

2023 年 6 月 99 日

## PostgreSQL 16 新機能検証結果 (Beta 1)

日本ヒューレット・パカード合同会社  
篠田典良

## 目次

目次.....	2
1. 本文書について.....	5
1.1. 本文書の概要.....	5
1.2. 本文書の対象読者 .....	5
1.3. 本文書の範囲.....	5
1.4. 本文書の対応バージョン .....	5
1.5. 本文書に対する質問・意見および責任 .....	6
1.6. 表記 .....	6
2. PostgreSQL 16 における変更点概要 .....	8
2.1. 大規模環境に対応する新機能 .....	8
2.2. 信頼性向上に関する新機能.....	8
2.3. 運用性向上に関する新機能.....	8
2.4. プログラミングに関する新機能 .....	9
2.5. 将来の新機能に対する準備.....	9
2.6. 非互換 .....	10
2.6.1. サポート終了.....	10
2.6.2. 起動ユーザー属性.....	11
2.6.3. アーカイブ.....	11
2.6.4. プロモーション .....	12
2.6.5. エクステンション.....	12
2.6.6. ASCII 専用文字列 .....	12
2.6.7. ALTER DEFATULT PRIVILEGE 文 .....	13
2.6.8. CREATEROLE 属性 .....	13
2.6.9. RESET 文 .....	14
2.6.10. pg_stat_get_backend_idset 関数.....	14
2.6.11. PL/Python.....	15
2.6.12. postmaster.....	16
2.6.13. VACUUM.....	16
3. 新機能解説.....	17
3.1. アーキテクチャの変更 .....	17
3.1.1. システムカタログの変更.....	17
3.1.2. ロジカル・レプリケーションの拡張 .....	20
3.1.3. パラレル・クエリー .....	21
3.1.4. 設定ファイル.....	22



3.1.5. データ型 .....	24
3.1.6. WAL sender プロセス .....	26
3.1.7. トリガー .....	26
3.1.8. ログ .....	27
3.1.9. libpq .....	27
3.1.10. 待機イベント .....	28
3.1.11. 事前定義ロール .....	29
3.1.12 VACUUM .....	29
3.1.14. MATERIALIZED VIEW .....	30
3.1.13. Meson 対応 .....	30
3.1.14. UNICODE .....	30
3.1.15. LLVM .....	30
3.1.16. ロケール .....	31
3.1.17. PL/pgSQL .....	31
3.1.6. WAL .....	31
3.1.6. 確認中 .....	31
3.1.6. 確認中 .....	31
3.1.6. 確認中 .....	31
3.1.6. 確認中 .....	32
3.2. SQL 文の拡張 .....	33
3.2.1. ALTER ROLE .....	33
3.2.2. COPY .....	34
3.2.3. CREATE ROLE/USER .....	34
3.2.4. CREATE STATISTICS .....	34
3.2.5. CREATE TABLE .....	35
3.2.6. GRANT .....	36
3.2.7. REINDEX .....	38
3.2.8. VACUUM .....	38
3.2.9. サブクエリー .....	39
3.2.10. ウィンドウ関数 .....	39
3.2.11. SELECT DISTINCT .....	40
3.2.12. 関数 .....	41
3.2.X. EXPLAIN .....	46
3.3. パラメーターの変更 .....	47
3.3.1. 追加されたパラメーター .....	47
3.3.2. 変更されたパラメーター .....	49



3.3.3. デフォルト値が変更されたパラメーター .....	50
3.3.4. 削除されたパラメーター .....	50
3.4. ユーティリティの変更 .....	51
3.4.1. configure .....	51
3.4.2. createuser .....	51
3.4.3. pgindent .....	52
3.4.4. pg_dump .....	52
3.4.5. pg_receivewal / pg_recvlogical .....	53
3.4.6. psql .....	53
3.4.7. vacuumdb .....	56
3.5. Contrib モジュール .....	57
3.5.1. auto_explain .....	57
3.5.2. ltree .....	57
3.5.3. pageinspect .....	58
3.5.4. pg_buffercache .....	58
3.5.5. pg_stat_statements .....	59
3.5.6. pg_upgrade .....	60
3.5.7. pg_waldump .....	60
3.5.8. pg_walinspect .....	61
3.5.9. postgres_fdw .....	62
3.5.X. vacuumdb .....	64
3.5.X. amcheck .....	64
参考にした URL .....	65
変更履歴 .....	66



## 1. 本文書について

### 1.1. 本文書の概要

本文書はオープンソース RDBMS である PostgreSQL 16 (16.0) Beta 1 の主な新機能について検証した文書です。

### 1.2. 本文書の対象読者

本文書は、既にある程度 PostgreSQL に関する知識を持っているエンジニア向けに記述しています。インストール、基本的な管理等は実施できることを前提としています。

### 1.3. 本文書の範囲

本文書は PostgreSQL 15 (15.3) と PostgreSQL 16 (16.0) Beta 1 の主な差分を記載しています。原則として利用者が見て変化がわかる機能について調査しています。すべての新機能について記載および検証しているわけではありません。特に以下の新機能は含みません。

- バグ解消
- 内部動作の変更によるパフォーマンス向上
- レグレッション・テストの改善
- psql コマンドのタブ入力による操作性改善
- pgbench コマンドの改善
- ドキュメントの改善、ソース内の Typo 修正
- 動作に変更がないリファクタリング

### 1.4. 本文書の対応バージョン

本文書は以下のバージョンとプラットフォームを対象として検証を行っています。

表 1 対象バージョン

種別	バージョン
データベース製品	PostgreSQL <b>15.3</b> (比較対象) PostgreSQL 16 (16.0) Beta 1 (2023/06/99 99:99:99)
オペレーティング・システム	Red Hat Enterprise Linux 8 Update <b>5</b> (x86-64)
Configure オプション	<b>--with-ssl=openssl --with-python --with-lz4 --with-zstd</b> <b>--with-llvm --with-icu</b>

## 1.5. 本文書に対する質問・意見および責任

本文書の内容は日本ヒューレット・パカード合同会社の公式見解ではありません。また内容の間違いにより生じた問題について作成者および所属企業は責任を負いません。本文書で検証した仕様は後日変更される場合があります。本文書に対するご意見等ありましたら作成者 篠田典良 (Mail: [noriyoshi.shinoda@hpe.com](mailto:noriyoshi.shinoda@hpe.com)) までお知らせください。

## 1.6. 表記

本文書内にはコマンドや SQL 文の実行例および構文の説明が含まれます。実行例は以下のルールで記載しています。

表 2 例の表記ルール

表記	説明
#	Linux root ユーザーのプロンプト
\$	Linux 一般ユーザーのプロンプト
太字	ユーザーが入力する文字列
postgres=#	PostgreSQL 管理者が利用する psql コマンド・プロンプト
postgres=>	PostgreSQL 一般ユーザーが利用する psql コマンド・プロンプト
<u>下線部</u>	特に注目すべき項目
<<以下省略>>	より多くの情報が出力されるが文書内では省略していることを示す
<<途中省略>>	より多くの情報が出力されるが文書内では省略していることを示す
<<パスワード>>	パスワードの入力を示す

構文は以下のルールで記載しています。



表 3 構文の表記ルール

表記	説明
斜体	ユーザーが利用するオブジェクトの名前やその他の構文に置換
[]	省略できる構文であることを示す
{A   B}	A または B を選択できることを示す
...	旧バージョンと同一である一般的な構文



## 2. PostgreSQL 16 における変更点概要

PostgreSQL 16 には 200 以上の新機能が追加されました。本章では代表的な新機能と利点の概要について説明します。新機能の詳細は「3. 新機能解説」で説明します。

### 2.1. 大規模環境に対応する新機能

大規模環境に適用できる以下の機能が追加されました。

□ パラレル・クエリーの拡張

...

□ 圧縮アルゴリズムの拡張

...

□ 稼働統計の拡張

...

### 2.2. 信頼性向上に関する新機能

信頼性を向上させるために以下の拡張が実装されました。

□ アーカイブ・ライブラリ

...

□ チェックポイントのログ

...

### 2.3. 運用性向上に関する新機能

運用性を向上できる以下の機能が追加されました。



□ パラメータ・ファイルの拡張

`pg_hba.conf` ファイルに記述されるユーザー名とデータベース名に正規表現が使えるようになりました。また `pg_hba.conf` と `pg_ident.conf` ファイルには `postgresql.conf` と同じように外部のファイルをインクルードすることができるようになりました。

□ セキュリティ

事前定義ロール `pg_maintain` が追加されました。このロールを付与されたユーザーは所有者以外のテーブルに対しても `ANALYZE` 文、`VACUUM` 文、`REINDEX` 文等を実行できます。またテーブル毎に上記 `SQL` 文の実行を他のユーザーに対して許可できるようになりました。

## 2.4. プログラミングに関する新機能

`SQL` 文に以下の機能が追加されました。

□ JSON 関連

...

□ MERGE 文

...

## 2.5. 将来の新機能に対する準備

将来のバージョンで提供される機能の準備が進みました。

□ テーブル・アクセス・メソッドの変更

...

## 2.6. 非互換

PostgreSQL 16 は PostgreSQL 15 から以下の仕様が変更されました。

### 2.6.1. サポート終了

PostgreSQL 16 では以下のプラットフォームやツール向けのサポート・バージョンが変更されました。

サポートが終了したプラットフォームとツールは以下の通りです。

- HP-UX
- HP/Intel Itanium プロセッサ
- Windows 10 より前の Microsoft Windows
- Microsoft Visual Studio 2013

PostgreSQL 16 のビルドに必要なコンポーネントのサポート・バージョンの変化は以下の通りです。

- Bison 2.3 以降
- Flex 2.5.35 以降
- Perl 5.14 以降

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=9db300ce6e38411144f1e36dba345a5f91bbdee4>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=495ed0ef2d72a6a74def296e042022479d5d07bd>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6203583b72b58272010f8d06999811ff39922acf>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=8efefa748777baf6a61a0e5ec3858a2b90fd8e84>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=8b878bffa8d8ac7e13508025c3ca5e6101e3a5e8>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b086a47a270fba133969e78f1fb9e264725d97ae>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=4c1532763a00c21cbb737bc3855e9a31374b119d>

## 2.6.2. 起動ユーザー属性

ブートストラップ・ユーザーから SUPERUSER 属性を削除できなくなりました。

### 例 1 ブートストラップ・ユーザーの属性変更

```
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: bootstrap user must be superuser
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e530be2c5ce77475d56ccf8f4e0c4872b666ad5f>

## 2.6.3. アーカイブ

アーカイブ WAL について以下の非互換があります。

### □ パラメーター設定

パラメーター `archive_command` と `archive_library` は同時に値を設定することができなくなりました。PostgreSQL 15 では `archive_library` が優先されていました。同時に設定するとログに以下のエラーが出力され、WAL アーカイブは出力されなくなります。

### 例 2 アーカイブ設定の重複エラー

```
$ tail -2 data/log/postgresql-2022-11-15_203015.log
FATAL:  both archive_command and archive_library set
DETAIL:  Only one of archive_command, archive_library may be set.
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d627ce3b706de16279d8eb683bfeda34ad7197fe>

~~□ アーカイブ・モジュール~~

~~Contrib モジュール basic\_archive は名前が basic\_wal\_module に変更されました。~~

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=0ad3c60caf5f77edfefaf8850fbb5ea4fe28640>

⇒ Revert

## 2.6.4. プロモーション

トリガー・ファイルの作成によるスタンバイ・データベースのプロモーション方法はサポートされなくなりました。これに伴いパラメーター `promote_trigger_file` は削除されました。プロモーションを実行するには `pg_ctl promote` コマンドまたは `pg_promote` 関数を使用します。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=cd4329d9393f84dce34f0bd2dd936adc8ffaa213>

## 2.6.5. エクステンション

以前はエクステンション・スクリプトが `CREATE OR REPLACE` 文を実行し、拡張に属さない既存のオブジェクトが存在する場合、オブジェクトを上書きしていました。すでにエクステンションに属していない限り、既存のオブジェクトの `CREATE OR REPLACE` 文の実行は禁止されます。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b9b21acc766db54d8c337d508d0fe2f5bf2daab0>

## 2.6.6. ASCII 専用文字列

GUC `application_name` や `cluster_name` 等、ASCII 専用の文字列に対して非 ASCII 文字を指定した場合の変換ルールが変更されました。従来はバイト単位にクエスチョン記号 (?) で変換していましたが、PostgreSQL 16 では 16 進数の文字列に変換されます。

表 4 ASCII 変換ルール

変更前 (UTF-8)	PostgreSQL 15	PostgreSQL 16
Abc 漢字	Abc?????	Abc¥xe6¥xbc¥xa2¥xe5¥xad¥x97



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=45b1a67a0fcb3f1588df596431871de4c93cb76f>

## 2.6.7. ALTER DEFATULT PRIVILEGE 文

仕様変更？

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=48a257d444a787941ba3da24d65e6cbe31461d0a>

<https://pgpedia.info/postgresql-versions/postgresql-16.html>

## 2.6.8. CREATEROLE 属性

従来のバージョンでは CREATEROLE 属性を持つユーザーは他のユーザーに対してほとんどの変更を実行することができました。PostgreSQL 16 では CREATEROLE 属性を持っていても、他のユーザーに対して WITH ADMIN オプションを持たないロールを付与することはできなくなりました。下記の例にある最初の GRANT 文は PostgreSQL 15 では成功します。

### 例 3 GRANT 文の失敗

```
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:  must have admin option on role "pg_monitor"
postgres=> \connect postgres 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
```



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=cf5eb37c5ee0cc54c80d95c1695d7fca1f7c68cb>

### 2.6.9. RESET 文

下記のパラメーターに対する RESET 文は失敗します。

- transaction\_read\_only
- transaction\_deferrable
- seed
- transaction\_isolation

#### 例 4 RESET 文の失敗

```
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=385366426511399a91da327c0f04765bbcf5322>

### 2.6.10. pg\_stat\_get\_backend\_idset 関数

pg\_stat\_get\_backend\_idset 関数はバックエンド・プロセスの ID を返すようになりました。旧バージョンでは 1 から始まる順番を返していました。



#### 例 5 PostgreSQL 15 の動作

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

#### 例 6 PostgreSQL 16 の動作

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d7e39d72ca1c6f188b400d7d58813ff5b5b79064>

### 2.6.11. PL/Python

{インストール先}/lib/pgxs/src/pl/plpython ディレクトリは作成されなくなりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=7d5852ca83f4103d806ee7501f70b9354345f65d>



## 2.6.12. postmaster

-T オプションは SIGSTOP シグナルではなく SIGABRT シグナルを送信します。また -n オプションは削除されました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=51b5834cd53f0bd068729043b55f7da3ca6bb15f>

## 2.6.13. VACUUM

仕様変更？

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c3ffa731a5f99c4361203015ce2219d209fea94c>



## 3. 新機能解説

### 3.1. アーキテクチャの変更

#### 3.1.1. システムカタログの変更

以下のシステムカタログやビューが変更されました。

表 5 追加されたシステムカタログ／ビュー

カタログ／ビュー名	説明

表 6 列が追加されたシステムカタログ／ビュー

カタログ／ビュー名	追加列名	データ型	説明
pg_auth_members	oid	oid	Object ID
	inherit_option	boolean	継承の許可
	set_option	boolean	SET ROLE 文の許可
pg_db_role_setting	setuser	bool[]	
pg_hba_file_rules	rule_number	integer	ルール番号
	file_name	text	設定ファイル名
pg_ident_file_mappings	map_number	integer	マッピング番号
	file_name	text	設定ファイル名
pg_prepared_statements	result_types	regtype[]	ステートメントから返される列のタイプ
pg_stat_subscription	leader_pid	integer	パラレル適用リーダー・プロセスの ID
pg_stat_*_indexes	last_idx_scan	timestamp with time zone	インデックス・アクセス最終時刻
pg_stat_*_tables	last_seq_scan	timestamp with time zone	シーケンシャル・アクセス最終時刻
	last_idx_scan	timestamp with time zone	インデックス・アクセス最終時刻
pg_subscription	suborigin	text	PUBLICATION へのデータ送信依頼種別

カタログ／ビュー名	追加列名	データ型	説明

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d540a02a724b9643205abce8c5644a0f0908f6e3>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=84ad713cf85aeffee5dd39f62d49a1b9e34632da>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c037471832e1ec3327f81eebbd8892e5c1042fe0>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=366283961ac0ed6d89014444c6090f3fd02fce0a>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6566133c5f52771198aca07ed18f84519fac1be7>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e3ce2de09d814f8770b2e3b3c152b7671bcd83f>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c591300a8f54d9711157d9a8866f022a257ec4ee>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=blobdiff;f=doc/src/sgml/system-views.sgml;h=d38b42c5cd5c5c9c1712637c80b45560526985d3;hp=7c716fe32762a86428b04dce9cc47db3020602cc;hb=a54b658ce77b6705eb1f997b416c2e820a77946c;hpb=d09deb9bde6b9faabd30e887eff4493331d6424>

Add connection active, idle time to pg\_stat\_activity

<https://commitfest.postgresql.org/39/3405/>

表 7 列が削除されたシステムカタログ／ビュー

カタログ名	削除列名	説明

表 8 内容が変更されたシステムカタログ／ビュー

カタログ／ビュー名	説明
<b>pg_class</b>	<del>relfilenode 列のデータ型が bigint に変更されました。また列の定義順が変更されました。Revert</del>
pg_locks	locktype 列に applytransaction が出力されるようになりました。
pg_subscription	substream 列が bool 型から char 型に変更されました。

pg_stat_subscription	SUBSCRIPTION 作成直後にレコードが作成されます。従来は最初の統計情報が受信された時点で作成されていました。
----------------------	---

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=216a784829c2c5f03ab0c43e009126cbb819e9b2>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=05d4cbf9b6ba708858984b01ca0fc56d59d4ec7c>

⇒ Revert

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e0b0142959066f3a9de1c6867f4ec8d41d104f33>

追加されたシステムカタログやビューから、主要なものについて詳細を以下に記載します。

#### □ pg\_ident\_file\_mappings

pg\_ident.conf ファイルの内容を SQL 文から検索するためのビューです。

**表 9 pg\_ident\_file\_mappings カタログ**

列名	データ型	説明
line_number	integer	pg_ident.conf ファイルの行番号
map_name	text	マップ名
sys_name	text	システム名
pg_username	text	PostgreSQL ユーザー名
error	text	エラー・メッセージ

#### 例 7 pg\_ident\_file\_mappings ビューの検索

```
postgres=# SELECT * FROM pg_ident_file_mapping ORDER BY 1 ;
 line_number | map_name | sys_name | pg_username | error
-----+-----+-----+-----+-----
          43 | map1    | sysname1 | pgname1    |
          44 | map2    | sysname2 | pgname2    |
(2 rows)
```

### 3.1.2. ロジカル・レプリケーションの拡張

ロジカル・レプリケーションには以下の新機能が実装されました。

#### □ パラレル適用

従来は大規模なトランザクションは一時ファイルに保存された後に適用されていたが、複数のワーカー・プロセスによってトランザクションの差分を直接適用できるようになりました。この機能を有効にするためには `CREATE SUBSCRIPTION` 文のオプション `streaming` に `parallel` を指定します。これに伴い `pg_subscription` カタログの `substream` 列は `bool` 型から `char` 型に変更され、`stream` オプションが `parallel` に指定された場合は 'p' が出力されます。サブスクリプションが使用するワーカー・プロセスの最大値はパラメーター `max_parallel_apply_workers_per_subscription` で決定されます。

#### 例 8 パラレル適用の設定

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=216a784829c2c5f03ab0c43e009126cbb819e9b2>

#### □ SUBSCRIPTION のオプション

SUBSCRIPTION のオプションに `origin` を指定できるようになりました。このパラメーターには PUBLICATION に対して送信要求を行う変更の種類を指定します。このオプション情報を格納するために `pg_subscription` カタログに `suborigin` 列が追加されました。

表 10 origin オプション設定値

設定値	説明	備考
none	SUBACRIPTION は PUBLICATION に対してオリジンに関連しない変更のみを要求する。	
any	PUBLICATION はオリジンに関係なくすべての変更内容を SUBSCRIPTION に送信する。	デフォルト値

例 9 origin オプションの設定

```
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)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=366283961ac0ed6d89014444c6090f3fd02fce0a>

#### □ REPLICA IDENTITY

インデックスが INVALID な状態でも ALTER TABLE REPLICA IDENTITY USING 文が成功するようになりました。これは pg\_dump コマンドがテーブル定義を出力する順番によるエラーを回避するための修正です。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c9f7f926484d69e2806e35343af7e472fadfed7>

### 3.1.3. パラレル・クエリー

string\_agg 関数および array\_agg 関数の実行時に集約処理を並列に実行できるようになりました。パラレル・クエリーが実行される場合には実行計画に Partial Aggregate が出力されます。



#### 例 10 string\_agg 関数の実行

```
postgres=> EXPLAIN SELECT string_agg(c2, ':') FROM data1 ;
               QUERY PLAN
-----
Finalize Aggregate  (cost=13785.56..13785.57 rows=1 width=32)
  -> Gather  (cost=13785.34..13785.55 rows=2 width=32)
        Workers Planned: 2
        -> Partial Aggregate  (cost=12785.34..12785.35 rows=1 width=32)
              -> Parallel Seq Scan on data1  (cost=0.00..11743.67 rows=416667
width=2)
(5 rows)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=67c5b8840fcad07eeecd9af517b9b1ff09a3cf8e>

#### ☐ Parallel Hash Full Join

<https://commitfest.postgresql.org/39/2903/>

### 3.1.4. 設定ファイル

いくつかの設定ファイルについて記述方法が改善されました。

#### ☐ ホスト名とユーザー名の正規表現

pg\_hba.conf ファイルではホスト名とユーザー名を正規表現で記述できるようになりました。スラッシュ (/) から始まるユーザー名とデータベース名は正規表現とみなされます。

#### 例 11 正規表現を使った pg\_hba.conf ファイルのユーザー名とデータベース名

#	TYPE	DATABASE	USER	ADDRESS	METHOD
host		/^demodb%d{1,3}	all	192.168.1.0/24	scram-sha-256
host		demodb4	/^user.*\$	192.168.1.0/24	scram-sha-256

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=8fea86830e1d40961fd3cba59a73fca178417c78>

正規表現は pg\_ident.conf ファイルのデータベース・ユーザー項目にも使用できます。



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=efb6f4a4f9b627b9447f5cd8e955d43a7066c30c>

□ ファイルのインクルード

pg\_hba.conf ファイルと pg\_ident.conf ファイルには postgresql.conf ファイルと同様に他のファイルをインクルードすることができるようになりました。

表 11 追加された構文

構文	説明
include ファイル名	ファイルをインクルードする
include_if_exists ファイル名	ファイルが存在すればインクルードする
include_dir ディレクトリ名	指定されたディレクトリ以下の全ファイルをインクルードする

例 12 ファイルをインクルードする設定

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

pg\_hba\_file\_rules カタログ、pg\_ident\_file\_mappings カタログには設定ファイル名を示す file\_name 列が追加されました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a54b658ce77b6705eb1f997b416c2e820a77946c>

□ pg\_ident.conf ファイル

PostgreSQL ユーザー名部分に対して以下のように pg\_hba.conf ファイルと同様の設定が許可されます。

- 全データベース・ユーザーを意味する all
- プラス (+) を使ったメンバーチェック
- データベース・ユーザー名の正規表現

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=efb6f4a4f9b627b9447f5cd8e955d43a7066c30c>

[d8e955d43a7066c30c](#)

### 3.1.5. データ型

データ型には以下の拡張が実装されました。

#### □ +infinity 表記

date 型、timestamp 型、timestamp with time zone 型に+infinity 値を指定できるようになりました。+infinity は infinity と同じとみなされます。

#### 例 13 +Infinity 指定

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

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=2ceea5adb02603ef52579b568ca2c5aebd87358>

#### □ 整数リテラル

整数リテラルに 16 進数、8 進数、2 進数の表現ができるようになりました。それぞれ以下のように表現します。先頭の'0'の次に底を示す記号を指定します。大文字と小文字は区別されません。

表 12 整数リテラルの記述

指定方式	例
16 進数	0x42F
8 進数	0o273
2 進数	0b100101





#### 例 14 整数リテラルの表記方法

```
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)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6fcda9aba83449082124825b6d375c0a61e21c42>

#### □ numeric 型

numeric 型は 16 進数、8 進数、2 進数の表記をサポートします。ただし小数点以下の記述はできません。

#### 例 15 numeric 型の 16 進数表記

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

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6dfacbf72b53b775e8442a7fd2fca7c24b139773>



### 3.1.6. WAL sender プロセス

論理レプリケーション環境の WAL sender プロセスのプロセス名にデータ提供元 (PUBLICATION 側) のデータベース名が出力されるようになりました。

例 16 WAL sender プロセス名

```
$ ps -ef|grep walsender | grep -v grep
postgres    23488    23127    0 18:10 ?                00:00:00 postgres: walsender
postgres    demodb    ::1 (48688)  START_REPLICATION
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=af205152ef57cf54730c38342878cb6b9f8ce7a1>

### 3.1.7. トリガー

トリガーには以下の拡張が実装されました。

#### ☐ TRUNCATE トリガー

外部テーブルに対する TRUNCATE 文実行時に TRUNCATE トリガーが実行されるようになりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3b00a944a9b3847fb02dae7c9ea62fe0b211b396>

#### ☐ イベント・トリガー

ALTER MATERIALIZED VIEW 文の実行でイベント・トリガーが実行されるようになりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=93f2349c36a7555a81d5e26edf9a0213fb3d323a>

#### ☐ ログイン・トリガー

<https://commitfest.postgresql.org/39/2900/>

### 3.1.8. ログ

いくつかの場面でログに出力される情報が追加されました。

#### □ チェックポイント

パラメーターlog\_checkpoint を on に設定した場合のログに LSN の情報が追加されます。

#### 例 17 チェックポイントのログ

```
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=62c46eee2279eb0300ab7ffe393d0d0dcfab157>

#### □ 自動 VACUUM

自動 VACUUM のログにフリーズされたテーブルの情報が追加されるようになりました。

#### 例 18 自動 VACUUM のログ

```
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
<<<以下省略>>>
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d977ffd923d207164eef78ed107d5293aee6c660>

### 3.1.9. libpq

CopySendEndOfRow API にコールバック関数を指定できるようになりました。この機

能はエクステンションが COPY TO 文を実行するのに役立ちます。COPY FROM 文については既に同様のコールバック機能が提供されています。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=9fcdf2c787ac6da330165ea3cd50ec5155943a2b>

Add non-blocking version of PQcancel  
<https://commitfest.postgresql.org/38/3511/>

### 3.1.10. 待機イベント

以下の待機イベントが追加されました。

表 13 追加された待機イベント

イベント名	タイプ	説明
DSMAllocate	IO	動的共有メモリーの取得待ち
LogicalParallelApplyMain	Activity	パラレル適用待ち
LogicalParallelApplyState Change	Activity	パラレル適用のステータス変更待ち
LogicalRepLauncherDSA	LWLock	ランチャー・プロセスの DSA メモリー・アロケータへのアクセス待ち
LogicalRepLauncherHash	LWLock	ランチャー・プロセスの共有ハッシュ・テーブルへのアクセス待ち
RelationMapReplace	IO	リレーションマップ・ファイルの置き換え待ち。 RelationMapSync から変更
SpinDelay	Timeout	pg_usleep 実行待ち時間

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=7bae3bbf62d63cdd49ae4ca4a851cef0cdbe6ab5>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=216a784829c2c5f03ab0c43e009126cbb819e9b2>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=5a3a95385bd5a8f1a4fd50545b7efe9338581899>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d8cd0c6c95c0120168df93aae095df4e0682a08a>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=92daeca45df6551dd85f9>

[2f7369eaa57a35fb8a9](#)

待機イベント `PgStatMain`、`LogicalChangesRead`、`LogicalChangesWrite`、`LogicalSubxactRead`、`LogicalSubxactWrite` は削除されました。

### 3.1.11. 事前定義ロール

PostgreSQL 16 には以下の事前定義ロールが追加されました。

表 14 追加されたロール

ロール名	説明
<code>pg_maintain</code>	すべてのテーブルに対するメンテナンス文の実行を許可
<code>pg_use_reserved_connections</code>	パラメーター <code>reserved_connections</code> に割り当てられたコネクションを利用できる

`pg_maintain` ロールを付与されたユーザーは他のユーザーが所有するテーブルやマテリアライズドビューに対して `ANALYZE` 文、`VACUUM` 文、`REINDEX` 文、`CLUSTER` 文、`REFRESH MATERIALIZED VIEW` 文を実行できます。従来は `SUPERUSER` 属性を持つユーザーに限られていました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=4441fc704d7048b2f1f039cc74b72bd23e7e36d0>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=60684dd834a222fefedd49b19d1f0a6189c1632e>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6e2775e4d4e47775f0d933e4a93c148024a3bc63>

⇒ `GRANT` 文の説明へ

### 3.1.12 VACUUM

□ `FREEZE`

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=4d417992613949af3553>



[0b4e8e83670c4e67e1b2](#)

☐ ページレベル FREEZE

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1de58df4fec7325d91f5a8345757314be7ac05da>

### 3.1.14. MATERIALIZED VIEW

述語ロック (Predicate Lock) に対応します。REFRESH CONC... に対応。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=43351557d0d2b9c5e20298b5fee2849abef86aff>

### 3.1.13. Meson 対応

ビルド・システムとして Meson (<https://mesonbuild.com/>) が利用できるようになりました。ソースコード内の各ディレクトリに meson.build ファイルが配置されています。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e6927270cd18d535b77cbe79c55c6584351524be>

### 3.1.14. UNICODE

対応する Unicode のバージョンが 14.0.0 から 15.0.0 に変更されました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1091b48cd761abc7f697fa9c4e018c46763b46fb>

### 3.1.15. LLVM

LLVM 15 をサポートします。この変更は旧バージョンにもバックポートされます。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c2ae01f695b1605bc5e3908ff52b24fce6636caa>



### 3.1.16. ロケール

locale コマンドが存在しない場合でも initdb コマンドは正常に終了するようになりました。従来は locale -a コマンドの実行が失敗するとエラーになっていました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=af3abca029b39ae6bdb683a758b11375e9839631>

### 3.1.17. PL/pgSQL

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d747dc85aec536c471fd7c739695e155627b08fd>

### 3.1.6. WAL

FREEZE 処理の WAL 出力量が削減されました。

Deduplicate freeze plans in freeze WAL records.

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=9e5405993c1e242ae6fb7561353e437241244ac1>

### 3.1.6. 確認中

Allow left join removals and unique joins on partitioned table

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3c569049b7b502bb4952483d19ce622ff0af5fd6>

### 3.1.6. 確認中

Improve GIN cost estimation

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=cd9479af2af25d7fa9bfd24dd4dcf976b360f077>

### 3.1.6. 確認中

Replace SQLValueFunction by COERCE\_SQL\_SYNTAX

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=f193883fc9cebe8fa20359>



[b0797832837a788112](#)

### 3.1.6. 確認中

use has\_privs\_of\_role() for pg\_hba.conf

<https://commitfest.postgresql.org/39/3609/>





## 3.2. SQL 文の拡張

ここでは SQL 文に関する新機能を説明しています。

### 3.2.1. ALTER ROLE

ALTER ROLE 文には USER SET 句が指定できるようになりました。これは一般ユーザーにもカスタム・パラメーターに対する ALTER ROLE SET 文の実行を許可します。設定した内容は pg\_db\_role\_setting カタログの setuser 列で確認できます。

#### 例 19 SET USER 句の指定

```
postgres=# ALTER ROLE demo SET prefix1.foo1 TO 'abc' USER SET ;
ALTER ROLE
postgres=# \connect postgres demo
You are now connected to database "postgres" as user "demo".
postgres=> SHOW prefix1.foo1 ;
prefix1.foo1
-----
abc
(1 row)

postgres=> ALTER ROLE demo SET prefix1.foo1 TO 'def' USER SET ;
SET
postgres=> SHOW prefix1.foo1 ;
prefix1.foo1
-----
def
(1 row)
```

ALTER ROLE / ALTER USER / ALTER DATABASE SET USER SET

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=096dd80f3ccc103c8e078fca05e6ccfb2071aa91>

[https://qiita.com/nuko\\_yokohama/items/bd9844c27e0b0cbf99fd](https://qiita.com/nuko_yokohama/items/bd9844c27e0b0cbf99fd)

### 3.2.2. COPY

外部テーブルに対する COPY 文の実行時に、外部サーバー（FOREIGN SERVER）や外部テーブル（FOREIGN TABLE）の batch\_size オプションを利用できるようになりました。従来は INSERT 文実行時にのみ有効でした。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=97da48246d34807196b404626f019c767b7af0df>

### 3.2.3. CREATE ROLE/USER

CREATEROLE 属性を持つロール／ユーザーは、自身が REPLICATION 属性、BYPASSRLS 属性を持つ場合に同じ属性を他のユーザーに設定できるようになります。CRETEDB 属性は旧バージョンで同じ動作になっています。

例 20 CREATEROLE 属性ユーザーによるユーザー作成

```
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=f1358ca52dd7b8cedd29c6f2f8c163914f03ea2e>

### 3.2.4. CREATE STATISTICS

統計情報の名前は必須ではなくなりました。名前を省略した場合の統計情報名は元のテーブル名と列名から自動生成されます。



#### 例 21 CREATE STATISTICS 文の名前省略

```
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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=624aa2a13bd02dd584bb0995c883b5b93b2152df>

### 3.2.5. CREATE TABLE

CREATE TABLE 文に STORAGE 句が指定できるようになりました。従来は ALTER TABLE 文で設定する必要がありました。

#### 例 22 CREATE TABLE 文の STORAGE 句

```
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)				<u>plain</u>	...
c3	text				<u>extended</u>	...

Indexes:

"data1\_pkey" PRIMARY KEY, btree (c1)

Access method: heap



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=784cedda0604ee4ac731fd0b00cd8b27e78c02d3>

CREATE TABLE 文、ALTER TABLE 文の STORAGE 句にはデフォルトのストレージ形式を示す DEFAULT を指定できます。

#### 例 23 ALTER TABLE 文の STORAGE DEFAULT 句

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b9424d014e195386a83b0f1fe9f5a8e5727e46ea>

### 3.2.6. GRANT

GRANT 文には以下の機能が追加されました。

#### □ テーブルに対する権限の追加

テーブルおよびマテリアライズドビューのメンテナンス文（VACUUM、ANALYZE、REINDEX、REFRESH MATERIALIZED VIEW、CLUSTER、LOCK TABLE）の実行をテーブルに対して許可する MAINTAIN 権限が追加されました。psql コマンドでアクセス権限を確認すると'm'と出力されます。⇒ **pg\_class.relaccl**

#### 例 24 メンテナンス文の実行権限

```
postgres=> CREATE TABLE data1(c1 INT, c2 VARCHAR(10)) ;
CREATE TABLE
postgres=> GRANT MAINTAIN ON data1 TO demo ;
GRANT
postgres=> \dp+ data1
```

Access privileges					
Schema	Name	Type	Access privileges	Column privileges	
public	data1	table	postgres=arwdDxtvz/postgres+		...
			demo=vz/postgres		...

(1 row)

全テーブルに対して同様の操作を許可する事前定義ロール `pg_maintain` が追加されています。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b5d6382496f2b8fc31abd92c2654a9a67aca76c6>

#### □ WITH INHERIT 句

権限を継承する `WITH INHERIT TRUE` 句と継承しない `WITH INHERIT FALSE` 句が指定できます。デフォルトは `WITH INHERIT TRUE` です。設定した値は `pg_auth_members` カタログの `inherit_option` 列で確認できます。

#### 例 25 WITH INHERIT

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e3ce2de09d814f8770b2e3b3c152b7671bcd83f>

#### □ WITH SET 句

`SET ROLE` 文の使用を許可する `WITH SET TRUE` 句と、許可しない `WITH SET FALSE` 句を指定できます。デフォルトは `WITH SET TRUE` です。設定した値は `pg_auth_members` カタログの `set_option` 列で確認できます。

#### 例 26 WITH SET

```
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"
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3d14e171e9e2236139e8976f3309a588bcc8683b>



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=ce6b672e4455820a0348214be0da1a024c3f619f>

### 3.2.7. REINDEX

REINDEX SYSTEM 文と REINDEX DATABASE 文ではデータベース名を省略できるようになりました。データベース名を省略するとカレントのデータベースに対してコマンドが実行されます。また REINDEX DATABASE 文ではシステムカタログに対する REINDEX 処理は実行されなくなります。

#### 例 27 REINDEX 文でデータベース名の省略

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=2cbc3c17a5c11d13c0ac92fe7557c56408f8f3d0>

### 3.2.8. VACUUM

VACUUM 文にデータベース統計情報の更新を抑止する SKIP\_DATABASE\_STATS オプションが追加されました。このオプションを TRUE に設定すると pg\_class カタログへのスキャンが抑止され、同時実行性が向上します。データベース統計情報のみを取得する ONLY\_DATABASE\_STATS オプションも利用できるようになりました。

#### 例 28 VACUUM 文の実行

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

ONLY\_DATABASE\_STATS / SKIP\_DATABASE\_STATS

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a46a7011b27188af526047a111969f257aaf4db8>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d977ffd923d207164eef78ed107d5293aee6c660>

### 3.2.9. サブクエリー

FROM 句のサブクエリーにエイリアス名を指定する必要がなくなりました。

#### 例 29 サブクエリーの FROM 句

```
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)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=bcedd8f5fce0b69970cf0cee7bca560833d05869>

### 3.2.10. ウィンドウ関数

同一の OVER 句に対して実行計画を共有できるようになりました。以下の SELECT 文を実行した際に実行計画が最適化されます。

#### 例 30 テスト用 SELECT 文

```
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 ;
```



例 31 PostgreSQL 15 の実行計画

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

例 32 PostgreSQL 16 の実行計画

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=ed1a88ddaccfe883e4cf74d30319accfeae6cfe5>

### 3.2.11. SELECT DISTINCT

DISTINCT 句付きの SELECT 文にもインクリメンタル・ソートが考慮されるようになりました。パラメーター `enable_presorted_aggregate` を off に設定することで PostgreSQL 15 と同じ動作になります。





#### 例 33 PostgreSQL 15 の実行計画

```
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)
```

#### 例 34 PostgreSQL 16 の実行計画

```
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 width=4)
(2 rows)
```

Have the planner consider Incremental Sort for DISTINCT

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3c6fc58209f24b959ee18f5d19ef96403d08f15c>

### 3.2.12. 関数

以下の関数が追加／拡張されました。

#### □ random\_normal

正規分布の乱数を発生させる関数 `random_normal` が追加されました。平均と標準偏差を指定します。エクステンション `tablefunc` には既に `normal_rand` 関数が提供されていますが、修正されてシステム関数として提供されました。

構文

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



#### 例 35 random\_normal 関数の実行

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=38d81760c4d7e22b95252e3545596602c9e38806>

#### ☐ pg\_read\_file / pg\_read\_binary\_file

ファイル名とファイルが存在しない場合の挙動を示す `missing_ok` パラメーターを持つバージョンが追加されました。

#### 構文

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

#### 例 36 pg\_read\_file 関数の実行

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=283129e325b721a5a62227f20d7e3d149b379c73>

#### ☐ system\_user

`SYSTEM_USER` は SQL 仕様の予約語であり、認証方法と認証 ID を返す関数です。`TRUST` 認証の場合には `NULL` を返します。



構文

```
text system_user()
```

例 37 `system_user` 関数の実行

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

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=0823d061b0b7f1e20fbfd48bef3c2e093493dbd4>

□ `pg_split_walfile_name`

WAL ファイル名からシーケンス番号とタイムライン ID を返します。

構文

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

例 38 `pg_split_walfile_name` 関数の実行

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=cca186348929cd75f23ef1b25922386bf38cf99c>



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=13e0d7a603852b8b05c03b45228daabffa0cced2>

□ `pg_stat_get_backend_subxact`

特定のバックエンドのキャッシュ内のサブトランザクションの情報を取得します。

構文

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

例 39 `pg_dissect_walfile_name`

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=10ea0f924a2788f9e701d6213745aaa5ca3efb8a>

□ `pg_input_is_valid`

`pg_input_is_valid` 関数は指定した値がデータ型に合致するかをチェックできます。桁溢れや不適切な日付文字列を事前に確認できます。入力データとデータ型を文字列で指定します。

構文

```
boolean pg_input_is_valid(text, text)
```



例 40 pg\_input\_is\_valid 関数の実行

```
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)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1939d26282b27b4b264c6930830a7991ed83917a>

□ pg\_input\_error\_message

pg\_input\_error\_message 関数は入力データが指定したデータ型として適切でない場合のエラー・メッセージを取得できます。

構文

```
text pg_input_error_message(text, text)
```

例 41 pg\_input\_error\_message 関数の実行

```
postgres=> SELECT pg_input_error_message(' 1234. 48', 'numeric(5, 2)') ;
pg_input_error_message
-----
numeric field overflow
(1 row)

postgres=> SELECT pg_input_error_message(' 2022/12/32 09:30:60', 'timestamp') ;
pg_input_error_message
-----
date/time field value out of range: "2022/12/32 09:30:60"
(1 row)
```



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1939d26282b27b4b264c6930830a7991ed83917a>

☐ `pg_import_system_collations`

Microsoft Windows 環境でも利用できるようになりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=bf03cfd162176d543da79f9398131abc251ddbb9>

### **3.2.X. EXPLAIN**

`explain analyze rows=%0f`

<https://commitfest.postgresql.org/39/3704/>

### 3.3. パラメーターの変更

PostgreSQL 16 では以下のパラメーターが変更されました。

#### 3.3.1. 追加されたパラメーター

以下のパラメーターが追加されました。

表 15 追加されたパラメーター

パラメーター	説明 (context)	デフォルト値
createrole_self_grant	()	"
enable_presorted_aggregate	ソート処理の最適化機能を使用する (user)	on
logical_replication_mode	論理レプリケーションの転送方法を設定する (user)	buffered
max_parallel_apply_workers_per_subscription	サブスクリプション単位のワーカー最大数 (sighup)	2
reserved_connections	一般ユーザー用の予約された接続数 (postmaster)	0
send_abort_for_crash	バックエンドがクラッシュした際に SIGABRT シグナルを送信する (sighup)	off
send_abort_for_kill	子プロセスがスタックした際に SIGABRT シグナルを送信する (sighup)	off
vacuum_freeze_strategy_threshold	—(user)—	4GB

Createrole\_self\_grant=SET,INHERIT

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=6e2775e4d4e47775f0d933e4a93c148024a3bc63>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=e5b8a4c098ad6add39626a14475148872cd687e0>



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3226f47282a05979483475d1e4a11aab8c1bfc39>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=5de94a041ed7a51b571db2030ba87600c7fc6262>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=216a784829c2c5f03ab0c43e009126cbb819e9b2>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=51b5834cd53f0bd068729043b55f7da3ca6bb15f>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1e8b61735cfb1a4feb72cb9ea83db690fedbfef1>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=4d417992613949af35530b4e8e83670c4e67e1b2>

⇒ Revert

☐ send\_abort\_for\_crash / send\_abort\_for\_kill

postmaster は子プロセスのクラッシュ時にまず SIGQUIT シグナルを送信し、その後 SIGKILL シグナルを送信します。send\_abort\_for\_crash と send\_abort\_for\_kill はそれぞれ on に設定することでシグナルを SIGABRT に変更します。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3226f47282a05979483475d1e4a11aab8c1bfc39>

☐ enable\_presorted\_aggregate

⇒ 3.2.X SELECT DISTINCT へ

ORDER BY 句または DISTINCT 句を持つ集計関数をより効率的に実行できるようになりました。enable\_presorted\_aggregate パラメーターのデフォルト値は on で、この機能は有効になっています。



#### 例 42 PostgreSQL 15 までの動作

```
postgres=> SET enable_presorted_aggregate = off ;
SET
postgres=> EXPLAIN SELECT SUM(c1 ORDER BY 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)
```

#### 例 43 PostgreSQL 16 の動作

```
postgres=> SET enable_presorted_aggregate = on ;
SET
postgres=> EXPLAIN SELECT SUM(c1 ORDER BY 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)
```

jit\_warn\_above\_fraction

<https://commitfest.postgresql.org/39/3573/>

### 3.3.2. 変更されたパラメーター

以下のパラメーターは設定範囲や選択肢が変更されました。

表 16 変更されたパラメーター

パラメーター	変更内容
wal_sync_method	Windows 環境の NTFS 上で fdatasync を指定できるようになりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=9430fb407bb64cc842e56fb5844265a9343dafba>

### 3.3.3. デフォルト値が変更されたパラメーター

以下のパラメーターはデフォルト値が変更されました。

表 17 デフォルト値が変更されたパラメーター

パラメーター	PostgreSQL 15	PostgreSQL 16	備考
server_version	15.3	16beta1	
server_version_num	150003	160000	

### 3.3.4. 削除されたパラメーター

以下のパラメーターは削除されました。

表 18 削除されたパラメーター

パラメーター	理由
promote_trigger_file	5 秒毎のプロセス起動が無駄と判断され、トリガー・ファイルがサポートされなくなったため。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=cd4329d9393f84dce34f0bd2dd936adc8ffaa213>

### 3.4. ユーティリティの変更

ユーティリティ・コマンドの主な機能拡張点を説明します。

#### 3.4.1. configure

セグメントサイズをブロック単位で指定する `--with-segsize=blocks` オプションが追加されました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d3b111e3205b6e681e16b4f8e6ed01f67142ce7b>

#### 3.4.2. createuser

`createuser` コマンドには複数のオプションが追加されました。

表 19 追加されたオプション

オプション	短縮形	説明
<code>--admin=ROLE</code>	<code>-a</code>	追加ユーザーが所属する ADMIN オプション付きロール
<code>--member=ROLE</code>	<code>-g</code>	追加ユーザーが所属するロール
<code>--valid-until=TIMESTAMP</code>	<code>-v</code>	パスワード有効期限
<code>--bypassrls</code>	<code>-</code>	Bypass RLS 属性を付与
<code>--no-bypassrls</code>	<code>-</code>	Bypass RLS 属性を付与しない

例 44 `createuser` コマンドの実行

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

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=08951a7c93cf0dd791ee6ac8a8cf5e4b152528e5>

### 3.4.3. pgindent

pg\_indent コマンドには以下のオプションが追加・拡張されました。

表 20 追加されたオプション

オプション	変更	説明
--show-diff	追加	変更点を表示する
--silent-diff	追加	変更点があった場合には戻り値 2 で終了する
--excludes	変更	複数回指定可能に

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b90f0b57474eac3980be8146f2e45f73a05c994f>

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a1c4cd6f2c8857fbb78d043a0b3c7d397e48ef2e>

### 3.4.4. pg\_dump

pg\_dump コマンドには以下の拡張が実装されました。

#### □ --compress オプション

--compress オプションには圧縮メソッドと圧縮レベルをコロン (:) で区切って指定できるようになりました。現状では圧縮メソッドに指定できるのは none または gzip です。

#### 例 45 pg\_dump コマンドの実行

```
$ pg_dump -d postgres --compress=gzip:9 --file=postgres.gz
$ file postgres.gz
postgres.gz: gzip compressed data, max compression, from Unix, original size
11963
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=5e73a6048849bd7bda4947e39570b9011734114d>

#### □ --large-objects オプション

従来は--blob と--no-blob オプションは--large-objects と--no-large-objects に変更されました。従来のオプションも利用できますが、deprecated 扱いとなります。



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=35ce24c333cf6dee3c92bc5f67553c7720bd9988>

<https://commitfest.postgresql.org/38/2573/>

### 3.4.5. pg\_receivewal / pg\_recvlogical

プログラムの終了条件として従来の SIGINT シグナル以外に SIGTERM シグナルが追加されました。これは systemd の終了シグナル (KillSignal) のデフォルト値に対応する変更です。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=8b60db774356117fab2eb53fb37160fa3e173cdb>

### 3.4.6. psql

psql コマンドには以下の拡張が実装されました。

#### □ ¥d+オプション

パーティション・テーブルに対して ¥d+ コマンドを実行すると、パーティションが外部テーブルである場合に「FOREIGN」が出力されます。

#### 例 46 外部テーブルのパーティション情報

```
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 ...
-----+-----+-----+-----+-----+-----+
mpression | Stats target | Description
-----+-----+-----+-----+-----+
 c1      | integer       |           |          |         | plain    ...
 c2      | character varying(10) |           |          |         | extended...
Partition key: RANGE (c1)
Partitions: remote1 FOR VALUES FROM (0) TO (100), FOREIGN
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=bd95816f74ad4cad3d2a3c160be426358d6cea51>

#### □ ¥bind オプション

SQL 文内のバインド変数に値をセットする¥bind オプションが追加されました。

#### 例 47 バインド変数の設定

```
postgres=> SELECT $1, $2 ¥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)
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=5b66de3433e2110b38a2b32aaaa0b9cdac8aacdb>

#### □ ¥pset オプション

¥pset コマンドに指定できるオプション xheader\_width が追加されました。このオプションは拡張テーブル形式で出力する場合ヘッダの幅を制限できます。指定できる値は以下の通りです。

**表 21 指定できる値**

設定値	説明
full	最も幅の広い行の長さ (デフォルト)
column	最初の列の幅に切り捨て
page	ターミナル・ページの幅に切り捨て



数字	指定された数値に切り捨て
----	--------------

#### 例 48 xheader\_width オプションの指定

```
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a45388d6e0984abb02074f0300cd9c5cbda13848>

#### □ \dpS オプション

\dp コマンドと\z コマンドに'S'が指定できるようになりました。これに伴い旧バージョンから\dp または\z コマンドの動作が変更されました。一時オブジェクトが一覧に含まれ、information\_schema スキーマのオブジェクトが対象外になりました。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d913928c9c5e905d0062d1e7237b7fb5fbde61ed>



### 3.4.7. vacuumdb

vacuumdb コマンドには以下のオプションが追加されました。

□ --schema オプション

VACUUM を実行するテーブルが所属するスキーマ名を指定します。短縮形は **-n** です。このオプションは複数回指定できます。--exclude-schema オプションと同時に使用できません。

□ --exclude-schema オプション

VACUUM を実行しないテーブルが所属するスキーマ名を指定します。短縮形は **-N** です。--schema オプションとは同時に使用できません。

#### 例 49 スキーマ指定の VACUUM

```
$ 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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=7781f4e3e711ace6bcf9b6253a104b180cb78fcf>

データベース属性で実行コマンド変化？

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=a46a7011b27188af526047a111969f257aaf4db8>



### 3.5. Contrib モジュール

Contrib モジュールに関する新機能を説明しています。

#### 3.5.1. auto\_explain

□ パラメーター `log_parameter_max_length`

パラメーター `log_parameter_max_length` が追加されました。このパラメーターはログ出力の最大値を制限します。デフォルト値は -1 で制限を設けません。0 に指定するとロギングを無効にします。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=d4bfe41281705c1bcb7093b3d07ce5ff1114341b>

□ Query ID の出力

パラメーター `auto_explain.log_verbose` を on に指定し、かつ `compute_query_id` を on に指定した場合 Query ID が出力されます。

例 50 Query ID の出力

```
$ tail -5 postgresql-2023-01-26_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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=9d2d9728b8d546434aade4f9667a59666588edd6>

#### 3.5.2. ltree

ラベルにハイフン (-) を利用できるようになりました。以前はアルファベット、数字、アンダーライン (\_) のみ許可されていました。ラベルの最大サイズが 255 文字から 1,000 文字に拡大されました。



<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=b1665bf01e5f4200d37addfc2ddc406ff7df14a5>

### 3.5.3. pageinspect

B-Tree インデックス・ページの要約を出力する関数 `bt_multi_page_stats` が追加されました。`bt_page_stats` 関数の複数ページ版です。先頭ブロック番号とページ数を指定します。

例 51 `bt_multi_page_stats` 関数の実行

```
postgres=# SELECT * FROM bt_multi_page_stats('idx1_data1', 2, 1) ;
-[ RECORD 1 ]-+-----
blkno          | 2
type           | I
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=1fd3dd2048991a164c287dc29fdb78b1f7e6e14e>

### 3.5.4. pg\_buffercache

バッファキャッシュのサマリーを SQL 文で確認できる `pg_buffercache_summary` 関数が追加されました。

例 52 pg\_buffercache\_summary 関数の実行

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

~~pg\_buffercache ビューの relfilenode 列のデータ型が oid から bigint に変更されました。~~

~~<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=05d4cbf9b6ba708858984b01ea0fe56d59d4ee7e>~~

表 22 pg\_buffercache\_summary 実行結果

列名	説明
buffers_used	使用されているバッファ総数
buffers_unused	使用されていないバッファ数
buffers_dirty	ダーティバッファ数
buffers_pinned	ピン留めされているバッファ数
usagecount_avg	使用済バッファの平均使用数？

### 3.5.5. pg\_stat\_statements

SQL 文の一般化方法が更新されました。SET 文、CHECKPOINT 文、CREATE 文等が大文字／小文字の区別なく同一のクエリーID として収集されます。



例 53 pg\_stat\_statements ビューの検索

```
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)
```

Generate code for query jumbling through gen\_node\_support.pl

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=3db72ebcbe20debc6552500ee9ccb4b2007f12f8>

<https://commitfest.postgresql.org/39/3048/>

### 3.5.6. pg\_upgrade

オプション `--copy` が追加されました。これはデフォルトの動作です。

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=746915c6866953506379e996ce5198bc9e9e5b94>

### 3.5.7. pg\_waldump

全ページのイメージを保存するディレクトリを指定する `--save-fullpage` オプションが追加されました。指定したディレクトリは空である必要があります。フルページのイメージは以下のファイル名で作成されます。

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

表 23 ファイル名の説明

表記	説明	例
<lsn>	LSN の 16 進数フォーマット	00000000-05AD6E48
<tablespace>	テーブルスペースの OID	1663
<database>	データベースの OID	5
<relation>	リレーション filenode	16388
<block>	ブロック番号	20
<forkname>	フォーク名	main

例 54 --save-fullpage オプションの指定

```
$ pg_waldump --save-fullpage=$HOME/fullpage data/pg_wal/
rmgr: Heap      len (rec/tot):    65/    65, tx:      740, lsn: ...
rmgr: Heap      len (rec/tot):    65/    65, tx:      740, lsn: ...
...
$ ls -l $HOME/fullpage
00000000-05A26608.1663.5.16388.0_vm
00000000-05A7D2E8.1663.5.2619.18_main
00000000-05A7E920.1663.5.2696.1_main
00000000-05A80B20.1663.5.16388.0_main
00000000-05A853B0.1663.5.16388.1_main
...
```

### 3.5.8. pg\_walinspect

pg\_get\_wal\_fpi\_info 関数が追加されました。この関数は指定された LSN 間のフルページイメージを抽出します。

構文

```
record pg_get_wal_fpi_info(start_lsn pg_lsn, end_lsn pg_lsn)
```

例 55 pg\_get\_wal\_fpi\_info 関数の実行

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

この関数を実行すると以下の情報が返ります。

表 24 関数の戻り値

列名	データ型	説明
lsn	pg_lsn	LSN
reltablespace	oid	テーブル空間の OID
reldatabase	oid	データベースの OID
relfilenode	oid	ファイルの OID
relblocknumber	bigint	ブロック番号
forkname	text	フォーク名 (main, fsm, vm, init)
fpi	bytea	ブロックイメージ

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=c31cf1c03d01ce86f20bef8c980fe56a257b3b4b>

### 3.5.9. postgres\_fdw

postgres\_fdw には以下の拡張が実装されました。

#### □ COPY FROM 文

COPY FROM 文で外部テーブルの batch\_size オプションが利用されるようになりました。下記は実テーブルを保有するインスタンスの、log\_statement パラメーターを利用したログです。複数レコードが一括で挿入されていることがわかります。

例 56 外部テーブル側の実行ログ

```
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
```

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=97da48246d34807196b404626f019c767b7af0df>

□ ANALYZE 文

FOREIGN SERVER または FOREIGN TABLE のオプションに `analyze_sampling` が追加されました。従来のバージョンでは統計情報の取得のために外部テーブルのレコード全体を取得していました。外部テーブルの情報取得時に `TABLESAMPLE` 句を使用することでリモートデータベースの負荷を削減することができます。デフォルト値は `auto` で、PostgreSQL 9.5 未満では `random`、PostgreSQL 9.5 以上では `bernoulli` とみなされます。設定できる値は以下の通りです。

表 25 `analyze_sampling` オプションの設定値

設定値	説明
off	サンプリングを行いません。
auto	バージョンによってサンプリング方法を決定します (デフォルト値)
random	random 関数を使ってサンプリングを行います。
system	TABLESAMPLE SYSTEM 句を使ってサンプリングを行います。
bernoulli	TABLESAMPLE BERNOULLI 句を使ってサンプリングを行います。



#### 例 57 ANALYZE 文の実行

```
postgres=> CREATE FOREIGN TABLE foreign1(c1 INT, c2 VARCHAR(10)) SERVER svr5433
           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
```

外部テーブルの元テーブルで ANALYZE 文が実行されていない場合にはサンプリングは行われません。

サンプリング割合は？ (postgresAcquireSampleRowsFunc)

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=8ad51b5f446b5c19ba2c0033a0f7b3180b3b6d95>

☐ Allow batching of inserts during cross-partition updates.

<https://git.postgresql.org/gitweb/?p=postgresql.git;a=commit;h=594f8d3776af4699c5c32443b9d6a486f44beabf>

### 3.5.X. vacuumdb

<https://commitfest.postgresql.org/38/3612/>

### 3.5.X. amcheck

<https://commitfest.postgresql.org/39/3464/>





## 参考にした URL

本資料の作成には、以下の URL を参考にしました。

- **Release Notes**  
<https://www.postgresql.org/docs/15/release-15.html>
- Commitfests  
<https://commitfest.postgresql.org/>
- **PostgreSQL 16 Manual**  
<https://www.postgresql.org/docs/15/index.html>
- Git  
<git://git.postgresql.org/git/postgresql.git>
- GitHub  
<https://github.com/postgres/postgres>
- **PostgreSQL 16 Beta 1 のアナウンス**  
<https://www.postgresql.org/about/news/postgresql-15-beta-1-released-2453/>
- Postgres Professional  
<https://habr.com/ru/company/postgrespro/blog/541252/>
- PostgreSQL 16 Open Items  
[https://wiki.postgresql.org/wiki/PostgreSQL\\_16\\_Open\\_Items](https://wiki.postgresql.org/wiki/PostgreSQL_16_Open_Items)
- Qiita (ぬこ@横浜さん)  
[http://qiita.com/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](https://wiki.postgresql.org/wiki/Development_information)
- pgPedia  
<https://pgpedia.info/postgresql-versions/postgresql-16.html>
- **SQL Notes**  
<https://sql-info.de/postgresql/postgresql-15/articles-about-new-features-in-postgresql-15.html>
- Slack - postgresql-jp  
<https://postgresql-jp.slack.com/>

## 變更履歷

## 變更履歷

[illegible]

以上

