Hewlett Packard Enterprise

May 28, 2023

PostgreSQL 16 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 16 Beta 1 (16.0).

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 15 (15.3) and PostgreSQL 16 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 15.3 (for comparison) PostgreSQL 16 Beta 1 (16.0) (May 22, 2023 21:20:39)
Operating System	Red Hat Enterprise Linux 8 Update 5 (x86-64)
Configure options	--with-ssl=openssl --with-python --with-lz4 --with-zstd --with-llvm --with-icu --with-libxml

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 16. 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 query is now available for execution of FULL OUTER JOIN syntax, RIGHT OUTER JOIN syntax, string_agg function and array_agg function.

□ Extended Statistics

Add the pg_stat_io view to get detailed I/O statistics. In addition, the pg_stat_all_tables view and the pg_stat_all_indexes view have added a column that shows the last access time for an object.

□ Improved batch processing performance

It is now possible to batch insert multiple records with the COPY statement for a foreign table using the postgres_fdw module.

2.2. Improved reliability

PostgreSQL 16 implements the following enhancements to improve reliability.

□ Improvements in logical Replication

Logical replication is one of the major enhancements in this version. Improved apply performance for long-running transactions, limited bi-directional replication, roles that allow SUBSCRIPTION creation, improved initial data transfer performance, logical decoding on standby instances, and much more.

□ Backup compression

LZ4 compression and Zstandard compression are now available for the output of the pg_dump command.

2.3. Improved maintainability

The following features that can improve operability have been added.

□ Extension of the parameter file

Regular expressions can now be used for some setting items in the pg_hba.conf file and pg_ident.conf file. Also, the pg_hba.conf and pg_ident.conf files can now include external files, just like postgresql.conf.

□ Security

GSSAPI/Kerberos credential delegation is now supported. Add predefined role pg_maintain. A user granted this role can execute the ANALYZE, VACUUM, REINDEX, etc. statements for tables other than the owner. In addition, it is now possible to grant privileges to other roles to execute the above SQL statements for each table. Add pg_use_reserved_connections to reserve reserved sessions for general users.

2.4. Improvements in Programming

The following functions have been added to SQL statements.

□ JSON-related

A function conforming to the SQL standard has been added to generate SQL/JSON. Add IS JSON clause that can check JSON syntax.

□ Subquery

The alias name is now optional if the FROM clause contains a subquery.

□ Literals

Numeric literals can now also contain binary, octal, and hexadecimal numbers. Underscores in numeric strings are supported.

2.5. Preparing for future new features

PostgreSQL 16 is now ready for features that will be provided in future versions.

□ Extension of ICU locales

ICU is now the standard locale provider. Also, adding original rules to the ICU locale is supported,

and the UNICODE locale has been added.

□ Standby instance

Logical decoding is now possible on the standby instance.

2.6. Incompatibility

In PostgreSQL 16, the following specifications have been changed from PostgreSQL 15.

2.6.1. End of support

PostgreSQL 16 changed the supported versions for the following platforms and tools. [9db300c, 495ed0e, 6203583, 8efefa7, 8b878bf, b086a47, 4c15327, 8efefa7, e692727]

Platforms and tools that are no longer supported are listed below.

- HP-UX
- HP/Intel Itanium processor
- M68K, M88K, M32R and SuperH processors
- Microsoft Windows prior to Windows 10
- Microsoft Visual Studio 2013

Changes in supported versions of components required to build PostgreSQL 16 are as follows.

- Bison 2.3 or later
- Flex 2.5.35 or later
- Perl 5.14 or later
- GNU make 3.81 or later

2.6.2. Promotion

Standby database promotion by trigger file creation is no longer supported. The parameter promote_trigger_file has been removed accordingly. Use the pg_ctl promote command or pg_promote function to execute promotion. [cd4329d]

2.6.3. Bootstrap user attribute

The SUPERUSER attribute can no longer be removed from the bootstrap user. [e530be2]

Example 1 Modify attributes of the bootstrap user

```
postgres=# \du
List of roles
-[ RECORD 1 ]--------------------------------------------------------------
Role name  | postgres
Attributes | Superuser, Create role, Create DB, Replication, Bypass RLS
Member of  | {}

postgres=# ALTER USER postgres NOSUPERUSER ;
ERROR:  permission denied to alter role
DETAIL:  The bootstrap user must have the SUPERUSER attribute.
```

2.6.4. WAL Archive

The WAL archive of PostgreSQL 16 has the following incompatibilities: [d627ce3, 35739b8]

□ Parameter settings

The archive_command parameters and the archive_library can no longer be set at the same time. In PostgreSQL 15 archive_library was preferred. If set at the same time, the following error will be output to the log and the WAL archive will not be output.

Example 2 Duplicate error in WAL archive settings

```
$ tail -2 data/log/postgresql-2023-05-25_225124.log
FATAL:  both archive_command and archive_library set
DETAIL:  Only one of archive_command, archive_library may be set.
```

□ Specification change of the WAL archive module

The specification of the callback function of the shared library specified in the parameter archive_module has been changed. An address of type ArchiveModuleState has been added to the parameters of each callback.

Table 4 WAL archive module API

Callback	Modify	Call definition in PostgreSQL 16
Initialize	Add	(*ArchiveStartupCB) (ArchiveModuleState *state)
Check	Modify	(*ArchiveCheckConfiguredCB) (ArchiveModuleState *state)
WAL archive	Modify	(*ArchiveFileCB) (ArchiveModuleState *state, const char *file, const char *path)
Shutdown	Modify	(*ArchiveShutdownCB) (ArchiveModuleState *state)

2.6.5. Extension

Execution of the CREATE OR REPLACE statement on an existing object that does not belong to an extension is prohibited. Previously, an extension script would execute the CREATE OR REPLACE statement, overwriting an existing object if one existed that did not belong to the extension. [b9b21ac]

2.6.6. Datetime string format

Date-time strings combining "epoch" and "infinity" with other date-time fields are prohibited in PostgreSQL 16. Also, ISO 8601 will be strictly enforced for string formatting. [bcc704b, 5b3c595]

Example 3 Datetime expression that causes an error in PostgreSQL 16

```
postgres=> SELECT date '1995-08-06 epoch' ;
ERROR:  invalid input syntax for type date: "1995-08-06 epoch"
LINE 1: SELECT date '1995-08-06 epoch' ;
                    ^

postgres=> SELECT timestamp '1995-08-06 infinity' ;
ERROR:  invalid input syntax for type timestamp: "1995-08-06 infinity"
LINE 1: SELECT timestamp '1995-08-06 infinity';

postgres=> SELECT date '1995-08-06  J J J' ;
ERROR:  invalid input syntax for type date: "1995-08-06  J J J"
LINE 1: SELECT date '1995-08-06  J J J' ;
```

2.6.7. ASCII string conversion

Changed the conversion rules when non-ASCII characters are specified for ASCII-only strings such as parameters application_name and cluster_name. Previously, it was converted in byte units with a question mark (?), but in PostgreSQL 16, it is converted to a hexadecimal string. [45b1a67]

Table 5 ASCII Conversion rule

Before (UTF-8)	PostgreSQL 15	PostgreSQL 16	Note
Abc 漢字	Abc??????	Abc\xe6\xbc\xa2\xe5\xad\x97	

2.6.8. ALTER DEFAULT PRIVILEGES statement

Execution of ALTER DEFAULT PRIVILEGES requires explicit privileges, not membership in a role. The ALTER DEFAULT PRIVILEGES statement included in the following example will succeed in PostgreSQL 15. [48a257d]

Example 4 PostgreSQL 16 behavior

```
postgres=# CREATE USER role1 ;
CREATE ROLE
postgres=# CREATE USER role2 IN ROLE role1 NOINHERIT ;
CREATE ROLE
postgres=# \connect postgres role2
You are now connected to database "postgres" as user "role2".
postgres=> ALTER DEFAULT PRIVILEGES FOR ROLE role1 IN SCHEMA public GRANT SELECT
ON TABLES TO public ;
ERROR:  permission denied to change default privileges
```

2.6.9. ALTER TABLE statement

The unique indexes with the NULLS NOT DISTINCT clause can no longer be specified as primary keys for tables. [d959523]

Example 5 Failure of the ALTER TABLE statement

```
postgres=> CREATE TABLE pktest1(c1 INTEGER, c2 INTEGER) ;
CREATE TABLE
postgres=> CREATE UNIQUE INDEX idx1_pktest1 ON pktest1 (c1) NULLS NOT DISTINCT ;
CREATE INDEX
postgres=> ALTER TABLE pktest1 ADD PRIMARY KEY USING INDEX idx1_pktest1 ;
ERROR:  primary keys cannot use NULLS NOT DISTINCT indexes
```

2.6.10. CREATE RULE statement

In previous versions, it was possible to create an ON SELECT rule for a table and convert it to a view, but PostgreSQL 16 prohibits it. [b23cd18]

2.6.11. CREATE TABLE statement

System columns can no longer be used as FOREIGN KEYs. This change will be backported to the previous version. [f0d65c0]

Example 6 System column foreign key specification error

```
postgres=> CREATE TABLE pktable (c1 INT PRIMARY KEY, c2 OID) ;
CREATE TABLE
postgres=> CREATE TABLE fktable (c1 INT, CONSTRAINT fk1 FOREIGN KEY (tableoid)
               REFERENCES pktable(c2)) ;
ERROR:  system columns cannot be used in foreign keys
```

2.6.12. CREATEROLE attribute

In PostgreSQL 16, it is no longer possible to grant roles without the WITH ADMIN option to other users, even if they have the CREATEROLE attribute. In previous versions, users with the CREATEROLE attribute could perform most changes to other users. The first GRANT statement in the example below succeeds in PostgreSQL 15. [cf5eb37]

Example 7 Failure of the GRANT statement

```
postgres=# CREATE USER useradm1 PASSWORD '<<PASSWORD>>' CREATEROLE ;
CREATE ROLE
postgres=# \connect postgres useradm1
You are now connected to database "postgres" as user "useradm1".
postgres=> CREATE USER monitor1 PASSWORD '<<PASSWORD>>' ;
CREATE ROLE
postgres=> GRANT pg_monitor TO monitor1 ;
ERROR:  permission denied to grant role "pg_monitor"
DETAIL:  Only roles with the ADMIN option on role "pg_monitor" may grant this
role.
postgres=> \connect postgres
You are now connected to database "postgres" as user "postgres".
postgres=# GRANT pg_monitor TO useradm1 WITH ADMIN OPTION ;
GRANT ROLE
postgres=# \connect postgres useradm1
You are now connected to database "postgres" as user "useradm1".
postgres=> GRANT pg_monitor TO monitor1 ;
GRANT ROLE
```

2.6.13. POWER function

In PostgreSQL 16, the accuracy when specifying an integer to the POWER function is improved, and the effective accuracy matches when specifying a non-integer. This fix will not be backported to previous versions. [40c7fcb]

Example 8 PostgreSQL 15 behavior

```
postgres=> SELECT power(0.084738, 16) ;
       power
--------------------
 0.0000000000000000
(1 row)
```

Example 9 PostgreSQL 16 behavior

```
postgres=> SELECT power(0.084738, 16) ;
                power
--------------------------------------
 0.00000000000000000707264604102046
(1 row)
```

2.6.14. RESET statement

The RESET statement for the parameters below will fail. [3853664]

- transaction_read_only
- transaction_deferrable
- seed
- transaction_isolation

Example 10 Failure of the RESET statement

```
postgres=> RESET transaction_read_only ;
ERROR:  parameter "transaction_read_only" cannot be reset
postgres=> RESET transaction_deferrable ;
ERROR:  parameter "transaction_deferrable" cannot be reset
postgres=> RESET seed ;
ERROR:  parameter "seed" cannot be reset
postgres=> BEGIN ;
BEGIN
postgres=*> RESET transaction_isolation ;
ERROR:  parameter "transaction_isolation" cannot be reset
postgres=!> ROLLBACK ;
ROLLBACK
```

2.6.15. Pg_stat_get_backend_idset function

The pg_stat_get_backend_idset function now returns the ID of the backend process. Until the previous version, it returned the order starting from 1. [d7e39d7]

Example 11 PostgreSQL 15 behavior

```
postgres=> SELECT pg_stat_get_backend_idset() ;
 pg_stat_get_backend_idset
---------------------------
                         1
                         2
                         3
                         4
                         5
                         6
(6 rows)
```

Example 12 PostgreSQL 16 behavior

```
postgres=> SELECT pg_stat_get_backend_idset() ;
 pg_stat_get_backend_idset
---------------------------
                         1
                         2
                         4
                       124
                       126
                       127
(6 rows)
```

2.6.16. Pg_walinspect extension

Pg_get_wal_records_info function, pg_get_wal_stats function, and pg_get_wal_block_info function of pg_walinspect extension no longer generate an error even if the end LSN specified is higher than the current LSN. The pg_get_wal_records_info_till_end_of_wal and pg_get_wal_stats_till_end_of_wal functions, which are no longer needed due to this change, have been removed. [5c1b662]

2.6.17. Postmaster

The -T option may send a SIGABRT signal instead of a SIGSTOP signal by setting the send_abort_for_crash parameter. Also, the -n option has been removed. Symbolic links pointing to the postmaster are no longer created. [51b5834, 37e2673]

2.6.18. Psql

The following specifications of the psql command have been changed. [3dfae91, 00beecf]

□ The \df+ meta-command

The \df+ meta-command output column name "Source code" changed to "Internal name".

Example 13 PostgreSQL 16 behavior

```
postgres=> \df+ scale
List of functions
-[ RECORD 1 ]-------+-------------------------------------------------
Schema              | pg_catalog
Name                | scale
Result data type    | integer
Argument data types | numeric
Type                | func
Volatility          | immutable
Parallel            | safe
Owner               | postgres
Security            | invoker
Access privileges   |
Language            | internal
Internal name       | numeric_scale
Description         | number of decimal digits in the fractional part
```

□ The \watch meta-command

Negative numbers and strings specified in the \watch meta command will result in an error. Previous versions assumed 1 second. If 0 is specified, the command will be re-executed without an interval.

Example 14 PostgreSQL 16 behavior

```
postgres=> \watch -1
\watch: incorrect interval value '-1'
postgres=> \watch abc
\watch: incorrect interval value 'abc'
```

2.6.19. Libpq

Removed authorization code with SCM credentials. The backend code was removed in PostgreSQL 9.1, but the libpq code remained. [98ae2c8]

2.6.20. PL/Python

The {installation destination}/lib/pgxs/src/pl/plpython directory is no longer created. [7d5852c]

2.6.21. ECPG

The SQL keywords used in EXEC SQL commands can no longer be declared as typedefs. The following example will result in a syntax error. [83f1c7b]

Example 15 ECPG program with error

```
EXEC SQL BEGIN DECLARE SECTION;
    typedef int start;
EXEC SQL END DECLARE SECTION;
...
EXEC SQL START TRANSACTION;
```

Hewlett Packard
Enterprise

3. New Feature Detail

3.1. Architecture

3.1.1. Modified System catalogs

The following catalogs have been changed. [d540a02, 84ad713, c037471, 3662839, 6566133, e3ce2de, c591300, ae4fdde, 90189ee, 216a784, e0b0142, e056c55, 0c67946]

Table 6 Add system catalogs and views

Catalog/View name	Description
pg_stat_io	Provides detailed I/O statistics for database clusters.

Table 7 System catalogs/views with additional columns

Catalog/View name	Add column	Data type	Description
pg_auth_members	oid	oid	Object ID
	inherit_option	boolean	Allow inheritance
	set_option	boolean	Allow SET ROLE Statement
pg_collation	collicurules	text	ICU custom rule
pg_database	daticurules	text	ICU custom rule
pg_hba_file_rules	rule_number	integer	Rule number
	file_name	text	Configuration file name
pg_ident_file_mappings	map_number	integer	Mapping number
	file_name	text	Configuration file name
pg_prepared_statements	result_types	regtype[]	Types of columns returned by the statement
pg_replication_slots	conflicting	boolean	Conflict with recovery
pg_stat_database_conflicts	confl_active_logicalslot	bigint	Number of logical slot uses that have been canceled
pg_stat_gssapi	credentials_delegated	boolean	Is it a GSSAPI delegated session?
pg_stat_subscription	leader_pid	integer	The ID of the parallel apply reader process

Catalog/View name	Add column	Data type	Description
pg_stat_*_indexes	last_idx_scan	timestamp with time zone	Index access last time
pg_stat_*_tables	last_seq_scan	timestamp with time zone	Sequential access last time
	last_idx_scan	timestamp with time zone	Index access last time
	n_tup_newpage_upd	bigint	Number of rows that successor versions have moved to new heap pages
pg_subscription	suborigin	text	Type of request to send data to PUBLICATION
	subpasswordrequired	boolean	Password required for authentication
	subrunasowner	boolean	Execution by SUBSCRIPTION owner

Table 8 System catalogs/views with modified output

Catalog / View name	Description
pg_attribute	The data type of attndims, attstattarget, attinhcount columns changed from integer to smallint. Also, the order of the columns has changed.
pg_constraint	The data type of the coninhcount column has changed from integer to smallint.
pg_locks	'applytransaction' is now output in the locktype column.
pg_stat_activity	Now standalone backends are output in the backend_type column.
pg_stat_subscription	A tuple is created immediately after the SUBSCRIPTION is created. In previous versions, it was created when the first stat was received.
pg_subscription	Substream column changed from bool type to char type. 'p' is output when applying in parallel.

Details about the add the pg_stat_io views are described below. [a9c70b4, 8aaa04b, ac8d53d, ac8d53d, 0ecb87e, 093e5c5]

Table 9 The pg_stat_io view

Column name	Data type	Description
backend_type	text	Type of backend process. The following values are output • background worker • client backend • walsender • standalone backend • autovacuum worker • autovacuum launcher • background writer • startup • checkpointer etc.
object	text	Type of target relation (relation, temp relation)
context	text	Type of I/O operation (normal, vacuum, bulkread, bulkwrite)
reads	bigint	Number of reads of the size indicated by the io_bytes column
read_time	double precision	Total loading time
writes	bigint	Number of writes of the size indicated in the io_bytes column
write_time	double precision	Total write time
writebacks	bigint	Number of units of size op_bytes which the process requested the kernel write out to storage.
writeback_time	double precision	Total writeback time
extend_time	double precision	Total Extended Time
op_bytes	bigint	Number of bytes for one I/O processing
hits	bigint	Number of cache hits
evictions	bigint	Number of times written out from shared or local buffers so that they can be used for other purposes

Column name	Data type	Description
reuses	bigint	Number of times an I/O operation reused an existing ring buffer outside of shared buffers
fsyncs	bigint	Number of fsync system calls executed
fsync_time	double precision	fsync system call execution time
stats_reset	timestamp with time zone	view reset time

Example 16 Reference to the pg_ident_file_mapping view.

```
postgres=> SELECT backend_type, context, reads, writes FROM pg_stat_io WHERE
          backend_type='autovacuum worker' ;
   backend_type    | context  | reads | writes
-------------------+----------+-------+--------
 autovacuum worker | bulkread |     0 |      0
 autovacuum worker | normal   |   548 |      0
 autovacuum worker | vacuum   |   145 |      0
(3 rows)
```

Parameter track_io_timing should be set to 'on' to output read_time column, write_time column, extend_time column and fsync_time column. Execute the pg_stat_reset_shared function to reset the contents of the view.

Example 17 Reset of the pg_stat_io view

```
postgres=# SELECT pg_stat_reset_shared('io') ;
pg_stat_reset_shared
---------------------


(1 row)
```

3.1.2. Logical Replication Enhancements

The following features have been added to Logical Replication. [216a784, 3662839, 4826759, 1e10d49, ecb6965, c3afe8c, 89e46da, 0fdab27, 5de94a0, 9f2213a, 2666975, c9f7f92]

□ Parallel apply

In previous versions, the large transactions were applied after being saved to a temporary file. PostgreSQL 16 now allows transaction deltas to be applied directly by multiple worker processes. To enable this feature, specify 'parallel' for the option streaming in the CREATE SUBSCRIPTION statement. Along with this, the substream column of the pg_subscription catalog is changed from bool type to char type, and 'p' is output when 'parallel' is specified for the stream option of the CREATE SUBSCRIPTION statement. The maximum number of worker processes a subscription uses is determined by the parameter max_parallel_apply_workers_per_subscription.

Example 18 Setting for parallel apply

```
postgres=# CREATE SUBSCRIPTION sub1 CONNECTION 'port=5432 dbname=postgres'
             PUBLICATION pub1 WITH (streaming=parallel) ;
NOTICE:  created replication slot "sub1" on publisher
CREATE SUBSCRIPTION
postgres=# SELECT subname, substream FROM pg_subscription ;
 subname | substream
---------+-----------
 sub1    | p
(1 row)
```

□ Origin option for SUBSCRIPTION

The CREATE /ALTER SUBSCRIPTION statements now accept the 'origin' option. This parameter specifies the type of change to request to PUBLICATION. A suborigin column has been added to the pg_subscription catalog to store this option information.

Table 10 The origin option setting

Value	Description	Note
none	SUBSCRIPTION requests only non-origin related changes to PUBLICATION	
any	PUBLICATION will send all changes to SUBSCRIPTION regardless of origin	Default value

Setting the origin option to 'none' allows bi-directional replication for the same table with some restrictions.

Example 19 The origin option setting

```
postgres=# CREATE SUBSCRIPTION sub1 CONNECTION 'port=5432 dbname=postgres
user=postgres' PUBLICATION pub1 WITH (origin=none) ;
NOTICE:  created replication slot "sub1" on publisher
CREATE SUBSCRIPTION
postgres=# SELECT subname, suborigin FROM pg_subscription ;
 subname | suborigin
---------+-----------
 sub1    | none
(1 row)
```

□ Run_as_owner option for SUBSCRIPTION

In previous versions of logical replication, SUBSCRIPTION updated the table with the privileges of the SUBSCRIPTION owner. By default, in PostgreSQL 16, apply is done with the privileges of the table owner. Setting the run_as_owner option to 'true' allows the same behavior as in previous versions. The subrunasowner column has been added to the pg_subscription catalog to store the information of this option.

□ BINARY transfer for initial data synchronization

When both PUBLICATION / SUBSCRIPTION are PostgreSQL 16 or higher and binary option of the SUBSCRIPTION is enabled, initial data is now transferred in binary format. The following example is the COPY statement log executed for initial data synchronization on the PUBLICATION side.

Example 20 Log of the COPY statements executed in initial data synchronization

```
LOG:  statement: COPY public.data1 (c1, c2) TO STDOUT WITH (FORMAT binary)
LOG:  statement: COMMIT
```

□ Index with REPLICA IDENTITY FULL

Even in an environment where REPLICA IDENTITY and primary keys cannot be used at the publisher, the subscriber can use indexes other than primary keys and REPLICA IDENTITY. The

indexes that can be used are B-tree indexes, not partial indexes, and require at least one column reference.

□ SUBSCRIPTION creation role

General users who are granted the pg_create_subscription role will be able to create SUBSCRIPTIONs. However, the password is required in the CONNECTION clause. Only users with the SUPERUSER attribute can turn off the password_required attribute. The subpasswordrequired column has been added to the pg_subscription catalog with this option.

Example 21 SUBSCRIPTION creation role

```
postgres=# GRANT pg_create_subscription TO demo ;
GRANT ROLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> CREATE SUBSCRIPTION sub1 CONNECTION 'port=5432 dbname=postgres'
PUBLICATION pub1 ;
ERROR:  password is required
DETAIL:  Non-superusers must provide a password in the connection string.
postgres=> CREATE SUBSCRIPTION sub1 CONNECTION 'port=5432 dbname=postgres
        user=postgres' PUBLICATION pub1 WITH (password_required=false) ;
ERROR:  password_required=false is superuser-only
HINT:  Subscriptions with the password_required option set to false may only be
created or modified by the superuser.
```

□ Logical decoding on standby instance

Logical decoding can now be executed on the standby instance. The pg_create_logical_replication_slot function can be executed on the standby instance. To speed up the creation of logical replication slots on the standby instance, the pg_log_standby_snapshot function can be executed on the primary instance.

Example 22 Execute pg_log_standby_snapshot function (Primary instance)

```
postgres=# SELECT pg_log_standby_snapshot() ;
 pg_log_standby_snapshot
-------------------------
 0/5000098
(1 row)
```

□ Replication mode

Add parameter logical_replication_mode that allows changing the transfer mode for logical decoding. The default value is 'buffered'. This parameter can have the following values:

Table 11 Replication mode setting (PUBLISHER side)

Value	Description
buffered	It will be serialized when the amount of changed data reaches logical_decoding_work_mem.
immediate	Streaming occurs if the 'streaming' option of the CREATE SUBSCRIPTION statement is enabled.

Table 12 Replication mode setting (SUBSCRIBER side)

Value	Description
buffered	If the 'streaming' option is set to 'parallel', send data to parallel workers through a shared memory queue.
immediate	Serializes all data to a file and notifies parallel workers to apply it at the end of the transaction.

□ Conflict detection

Add confl_active_logicalslot column to the pg_stat_database_conflicts view. This column outputs the number of uses canceled in logical replication slots.

□ ALTER TABLE REPLICA IDENTITY statement

The ALTER TABLE REPLICA IDENTITY USING statement now succeeds even if the index is INVALID. This is a fix to avoid errors due to the order in which the pg_dump command outputs table definitions.

3.1.3. Parallel Query Enhancements

There are more syntaxes and functions for parallel query execution. [11c2d6f, 16fd03e]

□ Parallel Hash Full Join

Parallel query is now supported for RIGHT OUTER JOIN and FULL OUTER JOIN syntax. This behavior can be controlled with the parameter enable_parallel_hash.

Example 23 Parallel Hash Full Join execution plan

```
postgres=> EXPLAIN (COSTS OFF) SELECT COUNT(*) FROM data1 d1 FULL OUTER JOIN
                data2 d2 USING (c1) ;
                        QUERY PLAN
-----------------------------------------------------------------
 Finalize Aggregate
   ->  Gather
         Workers Planned: 2
         ->  Partial Aggregate
               ->  Parallel Hash Full Join
                     Hash Cond: (d1.c1 = d2.c1)
                     ->  Parallel Seq Scan on data1 d1
                     ->  Parallel Hash
                           ->  Parallel Seq Scan on data2 d2
(9 rows)
```

□ Functions

Aggregation processing can now be executed in parallel when executing the string_agg and array_agg functions. Partial Aggregate is output in the execution plan when the parallel query is executed.

Example 24 Execute string_agg function

```
postgres=> EXPLAIN SELECT string_agg(c2, ':') FROM data1 ;
                                    QUERY PLAN
-------------------------------------------------------------------------------
Finalize Aggregate  (cost=11614.56..11614.57 rows=1 width=32)
   ->  Gather  (cost=11614.34..11614.55 rows=2 width=32)
         Workers Planned: 2
         ->  Partial Aggregate  (cost=10614.34..10614.35 rows=1 width=32)
               ->  Parallel Seq Scan on data1  (cost=0.00..9572.67 rows=416667
width=6)
(5 rows)
```

3.1.4. Configuration Files

Some configuration files have been written better. [8fea868, efb6f4a, a54b658, efb6f4a]

□ Regular expression for hostname and username

The pg_hba.conf file now supports regular expressions for database names and user names. Usernames and database names starting with a slash (/) are considered regular expressions.

Example 25 Pg_hba.conf file with regular expressions

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    /^demodb[1-3]   all             192.168.1.0/24          scram-sha-256
host    demodb4         /^user.*$       192.168.1.0/24          scram-sha-256
```

Regular expressions can also be used in database user entries in the pg_ident.conf file.

□ Include another file

The pg_hba.conf and the pg_ident.conf files can now include other files just as the postgresql.conf file can.

Table 13 Add syntaxes

Syntax	Description
include *file_name*	Include the file
include_if_exists *file_name*	Include the file if it exists
include_dir *directory_name*	Include all files with extension .conf under the specified directory

Example 26 Settings to include files

```
$ cat pg_hba.conf
include pg_hba_1.conf
include_if_exists pg_hba_2.conf
include_dir hba_dir
```

The file_name column that indicates the configuration file name has been added to the pg_hba_file_rules and the pg_ident_file_mappings catalogs.

□ Pg_ident.conf file

The same setting as in the pg_hba.conf file is allowed for the PostgreSQL user name part as follows.

- 'all' for all database users
- Member check using plus (+)
- Database username regular expression

3.1.5. ICU Locale

Enhanced the ability to use the ICU locale. [fcb21b3, 27b6237, c1f1c1f, 5cd1a5a, c45dc7f, 30a53b7, cd42785, 0d21d4b]

□ Default locale provider

The default locale provider has been changed from libc to icu in environments where ICU is available. The ICU library is also included by default when building from source code. The --with-icu option of the 'configure' command has been abolished, and the --without-icu option has been added to specify when not using ICU. However, if only --no-locale is specified in the initdb command, LIBC will be used as the locale provider as in previous versions.

Example 27 Change Default locale provider (initdb)

```
$ export LANG=en_GB.utf8
$ initdb data
This user must also own the server process.

Using default ICU locale "en".
Using language tag "en" for ICU locale "en".
The database cluster will be initialized with this locale configuration:
  provider:    icu
  ICU locale:  en
  LC_COLLATE:  en_GB.utf8
...
```

Example 28 Change default locale provider (psql)

```
postgres=# \l
                                                List of databases
   Name    |  Owner   | Encoding | Locale Provider | Collate    | Ctype      | ICU Locale |…
-----------+----------+----------+-----------------+------------+------------+------------+…
 postgres  | postgres | UTF8     | icu             | en_GB.utf8 | en_GB.utf8 | en         |…
 template0 | postgres | UTF8     | icu             | en_GB.utf8 | en_GB.utf8 | en         |…
           |          |          |                 |            |            |            |…
 template1 | postgres | UTF8     | icu             | en_GB.utf8 | en_GB.utf8 | en         |…
           |          |          |                 |            |            |            |…
(3 rows)
```

□ Default encoding

The default encoding when using the ICU locale is UTF8.

□ Specify custom rules

Custom rules can now be specified for the ICU locales. Add RULES clause to the CREATE COLLATION statement and the ICU_RULES clause to the CREATE DATABASE statement to specify rules. See the URL below for the detailed syntax of the rules.

https://unicode-org.github.io/icu/userguide/collation/customization/

Syntax 1 CREATE COLLATION / CREATE DATABASE statement

```
CREATE COLLATION [ IF NOT EXISTS ] name (
  [ RULES = rules ]
)

CREATE DATABASE name
  [ WITH ] [ ICU_RULES [=] icu_rules ]
```

In the example below, a unique rule "&a < g" is added to the locale "en_US" when creating the database. The table to which the rule is applied is output in the order of apple, green, and bird when sorted.

Example 29 Specifying custom rules

```
postgres=# CREATE DATABASE en_db1 LOCALE_PROVIDER icu ICU_LOCALE 'en_US'
                ICU_RULES '&a < g' TEMPLATE template0 ;
CREATE DATABASE
postgres=# \connect en_db1 postgres
You are now connected to database "en_db1" as user "postgres".
en_db1=# CREATE TABLE data1(c1 text) ;
CREATE TABLE
en_db1=# INSERT INTO data1 values ('apple'),('bird'),('green') ;
INSERT 0 3
en_db1=# SELECT c1 FROM data1 ORDER BY c1 ;
  c1
-------
 apple
 green
 bird
(3 rows)
```

The contents of custom rules can be checked in the daticurules column of the pg_database catalog, the collicurules column of the pg_collation catalog, and the object definition of the psql command.

Example 30 Check custom rules

```
postgres=# \l en_db1
List of databases
-[ RECORD 1 ]-----+---------
Name              | en_db1
Owner             | postgres
Encoding          | UTF8
Locale Provider   | icu
Collate           | C
Ctype             | C
ICU Locale        | en_US
ICU Rules         | &a < g
Access privileges |
```

□ New ICU locale

Add UNICODE to the ICU locale.

Example 31 Check UNICODE locale

```
postgres=> SELECT * FROM pg_collation WHERE collname = 'unicode' ;
-[ RECORD 1 ]-------+--------
oid                | 12344
collname           | unicode
collnamespace      | 11
collowner          | 10
collprovider       | i
collisdeterministic | t
collencoding       | -1
collcollate        |
collctype          |
colliculocale      | und
collicurules       |
collversion        | 153.80
```

3.1.6. Predefined Roles

PostgreSQL 16 has been added the following predefined roles: [c3afe8c, 60684dd, 6e2775e]

Table 14 Add predefined roles

Role name	Description
pg_create_subscription	Allows creation of SUBSCRIPTION
pg_maintain	Allows execution of maintenance statements on all tables
pg_use_reserved_connections	Connections assigned to parameter reserved_connections can be used

Users granted the pg_maintain role can execute the ANALYZE, VACUUM, REINDEX, CLUSTER, REFRESH MATERIALIZED VIEW, and LOCK TABLE statements on tables and materialized views owned by other users. Previously, unauthorized operations on other user-owned objects were restricted to users with the SUPERUSER attribute.

3.1.7. Wait Events

The following wait events have been added or changed. [7bae3bb, fce003c, 216a784, 5a3a953, d8cd0c6, 92daeca, 8fba928]

Table 15 Add Wait Events

Event name	Type	Description
DSMAllocate	IO	Waiting for dynamic shared memory allocation
LogicalApplySendData	IPC	Waiting to complete sending data to parallel apply workers
LogicalParallelApplyMain	Activity	Waiting for parallel application
LogicalParallelApplyStateChange	Activity	Waiting for Parallel Apply status change
LogicalRepLauncherDSA	LWLock	Waiting for access to the DSA memory allocator in the launcher process
LogicalRepLauncherHash	LWLock	Waiting for access to the shared hash table of the launcher process
RelationMapReplace	IO	Waiting for replacement of relation map file. Changed from RelationMapSync
SLRUFlushSync	IO	Waiting for SLRU data to reach storage during checkpoint or database shutdown.
SpinDelay	Timeout	pg_usleep execution wait time

Table 16 Renamed Wait Event

Old name	New name	Note
HashGrowBatchesAllocate	HashGrowBatchesReallocate	
HashGrowBucketsAllocate	HashGrowBucketsReallocate	

3.1.8. Triggers

The following enhancements have been implemented for triggers: [3b00a94, 93f2349]

□ TRUNCATE Trigger

The TRUNCATE triggers are now executed when the TRUNCATE statement is executed for a foreign table.

□ Event Trigger

Execution of the ALTER MATERIALIZED VIEW statement now executes event triggers.

3.1.9. Log files

Add information that is printed to the log in some cases. [62c46ee, d977ffd, 71cb84e]

□ Checkpoint

The LSN information is added to the log when the parameter log_checkpoints is set to 'on'.

Example 32 Checkpoint log

```
LOG:  checkpoint starting: time
LOG:  checkpoint complete: wrote 44 buffers (0.3%); 0 WAL file(s) added, 0 removed, 0 recycled; write=4.134 s, sync=0.005 s, total=4.143 s; sync files=11, longest=0.002 s, average=0.001 s; distance=258 kB, estimate=258 kB; lsn=0/153EC70, redo lsn=0/153EC38
```

□ Auto vacuum

Add frozen table information to autovacuum logs.

Example 33 Auto vacuum log

```
LOG:  automatic vacuum of table "postgres.public.data1": index scans: 1
        pages: 0 removed, 541 remain, 541 scanned (100.00% of total)
        tuples: 50000 removed, 50000 remain, 0 are dead but not yet removable
        removable cutoff: 750, which was 0 XIDs old when operation ended
        new relfrozenxid: 747, which is 1 XIDs ahead of previous value
        frozen: 0 pages from table (0.00% of total) had 0 tuples frozen
        avg read rate: 0.414 MB/s, avg write rate: 0.829 MB/s
...
```

□ WAL

The LSN information is now outputted in error messages that occur while validating WAL headers.

3.1.10. Kerberos Credential Delegation

Supports GSSAPI/Kerberos authentication delegated by the client to the server. This allows users who authenticate to PostgreSQL with Kerberos credentials to delegate their credentials to PostgreSQL. The server will now be able to connect to other services using the delegated credentials. GSSAPI delegate acceptance is enabled by changing the parameter gss_accept_delegation to 'on'. Sessions with delegated GSSAPI credentials can be seen in the credentials_delegated column of the pg_stat_gssapi view.

This new feature adds the gssdeleg to the libpq connection string. The default setting value is 'disable'. Specify 'enable' in the connection string to enable GSS authentication delegation. The environment variable PGGSSDELEG can also be used instead of the connection string. This setting can also be used with the postgres_fdw module, the dblink module, etc. Add PQconnectionUsedGSSAPI as an API to check the authentication method. [6633cfb, 1f9f6aa]

Syntax 2 PQconnectionUsedGSSAPI function

```
int PQconnectionUsedGSSAPI(const PGconn *conn);
```

3.1.11. Libpq

The following extensions have been implemented in libpq, the API that using PostgreSQL. [3a465cc, 36f40ce, 7f5b198, 8eda731, 9fcdf2c, 419a8dd, 19d8e23]

□ Connection string require_auth

Add a connection string require_auth that specifies the list of required authentication methods. For this parameter, specify the following connection methods separated by commas (,). If there are no authentication methods that the server can provide, the connection will fail. This connection string can also be specified with the environment variable PGREQUIREAUTH.

Table 17 Connection string require_auth

Authentication method	Description
password	Password authentication
md5	Authentication with MD5 password
gss	GSSAPI authentication
sspi	SSPI authentication
scram-sha-256	SCRAM-SHA-256 authentication
none	No required authentication method

Example 34 Setting the required authentication method

```
$ grep demo data/pg_hba.conf
host    all             demo            127.0.0.1/32            scram-sha-256
$ export PGREQUIREAUTH=scram-sha-256
$ psql -h 127.0.0.1 -d postgres -U demo
Password for user demo:
psql (16beta1)
Type "help" for help.

postgres=> ¥q
$ psql -h 127.0.0.1 -d postgres -U postgres
psql: error: connection to server on socket "/tmp/.s.PGSQL.5432" failed: auth method "scram-sha-256" requirement failed: server did not complete authentication
```

□ Connection string sslcertmode

The connection string sslcertmode determines if the server will send the certificate from the client. This connection string can also be specified in the environment variable PGSSLCERTMODE.

Table 18 Setting value

Value	Description
allow	Client certificates will be sent if the server requests them. This behavior is the same as in previous versions and is the default value.
disable	Refuse to send even if a client certificate is available.
require	No client certificate is sent and the connection fails. It can be useful for troubleshooting.

□ Connection string load_balance_hosts

The connection string load_balance_hosts automatically selects a specific instance from the destination list. This connection string can also be specified in the environment variable PGLOADBALANCEHOSTS. If the destination does not exist, another destination is automatically selected.

Table 19 Setting value

Value	Description
random	Randomly selects a connection point from multiple specified hosts and ports.
disable	Disables random selection of destinations (default).

Example 35 Random selection of connection destination

```
$ psql "load_balance_hosts=random host=localhost,localhost port=5432,5433
dbname=postgres user=postgres" -c "\conninfo"
You are connected to database "postgres" as user "postgres" on host "localhost"
(address "::1") at port "5432".
$ psql "load_balance_hosts=random host=localhost,localhost port=5432,5433
dbname=postgres user=postgres" -c "\conninfo"
You are connected to database "postgres" as user "postgres" on host "localhost"
(address "::1") at port "5433".
```

□ Connection string sslrootcert

The connection string sslrootcert can now specify 'system' as its configuration value. This configuration value loads the system's trusted CA root certificate for certificate validation.

□ COPY statement callback

The CopySendEndOfRow API now supports a callback function. This feature helps the extensions execute the COPY TO statements. A similar callback function is already provided for the COPY FROM statements.

□ Hook

Add a hook to change the password specified in ldapbindpasswd in the pg_hba.conf file for LDAP authentication. Update the ldap_password_hook defined in the src/backend/libpq/auth.c file.

□ Index Access Method

Add "bool amsummarizing;" flag to IndexAmRoutine struct. This flag indicates whether the access method should summarize tuples.

3.1.12. PL/pgSQL

Add PG_ROUTINE_OID to get OID of self-procedure. [d3d53f9]

Example 36 Get PG_ROUTINE_OID

```
postgres=> CREATE OR REPLACE FUNCTION current_function()
             RETURNS regprocedure AS $$
             DECLARE
                 fnc_oid regprocedure;
             BEGIN
                 GET DIAGNOSTICS fnc_oid = PG_ROUTINE_OID;
             RETURN fnc_oid;
           END; $$ LANGUAGE plpgsql;
CREATE FUNCTION
postgres=> SELECT current_function() ;
  current_function
--------------------
 current_function()
(1 row)
```

3.1.13. MATERIALIZED VIEW

Supports Predicate Lock to prevent serialization anomaly when executing the REFRESH MATERIALIZED VIEW CONCURRENTLY statement. [4335155]

3.1.14. WAL sender process

The database name of the data provider (PUBLICATION side) is now output as the process name of the WAL sender process in the logical replication environment. [af20515]

Example 37 The WAL sender process name

```
$ ps -ef|grep walsender | grep -v grep
postgres    23488    23127   0 18:10 ?          00:00:00 postgres: walsender
postgres demodb [local] START_REPLICATION
```

3.1.15. GIN Index cost

CPU-based cost is now taken into account as part of the GIN index cost calculation. This increases the cost estimate for using GIN indexes. [cd9479a]

3.1.16. Meson

Meson (https://mesonbuild.com/) is now available as a build system. The meson.build file is located in each directory in the source code. [e692727]

3.1.17. UNICODE

The supported Unicode version has been changed from 14.0.0 to 15.0.0. [1091b48]

3.1.18. LLVM

Supports LLVM 15. This change will also be backported to previous versions. [c2ae01f]

3.1.19. Extension

Add the @extschema:{name}@ and no_relocate options to extensions. @extschema:{name}@ can extend the existing @extschema@ functionality and insert an extended schema name. The no_relocate option specifies a list of extension names that this extension depends on. [72a5b1f]

Example 38 Part of the test_ext_req_schema2--1.0.sql

```
CREATE FUNCTION dep_req2() RETURNS text
BEGIN ATOMIC
  SELECT @extschema:test_ext_req_schema1@.dep_req1() || ' req2';
END;
```

Example 39 Part of the test_ext_req_schema3.control

```
relocatable = true
requires = 'test_ext_req_schema1, test_ext_req_schema2'
no_relocate = 'test_ext_req_schema1'
```

3.1.20. Reduction of WAL output

Reduced amount of WAL output for FREEZE operations. [9e54059]

3.2. SQL Statement

This section explains new features related to SQL statements.

3.2.1. Data Types

The following extensions have been implemented for data types: [2ceea5a, 6fcda9a, 102a5c1, 6dfacbf, faff8f8, 096dd80]

□ +infinity notation

The +infinity value can now be specified for date, timestamp, and timestamp with time zone types. +infinity is considered the same as infinity.

Example 40 +Infinity notation

```
postgres=> SELECT '+infinity'::timestamp ;
 timestamp
-----------
 infinity
(1 row)

postgres=> SELECT '+infinity'::timestamptz ;
 timestamptz
-------------
 infinity
(1 row)
```

□ Integer Literals

Integer literals can now be represented in hexadecimal, octal, and binary. Each is represented as follows. Specify a leading '0' followed by a symbol indicating the base. The case is not sensitive. This change corresponds to the SQL:202x draft.

Table 20 Integer literal description

Designation method	Example	Note
Hexadecimal	0x42F	
Octal	0o273	
Binary number	0b100101	

Example 41 Notation for integer literals

```
postgres=> SELECT * FROM data1 WHERE c1 IN ( 100, 0b11001000, 0o454, 0x190 )
                ORDER BY 1 ;
 c1  |  c2
-----+-------
 100 | data1
 200 | data2
 300 | data3
 400 | data4
(4 rows)
```

□ JSONPATH literal

Hexadecimal, octal, and binary representations can also be specified in JSONPATH.

Example 42 Integer literal in the JSONPATH

```
postgres=> SELECT '0b100101'::jsonpath ;
 jsonpath
----------
 37
(1 row)

postgres=> SELECT '0o273'::jsonpath ;
 jsonpath
----------
 187
(1 row)

postgres=> SELECT '0x42F'::jsonpath ;
 jsonpath
----------
 1071
(1 row)
```

□ Numeric type

The numeric type supports hexadecimal, octal, and binary representations. However, it cannot be written below the decimal point.

Example 43 Hexadecimal notation of type NUMERIC

```
postgres=> SELECT pg_input_is_valid('0x1234', 'numeric') ;
 pg_input_is_valid
-------------------
 t
(1 row)

postgres=> SELECT message FROM
        pg_input_error_info('0x1234.56', 'numeric(10, 3)') ;
                        message
-----------------------------------------------------
 invalid input syntax for type numeric: "0x1234.56"
(1 row)
```

□ Numeric literals and underscores

Underscores (_) can be specified in numeric literals. This change corresponds to the SQL:202x draft.

Example 44 Specifying underscores in the numeric literals

```
postgres=> SELECT 1_000 dec, 2_000.234 dec2, 0x100_200 hex,
                0o2_467 oct, 0b01_101 bin1, 0b_1001_0001 bin2 ;
-[ RECORD 1 ]--
dec  | 1000
dec2 | 2000.234
hex  | 1049088
oct  | 1335
bin1 | 13
bin2 | 145
```

3.2.2. COPY

The following features have been implemented for the COPY statement: [9f8377f, 97da482]

□ Insert default value

The option DEFAULT have been added to the COPY FROM statement. If the value specified for this option matches the input value, then the DEFAULT value for the column will be entered. If there is no DEFAULT specification for the column and there is data that matches the DEFAULT option of the COPY statement, an error will occur.

Example 45 Specify the DEFAULT option

```
postgres=> CREATE TABLE data1(c1 INT PRIMARY KEY, c2 TEXT DEFAULT 'DefVal1') ;
CREATE TABLE
postgres=> COPY data1 FROM stdin WITH (format csv, default '\D') ;
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.
>> 1,data1
>> 2,\D
>> 3,data3
>> COPY 3
postgres=> SELECT * FROM data1 ;
 c1 |   c2
----+---------
  1 | data1
  2 | DefVal1
  3 | data3
(3 rows)
```

This option is also available in extension file_fdw. When referencing a FOREIGN TABLE and the value in the file matches the DEFAULT option, the DEFAULT value for the column is referenced.

Example 46 Specifying the DEFAULT option (file_fdw)

```
postgres=# \! cat /tmp/data1.csv
1,data1
2,\D
3,data3
postgres=# CREATE FOREIGN TABLE file1(c1 INT, c2 TEXT DEFAULT 'DefVal1')
  SERVER fsvr1 OPTIONS (filename '/tmp/data1.csv', format 'csv', default '\D') ;
CREATE FOREIGN TABLE
postgres=# SELECT * FROM file1 ;
 c1 |   c2
----+---------
  1 | data1
  2 | DefVal1
  3 | data3
(3 rows)
```

□ Effective range of the batch_size option

The batch_size option of foreign server (FOREIGN SERVER) and foreign table (FOREIGN TABLE) can now be used when executing the COPY statement for a foreign table. Previously, it was valid only when executing the INSERT statement.

3.2.3. CREATE ROLE/USER

Roles/users with the CREATEROLE attribute will be able to grant other users only their own attributes and roles granted with the ADMIN OPTION. The CREATEDB attribute has the same behavior as in older versions. [f1358ca]

Example 47 User creation by the CREATEROLE attribute user

```
postgres=# CREATE USER usradm1 CREATEROLE REPLICATION CREATEDB BYPASSRLS ;
CREATE ROLE
postgres=# \connect postgres usradm1
You are now connected to database "postgres" as user "usradm1".
postgres=> CREATE USER user_repl1 REPLICATION ;
CREATE ROLE
postgres=> CREATE USER user_rls1 BYPASSRLS ;
CREATE ROLE
postgres=> CREATE USER user_db1 CREATEDB ;
CREATE ROLE
```

The user who creates a role automatically becomes a member of the created role. However, it is not possible to automatically inherit (or SET ROLE) privileges. To change this behavior, specify SET and INHERIT in the createrole_self_grant parameter separated by a comma (,).

Example 48 SET ROLE statement for creation role

```
postgres=> CREATE ROLE role1 ;
CREATE ROLE
postgres=> SET ROLE role1 ;
ERROR:  permission denied to set role "role1"
postgres=> SET createrole_self_grant = 'SET' ;
SET
postgres=> CREATE ROLE role2 ;
CREATE ROLE
postgres=> SET ROLE role2 ;
SET
```

3.2.4. CREATE STATISTICS

The name of the statistic is no longer mandatory. If the name is omitted, the statistic name is automatically generated from the original table name and column name. [624aa2a]

Example 49 Name abbreviation in the CREATE STATISTICS statement

```
postgres=> CREATE STATISTICS ON c1, c2 FROM data1 ;
CREATE STATISTICS
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)
Statistics objects:
    "public.data1_c1_c2_stat" ON c1, c2 FROM data1
```

3.2.5. CREATE TABLE

The following new features have been added to the CREATE TABLE statement. [784cedd, b9424d0]

□ Specification of the STORAGE clause

The STORAGE clause can now be specified in the CREATE TABLE statement. In previous versions, it had to be set with the ALTER TABLE statement.

Example 50 STORAGE clause of the CREATE TABLE statement

```
postgres=> CREATE TABLE data1(c1 INT PRIMARY KEY,
    c2 VARCHAR(10) STORAGE PLAIN, c3 TEXT STORAGE EXTENDED) ;
CREATE TABLE
postgres=> \d+ data1
                                                Table "public.data1"
 Column |         Type          | Collation | Nullable | Default | Storage  |…
--------+-----------------------+-----------+----------+---------+----------+…
 c1     | integer               |           | not null |         | plain    |…
 c2     | character varying(10) |           |          |         | plain    |…
 c3     | text                  |           |          |         | extended |…
Indexes:
    "data1_pkey" PRIMARY KEY, btree (c1)
Access method: heap
```

□ DEFAULT Storage

The STORGE clause of the CREATE TABLE and ALTER TABLE statements can be DEFAULT to indicate the default storage format.

Example 51 STORAGE DEFAULT clause in the ALTER TABLE statement

```
postgres=> ALTER TABLE data1 ALTER COLUMN c2 SET STORAGE DEFAULT ;
ALTER TABLE
```

3.2.6. EXPLAIN

Add GENERIC_PLAN option to show the execution plan for parameterized query strings. [3c05284]

Example 52 Setting the GENERIC_PLAN option

```
postgres=> EXPLAIN (GENERIC_PLAN) SELECT * FROM data1 WHERE c1=$1 ;
                                QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using data1_pkey on data1  (cost=0.43..8.45 rows=1 width=10)
   Index Cond: (c1 = $1)
(2 rows)

postgres=> EXPLAIN (GENERIC_PLAN FALSE) SELECT * FROM data1 WHERE c1=$1 ;
ERROR:  there is no parameter $1
LINE 1: ...XPLAIN (GENERIC_PLAN FALSE) SELECT * FROM data1 WHERE c1=$1 ;
                                                                 ^
```

3.2.7. GRANT

The following features have been added to the GRANT statement. [b5d6382, e3ce2de, 3d14e17, ce6b672, c44f633]

□ MAINTAIN privilege

Add MAINTAIN privilege to allow execution of table and materialized view maintenance statements (VACUUM, ANALYZE, REINDEX, REFRESH MATERIALIZED VIEW, CLUSTER and LOCK TABLE) on a table. The psql command now outputs 'm' in the permissions display.

Example 53 Permission to execute maintenance statements

```
postgres=> CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> GRANT MAINTAIN ON data1 TO demo2 ;
GRANT
postgres=> \dp+ data1
                              Access privileges
 Schema | Name  | Type  | Access privileges  | Column privileges | Policies
--------+-------+-------+--------------------+-------------------+----------
 public | data1 | table | demo=arwdDxtm/demo+|                   |
        |       |       | demo2=m/demo       |                   |
(1 row)
```

The pg_maintain has been added as a predefined role that permits similar operations on all tables.

□ LOCK TABLE statement

The execution privilege of the LOCK TABLE statement has been simplified. Table owners, superusers, and holders of the pg_maintain role can execute the LOCK TABLE statement in any mode. The table below shows the relationship between table access privileges and permitted lock modes.

Table 21 Access privileges and the possible lock modes

Table access privileges	Allowed lock mode
MAINTAIN, UPDATE, DELETE, TRUNCATE	Any lock mode
INSERT	ROW EXCLUSIVE MODE
SELECT	ACCESS SHARE MODE

□ WITH INHERIT clause

It is now possible to specify the WITH INHERIT OPTION clause (or WITH INHERI TRUE) that inherits privileges and the WITH INHERIT FALSE clause that does not inherit privileges. The default is WITH INHERIT TRUE. The specified value can be checked in the inherit_option column of the pg_auth_members catalog.

Example 54 WITH INHERIT clause

```
postgres=# GRANT pg_read_all_stats TO demo WITH INHERIT FALSE ;
GRANT ROLE
```

□ WITH SET clause

It is now possible to specify the WITH SET OPTION clause (or WITH SET OPTION), which allows the use of the SET ROLE statement, and the WITH SET FALSE clause, which disallows it. The default is WITH SET TRUE. The specified value can be confirmed in the set_option column of the pg_auth_members catalog.

Example 55 WITH SET clause

```
postgres=# GRANT role1 TO demo WITH SET FALSE ;
GRANT ROLE
postgres=# SET SESSION AUTHORIZATION demo ;
SET
postgres=> SET ROLE role1 ;
ERROR:  permission denied to set role "role1"
```

3.2.8. JSON

Enhanced JSON related syntax. [7081ac4, 6ee3020]

□ JSON Constructor

JSON type constructors that are compliant with the SQL/JSON standard have been added. The following functions are available

Syntax 3 JSON Constructor

```
json_array ( [ { value_expression [ FORMAT JSON ] } [, ...] ] [ { NULL | ABSENT }
ON NULL ] [ RETURNING data_type [ FORMAT JSON [ ENCODING UTF8 ] ] ])

json_arrayagg ( [ value_expression ] [ ORDER BY sort_expression ] [ { NULL |
ABSENT } ON NULL ] [ RETURNING data_type [ FORMAT JSON [ ENCODING UTF8 ] ] ])

json_object ( [ { key_expression { VALUE | ':' } value_expression [ FORMAT JSON
[ ENCODING UTF8 ] ] }[, ...] ] [ { NULL | ABSENT } ON NULL ] [ { WITH | WITHOUT }
UNIQUE [ KEYS ] ] [ RETURNING data_type [ FORMAT JSON [ ENCODING UTF8 ] ] ])

json_objectagg ( [ { key_expression { VALUE | ':' } value_expression } ] [ { NULL
| ABSENT } ON NULL ] [ { WITH | WITHOUT } UNIQUE [ KEYS ] ] [ RETURNING data_type
[ FORMAT JSON [ ENCODING UTF8 ] ] ])
```

Example 56 JSON Constructor

```
postgres=> SELECT JSON_ARRAY('a',  NULL, 'b' NULL   ON NULL RETURNING jsonb) ;
    json_array
------------------
 ["a", null, "b"]
(1 row)

postgres=> SELECT JSON_ARRAYAGG(i ORDER BY i DESC) FROM generate_series(1,5)
i ;
  json_arrayagg
-----------------
 [5, 4, 3, 2, 1]
(1 row)

postgres=> SELECT JSON_OBJECT('a': 2 + 3) ;
 json_object
-------------
 {"a" : 5}
(1 row)

postgres=> SELECT JSON_OBJECTAGG(k: v ABSENT ON NULL WITH UNIQUE KEYS) FROM
(VALUES (1, 1), (0, NULL), (3, NULL), (2, 2), (4, NULL)) foo(k, v) ;
    json_objectagg
----------------------
 { "1" : 1, "2" : 2 }
(1 row)
```

□ IS JSON Syntax

Add IS JSON syntax to check conformance to JSON syntax. Multiple syntaxes are provided depending on the type of JSON format.

Syntax 4 IS JSON clause

```
item IS [NOT] JSON
item IS JSON VALUE
item IS JSON OBJECT
item JS JSON ARRAY
item IS JSON SCALAR
item IS JSON WITHOUT UNIQUE KEYS
item IS JSON WITH UNIQUE KEYS
```

Example 57 IS JSON Clause

```
postgres=> SELECT '[1, 2, 3]'::jsonb IS JSON ARRAY ;
 ?column?
----------
 t
(1 row)

postgres=> SELECT '{"a": 1, "b": null}'::jsonb IS JSON OBJECT ;
 ?column?
----------
 t
(1 row)
```

3.2.9. REINDEX

The database name can now be omitted in the REINDEX SYSTEM and REINDEX DATABASE statements. If the database name is omitted, the command is executed for the current database. Also, in the REINDEX DATABASE statement, REINDEX processing for the system catalog will not be executed. [2cbc3c1]

Example 58 Omitting the database name

```
postgres=# REINDEX DATABASE ;
REINDEX
postgres=# REINDEX SYSTEM ;
REINDEX
```

3.2.10. VACUUM

The VACUUM statement has the following new features: [a46a701, d977ffd, 4211fbd, 1cbbee0, 1de58df]

□ Database Statistics

Add the SKIP_DATABASE_STATS option to the VACUUM statement to suppress updating database statistics. Setting this option to TRUE suppresses scans into the pg_class catalog, improving concurrency. The ONLY_DATABASE_STATS option is now available to retrieve database statistics only.

Example 59 Execute the VACUUM statement

```
postgres=# VACUUM (SKIP_DATABASE_STATS TRUE) ;
VACUUM
postgres=# VACUUM (ONLY_DATABASE_STATS TRUE) ;
VACUUM
```

□ TOAST only VACUUM

By specifying the PROCESS_MAIN FALSE, only TOAST data can be VACUUMed. The PROCESS_MAIN option is TRUE by default.

Example 60 TOAST only VACUUM

```
postgres=> VACUUM (PROCESS_MAIN FALSE, VERBOSE) data1 ;
INFO:  vacuuming "postgres.pg_toast.pg_toast_16392"
INFO:  finished vacuuming "postgres.pg_toast.pg_toast_16392": index scans: 0
pages: 0 removed, 0 remain, 0 scanned (100.00% of total)
tuples: 0 removed, 0 remain, 0 are dead but not yet removable
removable cutoff: 739, which was 0 XIDs old when operation ended
new relfrozenxid: 739, which is 1 XIDs ahead of previous value
frozen: 0 pages from table (100.00% of total) had 0 tuples frozen
...
avg read rate: 0.000 MB/s, avg write rate: 0.000 MB/s
buffer usage: 2 hits, 0 misses, 0 dirtied
WAL usage: 1 records, 0 full page images, 188 bytes
system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
VACUUM
```

□ Shared buffer limit

Add the BUFFER_USAGE_LIMIT option to specify the shared buffer usage when executing VACAUUM and ANALYZE statements. The default value if this option is not specified is determined by the parameter vacuum_buffer_usage_limit (default value 256kB).

Example 61 The BUFFER_USAGE_LIMIT option

```
postgres=> SHOW vacuum_buffer_usage_limit ;
-[ RECORD 1 ]------------+------
vacuum_buffer_usage_limit | 256kB

postgres=> VACUUM (BUFFER_USAGE_LIMIT '1MB') data1 ;
VACUUM
```

□ Page-level FREEZE

The FREEZE processing can now be executed on a page-level. This will perform WAL deduplication and reduce the amount of writes.

3.2.11. Subquery

It is no longer necessary to specify an alias name for subqueries in the FROM clause. [bcedd8f]

Example 62 Subquery with the FROM clause

```
postgres=> INSERT INTO data1 SELECT * FROM (SELECT * FROM data2 WHERE c1<100) ;
INSERT 0 99
postgres=> SELECT COUNT(*) FROM (SELECT c2 FROM data1 WHERE c1 = 90) ;
 count
-------
     1
(1 row)
```

3.2.12. Execution Plan

Faster execution plans are now chosen in the following cases: [3c6fc58, ed1a88d, 16dc270, 9bfd282]

□ SELECT DITINCT statement

Aggregate functions with ORDER BY or DISTINCT clauses can now run more efficiently. This behavior can be controlled with the parameter enable_presorted_aggregate parameter. Setting the value of this parameter to 'off' will make it behave the same as PostgreSQL 15.

Example 63 PostgreSQL 16 behavior

```
postgres=> EXPLAIN SELECT COUNT(DISTINCT c1) FROM data1 ;
                                      QUERY PLAN
---------------------------------------------------------------------------------------
 Aggregate  (cost=28480.42..28480.43 rows=1 width=8)
   ->  Index Only Scan using data1_pkey on data1  (cost=0.42..25980.42 rows=1000
000 width=4)
(2 rows)
```

Example 64 PostgreSQL 15 behavior

```
postgres=> SET enable_presorted_aggregate = off ;
SET
postgres=> EXPLAIN SELECT COUNT(DISTINCT c1) 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=4)
(2 rows)
```

□ WINDOW functions

Execution plans can now be shared for the same OVER clause. The execution plan is optimized when executing the following the SELECT statements.

Example 65 SELECT statement for the example

```
SELECT
  empno,
  depname,
  ROW_NUMBER() OVER (PARTITION BY depname ORDER BY enroll_date) rn,
  RANK() OVER (PARTITION BY depname ORDER BY enroll_date ROWS BETWEEN
    UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) rnk,
  DENSE_RANK() OVER (PARTITION BY depname ORDER BY enroll_date RANGE BETWEEN
    CURRENT ROW AND CURRENT ROW) drnk
FROM
    empsalary ;
```

Example 66 Execution plan in PostgreSQL 15

```
                    QUERY PLAN
-----------------------------------------------------
 WindowAgg
   ->  WindowAgg
         ->  WindowAgg
               ->  Sort
                     Sort Key: depname, enroll_date
                     ->  Seq Scan on empsalary
(6 rows)
```

Example 67 Execution plan in PostgreSQL 16

```
               QUERY PLAN
---------------------------------------
 WindowAgg
   ->  Sort
         Sort Key: depname, enroll_date
         ->  Seq Scan on empsalary
(4 rows)
```

□ HASH Join / MERGE Join

Hash join and merge join are now supported for the RIGHT ANTI JOIN on non-NULL-valued inputs.

Example 68 Execution plan in PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT t1.* FROM prt1_adv t1 WHERE NOT EXISTS
(SELECT 1 FROM prt2_adv t2 WHERE t1.a = t2.b) AND t1.b = 0 ORDER BY t1.a ;
                       QUERY PLAN
-------------------------------------------------------
 Sort
   Sort Key: t1.a
   ->  Hash Right Anti Join
         Hash Cond: (t2.b = t1.a)
         ->  Append
               ->  Seq Scan on prt2_adv_p1 t2_1
               ->  Seq Scan on prt2_adv_p2 t2_2
               ->  Seq Scan on prt2_adv_p3 t2_3
         ->  Hash
               ->  Append
                     ->  Seq Scan on prt1_adv_p1 t1_1
                           Filter: (b = 0)
                     ->  Seq Scan on prt1_adv_p2 t1_2
                           Filter: (b = 0)
                     ->  Seq Scan on prt1_adv_p3 t1_3
                           Filter: (b = 0)
(16 rows)
```

Example 69 Execution plan in PostgreSQL 15

```
                        QUERY PLAN
-----------------------------------------------------
 Sort
   Sort Key: t1.a
   ->  Hash Anti Join
         Hash Cond: (t1.a = t2.b)
         ->  Append
               ->  Seq Scan on prt1_adv_p1 t1_1
                     Filter: (b = 0)
               ->  Seq Scan on prt1_adv_p2 t1_2
                     Filter: (b = 0)
               ->  Seq Scan on prt1_adv_p3 t1_3
                     Filter: (b = 0)
         ->  Hash
               ->  Append
                     ->  Seq Scan on prt2_adv_p1 t2_1
                     ->  Seq Scan on prt2_adv_p2 t2_2
                     ->  Seq Scan on prt2_adv_p3 t2_3
(16 rows)
```

□ Memoize by UNION ALL

Memoize can now be used in the Append operation with the UNION ALL clause.

Example 70 Execution plan in PostgreSQL 16

```
postgres=> EXPLAIN (COSTS OFF) SELECT * FROM t1, (SELECT * FROM t2 UNION ALL
        SELECT * FROM t2) t3 WHERE t1.c1 = t3.c1 ;
                        QUERY PLAN
-----------------------------------------------------------
 Nested Loop
   ->  Seq Scan on t1
   ->  Memoize
         Cache Key: t1.c1
         Cache Mode: logical
         ->  Append
               ->  Index Scan using t2_c1_idx on t2
                     Index Cond: (c1 = t1.c1)
               ->  Index Scan using t2_c1_idx on t2 t2_1
                     Index Cond: (c1 = t1.c1)
(10 rows)
```

□ Improve performance for partition lookups

LIST / RANGE partition table now caches partition information if the same partition is found 16 times. Faster searches on tables with many partitions. [a61b1f7, a61b1f7]

3.2.13. Functions

The following functions have been added/enhanced. [888f2ea, 2ddab01, 38d8176, d5d5741, 75bd846, 75bd846, 283129e, 0823d06, cca1863, 13e0d7a, 10ea0f9, 1939d26, 1939d26, bf03cfd, 483bdb2]

□ Array_sample / array_shuffle

Two new functions have been added: array_sample, which returns a specified number of random elements from an array, and array_shuffle, which shuffles the elements of an array.

Syntax 5 Array_sample / array_shuffle function

```
anyarray array_sample(array anyarray, n integer)
anyarray array_shuffle(array anyarray)
```

Example 71 Array_sample / array_shuffle function

```
postgres=> SELECT array_sample(ARRAY[1,2,3,4,5,6], 3) ;
 array_sample
--------------
 {1,3,6}
(1 row)

postgres=> SELECT array_shuffle(ARRAY[[1,2],[3,4],[5,6]]) ;
    array_shuffle
--------------------
 {{3,4},{1,2},{5,6}}
(1 row)
```

□ Any_value

Add aggregate function any_value. This function non-deterministically returns an arbitrary value from the input values of the aggregate. This function is part of the SQL:2023 standard.

Syntax 6 Any_value function

```
anyelement any_value(anyelement)
```

Example 72 Any_value function

```
postgres=> SELECT * FROM data1 ;
 c1  |  c2
-----+-------
 100 | data1
 200 | data2
 300 | data2
(3 rows)

postgres=> SELECT ANY_VALUE(c1), MAX(c1), AVG(c1) FROM data1 ;
 any_value | max |         avg
-----------+-----+---------------------
       100 | 300 | 200.0000000000000000
(1 row)
```

□ Date_add / date_subtract

Add functions for adding and subtracting dates and times. The date_add function adds time and the date_subtract function subtracts time. The timezone parameter is optional.

Syntax 7 Date_add / date_subtract function

```
timestamp with time zone date_add(timestamp with time zone, interval [, text])
timestamp with time zone date_subtract(timestamp with time zone, interval [, text]
```

Example 73 Date_add / date_subtract function

```
postgres=> SELECT date_add('2021-10-31 00:00:00+02'::timestamptz,
                '1 day'::interval, 'Europe/Warsaw') ;
        date_add
------------------------
 2021-11-01 08:00:00+09
(1 row)

postgres=> SELECT date_subtract('2021-11-01 00:00:00+01'::timestamptz,
                '1 day'::interval, 'Europe/Warsaw') ;
     date_subtract
------------------------
 2021-10-31 07:00:00+09
(1 row)
```

□ Date_trunc

The immutability of functions of the form date_trunc(unit, timestamptz, text) has been changed from stable to immutable.

□ Erf / erfc

Add erf and erfc functions to compute the error function and complementary error.

Syntax 8 Erf / erfc functions

```
double precision erf(double precision)
double precision erfc(double precision)
```

Example 74 Erf / erfc functions

```
postgres=> SELECT erf(1.0), erfc(1.0) ;
        erf         |        erfc
--------------------+--------------------
 0.8427007929497149 | 0.15729920705028513
(1 row)
```

□ Generate_series

Add an interval version to the generate_series function.

Syntax 9 Generate_series function

```
timestamp without time zone generate_series(timestamp without time zone,
         timestamp without time zone, interval)
```

Example 75 Execution of generate_series function with specified interval

```
postgres=> SELECT * FROM
        generate_series('2001-10-22 00:00 -04:00'::timestamptz,
        '2001-10-24 00:00 -05:00'::timestamptz,
        '1 day'::interval, 'America/New_York') ;
    generate_series
------------------------
 2001-10-22 13:00:00+09
 2001-10-23 13:00:00+09
 2001-10-24 13:00:00+09
(3 rows)
```

□ Random_normal

Add function random_normal to generate normally distributed random numbers. Specify the mean and standard deviation. The extension tablefunc already provided the normal_rand function, which has been modified and provided as a system function.

Syntax 10 Random_normal function

```
double precision random_normal(mean double precision DEFAULT 0, stddev double
precision DEFAULT 1)
```

Example 76 Random_normal function

```
postgres=> SELECT random_normal(0.0, 1.0) ;
    random_normal
----------------------
 -0.1803089817060607
(1 row)
```

□ Pg_read_file / pg_read_binary_file

Add a version with a filename and the missing_ok parameter to indicate what to do if the file doesn't exist.

Syntax 11 Pg_read_file / Pg_read_binary_file functions

```
text pg_read_file(filename text, missing_ok boolean)
bytea pg_read_binary_file(filename text, missing_ok boolean)
```

Example 77 Pg_read_file function

```
postgres=# SELECT pg_read_file('PG_VERSION', true) ;
 pg_read_file
--------------
 16          +

(1 row)
```

□ Pg_split_walfile_name

This function returns the sequence number and timeline id from the WAL file name.

Syntax 12 Pg_split_walfile_name function

```
record pg_split_walfile_name(file_name text)
```

Example 78 Pg_split_walfile_name function

```
postgres=# SELECT * FROM pg_split_walfile_name ('000000010000000000000001') ;
 segment_number | timeline_id
----------------+-------------
              1 |           1
(1 row)
```

□ Pg_stat_get_backend_subxact

This function retrieves sub transaction information in a particular backend's cache.

Syntax 13 Pg_stat_get_backend_subxact function

```
record pg_stat_get_backend_subxact(bid integer)
```

Example 79 Pg_stat_get_backend_subxact function

```
postgres=# SELECT * FROM pg_stat_get_backend_subxact(5) ;
-[ RECORD 1 ]------+--
subxact_count      | 0
subxact_overflowed | f
```

□ Pg_input_is_valid

The pg_input_is_valid function can check whether a given value matches a data type. Overflow digits and improper date strings can be checked in advanced. The input data and data type are specified as strings.

Syntax 14 Pg_input_is_valid function

```
boolean pg_input_is_valid(string text, type text)
```

Example 80 Pg_input_is_valid function

```
postgres=> SELECT pg_input_is_valid('1234.48', 'numeric(5, 2)') ;
 pg_input_is_valid
-------------------
 f
(1 row)

postgres=> SELECT pg_input_is_valid('2022/12/32 09:30:60', 'timestamp') ;
 pg_input_is_valid
-------------------
 f
(1 row)
```

□ Pg_input_error_info

The pg_input_error_info function can get the error message, hint string, and error code when the input data is not appropriate for the specified data type.

Syntax 15 Pg_input_error_info function

```
record pg_input_error_info(value text, type_name text)
```

Example 81 Pg_input_is_valid / pg_input_error_info function

```
postgres=> SELECT message FROM pg_input_error_info('1234.48', 'numeric(5,
2)') ;
       message
-------------------------
 numeric field overflow
(1 row)

postgres=> SELECT hint FROM pg_input_error_info('2022/12/32 09:30:60',
'timestamp') ;
                      hint
-------------------------------------------------------
 Perhaps you need a different "datestyle" setting.
(1 row)
```

□ Pg_size_bytes

This function now accepts B (bytes) as the unit.

Example 82 Pg_size_bytes function

```
postgres=> SELECT pg_size_bytes('123456B') "B", pg_size_bytes('123456bytes')
"bytes" ;
-[ RECORD 1 ]-
B     | 123456
bytes | 123456
```

□ Pg_import_system_collations

Now available in Microsoft Windows environment.

□ System_user

The SYSTEM_USER function is a SQL standard reserved word and is a function that returns an authentication method and an authentication ID. Returns NULL for TRUST authentication.

Syntax 16 System_user function

```
text system_user()
```

Example 83 System_user function

```
$ psql -h dbsvr1 -d postgres -U demo
Password for user demo: <<PASSWORD>>
psql (16beta1)
Type "help" for help.

postgres=> SELECT system_user ;
    system_user
--------------------
 scram-sha-256:demo
(1 row)
```

□ Xmlserialize

Add the INDENT option (and NO INDENT) to pretty (or not pretty) XML documents.

Example 84 Specify the INDENT clause

```
postgres=> SELECT xmlserialize(DOCUMENT
        '<foo><bar><val x="y">42</val></bar></foo>' AS text INDENT) ;
      xmlserialize
--------------------------
 <foo>                  +
   <bar>                +
     <val x="y">42</val>+
   </bar>               +
 </foo>                 +

(1 row)

postgres=> SELECT xmlserialize(DOCUMENT
        '<foo><bar><val x="y">42</val></bar></foo>' AS text NO INDENT) ;
              xmlserialize
---------------------------------------
 <foo><bar><val x="y">42</val></bar></foo>
(1 row)
```

3.3. Configuration parameters

In PostgreSQL 16, the following parameters have been changed.

3.3.1. Added parameters

The following parameters have been added. [6e2775e, e5b8a4c, 1671f99, 3226f47, 5de94a0, 216a784, b577743, 51b5834, 1e8b617, 3d4fa22, 6de31ce, 9c0a0e2]

Table 22 Added parameters

Parameter name	Description (context)	Default value
debug_io_direct	Determine where direct I/O is used (postmaster)	''
createrole_self_grant	Options automatically specified in the GRANT statement (user)	''
enable_presorted_aggregate	Use the sort processing optimization function (user)	on
gss_accept_delegation	Whether to accept GSSAPI authentication delegation from the client (sighup)	off
icu_validation_level	ICU locale check log output level (user)	error
logical_replication_mode	Setting the transfer method for logical replication (user)	buffered
max_parallel_apply_workers_per_subscription	Maximum number of workers per subscription (sighup)	2
reserved_connections	Number of connections reserved for general users (postmaster)	0
scram_iterations	SCRAM authentication iteration count (user)	4096
send_abort_for_crash	Send a SIGABRT signal when the backend crashes (sighup)	off
send_abort_for_kill	Send SIGABRT signal when child process is stuck (sighup)	off
vacuum_buffer_usage_limit	Default shared buffer size used by VACUUM (user)	256kB

□ Debug_io_direct

Request the OS to use Direct I/O. Specify O_DIRECT on Linux/UNIX, F_NOCACHE on macOS, and FILE_FLAG_NO_BUFFERING on Windows when opening the file. For the parameter, specify multiple target areas separated by commas (,). [319bae9, d4e71df]

Table 23 Possible values

Value	Description
data	I/O to main data file
wal	I/O to WAL files
wal_init	I/O for WAL file initialization

This parameter is currently for developers (Developer Option category).

□ Send_abort_for_crash / Send_abort_for_kill

The postmaster process first sends a SIGQUIT signal and then a SIGKILL signal when a child process crashes. Send_abort_for_crash and send_abort_for_kill change the signal to SIGABRT by setting each to 'on'. [3226f47]

3.3.2. Modified Parameters

The following parameters have been changed in terms of setting range and choices. [9430fb4, 6c31ac0, 5352ca2, 0981846]

Table 24 Modified Parameters

Parameter name	Changes
archive_command	It can no longer be set at the same time as archive_library.
archive_library	It can no longer be set at the same time as archive_command.
wal_sync_method	It is now possible to specify fdatasync on NTFS in Windows environment.
shared_preload_libraries	Now also handled in single user mode (backported to PostgreSQL 15).
debug_parallel_query	Renamed from force_parallel_mode.

3.3.3. Parameters with default values changed

The following parameters have changed default values.

Table 25 Parameters with default values changed

Parameter name	PostgreSQL 15	PostgreSQL 16	Note
server_version	15.3	16beta1	
server_version_num	150003	160000	

3.3.4. Removed parameter

The following parameter have been removed. [cd4329d, 1118cd3]

Table 26 Removed parameters

Parameter name	Reason
vacuum_defer_cleanup_age	Removed due to difficulty in determining a reasonable setpoint and the availability of alternatives.
promote_trigger_file	Removed because process startup every 5 seconds was deemed useless and trigger files are no longer supported.

3.4. Utilities

Describes the major enhancements of utility commands.

3.4.1. Configure

Add the --with-segsize-blocks option to specify segment size in blocks. The --with-icu option has been removed, and the --without-icu option has been added to specify when not using ICU. [d3b111e, fcb21b3]

3.4.2. Createuser

Multiple options have been added to the createuser command. The --role option can still be used, but it is deprecated. [08951a7, 2dcd157, 381d19b]

Table 27 Add/Modify options

Option name	Short option	Description
--with-admin=*ROLE*	-a	ROLE will be a member of new role with admin option
--with-member=*ROLE*	-m	ROLE will be a member of new role
--valid-until=*TIMESTAMP*	-v	Password expiration
--bypassrls	-	Add Bypass RLS attribute
--no-bypassrls	-	Do not assign Bypass RLS attribute
--member-of=*ROLE*	-g	new role will be a member of ROLE (modified from --role option)

Example 85 Execute the createuser command

```
$ createuser --member=role1 user1 --echo
SELECT pg_catalog.set_config('search_path', '', false);
CREATE ROLE user1 NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT LOGIN ROLE role1;
```

3.4.3. Initdb

The following enhancements have been implemented to the initdb command: [27b6237, 30a53b7, 3e51b27, af3abca]

□ --locale-provider option

Default locale provider changed from libc to icu.

□ --icu-rules option

The --icu-rules option has been added to add a custom rule to the ICU locale.

Example 86 Execute initdb command with the --icu-rules option

```
$ initdb --icu-local=en_US --locale-provider=icu --icu-rules='&a < g' 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_US
  LC_COLLATE:  en_US.UTF-8
...
```

□ --set option

The --set option (shorthand option is -c) specifies the initial values of the parameters described in the postgresql.conf file. This option can be specified multiple times.

Example 87 initdb command with the --set option

```
$ initdb --set port=5433 --set listen_addresses='*' --locale-provider=libc --
no-locale --encoding=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 locale "C".
The default text search configuration will be set to "english".

Data page checksums are disabled.
...
```

□ Check locale command

The initdb command now exits successfully even if the 'locale' command does not exist. In previous versions, the initdb command would fail if the locale -a command failed.

3.4.4. Pgindent

The following options have been added/extended to the pgindent command. [b90f0b5, a1c4cd6, 068a243, b16259b]

Table 28 Add/change options

Option name	Change	Description
--build	Remove	Building the pg_bsd_indent command
--code-base	Remove	Specifying base directory for source code
--commit	Add	Target files changed in the specified commit
--excludes	Change	Can be specified multiple times
--help	Add	Output usage
--show-diff	Add	View changes
--silent-diff	Add	Exit with return value 2 if there are any changes

3.4.5. Pg_basebackup

It is now possible to specify 'long' for the level of the Zstandard compression method. This option improves the compression ratio for large data, but increases resource usage. [2820adf]

Example 88 Specifying the compression level to 'long'

```
$ pg_basebackup -D data.zstd --compress=zstd:long --format=tar
$ file data.zstd/*
data.zstd/backup_manifest: ASCII text
data.zstd/base.tar.zst:    Zstandard compressed data (v0.8+), Dictionary ID:
None
data.zstd/pg_wal.tar:      POSIX tar archive
```

3.4.6. Pg_bsd_indent

The BSD-style indentation tool the pg_bsd_indent command, developed in a separate project, has been incorporated into the core repository. [4e831f4]

Example 89 Execute the pg_bsd_indent command

```
$ pg_bsd_indent -v indent.c
There were 1273 output lines and 223 comments
(Lines with comments)/(Lines with code):  0.548
```

3.4.7. Pg_dump

The following enhancements have been implemented in the pg_dump command. [5e73a60, 0da243f, 84adc8e, 2820adf, 35ce24c, 2f80c95, a563c24, 5f53b42]

□ --compress option

The --compress option now accepts a compression method and compression level separated by a colon (:). The compression method can be none, gzip, lz4 or zstd. The value that can be specified for compression level differs depending on the compression method.

Table 29 Compression level range

Compression Method	Default level	Level Range	Note
none	-	-	
gzip	-1	1～9	
lz4	0	1～12	
zstd	3	-131072～22, long	

Example 90 Execute the pg_dump command

```
$ pg_dump --compress=gzip:9 --file=postgres.gz postgres
$ file postgres.gz
postgres.gz: gzip compressed data, max compression, from Unix, original size
11963
$ pg_dump --compress=lz4:12 --file=postgres.lz4 postgres
$ file postgres.lz4
postgres.lz4: LZ4 compressed data (v1.4+)
$ pg_dump --compress=zstd:22 --file=postgres.zstd postgres
$ file postgres.zstd
postgres.zstd: Zstandard compressed data (v0.8+), Dictionary ID: None
```

□ --large-objects option

The --blob and --no-blob options from previous versions have been changed to --large-objects and --no-large-objects. Legacy options are still available, but are deprecated.

Example 91 Help message of the pg_dump command

```
$ pg_dump --help
pg_dump dumps a database as a text file or to other formats.
...
Options controlling the output content:
  -a, --data-only              dump only the data, not the schema
  -b, --large-objects          include large objects in dump
  --blobs                      (same as --large-objects, deprecated)
  -B, --no-large-objects       exclude large objects in dump
  --no-blobs                   (same as --no-large-objects, deprecated)
  -c, --clean                  (drop) database objects before recreating
...
```

□ --table-and-children option

Basically, the same as the --table option. Includes tables matching the pattern and their child tables and partitions.

□ --exclude-table-and-children option

Basically, the same as the --exclude-table option. Exclude tables matching the pattern and their child tables and partitions.

□ --exclude-table-data-and-children option

Basically, the same as the --exclude-table-data option. Exclude data from tables matching the pattern and their child tables and partitions.

□ Lock acquisition

The pg_dump command executes a LOCK TABLE statement on the target table at startup. In PostgreSQL 16, multiple table names are described for one LOCK TABLE to reduce the amount of communication between the client and server.

3.4.8. Pg_receivewal / Pg_recvlogical

The SIGTERM signal has been added as a program exit condition in addition to the existing SIGINT signal. This change corresponds to the default value of the systemd exit signal (KillSignal). [8b60db7]

3.4.9. Pg_upgrade

Add the --copy option to the pg_upgrade command. This is the default behavior. Copies locale and encoding information from source to destination. [746915c, 9637bad]

3.4.10. Pg_verifybackup

Add the --progress option (shortened option -P). Report progress. [d07c294]

Example 92 Execute the pg_verifybackup command

```
$ pg_basebackup -D back.1 -v
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 0/5E000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: created temporary replication slot "pg_basebackup_120380"
pg_basebackup: write-ahead log end point: 0/5E0001B0
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: renaming backup_manifest.tmp to backup_manifest
pg_basebackup: base backup completed
$ pg_verifybackup back.1 --progress
887802/887802 kB (100%) verified
backup successfully verified
```

3.4.11. Pg_waldump

The following new features have been added to the pg_waldump command. [d497093, 4c8044c]

□ --save-fullpage option

Add the --save-fullpage option to specify the directory to save full page images. The specified directory must be empty. A full-page image is created with the following file name:

```
<lsn>.<tablespace>.<database>.<relation>.<block>_<forkname>
```

Table 30 File name explanation

Notation	Description	Example
<lsn>	LSN hexadecimal format	00000000-05AD6E48
<tablespace>	OID of the tablespace	1663
<database>	OID of the database	5
<relation>	Relation filenode	16388
<block>	Block number	20
<forkname>	Fork name	main

Example 93 Execute the pg_waldump command with the --save-fullpage option

```
$ pg_waldump --save-fullpage=$HOME/fullpage
        data/pg_wal/000000010000000000000004
rmgr: Heap        len (rec/tot):     65/    65, tx:        740, lsn: …
rmgr: Heap        len (rec/tot):     65/    65, tx:        740, lsn: …
…
$ ls -1 $HOME/fullpage
00000000-05A26608.1663.5.16388.0_vm
00000000-05A7D2E8.1663.5.2619.18_main
00000000-05A7E920.1663.5.2696.1_main
```

□ --timeline option

The --timeline option now accepts hexadecimal strings.

Example 94 Execute the pg_waldump command with the --timeline option

```
$ pg_waldump 000000010000000000000001  --timeline=0xA
rmgr: XLOG          len (rec/tot):    114/   114, tx:             0, lsn:
0/01000028, prev 0/00000000, desc: CHECKPOINT_SHUTDOWN redo 0/1000028; tli 1;
pr
ev tli 1; fpw true; xid 0:3; oid 10000; multi 1; offset 0; oldest xid 3 in DB
1; oldest multi 1 in DB 1; oldest/newest commit timestamp xid: 0/0; old
est running xid 0; shutdown
…
```

3.4.12. Psql

The following enhancements have been implemented to the psql command: [b0d8f2d, 31ae2aa, bd95816, 5b66de3, a45388d, d913928, 6f9ee74, 31ae2aa]

□ Command execution status

The execution status of \!, \g, \o, \w, and \copy metacommands can now be obtained with the following variables.

Table 31 Command execution status

Variable name	Description
SHELL_ERROR	'true' if command execution result is 0, 'false' if not 0
SHELL_EXIT_CODE	Command exit code

Example 95 Check command execution status

```
postgres=> \! ls postgresql.conf
postgresql.conf
postgres=> \echo :SHELL_ERROR
false
postgres=> \echo :SHELL_EXIT_CODE
0
postgres=> \! ls postgresql.conf.bad
ls: cannot access 'postgresql.conf.bad': No such file or directory
postgres=> \echo :SHELL_ERROR
true
postgres=> \echo :SHELL_EXIT_CODE
2
```

□ \d+ metacommand

Executing the \d+ meta command on a partitioned table will output "FOREIGN" if the partition is an external table.

Example 96 External table partition information

```
postgres=> CREATE FOREIGN TABLE remote1(c1 INT, c2 VARCHAR(10)) SERVER remsvr1 ;
CREATE FOREIGN TABLE
postgres=> CREATE TABLE part1(c1 INT, c2 VARCHAR(10)) PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> ALTER TABLE part1 ATTACH PARTITION remote1 FOR VALUES FROM (0) TO
(100) ;
ALTER TABLE
postgres=> \d+ part1
                                          Partitioned table "public.part1"
 Column |         Type          | Collation | Nullable | Default | Storage …
--------+-----------------------+-----------+----------+---------+-------- …
 c1     | integer               |           |          |         | plain   …
 c2     | character varying(10) |           |          |         | extended…
Partition key: RANGE (c1)
Partitions: remote1 FOR VALUES FROM (0) TO (100), FOREIGN
```

□ \bind metacommand

Add \bind metacommand to set values for bind variables in SQL statements. A semicolon (;) is not considered a line terminator, so specify something like \g.

Example 97 Binding variable settings

```
postgres=> SELECT $1, $2
postgres-> \bind 'value1' 'value2' \g
 ?column? | ?column?
----------+----------
 value1   | value2
(1 row)

postgres=> SELECT * FROM data1 WHERE c1=$1 \bind 100 \g
 c1  |  c2
-----+-------
 100 | data1
(1 row)
```

□ \pset metacommand

Add the xheader_width option to \pset metacommand. This option can limit the width of the header when outputting in extended table format. The values that can be specified are as follows.

Table 32 Possible values

Value	Description	Note
full	Widest line length (default)	
column	Truncate to Width of First Column	
page	Truncate to Terminal Page Width	
Numerical value	Truncate to Specified Number	

Example 98 Specifying the xheader_width option

```
postgres=> \x
Expanded display is on.
postgres=> CREATE TABLE data1(c1 char(10), c2 char(20)) ;
CREATE TABLE
postgres=> INSERT INTO data1 VALUES ('1234567890', '12345678901234567890') ;
INSERT 0 1
postgres=> \pset xheader_width
Expanded header width is 'full'.
postgres=> SELECT * FROM data1 ;
-[ RECORD 1 ]----------
c1 | 1234567890
c2 | 12345678901234567890

postgres=> \pset xheader_width column
Expanded header width is 'column'.
postgres=> SELECT * FROM data1 ;
-[ RECORD 1 ]
c1 | 1234567890
c2 | 12345678901234567890
```

□ \dpS metacommand

'S' can now be specified for \dp meta-command and \z meta-command. Along with this, the operation of the \dp or \z command has been changed from the previous version. Temporary objects were included in the list, and objects in the information_schema schema were excluded.

□ \watch metacommand

Add validation of entered values. Any non-numeric or negative value will result in an error. Specify zero (0) to rerun without an interval. Also, the number of reruns can now be specified. To limit the number of re-executions, specify the interval and the number of executions in the "i=interval c=number of times of execution" format.

Example 99 \watch metacommand with rerun count

```
postgres=> SELECT current_time ;
    current_time
--------------------
 16:58:33.012005+09
(1 row)

postgres=> \watch i=1 c=2
Fri 26 May 2023 04:58:34 PM JST (every 1s)

    current_time
--------------------
 16:58:34.850061+09
(1 row)

Fri 26 May 2023 04:58:35 PM JST (every 1s)

    current_time
--------------------
 16:58:35.850556+09
(1 row)
```

3.4.13. Vacuumdb

The following options have been added to the vacuumdb command. [7781f4e, 4211fbd, ae78cae]

□ --schema option

Specify the name of the schema to which the table to be VACUUMed belongs. The shorthand option is -n. This option can be specified multiple times. This option cannot be used with the --exclude-schema option.

□ --exclude-schema option

Specify the name of the schema to which the table that does not execute VACUUM belongs. The shorthand option is -N. This option cannot be used at the same time as the --schema option.

Example 100 vacuumdb command with specifying schema

```
$ vacuumdb -d postgres --schema=public
vacuumdb: vacuuming database "postgres"
$ vacuumdb -d postgres --exclude-schema=public
vacuumdb: vacuuming database "postgres"
$ vacuumdb -d postgres --schema=public --exclude-schema=public
vacuumdb: error: cannot vacuum all tables in schema(s) and exclude schema(s) at
the same time
```

□ --no-process-main option

VACUUM (PROCESS_MAIN FALSE) 文に対応するオプション--no-process-main が追加されました。

Example 101 With the --no-process-main option

```
$ vacuumdb --no-process-main --echo postgres
SELECT pg_catalog.set_config('search_path', '', false);
vacuumdb: vacuuming database "postgres"
RESET search_path;
SELECT c.relname, ns.nspname FROM pg_catalog.pg_class c
 JOIN  pg_catalog.pg_namespace  ns  ON  c.relnamespace  OPERATOR(pg_catalog.=)
ns.oid
 LEFT JOIN pg_catalog.pg_class t ON c.reltoastrelid OPERATOR(pg_catalog.=) t.oid
 WHERE c.relkind OPERATOR(pg_catalog.=) ANY (array['r', 'm'])
 ORDER BY c.relpages DESC;
SELECT pg_catalog.set_config('search_path', '', false);
VACUUM (PROCESS_MAIN FALSE, SKIP_DATABASE_STATS) pg_catalog.pg_proc;
...
```

□ --buffer-usage-limit option

Add the --buffer-usage-limit option for VACUUM (BUFFER_USAGE_LIMIT) statement.

Example 102 The vacuumdb command with --buffer-usage-limit option

```
$ vacuumdb --buffer-usage-limit=1MB --echo postgres
SELECT pg_catalog.set_config('search_path', '', false);
vacuumdb: vacuuming database "postgres"
...
```

3.5. Contrib modules

Describes new features related to the Contrib modules.

3.5.1. Auto_explain

The following features have been added to the auto_explain: [d4bfe41, 9d2d972]

□ Parameter log_parameter_max_length

Add the log_parameter_max_length parameter. This parameter limits the maximum log output width. The default value is -1, which imposes no limit. Specifying 0 for this parameter disables logging.

□ Output of the query ID

The query ID is output if the parameter auto_explain.log_verbose is set to 'on' and compute_query_id is set to 'on'.

Example 103 Output of the query ID

```
$ tail -5 data/log/ postgresql-2023-05-26_135719.log
LOG:  duration: 0.005 ms  plan:
        Query Text: SELECT 1;
        Result  (cost=0.00..0.01 rows=1 width=4)
          Output: 1
        Query Identifier: -1801652217649936326
```

3.5.2. Fuzzystrmatch

The daitch_mokotoff function has been added to the fuzzystrmatch module. This function is superior to the SOUNDEX function for non-English languages. [a290378]

Example 104 Execute the daitch_mokotoff function

```
postgres=> SELECT daitch_mokotoff('Müller') ;
daitch_mokotoff
-----------------
 {479000}
(1 row)
```

3.5.3. Ltree

Hyphens (-) are now allowed in labels. Previously, only letters, numbers, and underlines (_) were permitted. The maximum label size has been increased from 255 to 1,000 characters. [b1665bf]

3.5.4. Pageinspect

The following features have been added to the pageinspect: [1fd3dd2, 3581cbd, 428c0ca]

□ Bt_multi_page_stats function

Add function bt_multi_page_stats to print B-Tree index page summaries. This function is the multi-page version of the bt_page_stats function. Specify the first block number and number of pages.

Example 105 Execute the bt_multi_page_stats function

```
postgres=# SELECT * FROM bt_multi_page_stats('idx1_data1', 2, 1) ;
-[ RECORD 1 ]-+-----
blkno         | 2
type          | l
live_items    | 367
dead_items    | 0
avg_item_size | 16
page_size     | 8192
free_size     | 808
btpo_prev     | 1
btpo_next     | 4
btpo_level    | 0
btpo_flags    | 1
```

□ Brin_page_items function

The "empty" column has been added to the output of the brin_page_items function to indicate free blocks.

3.5.5. Pg_buffercache

The following functions have been added to the pg_buffercache module. [f3fa313, f3fa313]

□ Pg_buffercache_summary function

Add the pg_buffercache_summary function that can check buffer cache summary by SQL statement.

Example 106 Execute the pg_buffercache_summary function

```
postgres=# SELECT * FROM pg_buffercache_summary() ;
-[ RECORD 1 ]--+------------------
buffers_used   | 251
buffers_unused | 16133
buffers_dirty  | 62
buffers_pinned | 0
usagecount_avg | 3.3346613545816735
```

Table 33 Results of the pg_buffercache_summary function

Column name	Description	Note
buffers_used	Total number of buffers used	
buffers_unused	Number of free buffers	
buffers_dirty	Number of dirty buffers	
buffers_pinned	Number of pinned buffers	
usagecount_avg	Average amount of dirty buffers used	

□ Pg_buffercache_usage_counts function

Returns a summary of the state of all shared buffers. It is easier to check the information than by aggregating the pg_buffercache view.

Example 107 Execute the pg_buffercache_usage_counts function

```
postgres=# SELECT * FROM pg_buffercache_usage_counts() ;
 usage_count | buffers | dirty | pinned
-------------+---------+-------+--------
           0 |   16035 |     0 |      0
           1 |      76 |     6 |      0
           2 |      31 |     1 |      0
           3 |      13 |     6 |      0
           4 |       8 |     3 |      0
           5 |     221 |    48 |      0
(6 rows)
```

3.5.6. Pg_stat_statements

Updated how SQL statements are generalized. SET statements, CHECKPOINT statements, CREATE statements, etc. are collected as the same query ID regardless of case. [9ba37b2, daa8365]

Example 108 Select the pg_stat_statements view

```
postgres=> SET enable_seqscan = off ;
SET
postgres=> set enable_seqscan = off ;
SET
postgres=> CHECKPOINT;
CHECKPOINT
postgres=> checkpoint;
CHECKPOINT
postgres=> SELECT queryid, query FROM pg_stat_statements WHERE userid=16384 ;
       queryid        |          query
----------------------+-------------------------
 -4537744274534392533 | SET enable_seqscan = off
 -7184411855217750948 | CHECKPOINT
(2 rows)
```

3.5.7. Pg_walinspect

Add the pg_get_wal_block_info function. This function extracts a full-page image between specified LSNs. [c31cf1c, 9ecb134, 122376f]

Syntax 17 Pg_get_wal_block_info function

```
record  pg_get_wal_block_info(start_lsn  pg_lsn,  end_lsn  pg_lsn,  show_data
boolean)
```

Example 109 Execute the pg_get_wal_block_info function

```
postgres=# SELECT lsn, relfilenode, relblocknumber, forkname  FROM
              pg_get_wal_block_info('0/150A0A0', '0/23413F0', false) ;
    lsn     | relfilenode | relblocknumber | forkname
------------+-------------+----------------+----------
 0/150A0B0 |        2619 |              0 | main
 0/150C0B0 |        2619 |             16 | main
...
```

Executing this function will return the following information:

Table 34 Return value of the pg_get_wal_block_info function

Column name	Data type	Description	Note
start_lsn	pg_lsn	Start LSN	
end_lsn	pg_lsn	End LSN	
prev_lsn	pg_lsn	Previous LSN	
block_id	smallint	Block ID	
reltablespace	oid	OID of the tablespace	
reldatabase	oid	OID of the database	
relfilenode	oid	OID of the relation file node	
relforknumber	smallint	Relation fork number	
relblocknumber	bigint	Block number of the table	
xid	xid	Transaction ID	
resource_manager	text	Resource manager name	
record_type	text	WAL record type	
record_length	integer	WAL record length	
main_data_length	integer	Main data length	
block_data_length	integer	Block data length	
block_fpi_length	integer	Block FPI data length	
block_fpi_info	text[]	Block FPI information	
description	text	Description	
block_data	bytea	Block data	
block_fpi_data	bytea	FPI data	

3.5.8. Postgres_fdw

The following enhancements have been implemented in the postgres_fdw module. [97da482, 8ad51b5, 983ec23]

□ COPY FROM statement

The COPY FROM statement now uses the foreign table's batch_size option. Below is the log using the log_statement parameter for the instance that owns the table. It can be confirmed that multiple records are inserted in batch.

Example 110 Execution log on the external table side

```
LOG:  statement: START TRANSACTION ISOLATION LEVEL REPEATABLE READ
LOG:  statement: TRUNCATE public.data1 CONTINUE IDENTITY RESTRICT
LOG:  statement: COMMIT TRANSACTION
LOG:  statement: START TRANSACTION ISOLATION LEVEL REPEATABLE READ
LOG:  execute pgsql_fdw_prep_7: INSERT INTO public.data1(c1, c2) VALUES ($1, $2), ($3, $4), ($5, $6), ($7, $8)
DETAIL:  parameters: $1 = '1', $2 = 'data1', $3 = '2', $4 = 'data2', $5 = '3', $6 = 'data3', $7 = '4', $8 = 'data4'
LOG:  statement: DEALLOCATE pgsql_fdw_prep_7
LOG:  statement: COMMIT TRANSACTION
```

□ Analyze_sampling option

The analyze_sampling option has been added to FOREIGN SERVER and FOREIGN TABLE. Previous versions fetched the entire foreign table record for statistics. The load on the remote database can be reduced by using the TABLESAMPLE clause when retrieving information from external tables. The default value is auto, which is assumed to be random for PostgreSQL < 9.5 and bernoulli for PostgreSQL 9.5 and higher. The values that can be set are as follows.

Table 35 Possible values for the analyze_sampling option

Value	Description
Off	Do not sample.
auto	Determine sampling method by version (default value)
random	Sampling is done using the random function.
system	Sampling is done using the TABLESAMPLE SYSTEM clause.
bernoulli	Sampling is done using the TABLESAMPLE BERNOULLI clause.

Example 111 Execute the ANALYZE statement

```
postgres=> CREATE FOREIGN TABLE foreign1(c1 INT, c2 VARCHAR(10)) SERVER remsvr1
           OPTIONS (analyze_sampling 'bernoulli', table_name 'data1') ;
CREATE FOREIGN TABLE
postgres=> ANALYZE VERBOSE foreign1 ;
INFO:  analyzing "public.foreign1"
INFO:  "foreign1": table contains 1000000 rows, 30000 rows in sample
ANALYZE
```

No sampling is performed if no ANALYZE statement is executed on the source table of the external table.

□ Parallel_abort option

Postgres_fdw aborts remote transactions one at a time if it aborts the local transaction. Setting the option parallel_abort to 'true' aborts remote transactions in parallel, improving performance. The default value for this option is 'false'.

Hewlett Packard
Enterprise

URL list

The following websites are references to create this material.

□ Release Notes

https://www.postgresql.org/docs/16/release-16.html

□ Commitfests

https://commitfest.postgresql.org/

□ PostgreSQL 16 Manual

https://www.postgresql.org/docs/16/index.html

□ Git

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

□ GitHub

https://github.com/postgres/postgres

□ Announce of the PostgreSQL 16

https://www.postgresql.org/about/news/postgresql-16-beta-1-released-2643/

□ PostgreSQL 16 Open Items

https://wiki.postgresql.org/wiki/PostgreSQL_16_Open_Items

□ Michael Paquier - PostgreSQL committer

https://paquier.xyz/

□ 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-16.html

□ SQL Notes

https://sql-info.de/postgresql/postgresql-16/articles-about-new-features-in-postgresql-16.html

□ Slack - postgresql-jp

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

Hewlett Packard
Enterprise

Change history

Change history

Version	Date	Author	Description
0.1	Apr 25, 2023	Noriyoshi Shinoda	Create an internal review version. Reviewers: Tomoo Takahashi Akiko Takeshima (Hewlett Packard Enterprise Japan)
1.0	May 28, 2023	Noriyoshi Shinoda	Verification completed in PostgreSQL 16 Beta 1 environment.