR:  invalid configuration parameter name "plpgsql.bad_parameter_name"  
DETAIL:  "plpgsql" is a reserved prefix.
```

3.4. Utilities

Describes the major enhancements of utility commands.

3.4.1. Configure

Added `--with-zstd` option to support Zstandard compression method. Zstandard 1.4.0 and above are supported.

Example 67 For Zstandard compression

```
$ ./configure --help | grep -i zstd
--with-zstd          build with ZSTD support
ZSTD_CFLAGS C compiler flags for ZSTD, overriding pkg-config
ZSTD_LIBS linker flags for ZSTD, overriding pkg-config
```

3.4.2. Psql

The following enhancements have been implemented in the `psql` command.

□ Environment variable `PSQL_WATCH_PAGER`

The environment variable `PSQL_WATCH_PAGER`, which specifies the pager to be displayed when the `\watch` command is executed, is now available.

□ `\Dconfig` command

The `\dconfig` command can display the parameter settings of the instance. If no option is specified, a list of parameters that have changed from the default values is displayed. If a parameter name is specified, the specific parameter setting value can be checked. Wildcards can be used for parameter names.

Syntax

```
\dconfig [parameter_name]
```


Example 68 Execute the \dconfig command

```
postgres=> \dconfig work_mem
List of configuration parameters
Parameter | Value
-----+-----
work_mem  | 4MB
(1 row)

postgres=> \dconfig log_*connect*
List of configuration parameters
Parameter      | Value
-----+-----
log_connections | off
log_disconnections | off
(2 rows)

postgres=> \dconfig+ port
                List of configuration parameters
Parameter | Value | Type   | Context  | Access privileges
-----+-----+-----+-----+-----
port      | 5432  | integer | postmaster |
(1 row)
```

□ \Getenv command

Added the \getenv command to get environment variables. Specify the psql variable name and environment variable name to store the variable value. Specifying an environment variable name that does not exist does not cause an error.

Syntax

```
\getenv psql_variable_name Environment_variable_name
```

Example 69 Execute the \getenv command

```
postgres=> \getenv home
postgres=> \echo :home
/home/postgres
postgres=> \getenv val BADENV
postgres=> \echo :val
:val
```

□ Comments in SQL statements

Comments with two hyphen (-) in SQL statements are now sent to the backend. Previously it was removed before sending to the backend.

□ Large Object output

Access privileges are now output to the \dl+ command and \lo_list+ command.

Example 70 Output the large object access privilege

```
postgres=> \lo_import 'test.dat'
lo_import 16409
postgres=> GRANT ALL ON LARGE OBJECT 16409 TO PUBLIC ;
GRANT
postgres=> \dl+
          Large objects
 ID  | Owner | Access privileges | Description
-----+-----+-----+-----
 16409 | demo | demo=rw/demo      +|
      |      | =rw/demo          |
(1 row)

postgres=> \lo_list+
          Large objects
 ID  | Owner | Access privileges | Description
-----+-----+-----+-----
 16409 | demo | demo=rw/demo      +|
      |      | =rw/demo          |
(1 row)
```

- Specifying a database name

When specifying a table name with the `\d` command, etc., use "database_name.schema_name.table_name" format is allowed. However, the "database_name" part is valid only for the currently connected database. If other database names are specified, an error message will be displayed.

Example 71 Database name qualification

```
postgres=> CREATE TABLE public.data1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> \d postgres.public.data1
                Table "postgres.data1"
Column |          Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
c1     | integer                |           |          |
c2     | character varying(10) |           |          |

postgres=> \d demodb.public.data1
cross-database references are not implemented: demodb.data1
postgres=>
```

- Output of multiple SQL

PostgreSQL 15 displays all results if the server returns multiple result sets. This behavior can be changed with the "`\set SHOW_ALL_RESULTS off`" command.

3.4.3. Pg_amcheck

The `--install-missing` option has been added. If this option is specified, the `amcheck` extension will be installed automatically if it is not installed.

3.4.4. Pg_basebackup

The following options have been added / extended to the `pg_basebackup` command.

- `--target` option

The `--target` option to specify the output destination has been added. This option uses a new data transfer protocol (`BASE_BACKUP TARGET`). The following formats can be specified Specifying

server as the output destination allows base backups to be output to a server running a PostgreSQL instance. Server-side backups can be performed by a user with the SUPERUSER or pg_write_server_files roles.

Table 21 --Target option settings

Target	Description
blackhole	Nothing is output. Test options
server:/path	Output to the specified directory on the server. Currently, the output format is tar (output to /path in the example on the left).
shell[:DETAIL]	Execute a shell command.

The --wal-method option must be set to "fetch" or "none" to make this specification. When "shell" is specified in the --target option, it is necessary to set the Contrib module basebackup_to_shell.

□ --Compress option

The --compress option can now specify the compression method and compression level, whereas the previous version specified the compression level. The compression level is specified by specifying "level=number" after a colon (:); for Zstandard compression, the "workers=number" option for parallel processing can also be set. The same notation as in previous versions is also available for compatibility. If server-side compression (server-gzip) is specified for plain format (-Fp), only the transferred data will be compressed and decompressed and stored on the client side. This setting is useful when network bandwidth is small.

Table 22 --Compress option settings

Setting	Description	Note
none	Do not compress	
gzip	Gzip compression on the client if not targeted	
lz4	LZ4 compression on the client if not targeted	
zstd	Zstandard compression on the client if not targeted	
client-gzip	Gzip compression on the client side	
server-gzip	Gzip compression on the server side	
client-lz4	LZ4 compression on the client side	
server-lz4	LZ4 compression on the server side	
client-zstd	Zstandard compression on the client side	
server-zstd	Zstandard compression on the server side	

Example 72 Specify the compression method

```
$ pg_basebackup --compress=none --format=tar -D back.1
$ pg_basebackup --compress=zstd:level=9 --format=tar -D back.2
```

3.4.5. Pg_dump

The following enhancements have been implemented for the `pg_dump` command.

□ --No-table-access-method option

The `--no-table-access-method` option can now be specified to remove table access methods from the output DDL. This option is ignored if the dump file is output in non-text format. The same option can also be specified for the `pg_dumpall` and `pg_restore` commands.

□ Consideration of TOAST data volume

The `pg_dump` command takes into account the table size when running in parallel mode; PostgreSQL 15 now also takes into account the amount of TOAST data.

3.4.6. Pg_recvlogical

The `--two-phase` (or `-t`) option has been added to the `pg_recvlogical` command to support the addition of the `TWO_PHASE` option to the `CREATE_REPLICATION_SLOT` command for the replication protocol.

Example 73 Specify --two-phase option

```
$ pg_recvlogical --create-slot --slot=slot1 --two-phase --dbname=postgres
$ psql
postgres=# SELECT two_phase FROM pg_replication_slots WHERE slot_name='slot1' ;
 two_phase
-----
t
```

3.4.7. Pg_receivewal

The `--compress` option, which specifies how to compress the logs, has changed. Compression method, level, etc. can be specified.

Syntax

```
pg_receivewal --compress=METHOD[:DETAIL]
METHOD = { 'gzip', 'none', 'lz4' }
DETAIL = level=[1-9]
```

Valid values for compression are `gzip`, `lz4` or `none`. An optional compression level (`level`) can be specified. The range of compression levels is 1-9.

Example 74 Specify the `--compress` option

```
$ pg_receivewal -D wal --compress=lz4:level=9
^Cpg_receivewal: not renaming "000000010000000000000028.lz4.partial", segment
is not complete
$ ls wal
000000010000000000000022.lz4 000000010000000000000026.lz4
000000010000000000000023.lz4 000000010000000000000027.lz4
000000010000000000000024.lz4 000000010000000000000028.lz4.partial
000000010000000000000025.lz4
```

Must be compiled with the `--with-lz4` option to perform LZ4 compression.

3.4.8. Pg_resetwal

The `--oldest-transaction-id` option (short form `-u`) has been added to specify the oldest transaction ID.

3.4.9. Pg_rewind

Added the `--config-file` option to specify the path to the server configuration file (`postgresql.conf`). This option is useful when the configuration file is outside the data directory.

3.4.10. Pg_upgrade

The following enhancements have been implemented for the `pg_upgrade` command

□ --No-sync option

The --no-sync option (short form -N) has been added. When this option is specified, it does not wait for the completion of file writing. This is not to be used in a production environment.

□ Temporary Directories

Temporary files created during the upgrade will be created in the pg_upgrade_output.d directory of the new database cluster.

3.4.11. Pg_waldump

The following enhancements have been added to the pg_waldump command

□ Options

The following options have been added Multiple --rmgr options can now be specified.

Table 23 Added options

Option name	Short option	Description
--block=N	-B	Show the block of relations specified by the --relation option
--fork=N	-F	Show only forks with matching numbers
--relation=tblspc/db/rel	-R	Show only information about the relevant relationship
--fullpage	-w	Show only Full page write information

□ Signal

When the SIGINT signal is received and the process ends, summary information is output.

□ Addition of output information

The remote_apply information of transaction commit and the replication origin information of PREPARE TRANSACTION are added to the command output.



Example 75 Command output

```
$ pg_waldump data/pg_wal/000000100000000000000008 | grep origin
rmgr: Transaction len (rec/tot):      117/      117, tx:          775, lsn:
0/080025F0, prev 0/080025C0, desc: COMMIT 2022-10-14 10:01:46.390500 JST; inval
msgs: catcache 57 catcache 56 catcache 66; origin: node 2, lsn 0/0, at 2022-10-
14 10:01:46.382305 JST
rmgr: Transaction len (rec/tot):      85/       85, tx:          776, lsn:
0/08002770, prev 0/08002740, desc: COMMIT 2022-10-14 10:01:46.391231 JST; inval
msgs: catcache 66; origin: node 2, lsn 0/0, at 2022-10-14 10:01:46.382305 JST
...
```


3.5. Contrib modules

Describes new features related to the Contrib modules.

3.5.1. Amcheck

A sequence checking is now supported.

Example 76 Sequence checking

```
postgres=# CREATE SEQUENCE seq1 ;
CREATE SEQUENCE
postgres=# SELECT verify_heapam(relation=>'seq1') ;
verify_heapam
-----
(0 rows)
```

3.5.2. Basebackup_to_shell

This module is used when "shell" is specified as the target of the base backup. When using this module, specify `basebackup_to_shell` in the parameter `shared_preload_libraries` or `local_preload_libraries`. The following parameters can be specified:

Table 24 Available parameters

Parameter name	Description
<code>basebackup_to_shell.command</code>	Command path to be executed
<code>basebackup_to_shell.required_role</code>	Roles required for execution

3.5.3. File_fdw

The MATCH value can now be specified in the HEADER option of the CREATE FOREIGN TABLE statement. This option checks for a match between the header line and column name in the text file.

Example 77 Header match check

```
postgres=# \! cat /tmp/data1.csv
c1, c3
1, data1
2, data2
postgres=# CREATE EXTENSION file_fdw ;
CREATE EXTENSION
postgres=# CREATE SERVER filesvr1 FOREIGN DATA WRAPPER file_fdw ;
CREATE SERVER
postgres=# CREATE FOREIGN TABLE head1 (c1 int, c2 text) SERVER filesvr1
      OPTIONS (format 'csv', filename '/tmp/data1.csv', delimiter ',',
              header 'match') ;
CREATE FOREIGN TABLE
postgres=# SELECT * FROM head1 ;
ERROR: column name mismatch in header line field 2: got "c3", expected "c2"
CONTEXT: COPY head1, line 1: "c1,c3"
```

3.5.4. Pg_stat_statements

The pg_stat_statements view has additional items that can be monitored.

□ I/O information for temporary files

Read/write times for temporary files can now be monitored. The following columns have been added to the pg_stat_statements view.

Table 25 Added column

Column name	Data type	Description
temp_blk_read_time	double precision	Temporary block read time
temp_blk_write_time	double precision	Temporary block write time

□ Addition of JIT-related information

JIT-related information can now be monitored; the following columns have been added to the pg_stat_statements view.

Table 26 Added column

Column name	Data type	Description
jit_functions	bigint	Number of JIT-compiled function executions
jit_generation_time	double precision	JIT code generation time (milliseconds)
jit_inlining_count	bigint	Number of executions of inlined function
jit_inlining_time	double precision	Execution time of inlined function (milliseconds)
jit_optimization_count	bigint	Optimized statement execution count
jit_optimization_time	double precision	Optimized statement execution time (milliseconds)
jit_emission_count	bigint	Number of times the code was issued
jit_emission_time	double precision	Time (milliseconds) the code was used to issue

3.5.5. Pg_walinspect

The pg_walinspect module is a module for executing processing similar to the pg_waldump command with SQL statements. The following functions are provided. These functions can only be executed by users with the SUPERUSER attribute or the pg_read_server_files role.

Table 27 Provided functions

Function name	Description
pg_get_wal_record_info	Return WAL information for the specified LSN
pg_get_wal_records_info	Return WAL information between specified LSNs
pg_get_wal_records_info_till_end_of_wal	Return WAL information from the specified LSN to the end
pg_get_wal_stats	Return WAL statistics between specified LSNs
pg_get_wal_stats_till_end_of_wal	Return WAL statistics from the specified LSN to the end

Example 78 Execute the pg_get_wal_stats function

```
postgres=# CREATE EXTENSION pg_walinspect ;
CREATE EXTENSION
postgres=# SELECT * FROM pg_get_wal_stats('0/B4BB822'::pg_lsn ,
      '0/B4BB828'::pg_lsn) ;
-[ RECORD 1 ]-----+-----
resource_manager/record_type | XLOG
count                        | 0
count_percentage            | 0
...
```

3.5.6. Postgres_fdw

The following enhancements have been implemented in postgres_fdw:

- Specifying the application name

It is now possible to specify application_name for a remote connection. Change the postgres_fdw.application_name parameter.

Example 79 Specify application name

```
postgres=# SET postgres_fdw.application_name TO 'fdw_name1' ;
SET
postgres=# CREATE SERVER svr5432 FOREIGN DATA WRAPPER postgres_fdw OPTIONS
(host 'remhost1', port '5432', dbname 'postgres') ;
CREATE SERVER
```

The following escape sequences can be specified in the application name.

Table 28 Available escape sequences

Escape sequence	Description	Note
%a	Local application name	
%c	Session ID	
%C	Cluster name (cluster_name)	
%d	Local database name	
%p	Backend process ID	
%u	Local user name	
%%	% character	

□ Parallel_commit option

Added the parallel_commit option to control how remote transaction commits are executed. Setting this option to 'on' allows remote transaction commits in parallel. The default value is off, which causes commits to be executed serially.

Example 80 Specify parallel_commit option

```
postgres=# CREATE SERVER svr5433 FOREIGN DATA WRAPPER postgres_fdw OPTIONS
( host 'remhost1', port '5433', dbname 'postgres', parallel_commit 'true' ) ;
CREATE SERVER
postgres=# SELECT srvoptions FROM pg_foreign_server WHERE srvname='svr5433' ;
                srvoptions
-----
 {host=localhost,port=5433,dbname=postgres,parallel_commit=true}
(1 row)
```

□ CASE expression

CASE expressions are now pushed to remote instances.

Example 81 Pushed CASE expression

```
postgres=> EXPLAIN (ANALYZE, VERBOSE) SELECT COUNT(*) FROM remote1 WHERE
           CASE WHEN c1 > 100 THEN c1 END < 100 ;
           QUERY PLAN
-----
Foreign Scan (cost=102.84..164.07 rows=1 width=8) (actual time=1.364..1.365
rows=1 loops=1)
  Output: (count(*))
  Relations: Aggregate on (public.remote1)
  Remote SQL: SELECT count(*) FROM public.remote1 WHERE (((CASE WHEN (c1 >
100::numeric) THEN c1 ELSE NULL::numeric END) < 100::numeric))
  Planning Time: 0.077 ms
  Execution Time: 1.620 ms
(6 rows)
```

3.5.7. Sepgsql

The permissive/enforcing state has been added to the log, with permissive=0 or permissive=1 output at the end of the line.

URL list

The following websites are references to create this material.

- Release Notes
<https://www.postgresql.org/docs/15/release.html>
- Commitfests
<https://commitfest.postgresql.org/>
- PostgreSQL 15 Manual
<https://www.postgresql.org/docs/15/index.html>
- Git
`git://git.postgresql.org/git/postgresql.git`
- GitHub
<https://github.com/postgres/postgres>
- Announce of PostgreSQL 15
<https://www.postgresql.org/about/news/postgresql-15-released-2526/>
- Postgres Professional
<https://habr.com/ru/company/postgrespro/blog/541252/>
- PostgreSQL 15 Open Items
https://wiki.postgresql.org/wiki/PostgreSQL_15_Open_Items
- Qiita (Nuko@Yokohama)
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-15.html>
- SQL Notes
<https://sql-info.de/postgresql/postgresql-15/articles-about-new-features-in-postgresql-15.html>
- Slack - postgresql-jp
<https://postgresql-jp.slack.com/>

Change history

Change history

Version	Date	Author	Description
0.1	Apr 25, 2022	Noriyoshi Shinoda	Create an internal review version. Reviewers: Tomoo Takahashi Akiko Takeshima (Hewlett Packard Enterprise Japan)
1.0	May 24, 2022	Noriyoshi Shinoda	Modification completed according to PostgreSQL 15 Beta 1.
1.1	Oct 14, 2022	Noriyoshi Shinoda	Fixed for PostgreSQL 15 GA

