Hewlett Packard Enterprise

2019 年 5 月 24 日

PostgreSQL 12 新機能検証結果（Beta 1）

日本ヒューレット・パッカード株式会社
篠田典良

Hewlett Packard
Enterprise

目次

Hewlett Packard
Enterprise

1. 本文書について

1.1 本文書の概要

本文書は現在ベータ版が公開されているオープンソース RDBMS である PostgreSQL 12 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 11 (11.3) と PostgreSQL 12 (12.0) 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 11.3 (比較対象) PostgreSQL 12 (12.0) Beta 1 (2019/5/20 20:41:05)
オペレーティング・システム	Red Hat Enterprise Linux 7 Update 5 (x86-64)
Configure オプション	--with-llvm --with-openssl --with-perl

Hewlett Packard
Enterprise

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・パッカード株式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が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）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
下線部	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

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

Hewlett Packard
Enterprise

2. PostgreSQL 12 における変更点概要

PostgreSQL 12 には 150 以上の新機能が追加されました。代表的な新機能と利点について説明します。

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

パラレル・クエリーが適用される範囲が拡張されました。トランザクション分離レベルが SERIALIZABLE の場合でもパラレル・クエリーが実行される可能性があります。

□ パーティション・テーブルの拡張

パーティション・キーの値として固定値ではなく、計算値を指定できるようになりました。外部キーの参照先としてパーティション・テーブルを利用することができるようになりました。

2.2. 信頼性向上に関する新機能

PostgreSQL 12 では信頼性を向上させるために整合性のチェック・ツールが充実しました。

□ pg_checksums コマンド

PostgreSQL 11 で追加された pg_verify_checksums コマンドは pg_checksums コマンドに変更されました。整合性のチェックだけでなく、チェックサム機能の有効化／無効化をコマンドで変更することができるようになりました。

2.3. 運用性を向上させる新機能

運用性を向上できる以下の機能が追加されました。

□ recovery.conf ファイルの廃止

ストリーミング・レプリケーションのスタンバイ・インスタンスや、リカバリー時に使用する recovery.conf ファイルは、postgresql.conf ファイルに統合されました。

□ REINDEX 文の拡張

REINDEX 文に CONCURRENTLY 句が追加され、インデックスの再構成時のロック範囲が非常に小さくなりました。

□ モニタリング機能の強化

CLUSTER 文、VACUUM FULL 文や CREATE INDEX 文の実行状況をリアルタイムに確認できるカタログが追加されました。

□ 待機イベントの増加

待機イベントがいくつか拡張されました。pg_stat_activity カタログで確認できます。

□ pg_promote 関数

スタンバイ・インスタンスからプライマリー・インスタンスへの昇格を実行する関数 pg_promote が提供されました。

2.4. 将来の新機能に対する準備

PostgreSQL 12 では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。

□ 複数ストレージ・エンジン

複数のストレージ・エンジンを同時に利用できる PLUGGABLE STORAGE ENGINE の基本仕様が決定されました。今後 zheap 等の対応が行われることが期待されます。

□ 64 ビット・トランザクション ID

64 ビット長のトランザクション ID を取得する API が利用できるようになりました。

2.5. 非互換

PostgreSQL 12 は PostgreSQL 11 から以下の仕様が変更されました。

2.5.1. recovery.conf ファイルの廃止

データベースのリカバリーや、ストリーミング・レプリケーション環境のスタンバイ・インスタンスで作成されていた recovery.conf ファイルは廃止されました。recovery.conf ファイル内の各種パラメーターは postgresql.conf ファイルに統合されました。詳しくは「3.1.2. recovery.conf ファイルの廃止」で説明しています。

2.5.2. pg_checksums コマンド

PostgreSQL 11 で追加された pg_verify_checksums コマンドは、pg_checksums コマンドに名前が変更され、機能が追加されました。

2.5.3. to_timestamp / to_date 関数の仕様変更

テンプレート内の不要なスペースは無視されるようになりました。

例 1 PostgreSQL 11 の仕様

```
postgres=> SELECT TO_DATE('2019/03/23', ' YYYY/MM/DD') ;
  to_date
------------
 0019-03-23
(1 row)
```

例 2 PostgreSQL 12 の仕様

```
postgres=> SELECT TO_DATE('2019/03/23', ' YYYY/MM/DD') ;
  to_date
------------
 2019-03-23
(1 row)
```

2.5.4. WITH OIDS 句の削除

CREATE TABLE 文の WITH OIDS 句は禁止され、OID 付きのテーブルは作成できなくなりました。これに伴い、システムカタログの oid 列は隠し列ではなくなりました。WITHOUT OIDS 句は引き続き使用できます。この仕様に伴い、default_with_oids パラメーターは on に変更できなくなりました。この変更に伴い、pg_dump コマンドから--oids オプション（-o オプション）が削除されました。

例 3 PostgreSQL 12 の CREATE TABLE 文

```
postgres=> CREATE TABLE oid1(c1 INT) WITH OIDS ;
psql: ERROR:  syntax error at or near "OIDS"
LINE 1: CREATE TABLE oid1(c1 INT) WITH OIDS ;
```

例 4 PostgreSQL 12 の pg_tablespace カタログ仕様

```
postgres=> ¥d pg_tablespace
            Table "pg_catalog.pg_tablespace"
   Column   |   Type    | Collation | Nullable | Default
------------+-----------+-----------+----------+---------
 oid        | oid       |           | not null |
 spcname    | name      |           | not null |
 spcowner   | oid       |           | not null |
 spcacl     | aclitem[] |           |          |
 spcoptions | text[]    |           |          |
Indexes:
    "pg_tablespace_oid_index" UNIQUE, btree (oid), tablespace "pg_global"
    "pg_tablespace_spcname_index" UNIQUE, btree (spcname), tablespace
"pg_global"
Tablespace: "pg_global"
```

2.5.5. timetravel Contrib モジュールの削除

Contrib モジュール timetravel が削除されました。

2.5.6. データ型の削除

データ型 abstime、reltime、tinterval は削除されました。これに伴い、システムカタログ pg_shadow の valuntil 列のデータ型が timestamp with time zone に変更されました。

2.5.7. パーティション・テーブルに対する COPY FREEZE

パーティション・テーブルに対する COPY FREEZE 文は実行が禁止されました。この仕様は PostgreSQL 11.2 以降にもバックポートされました。

2.5.8. 外部キー制約名の変更

自動生成される外部キー名には、外部キーに含まれる全列を含む名前が生成されるようになりました。

例 5 PostgreSQL 12 の FOREIGN KEY 制約名

```
postgres=> CREATE TABLE ftable2(c1 INT, c2 INT, c3 VARCHAR(10), FOREIGN KEY (c1, c2) REFERENCES ftable1(c1, c2)) ;
CREATE TABLE
postgres=> ¥d ftable2
                    Table "public.ftable2"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | integer               |           |          |
 c2     | integer               |           |          |
 c3     | character varying(10) |           |          |
Foreign-key constraints:
    "ftable2_c1_c2_fkey" FOREIGN KEY (c1, c2) REFERENCES ftable1(c1, c2)
```

2.5.9. 関数

current_schema 関数と current_schemas 関数はパラレル・クエリーに対して安全ではないと設定されました（Parallel Unsafe）。

Hewlett Packard
Enterprise

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が変更されました。

表 4 追加されたシステムカタログ

カタログ名	説明
pg_stat_progress_cluster	CLUSTER 文の実行状況をトレースします。
pg_stat_progress_create_index	CREATE INDEX 文の実行状況をトレースします。

表 5 追加され情報スキーマ（information_schema）内のテーブル

カタログ名	説明
column_column_usage	特定の列に依存する列の情報（生成列を持つテーブルの情報が格納）

表 6 列が追加されたシステムカタログ

カタログ名	追加列名	データ型	説明
※1	oid	oid	通常列に表示属性を変更
pg_attribute	attgenerated	char	生成列の場合の値は's'
pg_collation	collisdeterministic	boolean	照合順序は決定的か
pg_statistic	stacol[1-5]	oid	Collation 情報
pg_statistic_ext	stxmcv	pg_mcv_list	MCV 統計情報
pg_stat_database	checksum_failures	bigint	チェックサム・エラーが検知されたブロック数
	checksum_last_failure	timestamp with time zone	チェックサム・エラーが最後に検知された日時
pg_stat_replication	reply_time	timestamp with time zone	スタンバイからのメッセージ時刻
pg_stat_ssl	client_serial	numeric	クライアント証明書のシリアル番号

カタログ名	追加列名	データ型	説明
	issuer_dn	text	クライアント証明書の発行者 DN

※1 多数のカタログが該当

表 7 列が削除されたシステムカタログ

カタログ名	削除列名	説明
pg_attrdef	adsrc	見てわかるデフォルト値の表現
pg_class	relhasoids	WITH OIDS 指定のテーブルかどうか
pg_constraint	consrc	見てわかる検査制約の表現

表 8 列が変更されたシステムカタログ

カタログ名	列名	説明
pg_shadow	valuntil	データ型が timestamp with time zone に変更
pg_stat_ssl	client_dn	列名が clientdn から変更

変更されたシステムカタログから、主なカタログの詳細を以下に記載します。

□ pg_stat_progress_cluster カタログ

pg_stat_progress_cluster カタログは、CLUSTER 文または VACUUM FULL 文の実行状況をトレースすることができます。

表 9 pg_stat_progress_cluster カタログ

列名	データ型	説明
pid	integer	バックエンドのプロセス ID
datid	oid	バックエンドが接続しているデータベース OID
datname	name	データベース名
relid	oid	クラスター化されているテーブルの OID
command	text	実行文
phase	text	実行フェーズ
cluster_index_relid	oid	インデックス・スキャン実行時の OID
heap_tuples_scanned	bigint	スキャンされたタプル数
heap_tuples_written	bigint	書込みを行ったタプル数
heap_blks_total	bigint	テーブル内のブロック数
heap_blks_scanned	bigint	スキャンされたブロック数

列名	データ型	説明
index_rebuild_count	bigint	インデックスのリビルド回数

□ pg_stat_progress_create_index カタログ

pg_stat_progress_create_index カタログは、CREATE INDEX 文の実行状況をトレースすることができます。REINDEX 文の実行時にも情報が反映されます。

表 10 pg_stat_progress_create_index カタログ

列名	データ型	説明
pid	integer	バックエンドのプロセス ID
datid	oid	バックエンドが接続しているデータベース OID
datname	name	データベース名
relid	oid	インデックスが作成されているテーブルの OID
index_relid	oid	インデックスの OID
phase	text	インデックス作成フェーズ
lockers_total	bigint	待機するロッカーの総数
lockers_done	bigint	既に待っているロッカー数
current_locker_pid	bigint	待機しているロッカーのプロセス ID
blocks_total	bigint	現在のフェーズで処理されるブロック総数
blocks_done	bigint	現在のフェーズで処理されたブロック数
tuples_total	bigint	現在のフェーズで処理されるタプル総数
tuples_done	bigint	現在のフェーズで処理されたタプル数
partitions_total	bigint	インデックスが作成されるパーティション数
partitions_done	bigint	インデックスが作成されたパーティション数

例 6 pg_stat_progress_create_index カタログの検索

```
postgres=> SELECT * FROM pg_stat_progress_create_index ;
-[ RECORD 1 ]------+----------------------------------------
pid                | 13233
datid              | 16385
datname            | postgres
relid              | 16386
index_relid        | 0
phase              | building index: loading tuples in tree
lockers_total      | 0
lockers_done       | 0
current_locker_pid | 0
blocks_total       | 0
blocks_done        | 0
tuples_total       | 10000000
tuples_done        | 6207353
partitions_total   | 0
partitions_done    | 0
```

□ pg_stat_replication カタログ

pg_stat_replication カタログには reply_time 列が追加されました。スタンバイ・インスタンスから受信した最後の返信メッセージの送信時刻が格納されます。この列は SUPERUSER 属性または pg_monitor ロールの保持者のみ表示できます。

例 7 pg_stat_replication カタログの検索

```
postgres=# SELECT * FROM pg_stat_replication ;
-[ RECORD 1 ]----+------------------------------
pid              | 12497
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2019-05-24 20:13:15.551032+09
backend_xmin     |
state            | streaming
sent_lsn         | 0/3000060
write_lsn        | 0/3000060
flush_lsn        | 0/3000060
replay_lsn       | 0/3000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 0
sync_state       | async
reply_time       | 2019-05-24 20:15:45.68363+09
```

□ pg_indexes カタログ

pg_indexes カタログにはパーティション・インデックスも含まれるようになりました。具体的には pg_class カタログの relkind 列が'I'のインデックスも含まれるようになりました。

□ pg_stat_database カタログ

ブロック・チェックサムのエラーが検知された場合に pg_stat_database カタログの checksum_failures 列と checksum_last_failure 列が更新されるようになりました。checksum_failures 列の値はチェックサム・エラーが検知される度に更新されるため、実際に破損したブロック数を示しているわけではありません。

例 8 pg_stat_database カタログの検索

```
postgres=# SELECT * FROM data1 ;
WARNING:  page verification failed, calculated checksum 27311 but expected 12320
ERROR:  invalid page in block 0 of relation base/13567/16384
postgres=#
postgres=# SELECT datname, checksum_failures , checksum_last_failure
        FROM pg_stat_database WHERE datname='postgres' ;
-[ RECORD 1 ]-----+------------------------------
datname           | postgres
checksum_failures | 1
checksum_last_failure | 2019-05-24 20:06:23.747741+09
```

3.1.2. recovery.conf ファイルの廃止

ストリーミング・レプリケーション環境のスタンバイ・インスタンス構築時や、バックアップからのリカバリーに使用する recovery.conf ファイルは廃止されました。recovery.conf ファイル内の各種設定は postgresql.conf ファイルに統合されました。一部のパラメーターが変更されています。

表 11 変更されたパラメーター名

recovery.conf	postgresql.conf	備考
standby_mode	-	廃止
trigger_file	promote_trigger_file	

リカバリーを行う場合には、データベース・クラスターに recovery.signal ファイルを作成してインスタンス起動を行います。リカバリー処理が完了するとこのファイルは削除されます。

ストリーミング・レプリケーション環境のスレーブ・インスタンスでは、データベース・クラスターに standby.signal ファイルを作成します。このファイルはプライマリーへの昇格が行われると削除されます。

データベース・クラスタ内に recovery.conf ファイルを配置した状態では、インスタンス起動が失敗します。

例 9 recovery.conf ファイルを配置してインスタンス起動

```
$ ls data/recovery.conf
data/recovery.conf
$ pg_ctl -D data start
waiting for server to start....2019-05-24 18:59:43.296 JST [41021] LOG:  starting PostgreSQL 12beta1 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-28), 64-bit
2019-05-24 18:59:43.297 JST [41021] LOG:  listening on IPv6 address "::1", port 5432
2019-05-24 18:59:43.297 JST [41021] LOG:  listening on IPv4 address "127.0.0.1", port 5433
2019-05-24 18:59:43.468 JST [41021] LOG:  listening on Unix socket "/tmp/.s.PGSQL.5432"
2019-05-24 18:59:43.709 JST [41022] LOG:  database system was interrupted; last known up at 2019-05-23 18:55:36 JST
2019-05-24 18:59:43.920 JST [41022] FATAL:  using recovery command file "recovery.conf" is not supported
2019-05-24 18:59:43.921 JST [41021] LOG:  startup process (PID 41022) exited with exit code 1
2019-05-24 18:59:43.921 JST [41021] LOG:  aborting startup due to startup process failure
2019-05-24 18:59:43.921 JST [41021] LOG:  database system is shut down
 stopped waiting
pg_ctl: could not start server
Examine the log output.
$
```

3.1.3. インスタンス起動時のログ

インスタンス起動時に、ログにバージョン番号が出力されるようになりました。パラメーターlogging_collector を on に指定している場合はログ・ファイルに出力されます。

例 10 起動時のログ

```
$ pg_ctl -D data start
waiting for server to start....2019-05-24 18:09:33.291 JST [89769] LOG:  starting PostgreSQL 12beta1 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-28), 64-bit
2019-05-24 18:09:33.291 JST [89769] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2019-05-24 18:09:33.291 JST [89769] LOG:  listening on IPv6 address "::", port 5432
<<以下省略>>
```

3.1.4. 最大接続数

レプリケーション用の接続数は、max_connections パラメーターには依存せず、max_wal_senders パラメーターを利用して取得されるようになりました。この修正に伴い、pg_controldata コマンドの出力に max_wal_senders の情報が追加されました。

例 11 pg_controldata コマンドの出力

```
$ pg_controldata -D data | grep max_wal_senders
max_wal_senders setting:              10
```

3.1.5. パラレル・クエリーの拡張

セッションのトランザクション分離レベルが SERIALIZABLE の場合でもパラレル・クエリーが動作するようになりました。下記の例は SERIALIZABLE 分離レベルのトランザクション内で実行した SQL 文の実行計画を、auto_explain モジュールを使って出力しています。

例 12 SELECT 文の実行

```
postgres=# LOAD 'auto_explain' ;
LOAD
postgres=# SET auto_explain.log_min_duration = 0 ;
SET
postgres=# BEGIN ISOLATION LEVEL SERIALIZABLE ;
BEGIN
postgres=# SELECT COUNT(*) FROM data1 ;
  count
---------
 1000000
(1 row)
postgres=# COMMIT ;
COMMIT
```

例 13 PostgreSQL 11 の実行計画

```
2019-05-24 21:52:48.785 JST [1789] LOG:  duration: 45.472 ms  plan:
        Query Text: SELECT COUNT(*) FROM data1 ;
        Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
          ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
```

例 14 PostgreSQL 12 の実行計画

```
2019-05-24 21:53:22.509 JST [79911] LOG:  duration: 58.141 ms  plan:
        Query Text: SELECT COUNT(*) FROM data1 ;
        Finalize Aggregate  (cost=11614.55..11614.56 rows=1 width=8)
          ->  Gather  (cost=11614.33..11614.54 rows=2 width=8)
                Workers Planned: 2
                ->    Partial  Aggregate    (cost=10614.33..10614.34  rows=1
width=8)
                      ->   Parallel  Seq  Scan  on  data1   (cost=0.00..9572.67
rows=4
```

3.1.6. jsonb 型と GIN インデックス

jsonb 型に作成された GIN インデックスに"jsonb @@ jsonpath"オペレーターと"jsonb @? jsonpath"オペレーターが追加されました。GIN インデックスのオペレータークラスは jsonb_ops と json_path_ops のどちらでも使うことができます。以下はマニュアルに記載された SELECT 文の実行計画です。

例 15 jsonb 型と GIN インデックス

```
postgres=> CREATE INDEX idxgin ON api USING GIN (jdoc) ;
CREATE INDEX
postgres=> EXPLAIN (COSTS OFF) SELECT jdoc->'guid', jdoc->'name' FROM api WHERE
jdoc @@ '$.tags[*] == "qui"' ;
                            QUERY PLAN
------------------------------------------------------------------
 Bitmap Heap Scan on api
   Recheck Cond: (jdoc @@ '($."tags"[*] == "qui")'::jsonpath)
   ->  Bitmap Index Scan on idxgin
         Index Cond: (jdoc @@ '($."tags"[*] == "qui")'::jsonpath)
(4 rows)

postgres=> EXPLAIN (COSTS OFF) SELECT jdoc->'guid', jdoc->'name' FROM api WHERE
jdoc @@ '$.tags[*] ? (@ == "qui")' ;
                             QUERY PLAN
--------------------------------------------------------------------
 Bitmap Heap Scan on api
   Recheck Cond: (jdoc @@ '$."tags"[*]?(@ == "qui")'::jsonpath)
   ->  Bitmap Index Scan on idxgin
         Index Cond: (jdoc @@ '$."tags"[*]?(@ == "qui")'::jsonpath)
(4 rows)
```

3.1.7. 待機イベント

pg_stat_activity カタログの wait_event 列に出力される待機イベントに以下の変更が加えられました。

表 12 変更された待機イベント

待機イベント	説明	変更
BackendRandomLock	乱数発生待ち	削除
GSSOpenServer	GSSAPI 接続待ち	追加
CheckpointDone	チェックポイント完了待ち	追加
CheckpointStart	チェックポイント開始待ち	追加
Promote	スタンバイ・インスタンスの昇格待ち	追加
WALSync	WAL ファイルの同期待ち	追加

3.1.8. ECPG

ECPG には以下の機能が追加されました。

□ DECLARE STATEMENT

SQL 文を DECLARE 文により変数宣言できるようになりました。

例 16 DECLARE STATEMENT

```
{
    EXEC SQL BEGIN DECLARE SECTION ;
        char *selectString = "SELECT to_char(current_date, 'YYYY/MM/DD') cd" ;
        char today[20] ;
        short today_ind = 0 ;
    EXEC SQL END DECLARE SECTION ;

    memset(today, 0, sizeof(today)) ;

    EXEC SQL DECLARE stmt_1 STATEMENT ;
    EXEC SQL PREPARE stmt_1 FROM :selectString ;
    EXEC SQL DECLARE cur_1 CURSOR FOR stmt_1 ;
    EXEC SQL OPEN cur_1 ;
    EXEC SQL FETCH cur_1 INTO :today :today_ind ;
    EXEC SQL CLOSE cur_1 ;
}
```

□ bytea 型対応

DECLARE SECTION に bytea 型の変数を定義し、データの入出力に利用できるようになりました。

例 17 DECLARE bytea

```
{
    EXEC SQL BEGIN DECLARE SECTION ;
        bytea data[1024] ;
        short data_ind = 0 ;
    EXEC SQL END DECLARE SECTION ;

    memset(data.arr, 0, sizeof(data.arr)) ;
    data.len = 0 ;

    EXEC SQL SELECT col1 INTO :data FROM data1 ;

    for (i = 0; i < data.len; i++)
    {
        printf("data[%d] = %c¥n", i, data.arr[i]) ;
    }
}
```

3.1.9. PLUGGABLE STORAGE ENGINE

複数のストレージ・エンジンを利用するための基本的な仕様が決定しました。テーブルに対するアクセスメソッドは CREATE ACCESS METHOD 文に TYPE TABLE 句を指定します。

構文

```
CREATE ACCESS METHOD am_name TYPE TABLE HANDLER handler_name
```

デフォルトのストレージ・エンジンは default_table_access_method パラメーターで指定します。このパラメーターのデフォルト値は heap です。この修正に伴い、pg_am カタログにタプルが追加されています。

例 18 pg_am カタログの検索

```
postgres=# SELECT amname, amhandler, amtype FROM pg_am ;
 amname |      amhandler        | amtype
--------+-----------------------+--------
 heap   | heap_tableam_handler | t
 btree  | bthandler             | i
 hash   | hashhandler           | i
 gist   | gisthandler           | i
 gin    | ginhandler            | i
 spgist | spghandler            | i
 brin   | brinhandler           | i
(7 rows)
```

□ テーブル作成時の指定

テーブルに対するストレージ・エンジンの指定は CREATE TABLE 文に USING 句を指定します。CREATE TABLE AS SELECT 文や、CREATE MATERIALIZED VIEW 文でも指定することができます。

例 19 CREATE TABLE 文

```
postgres=> CREATE TABLE data1(c1 NUMERIC, c2 VARCHAR(10)) USING heap ;
CREATE TABLE
postgres=> CREATE TABLE data2 USING heap AS SELECT * FROM data1 ;
SELECT 100000
postgres=> CREATE MATERIALIZED VIEW mview1 USING heap AS SELECT COUNT(*) cnt
FROM data1 ;
SELECT 1
```

□ psql コマンドによるテーブル定義表示

「¥d+ テーブル名」コマンドを実行すると、テーブルのストレージ・エンジン名が出力されます。

例 20 ¥d+コマンドの出力

```
postgres=> ¥d+ data1
                                        Table "public.data1"
 Column |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
--------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 c1     | numeric               |           |          |         | main     |              |
 c2     | character varying(10) |           |          |         | extended |              |
Access method: heap
```

psql 変数の HIDE_TABLEAM（デフォルト値 off）に on を指定すると、Access Method 項目の出力を抑制することができます。

例 21 ¥d+コマンドの出力

```
postgres=> ¥set HIDE_TABLEAM on
postgres=> ¥d+ data1
                                        Table "public.data1"
 Column |         Type          | Collation | Nullable | Default | Storage  | Stats target | Description
--------+-----------------------+-----------+----------+---------+----------+--------------+-------------
 c1     | numeric               |           |          |         | main     |              |
 c2     | character varying(10) |           |          |         | extended |              |
```

□ psql コマンドによるアクセスメソッドの表示

¥dA コマンドでテーブル・アクセス・メソッドも表示されるようになりました。
従来はインデックスのみでした。

例 22 ¥dA コマンドの出力

```
postgres=> ¥dA
List of access methods
  Name  | Type
--------+-------
 brin   | Index
 btree  | Index
 gin    | Index
 gist   | Index
 hash   | Index
 heap   | Table
 spgist | Index
(7 rows)
```

3.1.10. pg_hba.conf ファイル

pg_hba.conf ファイルには、以下の新機能が追加されました。

□ clientcert 項目の新パラメーター

clientcert パラメーターに新しい設定値 verfy-full を指定できるようになりました。このオプションの後者は証明書の cn（共通名）がユーザー名または適切なマッピングと一致することも保証します。

□ GSSAPI 認証

クライアント認証に GSSAPI(Generic Security Standard Application Programming Interface)を利用できるようになりました。pg_hba.conf ファイルに hostgssenc / hostnogssenc のエントリーを記述することができます。GSSAPI 認証を有効にするにはインストール時の configure コマンドのオプションに--with-gssapi を指定する必要があります。

3.1.11. テキスト検索

テキスト検索の対応言語が増えました。PostgreSQL 11 では 16 言語でしたが、PostgreSQL 12 では 22 言語に対応しています。

例 23 テキスト検索設定

```
postgres=# ¥dF
               List of text search configurations
   Schema   |    Name    |              Description
------------+------------+---------------------------------------
 pg_catalog | arabic     | configuration for arabic language    << new
 pg_catalog | danish     | configuration for danish language
 pg_catalog | dutch      | configuration for dutch language
 pg_catalog | english    | configuration for english language
 pg_catalog | finnish    | configuration for finnish language
 pg_catalog | french     | configuration for french language
 pg_catalog | german     | configuration for german language
 pg_catalog | hungarian  | configuration for hungarian language
 pg_catalog | indonesian | configuration for indonesian language << new
 pg_catalog | irish      | configuration for irish language      << new
 pg_catalog | italian    | configuration for italian language
 pg_catalog | lithuanian | configuration for lithuanian language << new
 pg_catalog | nepali     | configuration for nepali language
 pg_catalog | norwegian  | configuration for norwegian language  << new
 pg_catalog | portuguese | configuration for portuguese language
 pg_catalog | romanian   | configuration for romanian language
 pg_catalog | russian    | configuration for russian language
 pg_catalog | simple     | simple configuration
 pg_catalog | spanish    | configuration for spanish language
 pg_catalog | swedish    | configuration for swedish language
 pg_catalog | tamil      | configuration for tamil language     << new
 pg_catalog | turkish    | configuration for turkish language
(22 rows)
```

3.1.12. libpq API

PostgreSQL に対する C 言語インターフェースには以下の関数が追加されました。

□ PQresultMemorySize

size_t PQresultMemorySize(const PGresult *res) API が追加されました。この関数は PGresult が確保したメモリー量を返します。アプリケーションのメモリー管理に利用することができます。

□ GetForeignDataWrapperExtended

ForeignDataWrapper* GetForeignDataWrapperExtended(Oid fwdid, bits16 flags) API が追加されました。

□ GetForeignServerExtended

ForeignServer* GetForeignServerExtended(Oid fwdid, bits16 flags) API が追加されました。

3.1.13. トランザクション ID

64 ビット化されたトランザクション ID が利用できるようになりました。API GetTopFullTransactionId と GetCurrentFullTransactionId が提供されます。ただし現状の heap では利用されていません。

3.1.14. クライアント環境変数

環境変数 PG_COLOR と PG_COLORS が追加されました。PG_COLOR には診断メッセージの色を使うかを指定します。指定できる値は always、auto、または never です。PG_COLORS にはエスケープ・シーケンスのカテゴリーとコードをイコール (=) で連結して指定します。複数のカテゴリーを指定する場合にはコロン (:) で区切ります。

表 13 環境変数 PG_COLORS 設定値

カテゴリー	デフォルト値	備考
error	01;31	
warning	01;35	
locus	01	

3.2. SQL 文の拡張

ここでは SQL 文に関係する新機能を説明しています。

3.2.1. ALTER TABLE 文

一部のシステムカタログの属性を変更できるようになりました。

例 24 システムカタログの変更

```
postgres=# SHOW allow_system_table_mods ;
 allow_system_table_mods
-------------------------
 on
(1 row)

postgres=# ALTER TABLE pg_attribute SET (autovacuum_vacuum_scale_factor=0) ;
ALTER TABLE
```

3.2.2. ALTER TYPE ADD VALUE 文

ALTER TYPE ADD VALUE 文がトランザクション・ブロック内で使えるようになりました。ただし、追加した値はそのトランザクション・ブロック内では使えません。

例 25 トランザクション・ブロック内の ALTER TYPE ADD VALUE 文

```
postgres=> BEGIN ;
BEGIN
postgres=> ALTER TYPE t1 ADD VALUE 'v3' ;
ALTER TYPE
```

3.2.3. COMMIT/ROLLBACK AND CHAIN 文

トランザクションを確定（COMMIT）または破棄（ROLLBACK）直後に、新規のトランザクションを開始する CHAIN 句が追加できるようになりました。COMMIT 文または ROLLBACK 文に AND CHAIN 句を指定します。明示的に CHAIN 句を否定する場合には、「AND NO CHAIN」を指定します。これらの文は PL/pgSQL を使った PROCEDURE 内

でも使うことができます。

例 26 COMMIT AND CHAIN

```
postgres=> BEGIN ;
BEGIN
postgres=> INSERT INTO data1 VALUES (100, 'data1') ;
INSERT 0 1
postgres=> COMMIT AND CHAIN ;
COMMIT
postgres=> INSERT INTO data1 VALUES (200, 'data2') ;
INSERT 0 1
postgres=> ROLLBACK AND CHAIN ;
ROLLBACK
postgres=> INSERT INTO data1 VALUES (300, 'data3') ;
INSERT 0 1
postgres=> COMMIT ;
COMMIT
```

CHAIN 句を指定されて開始されたトランザクションでは、トランザクション分離レベル等の属性は前トランザクションから維持されます。

例 27 トランザクション属性

```
postgres=> SHOW transaction_isolation ;
 transaction_isolation
-----------------------
 read committed
(1 row)

postgres=> BEGIN ISOLATION LEVEL SERIALIZABLE ;
BEGIN
postgres=> COMMIT AND CHAIN ;
COMMIT
postgres=> SHOW transaction_isolation ;
 transaction_isolation
-----------------------
 serializable
(1 row)
postgres=> COMMIT ;
COMMIT
postgres=> SHOW transaction_isolation ;
 transaction_isolation
-----------------------
 read committed
(1 row)
```

3.2.4. COPY 文

COPY 文には以下の拡張が追加されました。

□ COPY FROM 文

COPY FROM 文で特定の条件に合致するデータのみテーブルに格納することができるようになりました。COPY TO 文と同様に WHERE 句を使って条件を指定します。

例 28 COPY FROM WHERE 文

```
postgres=# COPY data1 FROM '/home/postgres/data1.csv' CSV DELIMITER ','
                WHERE mod(c1, 2) = 0 ;
COPY 50000
```

psql コマンドの¥copy コマンドでも同様に実行できます。

例 29 ¥copy WHERE コマンド

```
postgres=> ¥copy data1 FROM '/home/postgres/data1.csv' CSV DELIMITER ','
                WHERE mod(c1, 2) = 0 ;
COPY 50000
```

□ COPY FREEZE 文

パーティション・テーブルに対する COPY FREEZE 文の実行はエラーになります。この仕様は PostgreSQL 11.2 にも適用されました。

例 30 PostgreSQL 12 の COPY FREEZE 文

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY
RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES FROM (0) TO (100) ;
CREATE TABLE
postgres=> CREATE TABLE part1v2 PARTITION OF part1 FOR VALUES FROM (100) TO
(200) ;
CREATE TABLE
postgres=# COPY part1 FROM '/home/postgres/part1.csv' CSV FREEZE ;
ERROR:  cannot perform FREEZE on a partitioned table
```

2.2.5. CREATE AGGREGATE 文

CREATE AGGREGATE 文に、OR REPLACE 句を使用できるようになりました。

構文

```
CREATE [ OR REPLACE ] AGGREGATE name ( [ argmode ] [ argname ] arg_data_type
[ , ... ] )
```

3.2.6. CREATE INDEX 文

□ GiST インデックスの作成

GiST インデックスでカバリング・インデックスが利用できるようになりました。

例 31 GiST カバリング・インデックス

```
postgres=> CREATE TABLE data1(c1 INT, c2 box, c3 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE INDEX idx1_data1 ON data1 USING gist (c2) INCLUDE (c1) ;
CREATE INDEX
postgres=> ¥d data1
                    Table "public.data1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | integer               |           |          |
 c2     | box                   |           |          |
 c3     | character varying(10) |           |          |
Indexes:
    "idx1_data1" gist (c2) INCLUDE (c1)
```

□ GiST インデックスと VACUUM

空きページが VACUUM により再利用されるようになりました。

□ GIN インデックス作成時の WAL

インデックス作成時の WAL 出力量が大幅に削減されました。

3.2.7. CREATE STATISTICS 文

CREATE STATISTICS 文に mcv 句を使用できるようになりました。この値は多変量 MCV（Multivariate most-common values）を示します。通常の MCV リストを拡張し、最も頻繁な値の組み合わせを追跡します。取得した統計値は pg_statistic_ext カタログの stxmcv 列に保存されます。

例 32 MCV 統計

```
postgres=> CREATE TABLE stat1 (c1 NUMERIC, c2 NUMERIC, c3 VARCHAR(10)) ;
CREATE TABLE
postgres=> CREATE STATISTICS mcv_stat1(mcv) ON c1, c2 FROM stat1 ;
CREATE STATISTICS
```

3.2.8. CREATE TABLE 文

CREATE TABLE 文には以下の拡張が行われました。

□ 生成列（GENERATED 列）

生成列は、テーブルに対して計算結果を基にした列を定義します。列定義時に、データ型に続いて GENERATED ALWAYS AS (計算式) STORED 句を指定します。

例 33 生成列の定義

```
postgres=> CREATE TABLE gen1(c1 VARCHAR(10), c2 VARCHAR(10), c3 VARCHAR(20)
GENERATED ALWAYS AS (c1 || c2) STORED) ;
CREATE TABLE
postgres=> ¥d gen1
                                    Table "public.gen1"
 Column |         Type          | Collation | Nullable |Default
--------+-----------------------+-----------+----------+-----------------------
 c1     | character varying(10) |           |          |
 c2     | character varying(10) |           |          |
 c3     | character varying(20) |           |          | generated always as
(((c1::text || c2::text))) stored
```

INSERT 文や UPDATE 文には生成列に直接値を指定できません。DEFAULT 句のみが有効です。

例 34 生成列の更新

```
postgres=> INSERT INTO gen1 VALUES ('AB', 'CD', 'EF') ;
psql: ERROR:  cannot insert into column "c3"
DETAIL:  Column "c3" is a generated column.
postgres=> INSERT INTO gen1 VALUES ('AB', 'CD', DEFAULT) ;
INSERT 0 1
```

生成列の値となる計算値は INSERT や UPDATE 文実行時に行われ、結果が物理的に保存されます。

例 35 生成列の保存

```
postgres=# SELECT heap_page_items(get_raw_page('gen1', 0)) ;
                              heap_page_items
-----------------------------------------------------------------------------
 (1,8152,1,35,524,0,0,"(0,1)",3,2050,24,,,"¥¥x07414207434406b41424344")
(1 row)
```

上記の例では、c1 列が'AB'(=0x4142)、c2 列が'CD'(=0x4344)に加えて、c3 列に'ABCD'(=0x41424344)が格納されていることがわかります。

生成列の情報は、pg_attrdef カタログに追加された attgenerated 列に「s」が格納されていることでわかります。また information_schema column_column_usage テーブルが新規に追加されました。

例 36 生成列の定義情報

```
postgres=> SELECT * FROM information_schema.column_column_usage ;
 table_catalog | table_schema | table_name | column_name | dependent_column
---------------+--------------+------------+-------------+-----------------
 postgres      | public       | gen1       | c1          | c3
 postgres      | public       | gen1       | c2          | c3
(2 rows)

postgres=> SELECT attname, attgenerated FROM pg_attribute WHERE attname IN ('c1',
'c2', 'c3') ;
 attname | attgenerated
---------+--------------
 c1      |
 c2      |
 c3      | s
(3 rows)
```

生成列は、パーティション・キーに指定することはできません。また、他の生成列に依存する生成列を定義することはできません。

例 37 生成列の制約

```
postgres=> CREATE TABLE pgen1(c1 INT, c2 INT, c3 INT GENERATED ALWAYS AS
        (c1 + c2) STORED) PARTITION BY RANGE(c3) ;
psql: ERROR:  cannot use generated column in partition key
LINE 1: ...NERATED ALWAYS AS (c1 + c2) STORED) PARTITION BY RANGE(c3) ;
                                                             ^

DETAIL:  Column "c3" is a generated column.

postgres=> CREATE TABLE gen2 (c1 INT, c2 INT GENERATED ALWAYS AS (c1*2) STORED,
        c3 INT GENERATED ALWAYS AS (c2*2) STORED) ;
psql: ERROR:  cannot use generated column "c2" in column generation expression
LINE 1: ...AYS AS (c1*2) STORED, c3 INT GENERATED ALWAYS AS (c2*2) STOR...
                                                             ^

DETAIL:  A generated column cannot reference another generated column.
```

□ パーティション・テーブル定義の TABLESPACE 句

パーティション・テーブルの作成時に指定される TABLESPACE 句が有効になりました。これまでのバージョンでは TABLESPACE 句は無視されていました。またパーティション・テーブルの TABLESPACE 句の値が、パーティション作成時の標準のテーブル空間となります。

例 38 パーティション・テーブルの作成と TABLESPACE 句

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY
               LIST(c1) TABLESPACE ts1 ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES IN (100) ;
CREATE TABLE
postgres=> ¥d part1
               Partitioned table "public.part1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | numeric               |           |          |
 c2     | character varying(10) |           |          |
Partition key: LIST (c1)
Number of partitions: 1 (Use ¥d+ to list them.)
Tablespace: "ts1"

postgres=> ¥d part1v1
                    Table "public.part1v1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | numeric               |           |          |
 c2     | character varying(10) |           |          |
Partition of: part1 FOR VALUES IN ('100')
Tablespace: "ts1"
```

□ パーティション・テーブルの FOR VALUES 句

パーティションの FOR VALUES 句にリテラルではなく計算式や関数を指定できるようになりました。指定された計算式は CREATE TABLE 文実行時に一度だけ実行され、テーブル定義には計算値が保存されます。

例 39 パーティションの作成と FOR VALUES 句

```
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES IN
        (power(2, 3)) ;
CREATE TABLE
postgres=> ¥d part1v1
                    Table "public.part1v1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | numeric               |           |          |
 c2     | character varying(10) |           |          |
Partition of: part1 FOR VALUES IN ('8')
Tablespace: "ts1"
```

□ パーティション・テーブルに対する外部キー参照

外部キーとしてパーティション・テーブルを参照できるようになりました。

例 40 参照テーブルとしてパーティション・テーブルを参照

```
 postgres=> CREATE TABLE fkey1(c1 INT PRIMARY KEY, c2 VARCHAR(10)) PARTITION
BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE fkey1v1 PARTITION OF fkey1 FOR VALUES FROM (0) TO
(1000000) ;
CREATE TABLE
postgres=> CREATE TABLE fkey1v2 PARTITION OF fkey1 FOR VALUES FROM (1000000)
TO (2000000) ;
CREATE TABLE
postgres=> CREATE TABLE ref1(c1 INT REFERENCES fkey1(c1), c2 VARCHAR(10)) ;
CREATE TABLE
```

PostgreSQL 11 では、ref1 テーブルを作成しようとすると「ERROR: cannot reference partitioned table "fkey1"」エラーが発生していました。

□ インデックスに対する VACUUM 処理

WITH 句に VACUUM_INDEX_CLEANUP = OFF を指定することで、インデックスに対する VACUUM 処理を無効にすることができるようになりました。デフォルト値は ON で、従来通り VACUUM が行われます。

例 41 インデックスに対する VACUUM の抑制

```
postgres=> CREATE TABLE vacuum1(c1 INT, c2 VARCHAR(10)) WITH
        (VACUUM_INDEX_CLEANUP = OFF) ;
CREATE TABLE
postgres=> ¥d+ vacuum1
                                          Table "public.vacuum1"
 Column |         Type          | Collation | Nullable | Default | Storage  |
Stats target | Description
--------+-----------------------+-----------+----------+---------+----------+-
-------------+---
 c1     | integer               |           |          |         | plain    |
|
 c2     | character varying(10) |           |          |         | extended |
|
Access method: heap
Options: vacuum_index_cleanup=off
```

□ テーブル終端の空きブロック開放

テーブルの属性に VACUUM_TRUNCATE が追加されました。VACUUM 実行時にテーブル終端の空きブロックを解放するかを決定します。デフォルト値は ON で、従来と同様に空き領域を解放します。OFF に指定するとこの動作を行いません。

例 42 VACUUM による終端ブロックの解放

```
postgres=> CREATE TABLE vacuum1(c1 INT, c2 VARCHAR(10)) WITH
        (VACUUM_TRUNCATE = OFF) ;
CREATE TABLE
postgres=> ¥d+ vacuum1
                                          Table "public.vacuum1"
 Column |         Type          | Collation | Nullable | Default | Storage
| Stats target | Description
--------+-----------------------+-----------+----------+---------+------------
--+----------------------+-----------+----------+---------+----+-------------
-+-------------
 c1     | integer               |           |          |         | plain
|           |
 c2     | character varying(10) |           |          |         | extended
|           |
Access method: heap
Options: vacuum_truncate=off
```

3.2.9. EXPLAIN 文

EXPLAIN 文に、SETTINGS ON オプションを指定できるようになりました。このオプションは、デフォルト値から変更されている実行計画に関係するパラメーターの情報を出力します。

例 43 EXPLAIN (SETTINGS ON)

```
postgres=> SET random_page_cost = 1.0 ;
SET
postgres=> EXPLAIN (SETTINGS ON) SELECT * FROM data1 WHERE c1=100 ;
                                QUERY PLAN
-------------------------------------------------------------------------
 Index Scan using idx1_data1 on data1  (cost=0.29..2.31 rows=1 width=12)
   Index Cond: (c1 = '100'::numeric)
 Settings: random_page_cost = '1'
(3 rows)
```

3.2.10. REINDEX CONCURRENTLY 文

REINDEX 文に CONCURRENTLY オプションが追加できるようになりました。ロック範囲を縮小することでアプリケーションの稼働とインデックスの再作成を併存できるようになります。一時的に新しいインデックス（{インデックス名}_ccnew）を作成し、古いインデックスと入れ替えることで実現されています。

例 44 REINDEX CONCURRENTLY 文

```
postgres=> REINDEX (VERBOSE) TABLE CONCURRENTLY data1 ;
psql: INFO:  index "public.idx1_data1" was reindexed
psql: INFO:  index "pg_toast.pg_toast_16385_index" was reindexed
psql: INFO:  table "public.data1" was reindexed
DETAIL:  CPU: user: 3.25 s, system: 0.69 s, elapsed: 13.27 s.
REINDEX
```

REINDEX 文の変更に合わせて、reindexdb コマンドにも--concurrently オプションが追加されました。

例 45 reindexdb コマンド

```
$ reindexdb --dbname postgres --echo --concurrently
SELECT pg_catalog.set_config('search_path', '', false);
REINDEX DATABASE CONCURRENTLY postgres;
WARNING:  concurrent reindex is not supported for catalog relations, skipping
all
$
```

3.2.11. PL/pgSQL 追加チェック

パラメーターplpgsql.extra_warnings に以下の値を指定することができるようになりました。どちらもファンクション実行時に追加の警告やエラーを出力することができます。

□ strict_multi_assignment 設定

SELECT INTO 文で出力される列数と入力変数の数が一致しない場合に警告を出力します。下記の例ではファンクション内で２つの警告が発生しています。

例 46 strict_multi_assignment 設定

```
postgres=> psql
SET
postgres=> CREATE OR REPLACE FUNCTION strict1()
  RETURNS void
  LANGUAGE plpgsql
AS $$
  DECLARE
     x INTEGER ;
     y INTEGER ;
  BEGIN
     SELECT 1 INTO x, y ;
     SELECT 1, 2, 3 INTO x, y ;
  END ;
$$ ;
CREATE FUNCTION
postgres=> SELECT strict1() ;
psql: WARNING:  number of source and target fields in assignment do not match
DETAIL:  strict_multi_assignment check of extra_warnings is active.
HINT:  Make sure the query returns the exact list of columns.
psql: WARNING:  number of source and target fields in assignment do not match
DETAIL:  strict_multi_assignment check of extra_warnings is active.
HINT:  Make sure the query returns the exact list of columns.
 strict1
---------


(1 row)
```

□ too_many_rows 設定

SELECT INTO 文で複数レコードが返った場合にエラーを発生させ、プロシージャの実行を停止します。

例 47 too_many_rows 設定

```
postgres=> SET plpgsql.extra_errors to 'too_many_rows' ;
SET
postgres=> DO $$
        DECLARE x INTEGER ;
        BEGIN
            SELECT generate_series(1,2) INTO x ;
            RAISE NOTICE 'test output' ;
        END ;
        $$ ;
psql: ERROR:  query returned more than one row
HINT:  Make sure the query returns a single row, or use LIMIT 1
CONTEXT:  PL/pgSQL function inline_code_block line 4 at SQL statement
```

3.2.12. VACUUM / ANALYZE 文

VACUUM 文、ANALYZE 文には以下の機能が追加されました。

□ SKIP_LOCKED 句

ロックされたテーブルに対して VACUUM 文や ANALYZE 文を実行した場合、従来はロックの解除を待っていました。PostgreSQL 12 では、ロックされたテーブルをスキップするオプション SKIP_LOCKED 句が追加されました。処理がスキップされた場合には WARNING レベル（自動 VACUUM の場合は LOG レベル）のログが出力されます。スキップされた場合でも SQLSTATE は成功とみなされます。

例 48 テーブルのロック

```
postgres=> BEGIN ;
BEGIN
postgres=> LOCK TABLE lock1 IN EXCLUSIVE MODE ;
LOCK TABLE
```

例 49 ロックされたテーブルのスキップ

```
postgres=> VACUUM (SKIP_LOCKED) lock1 ;
psql: WARNING:  skipping vacuum of "lock1" --- lock not available
VACUUM
postgres=> ¥echo :SQLSTATE
00000
```

□ オプション指定構文

VACUUM 文と ANALYZE 文には、実行する動作を TRUE / FALSE または ON / OFF でも指定できるようになりました。

例 50 ON / OFF による操作の指定

```
postgres=> VACUUM (VERBOSE OFF, FULL ON, ANALYZE ON) data1 ;
VACUUM
postgres=> VACUUM (VERBOSE TRUE, FULL TRUE, ANALYZE FALSE) data1 ;
psql: INFO:  vacuuming "public.data1"
psql: INFO:  "data1": found 0 removable, 1000000 nonremovable row versions in
5406 pages
DETAIL:  0 dead row versions cannot be removed yet.
CPU: user: 0.23 s, system: 0.28 s, elapsed: 0.72 s.
VACUUM
```

□ インデックスに対する VACUUM の抑制

VACUUM 文に INDEX_CLEANUP 句に OFF を指定することで、インデックスに対する VACUUM 処理を抑制できるようになりました。省略した場合は、テーブルの VACUUM_INDEX_CLEANUP 属性に依存します。

例 51 インデックスに対する VACUUM 抑制

```
postgres=> VACUUM (VERBOSE ON, INDEX_CLEANUP OFF) data1 ;
psql: INFO:  vacuuming "public.data1"
psql: INFO:  "data1": found 0 removable, 0 nonremovable row versions in 0 out of
0 pages
DETAIL:  0 dead row versions cannot be removed yet, oldest xmin: 493
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
0 tuples and 0 item identifiers are left as dead.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
VACUUM
```

□ テーブル終端の空きページ切り詰め処理の抑制

VACUUM 文に TRUNCATE 句が追加されました。この属性に OFF を指定することで、テーブル終端の空き領域削除処理を抑制することができます。省略した場合は、テーブルの VACUUM_TRUNCATE 属性に依存します。

例 52 テーブル終端の空きページに対する削除抑制

```
postgres=> VACUUM (VERBOSE ON, TRUNCATE OFF) data1 ;
psql: INFO:  vacuuming "public.data1"
psql: INFO:  "data1": removed 50000 row versions in 541 pages
psql: INFO:  "data1": found 50000 removable, 50000 nonremovable row versions in
541 out of 541 pages
<<途中省略>>
There were 0 unused item pointers.
Skipped 0 pages due to buffer pins, 0 frozen pages.
0 pages are entirely empty.
CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s.
VACUUM
```

3.2.13. WITH SELECT 文

WITH 句で指定された共通テーブル式(CTE)は、従来すべて実体化(MATERIALIZED)していました。PostgreSQL 12 では非実体化（NOT MATERIALIZED）がデフォルトの動作に変更されました。これらの動作を変更するために、WITH 句に MATERIALIZED または NOT MATERIALIZED を指定できるようになりました。NOT MATERIALIZED 句を指定すると、WHERE 句の指定が WITH 句内にプッシュダウンできるようになります。
下記の例では、NOT MATERIALIZED 句を使うとインデックス検索が選択され、コストが下がっていることがわかります。

例 53 WITH NOT MATERIALIZED

```
postgres=> EXPLAIN WITH s AS NOT MATERIALIZED (SELECT * FROM data1)
        SELECT * FROM s WHERE c1=100 ;
                                QUERY PLAN
-------------------------------------------------------------------------
 Index Scan using data1_pkey on data1  (cost=0.42..8.44 rows=1 width=12)
   Index Cond: (c1 = '100'::numeric)
(2 rows)
```

例 54 WITH MATERIALIZED

```
postgres=> EXPLAIN WITH s AS MATERIALIZED (SELECT * FROM data1) SELECT *
        FROM s WHERE c1=100 ;
                                QUERY PLAN
-------------------------------------------------------------------------
 CTE Scan on s  (cost=15406.00..37906.00 rows=5000 width=70)
   Filter: (c1 = '100'::numeric)
   CTE s
     ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=12)
(4 rows)
```

3.2.14. 関数

以下の関数が追加／拡張されました。

□ SQL/JSON

SQL 2016 標準で提唱された SQL/JSON に関する一部の関数が提供されています。

例 55 jsonb_path_query_array 関数

```
postgres=> SELECT jsonb_path_query_array('{"a":[1,2,3,4,5]}', '$.a[*] ? (@ >=
$min && @ <= $max)', '{"min":2,"max":4}') ;
 jsonb_path_query_array
------------------------
 [2, 3, 4]
(1 row)
```

以下の関数が追加されました。

表 14 JSON/SQL 関数

関数名	説明
jsonb_path_exists	JSON パスが指定された JSON 値の項目を返すかどうかを確認します。
jsonb_path_match	指定された JSON 値に対する JSON パスの述語結果を返します。 最初の結果項目のみが考慮されます。
jsonb_path_query	指定された JSON 値の JSON パスによって返されたすべての JSON 項目を取得します。
jsonb_path_query_array	指定された JSON 値の JSON パスによって返されたすべての JSON 項目を取得し、結果を配列にラップします。
jsonb_path_query_first	指定された JSON 値の JSON パスによって返される最初の JSON 項目を取得します。

□ pg_partition_tree

pg_partition_tree はパーティション・テーブルのツリー構造を表示する関数です。階層化されたパーティション構造にも対応しています。パラメーターにパーティション・テーブルを指定します。パーティション・テーブルやパーティション以外のオブジェクト名を指定すると NULL が返ります。

例 56 pg_partition_tree 関数

```
postgres=> SELECT * FROM pg_partition_tree('part1') ;
  relid  | parentrelid | isleaf | level
---------+-------------+--------+-------
 part1   |             | f      |     0
 part1v1 | part1       | t      |     1
 part1v2 | part1       | t      |     1
(3 rows)

postgres=> SELECT pg_size_pretty(sum(pg_relation_size(relid))) AS total FROM
pg_partition_tree('part1') ;
 total
-------
 84 MB
(1 row)
```

□ pg_partition_root

pg_partition_root は指定されたパーティションの最上位パーティション・テーブル名を返す関数です。下記の例ではサブ・パーティションを作成し、pg_partition_root 関数を実行しています。

例 57 pg_partition_root 関数

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 NUMERIC, c3 VARCHAR(10)) PARTITION
BY LIST(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES IN (100) PARTITION
BY LIST (c2) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1v2 PARTITION OF part1v1 FOR VALUES IN (200) ;
CREATE TABLE
postgres=> SELECT pg_partition_root('part1v1v2') ;
 pg_partition_root
-------------------
 part1
(1 row)
```

□ pg_partition_ancestors

pg_partition_ancestors 関数は、指定されたパーティションを含むパーティション・テーブルの親に向かって一覧を出力します。

例 58 pg_partition_ancestors 関数

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY LIST(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES IN (100) ;
CREATE TABLE
postgres=> SELECT pg_partition_ancestors('part1v1') ;
 pg_partition_ancestors
------------------------
 part1v1
 part1
(2 rows)
```

□ pg_promote

スタンバイ・インスタンスをプライマリー・インスタンスに昇格させる関数です。従来は pg_ctl promote コマンドの実行が必要でした。パラメーターとして待機を行うか（デフォルト true）と、待機秒数（デフォルト 60 秒）を指定できます。この関数は処理が失敗した場合や、待機時間内に昇格が完了しない場合には false を、それ以外の場合は true を返します。

例 59 pg_promote 関数

```
postgres=# SELECT pg_promote(true, 90) ;
 pg_promote
------------
 t
(1 row)
```

□ pg_ls_tmpdir

一時データが保存されたファイル名のリストを返す pg_ls_tmpdir 関数が追加されました。パラメーターにはテーブル空間の OID を指定します。省略した場合は pg_default が指定さ

れたとみなされます。この関数の実行には SUPERUSER 権限または pg_monitor ロールが必要です。

例 60 pg_ls_tmpdir 関数

```
postgres=# SELECT * FROM pg_ls_tmpdir() ;
       name       |   size    |      modification
------------------+-----------+------------------------
 pgsql_tmp36911.6 | 148955136 | 2019-05-24 11:57:36+09
 pgsql_tmp37050.0 | 139722752 | 2019-05-24 11:57:36+09
 pgsql_tmp37051.0 | 138403840 | 2019-05-24 11:57:36+09
(3 rows)
```

□ pg_ls_archive_statusdir

アーカイブ・ファイルのステータスを取得する pg_ls_archive_statusdir 関数が追加されました。この関数は${PGDATA}/pg_wal/archive_status ディレクトリ内を検索し、ファイル名、サイズ、更新日時を出力します。実際にアーカイブされた WAL ファイルの情報を出力するわけではありません。この関数の実行には SUPERUSER 権限または pg_monitor ロールが必要です。

例 61 pg_ls_archive_statusdir 関数

```
postgres=# SELECT * FROM pg_ls_archive_statusdir() ;
             name              | size |      modification
-------------------------------+------+------------------------
 000000010000000000000000D.done |    0 | 2019-05-24 19:33:00+09
 000000010000000000000000E.done |    0 | 2019-05-24 19:33:01+09
 000000010000000000000000F.done |    0 | 2019-05-24 19:33:02+09
(3 rows)
```

□ date_trunc

この関数には Timezone の設定ができるようになりました。

例 62 date_trunc 関数とタイムゾーン

```
postgres=> SELECT date_trunc('day', TIMESTAMP WITH TIME ZONE '2019-05-24
20:38:40+00', 'Asia/Tokyo') ;
       date_trunc
------------------------
 2019-05-25 00:00:00+09
(1 row)
```

□ 双曲線関数

SQL: standards 2016 に含まれる、以下の双曲線関数が追加されました。

表 15 双曲線関数

関数名	説明	備考
log10	10 を底とする数値の対数	
sinh	ハイパボリックサイン	
cosh	ハイパボリックコサイン	
tanh	ハイパボリックタンジェント	
asinh	ハイパボリックアークサイン	
acosh	ハイパボリックアークコサイン	
atanh	ハイパボリックアークタンジェント	

例 63 双曲線関数

```
postgres=> SELECT log10(20), sinh(1) ;
       log10        |        sinh
--------------------+--------------------
 1.3010299956639813 | 1.1752011936438014
(1 row)
```

□ レプリケーション・スロットのコピー

既存のレプリケーション・スロットのコピーを行う関数が提供されました。レプリケーション・スロットの種類に応じて pg_copy_physical_replication_slot 関数と、pg_copy_logical_replication_slot 関数が提供されています。コピーを行うには、レプリケーション・スロットが使用されていることが必要です。

例 64 レプリケーション・スロットのコピー

```
postgres=# SELECT pg_create_physical_replication_slot('slot_1') ;
 pg_create_physical_replication_slot
-------------------------------------
 (slot_1,)
(1 row)
postgres=# SELECT pg_copy_physical_replication_slot('slot_1', 'slot_c') ;
psql: ERROR:  cannot copy a replication slot that doesn't reserve WAL
```

3.3. パラメーターの変更

PostgreSQL 12 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 16 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
archive_cleanup_command	recovery.conf ファイルから移行（sighup）	-
data_sync_retry (11.2 にも追加)	fsync システムコール失敗時の動作（postmaster）	off
default_table_access_method	デフォルトのストレージエンジン	heap
log_statement_sample_rate	SQL 文をログに出力する割合（superuser）	1
log_transaction_sample_rate	トランザクション制御文をログに出力する割合（superuser）	0
plan_cache_mode	プリペアド文を実行する際の実行計画をキャッシュする動作を変更（user）	auto
primary_conninfo	recovery.conf から移行（postmaster）	-
primary_slot_name	recovery.conf から移行（postmaster）	-
promote_trigger_file	recovery.conf から移行（sighup）	-
recovery_end_command	recovery.conf から移行（sighup）	-
recovery_min_apply_delay	recovery.conf から移行（sighup）	0
recovery_target	recovery.conf から移行（postmaster）	-
recovery_target_action	recovery.conf から移行（postmaster）	pause
recovery_target_inclusive	recovery.conf から移行（postmaster）	on
recovery_target_lsn	recovery.conf から移行（postmaster）	-
recovery_target_name	recovery.conf から移行（postmaster）	-
recovery_target_time	recovery.conf から移行（postmaster）	-
recovery_target_timeline	recovery.conf から移行（postmaster）	latest
recovery_target_xid	recovery.conf から移行（postmaster）	-
restore_comand	recovery.conf から移行（postmaster）	-
shared_memory_type	共有メモリーの種別を指定	OS 依存

パラメーター	説明（context）	デフォルト値
	（postmaster）	
ssl_library	SSL 機能を提供するライブラリ名（internal）	-
ssl_max_protocol_version	サポートするSSLプロトコルの最高バージョン（sighup）	-
ssl_min_protocol_version	サポートするSSLプロトコルの最低バージョン（sighup）	TLSv1
tcp_user_timeout	TCP タイムアウトの指定（user）	0
wal_init_zero	WAL ファイルを 0 埋めするか？（superuser）	on
wal_recycle	WAL ファイルを再利用するか？（superuser）	on

□ primary_conninfo パラメーター

プライマリー・インスタンスに対する接続文字列を指定します。従来は recovery.conf ファイルで指定していました。application_name 項目のデフォルト値は従来 walreceiver でしたが、cluster_name パラメーターが指定されている場合は cluster_name の値がデフォルト値に変更されました。pg_stat_replication カタログの application_name 列の値が変化します。

□ ssl_library パラメーター

このパラメーターは SSL 機能を提供するライブラリの名称を示します。Red Hat Enterprise Linux 環境で configure コマンド実行時に--with-openssl を指定した場合のパラメーター値は「OpenSSL」になります。

例 65 ssl_library パラメーター

```
postgres=# SHOW ssl_library ;
 ssl_library
-------------
 OpenSSL
(1 row)
```

□ shared_memory_type パラメーター

このパラメーターは共有メモリー（shared_buffers 等）の種類を指定します。

表 17 shared_memory_type パラメーター

設定値	説明	システムコール
mmap	無名メモリー・マップを使用します。	mmap
sysv	System V 共有メモリーを使用します。	shmget
windows	Windows 共有メモリーを使用します。	CreateFileMapping

このパラメーターの Linux におけるデフォルト値は mmap です。これは PostgreSQL 9.3 以降と同じ動作です。ごく小さな System V 共有メモリーに加えて、大部分の共有メモリーをメモリー・マップ（mmap）を使って構成します。このパラメーターを sysv に設定すると、PostgreSQL 9.2 以前の動作に戻すことができます。すべての共有メモリーを System V 共有メモリーを使って構成します。

□ plancache_mode パラメーター

このパラメーターはプリペアド文（PREPARE 文で作成）の実行計画をキャッシュする方法について設定します。デフォルト値は auto で、これまでのバージョンと同じ動作です。通常 PREPARE 文により作成された SQL 文が EXECUTE 文で実行されると、そのたびに実行計画が生成されます。下記の例では、EXECUTE 文に指定されたパラメーターによって実行計画が変化することがわかります。C2 列に格納されたデータに偏りがあり、plan0 データが少なく（インデックス検索が有効）、plan1 データが多い（テーブル全体検索が有効）ことを示しています。

例 66 EXECUTE 文の実行ごとに実行計画が変わる

```
postgres=> PREPARE sel1(VARCHAR) AS SELECT * FROM plan1 WHERE c2=$1 ;
PREPARE
postgres=> EXPLAIN EXECUTE sel1('plan0') ;
                                QUERY PLAN
-------------------------------------------------------------------------
 Index Scan using idx1_plan1 on plan1  (cost=0.42..4.44 rows=1 width=12)
   Index Cond: ((c2)::text = 'plan0'::text)
(2 rows)
postgres=> EXPLAIN EXECUTE sel1('plan1') ;
                            QUERY PLAN
-------------------------------------------------------------------
 Seq Scan on plan1  (cost=0.00..17906.01 rows=1000001 width=12)
   Filter: ((c2)::text = 'plan1'::text)
(2 rows)
```

同一の SQL 文を 5 回以上実行すると、実行計画がキャッシュされ次回からはパラメーターが変更されてもキャッシュされた実行計画（一般的な実行計画）が利用され可能性があります。下記の例では 6 回目の EXPLAIN 文の実行で、実行計画内の表示がリテラル値から$1 に変化しています。

例 67 実行計画がキャッシュされた

```
postgres=> EXPLAIN EXECUTE sel1('plan1') ;       -- 5 回繰り返し
                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on plan1  (cost=0.00..23311.01 rows=1000001 width=12)
   Filter: ((c2)::text = 'plan1'::text)
(2 rows)
postgres=> EXPLAIN EXECUTE sel1('plan1') ;       -- 6 回目（実行計画が変化）
                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on plan1  (cost=0.00..23311.01 rows=1000001 width=12)
   Filter: ((c2)::text = ($1)::text)
(2 rows)
```

新規に追加されたパラメーターplan_cache_mode はこの動作を変更します。パラメーター値を force_custom_plan に設定すると、実行計画のキャッシュ機能が無効になります。一方でパラメーター値を force_generic_plan に設定するとすぐに実行計画のキャッシュが有効になります。

例 68 設定値 force_generic_plan

```
postgres=> SET plan_cache_mode = force_generic_plan ;
SET
postgres=> PREPARE sel1(VARCHAR) AS SELECT * FROM plan1 WHERE c2=$1 ;
PREPARE
postgres=> EXPLAIN EXECUTE sel1('plan0') ;
                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on plan1  (cost=0.00..23312.50 rows=1000120 width=12)
   Filter: ((c2)::text = ($1)::text)
(2 rows)
```

□ data_sync_retry パラメーター

チェックポイント中に発行される fsync システムコールが失敗した際の動作を決定します。従来のバージョンでは fsync 関数は再実行されていました（data_sync_retry=on）、新しいバージョンのデフォルトの動作（data_sync_retry=off）は、fsync システムコールが失敗すると PANIC によるインスタンス停止が発生します。このパラメーターは、PostgreSQL 11.2 以降から追加されました。

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 18 変更されたパラメーター

パラメーター	変更内容
client_min_messages	ERROR よりも上位レベルに設定することができなくなりました。
dynamic_shared_memory_type	設定値 none が削除されました。
log_autovacuum_min_duration	ログの出力内容が VACUUM 実行状況により変化するようになりました。
log_connections	ログに application_name の情報が追加されました。
plpgsql.extra_warnings	以下のパラメーター値が追加されました。 - too_many_rows - strict_multi_assignment
trace_sort	ログの出力メッセージが変更されました。
wal_level	起動時に適切なレベルであるかチェックされるようになりました。
wal_sender_timeout	コンテキストが sighup から user に変更されました。
default_with_oids	on には設定できなくなりました。
recovery_target_timeline	設定値として current が追加されました，デフォルト値は latest に変更されました。
autovacuum_vacuum_cost_delay	データ型が integer から real に変更されました。

default_with_oids パラメーターは pg_settings カタログからは参照できないようになっています。

例 69 default_with_oids パラメーター

```
postgres=> SHOW default_with_oids ;
default_with_oids
-------------------
 off
(1 row)
postgres=> SET default_with_oids = on ;
psql: ERROR:  tables declared WITH OIDS are not supported
postgres=> SELECT COUNT(*) FROM pg_settings WHERE name='default_with_oids' ;
 count
-------
     0
(1 row)
```

□ wal_sender_timeout パラメーター

このパラメーターはユーザーがセッション単位に変更できるようになりました。これにより、ストリーミング・レプリケーション環境ではスレーブ・インスタンスからの接続単位でパラメーターを変更できるようになりました。

例 70 wal_sender_timeout パラメーター

```
$ grep primary_conninfo data/postgresql.conf
primary_conninfo = 'host=svrhost1 port=5432 user=postgres password=password
options=''-c wal_sender_timeout=5000'''
```

□ log_connections パラメーター

このパラメーターを on に設定した場合に出力されるログに application_name パラメーターの値が追加されるようになりました。

例 71 log_connections パラメーター

```
・psql コマンドからの接続
LOG:  connection authorized: user=postgres database=postgres application_name=psql

・pg_basebackup コマンドからの接続
LOG:  replication connection authorized: user=postgres application_name=pg_baseback

・Streaming Replication による接続
LOG:  replication connection authorized: user=postgres application_name=walreceiver
```

□ trace_sort パラメーター

このパラメーターを on に設定した場合の出力ログ・フォーマットが変更されました。

例 72 PostgreSQL 11 のログ（一部）

```
LOG:  -1 switching to external sort with 16 tapes: CPU: user: 0.00 s, system:
0.00 s, elapsed: 0.00 s
LOG:  -1 using 3951 KB of memory for read buffers among 15 input tapes
LOG:  performsort of -1 done (except 15-way final merge): CPU: user: 0.15 s,
system: 0.01 s, elapsed: 0.16 s
```

例 73 PostgreSQL 12 のログ（一部）

```
LOG:  worker -1 switching to external sort with 16 tapes: CPU: user: 0.00 s,
system: 0.00 s, elapsed: 0.00 s
LOG: worker -1 starting quicksort of run 1: CPU: user: 0.00 s, system: 0.00 s,
elapsed: 0.00 s
```

□ wal_level パラメーター

インスタンス起動時に、レプリケーション・スロットが存在する場合はパラメーター wal_level が適切な値であるかチェックされるようになりました。必要なレベルが設定されてないとインスタンス起動が失敗します。下記の例では Logical Replication 環境で wal_level を replica に変更してインスタンスを再起動しています。

例 74 wal_level のチェック

```
postgres=# ALTER SYSTEM SET wal_level=minimal ;
ALTER SYSTEM
postgres=# ¥q
$
$ pg_ctl -D data restart
waiting for server to shut down.... done
server stopped
waiting for server to start....2019-05-24 23:09:17.918 JST [32486] FATAL:  WAL
archival cannot be enabled when wal_level is "minimal"
 stopped waiting
pg_ctl: could not start server
Examine the log output.
$
```

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 19 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 11	PostgreSQL 12	備考
autovacuum_vacuum_cost_delay	20	2	
extra_float_digits	0	1	
jit	off	on	
recovery_target_timeline	''	latest	
server_version	11.3	12beta1	
server_version_num	110003	120000	
transaction_isolation	default	read committed	

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強化点を説明します。

3.4.1. configure

PostgreSQL をソースコードからインストールする際に実行する configure コマンドから、「--disable-strong-random」オプションが削除されました。GSSAPI を利用するための --with-gssapi オプションが追加されました。

3.4.2. initdb

initdb コマンドはデータベース・クラスターのタイムゾーンを決定する際に /etc/localtime ファイルを参照するようになりました。環境変数 TZ が指定されていない場合にこのファイルを参照します。

3.4.3. oid2name

oid2name コマンドはオプションが見直され、長い名前のオプションが利用できるようになりました。

表 20 追加されたオプション

短いオプション	追加された長いオプション	説明
-f	--filenode	ファイルノードの指定
-i	--indexes	インデックスとシーケンスを含む
-o	--oid	OID を指定
-q	--quiet	ヘッダを省略
-s	--tablespaces	テーブル空間 OID を表示
-S	--system-objects	システム・オブジェクトを含む
-t	--table	テーブル名を指定
-x	--extended	追加情報を出力
-d	--dbname	接続データベース名
-h	--host	ホスト名（-H オプションは廃止）
-p	--port	接続ポート番号
-U	--username	ユーザー名

3.4.4. pg_basebackup

pg_basebackup コマンドは--write-recovery-conf パラメーター（-R）を指定された場合の動作が変更されました。バックアップ先のフォルダに standby.signal ファイルが自動的に作成され、postgresql.auto.conf ファイルに primary_conninfo パラメーターが追記されます。この動作は PostgreSQL 12 以降のインスタンスに接続した場合にのみ実行されます。

例 75 pg_basebackup コマンドの-R パラメーター

```
$ pg_basebackup -D back -R
$ ls back
backup_label        log              pg_ident.conf   pg_replslot    pg_stat_tmp
PG_VERSION            postgresql.conf
base                  pg_commit_ts   pg_logical      pg_serial      pg_subtrans
pg_wal                standby.signal
current_logfiles    pg_dynshmem      pg_multixact     pg_snapshots    pg_tblspc
pg_xact
global               pg_hba.conf    pg_notify       pg_stat         pg_twophase
postgresql.auto.conf
$ cat back/postgresql.auto.conf
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=postgres passfile=''/home/postgres/.pgpass'' port=5432
sslmode=disable sslcompression=0 gssencmode=disable target_session_attrs=any'
primary_slot_name = 'slot1'
$
```

3.4.5. pg_checksums

チェックサムの整合性を検査するコマンド pg_verify_checksums は、名前が pg_checksums に 変更されました。またチェックサムの整合性をチェックするだけでなく、チェックサムの有効化／無効化を変更できるようになりました。

例 76 pg_checksums コマンドの使い方

```
$ pg_checksums --help
pg_checksums enables, disables or verifies data checksums in a PostgreSQL
database cluster.

Usage:
  pg_checksums [OPTION]... [DATADIR]

Options:
 [-D, --pgdata=]DATADIR  data directory
  -c, --check            check data checksums (default)
  -d, --disable          disable data checksums
  -e, --enable           enable data checksums
  -N, --no-sync          do not wait for changes to be written safely to disk
  -P, --progress         show progress information
  -v, --verbose          output verbose messages
  -r RELFILENODE         check only relation with specified relfilenode
  -V, --version          output version information, then exit
  -?, --help             show this help, then exit

If no data directory (DATADIR) is specified, the environment variable PGDATA
is used.

Report bugs to <pgsql-bugs@lists.postgresql.org>.
$
```

ブロックの整合性をチェックするためには、--check オプションを指定します。整合性異常が確認された場合、コマンドは戻り値 1 で終了します。

例 77 整合性のチェック

```
$ pg_checksums -D data --check
Checksum operation completed
Files scanned:  973
Blocks scanned: 11187
Bad checksums:  0
Data checksum version: 1
$ echo $?
0
$
```

データベース・クラスターのチェックサムを有効にする場合は --enable オプション（無効化する場合は--disable オプション）を指定します。チェックサムを有効にする場合でも、整合性のチェックが行われます。

例 78 整合性チェックの有効化

```
$ pg_checksums -D data --enable
Checksum operation completed
Files scanned:  973
Blocks scanned: 11187
pg_checksums: syncing data directory
pg_checksums: updating control file
Checksums enabled in cluster
$
```

このコマンドはインスタンスを正常に停止した状態で実行する必要があります。インスタンスが異常終了した場合や、インスタンス起動中のデータベース・クラスターに対しては実行できません。

例 79 異常終了したインスタンスと整合性の有効化

```
$ pg_ctl -D data -m immediate stop
waiting for server to shut down.... done server stopped
$ pg_checksums -D data --check
pg_checksums: cluster must be shut down
$
```

処理の状況を表示するためには、--progress オプション（-P オプション）を指定します。

例 80 コマンドの実行状況を表示

```
$ pg_checksums -D data --check --progress
87/87 MB (100%) computed
Checksum operation completed
Files scanned:  973
Blocks scanned: 11187
Bad checksums:  0
Data checksum version: 1
```

3.4.6. pg_ctl

pg_ctl コマンドには以下の機能が追加されました。

□ logrotate オプション

pg_ctl コマンドにログ・ローテーションを行うパラメーターlogrotate が追加されました。従来は logger プロセスに SIGHUP シグナルを送信する必要がありました。メッセージを出力しない場合は-s パラメーターも同時に指定します。

例 81 pg_ctl logrotate コマンド

```
$ pg_ctl -D data logrotate
server signaled to rotate log file
$
```

3.4.7. pg_dump

pg_dump コマンドには以下のオプションが追加されました。

□--on-conflict-do-nothing オプション

このオプションは出力される INSERT 文に ON CONFLICT DO NOTHING 句を自動的に付与します。--inserts オプションまたは--column-inserts オプションと一緒に指定する必要があります。

例 82 --on-conflict-do-nothing オプション

```
$ pg_dump -t data1 --inserts --on-conflict-do-nothing
--
-- PostgreSQL database dump
<<途中省略>>
-- Data for Name: data1; Type: TABLE DATA; Schema: public; Owner: demo
--

INSERT INTO public.data1 VALUES (1, 'data1') ON CONFLICT DO NOTHING;
INSERT INTO public.data1 VALUES (2, 'data1') ON CONFLICT DO NOTHING;
<<以下省略>>
```

□ --extra-float-digits オプション

このパラメーターに整数値を指定すると、pg_dump コマンドによるデータ取得前に「SET extra_float_digits=指定値」文が実行されます。ダンプファイルには SET 文は含まれません。指定できる値の範囲は-15 から 3 です。数字以外の値を指定した場合は 0 とみなされます。

□ --rows-per-insert オプション

このオプションは--inserts オプションと同時に使用します。複数のタプルを単一の INSERT 文で挿入することができます。値の範囲は 1 から 2,147,483,647 です。

例 83 –rows-per-insert オプション

```
$ pg_dump -t data1 --inserts --rows-per-insert=2
--
-- PostgreSQL database dump
<<途中省略>>
-- Data for Name: data1; Type: TABLE DATA; Schema: public; Owner: postgres
--

INSERT INTO public.data1 VALUES
        (1, 'data1'),
        (2, 'data1');
INSERT INTO public.data1 VALUES
        (3, 'data1'),
        (4, 'data1');
<<以下省略>>
```

3.4.8. pg_dumpall

pg_dumpall コマンドには以下のオプションが追加されました。

□ --extra-float-digits オプション

このパラメーターに整数値を指定すると、pg_dumpall コマンドによるデータ取得前に「SET extra_float_digits=指定値」文が実行されます。ダンプファイルには SET 文は含まれません。指定できる値の範囲は-15 から 3 です。数字以外の値を指定した場合は 0 とみなされます。

□ --exclude-database オプション

PostgreSQL 12 では--exclude-database オプションが追加されました。このオプションはバックアップから除外するデータベースを指定することができます。複数のデータベースを指定する場合には、psql コマンドと同様のパターンによる指定を行います。同オプションを複数回指定することもできます。下記の例では、除外するデータベースとして、demodb1 と demodb2 を指定しています。

例 84 --exclude-database オプション

```
$ pg_dumpall --exclude-database='demodb[12]' -f alldump.sql
```

□ 追加コメント

ユーザー設定（ALTER USER SET 文）や、データベース設定について、出力ファイルにコメントが追加されました。

例 85 追加されるコメント

```
-- User Configurations
-- User Config {ユーザー名}
-- Databases
-- Database {データベース名} dump
```

3.4.9. pg_rewind

pg_rewind コマンドにはストレージに対する sync システムコールを実行しない、--no-sync オプションが追加されました。

例 86 pg_rewind コマンド

```
$ pg_rewind --help
pg_rewind resynchronizes a PostgreSQL cluster with another copy of the cluster.

Usage:
  pg_rewind [OPTION]...

Options:
  -D, --target-pgdata=DIRECTORY  existing data directory to modify
      --source-pgdata=DIRECTORY  source data directory to synchronize with
      --source-server=CONNSTR    source server to synchronize with
  -n, --dry-run                  stop before modifying anything
  -N, --no-sync                  do not wait for changes to be written
                                 safely to disk
  -P, --progress                 write progress messages
      --debug                    write a lot of debug messages
  -V, --version                  output version information, then exit
  -?, --help                     show this help, then exit

Report bugs to <pgsql-bugs@postgresql.org>.
```

3.4.10. pg_restore

データの出力先として、標準出力を指定する場合には、「-f -」と指定します。

3.4.11. pg_upgrade

pg_upgrade コマンドには以下のオプションが追加されました。

□ --socketdir オプション

--socketdir オプション（または-s オプション）はローカル・ソケット作成用のディレクトリを指定します。

□ --clone オプション

--clone オプションは、"reflink"機能を使って高速なクローニングを行います。この機能は一部のオペレーティング・システムとファイル・システム上でのみ利用可能です。

例 87 pg_upgrade コマンド

```
$ pg_upgrade --help
pg_upgrade upgrades a PostgreSQL cluster to a different major version.

Usage:
  pg_upgrade [OPTION]...

Options:
  -b, --old-bindir=BINDIR       old cluster executable directory
<<途中省略>>
  -r, --retain                  retain SQL and log files after success
  -s, --socketdir=DIR           socket directory to use (default CWD)
  -U, --username=NAME           cluster superuser (default "postgres")
  -v, --verbose                 enable verbose internal logging
  -V, --version                 display version information, then exit
  --clone                       clone instead of copying files to new cluster
  -?, --help                    show this help, then exit
<<以下省略>>
```

3.4.12. psql

psql コマンドには以下の機能が追加されました。

□ CSV 出力

psql コマンドからの出力フォーマットを CSV 形式に変更できるようになりました。以下のいずれかの方法で変更できます。

- psql コマンドのパラメーター--csv を指定する
- psql コマンド内から¥pset format csv コマンドを実行する

列区切り文字のデフォルト値はカンマ（,）ですが、¥pset csv_fieldsep コマンドで変更することができます。出力される値の中に区切り文字が含まれる場合、値はダブルクオーテーション（”）で囲まれます。列のタイトルは¥pset tuples_only on コマンドで出力を抑制することができます。

例 88 ¥pset format csv コマンド

```
postgres=> ¥pset format csv
Output format is csv.
postgres=> ¥pset csv_fieldsep
Field separator for CSV is ",".
postgres=> SELECT * FROM data1 ;
c1,c2              ← タイトル出力
1,ABC
2,"AB,C"           ← データに列セパレータが含まれる場合
2,"AB""C"          ← データにダブルクオーテーションが含まれる場合
```

□ パーティション・テーブルの表示

¥d コマンド実行時にパーティション・テーブルが明示されるようになりました。

例 89 ¥d コマンド

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY
LIST(c1) ;
CERATE TABLE
postgres=> ¥d part1
                Partitioned table "public.part1"
 Column |         Type          | Collation | Nullable | Default
--------+-----------------------+-----------+----------+---------
 c1     | numeric               |           |          |
 c2     | character varying(10) |           |          |
Partition key: LIST (c1)
Number of partitions: 0
```

パーティション・インデックスのテーブル空間が表示されるようになりました。

例 90 ¥d コマンド

```
postgres=> CREATE INDEX idx1_part1 ON part1(c2) TABLESPACE ts1 ;
CREATE INDEX
postgres=> ¥d idx1_part1
       Partitioned index "public.idx1_part1"
 Column |         Type          | Key? | Definition
--------+-----------------------+------+------------
 c2     | character varying(10) | yes  | c2
btree, for table "public.part1"
Tablespace: "ts1"
```

□ 権限の表示

オブジェクト権限の一覧にパーティション・テーブルが明示されるようになりました。

例 91 ¥z コマンド

```
postgres=> CREATE TABLE part1(c1 NUMERIC, c2 VARCHAR(10)) PARTITION BY LIST (c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES IN (10) ;
CREATE TABLE
postgres=> ¥z
                                Access privileges
 Schema |  Name   |       Type        | Access privileges | Column privileges | Policies
--------+---------+-------------------+-------------------+-------------------+----------
 public | part1   | partitioned table |                   |                   |
 public | part1v1 | table             |                   |                   |
(2 rows)
```

□ 接続情報の表示

¥conninfo コマンド実行時に TCP/IP アドレスが出力されるようになりました。

例 92 ¥conninfo コマンド

```
$ psql -p 5432 -h pghost1 -d postgres -U demo
Password for user demo: <<PASSWORD>>
psql (12devel)
Type "help" for help.

postgres=> ¥conninfo
You are connected to database "postgres" as user "demo" on host "pghost1"
(address "192.168.1.101") at port "5432".
postgres=>
```

□ VERBOSITY 項目に SQLSTATE 指定

¥set VERBOSITY コマンドに SQLSTATE を指定できるようになりました。

例 93 ¥set VERBOSITY command

```
postgres=> ¥set VERBOSITY sqlstate
postgres=> SELECT * FROM not_exists ;
psql: ERROR:  42P01
```

□ パーティション・テーブルのみの表示

パーティション・テーブルのみを表示する¥dP コマンドが追加されました。

例 94 ¥dP コマンド

```
ostgres=> CREATE TABLE part1(c1 INT, c2 VARCHAR(2)) PARTITION BY RANGE(c1) ;
CREATE TABLE
postgres=> CREATE TABLE part1v1 PARTITION OF part1 FOR VALUES FROM (0) TO (1000000) ;
CREATE TABLE
postgres=> ¥dP
            List of partitioned relations
 Schema | Name  | Owner |       Type        | On table
--------+-------+-------+-------------------+----------
 public | part1 | demo  | partitioned table |
(1 row)

postgres=> ¥dP+
                          List of partitioned relations
 Schema | Name  | Owner |       Type        | On table | Total size | Description
--------+-------+-------+-------------------+----------+------------+-------------
 public | part1 | demo  | partitioned table |          | 0 bytes    |
(1 row)
```

3.4.13. vacuumdb

vacuumdb コマンドには以下のオプションが追加されました。

□ --disable-page-skipping オプション

このオプションは、VACUUM (DISABLE_PAGE_SKIPPING)文をコマンドから実行するための指定です。このオプションは PostgreSQL 9.6 以降のインスタンスに対して指定できます。

□ --skip-locked オプション

このオプションは、VACUUM (SKIPP_LOCKED)文をコマンドから実行するための指定です。このオプションは PostgreSQL 12 以降のインスタンスに対して指定できます。

例 95 古いバージョンに対する実行

```
$ vacuumdb -h remhost11 -d postgres -U postgres --skip-locked
vacuumdb: cannot use the "skip-locked" option on server versions older than
PostgreSQL 12
```

□ --min-mxid-age オプション

少なくとも指定されたマルチトランザクション ID の age を持つテーブルに対してのみ Vacuum または Analyze を行います。

□ --min-xid-age オプション

トランザクション ID の期間が指定された値以上のテーブルに対してのみ Vacuum または Analyze を行います。

3.4.14. vacuumlo

vacuumlo コマンドはオプションが見直され、長い名前のオプションが利用できるようになりました。

表 21 追加されたオプション

短いオプション	追加された長いオプション	説明
-l	--limit	削除するラージオブジェクトの上限
-n	--dry-run	実際には実行しない
-v	--verbose	追加情報の出力
-h	--host	接続ホスト
-p	--port	接続ポート番号
-U	--username	接続ユーザー名
-w	--no-password	パスワード・プロンプトを出力しない
-W	--password	パスワード入力を強制

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auto_explain.log_level

パラメーターauto_explain.log_level（デフォルト値 LOG）が追加されました。このパラメーターには auto_explain モジュールが出力するログのレベルを指定することができます。従来はログレベル LOG に固定されていました。

例 96 log_level パラメーター

```
postgres=# LOAD 'auto_explain' ;
LOAD
postgres=# SET auto_explain.log_level = NOTICE ;
SET
postgres=# SET auto_explain.log_min_duration = 0 ;
SET
postgres=# SELECT * FROM data1 WHERE c1=1000 ;
psql: NOTICE:  duration: 0.025 ms  plan:
Query Text: SELECT * FROM data1 WHERE c1=1000 ;
Index Scan using data1_pkey on data1  (cost=0.42..8.44 rows=1 width=12)
  Index Cond: (c1 = '1000'::numeric)
  c1  |  c2
------+-------
 1000 | data1
(1 row)
```

□ JIT コンパイル情報の追加

JIT コンパイル情報がログに出力されるようになりました（PostgreSQL 11.2 にバックポートされています）。

例 97 JIT コンパイル情報ログ

```
2019-05-24 22:25:11.027 JST [17749] LOG:  duration: 47.828 ms  plan:
        Query Text: SELECT COUNT(*) FROM data1 ;
        Aggregate  (cost=17906.00..17906.01 rows=1 width=8)
          ->  Seq Scan on data1  (cost=0.00..15406.00 rows=1000000 width=0)
        JIT:
          Functions: 2
          Options: Inlining false, Optimization false, Expressions true,
Deforming true
```

3.5.2. citext

64 ビットのハッシュ値を求める citext_hash_extended が追加されました。第２パラメーターは SEED を指定します。

例 98 citext_hash_extended 関数

```
postgres=> INSERT INTO hash1 VALUES ('abc') ;
INSERT 0 1
postgres=> SELECT citext_hash_extended(c1, 0) FROM hash1 ;
 citext_hash_extended
----------------------
 -6747756470228489321
(1 row)

postgres=> SELECT citext_hash_extended(c1, 1) FROM hash1 ;
 citext_hash_extended
----------------------
  2125675926272891485
(1 row)
```

3.5.3. hstore

64 ビットのハッシュ値を求める hstore_hash_extended が追加されました。第２パラメーターは SEED を指定します。

例 99 hstore_hash_extended 関数

```
postgres=> INSERT INTO hstore1 VALUES (hstore('key1', 'val1')) ;
INSERT 0 1
postgres=> SELECT hstore_hash_extended(c1, 0) FROM hstore1 ;
 hstore_hash_extended
----------------------
  4863461342754690677
(1 row)

postgres=> SELECT hstore_hash_extended(c1, 1) FROM hstore1 ;
 hstore_hash_extended
----------------------
  1673650576531873159
(1 row)
```

3.5.4. pg_stat_statements

Pg_stat_statements モジュールの pg_stat_statements_reset 関数には、統計情報の削除範囲を限定するパラメーターが追加されました。データベース ID、ユーザーID、クエリー ID を指定して、統計情報を削除できます。これらのパラメーターを省略した場合（デフォルト値 0）は、従来通りすべての統計情報が破棄されます。

例 100 pg_stat_statements_reset 関数

```
postgres=> ¥dfS pg_stat_statements_reset
List of functions
-[ RECORD 1 ]-------+-----------------------------------------------------------
Schema              | public
Name                | pg_stat_statements_reset
Result data type    | void
Argument data types | userid oid DEFAULT 0, dbid oid DEFAULT 0, queryid bigint
DEFAULT 0
Type                | func
```

また、この関数は pg_read_all_stats ロールを保持するユーザーでは実行できなくなりました。

3.5.5. postgres_fdw

postgres_fdw モジュールには、リモートセッションで実行できる GUC オプションを指定する「options」オプションが追加されました。work_mem パラメーターや geqo パラメーター等の設定を変更できます。設定値は「-c パラメーター名=値」の形式で記述します。複数のパラメーターを指定する場合には、「-c パラメーター名=値」全体をスペース区切りで指定します。

例 101 options パラメーター

```
postgres=# CREATE SERVER remsvr1 FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS (host 'remhost1', options '-c work_mem=4MB') ;
CREATE SERVER
postgres=# ALTER SERVER remsvr1 OPTIONS
        (SET options '-c work_mem=16MB -c geqo=off') ;
ALTER SERVER
postgres=# SELECT * FROM pg_foreign_server ;
-[ RECORD 1 ]-----------------------------------------------------
srvname    | remsvr1
srvowner   | 10
srvfdw     | 16412
srvtype    |
srvversion |
srvacl     |
srvoptions | {host=remhost1, "options=-c work_mem=16MB -c geqo=off"}
```

Hewlett Packard
Enterprise

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/devel/static/release-12.html
- Commitfests
  https://commitfest.postgresql.org/
- PostgreSQL 12 Beta Manual
  https://www.postgresql.org/docs/12/index.html
- Git
  https://git.postgresql.org/gitweb/?p=postgresql.git;a=summary
- GitHub
  https://github.com/postgres/postgres
- Open source developer based in Japan（Michael Paquier さん）
  http://paquier.xyz/
- PostgreSQL 12 Open Items
  https://wiki.postgresql.org/wiki/PostgreSQL_12_Open_Items
- Qiita（ぬこ@横浜さん）
  http://qiita.com/nuko_yokohama
- PostgreSQL Deep Dive
  http://pgsqldeepdive.blogspot.jp/ （Satoshi Nagayasu さん）
- pgsql-hackers Mailing list
  https://www.postgresql.org/list/pgsql-hackers/
- PostgreSQL 12 Beta 1 のアナウンス
  https://www.postgresql.org/about/news/1943/
- Slack - postgresql-jp
  https://postgresql-jp.slack.com/

Hewlett Packard
Enterprise

変更履歴

変更履歴

版	日付	作成者	説明
0.1	2019/04/22	篠田典良	内部レビュー版作成 レビュー担当（敬称略）： 永安悟史（アップタイム・テクノロジーズ合同会社） 高橋智雄 竹島彰子 北山貴広
1.0	2019/05/24	篠田典良	PostgreSQL 12 Beta 1 公開版に合わせて修正完了

以上