PostgreSQL 17 packs big performance and feature enhancements

This latest release of the popular open-source DBMS delivers incremental backups, replication, and JSON tables among other improvements.

In a significant milestone for the 2024 Stack Overflow developer's favorite relational database management system (DBMS), the PostgreSQL Global Development Group has released PostgreSQL 17. This latest version of the open-source DBMS introduces a host of new features and performance improvements that further solidify PostgreSQL's position as a leading database solution for enterprises and developers alike.

One of the most anticipated new features in PostgreSQL 17 is native support for incremental backups. Previously, you had to use third-party programs for this; now, it's baked into the server. This should reduce storage requirements and recovery times, making it particularly valuable for enterprise database environments where full backups are time-consuming and resource-intensive.

PostgreSQL 17 significantly expands its SQL/JSON capabilities, introducing new functions like JSON_TABLE(), JSON_EXISTS(), JSON_QUERY(), and JSON_VALUE(). These additions bring PostgreSQL more in line with the SQL:2023 standard, offering developers powerful tools for interacting with JSON documents in a SQL-friendly manner.

The new PostgreSQL also introduces the ability to configure the Simple Least Recently Used (SLRU) cache, which is crucial for handling subtransactions. This feature allows applications with high transaction volumes to run more efficiently without requiring significant rework of their transaction handling.

PostgreSQL also -- finally -- improved logical replication, which enables you to replicate data between PostgreSQL databases. Previously, if you used logical replication to start running a standby database in case of a failure, you had to resync the replication. This ate up a lot of time.

Database administrators won't have to deal with this delay from this release on.

"Our work on PostgreSQL 17," noted Jozef de Vries, chief product engineering officer of EnterpriseDB, a leading PostgresSQL company, "underscores a belief that Postgres isn't just a database -- it's a data platform supporting your most critical business systems. By introducing features like incremental backups and advancing features like JSON functionality and performance, EDB is doubling down on our commitment to the open source project while enabling enterprises to 'just use Postgres' to solve their most pressing data challenges."

In addition to new features, PostgreSQL 17 comes with several performance enhancements. The most significant of these are:

- **Improved handling of write-ahead log (WAL) locks, with some tests showing up to a 2x performance improvement for workloads with highly concurrent changes.**
- **Optimization of queries using the IN clause with B-tree indexes. These are PostgreSQL's standard index type. This will provide significantly improved execution times.**
- **Support for Intel AVX-512 chip instructions for certain functions.**

With its focus on advanced features and better performance, PostgreSQL 17 represents a substantial step forward for the database platform. As organizations rely increasingly on data-driven decision-making and AI applications, the enhancements in this release are likely to cement PostgreSQL's position as a go-to DBMS for modern database needs.

PostgreSQL 17 Released!

Posted on 2024-09-26 by PostgreSQL Global Development Group **PostgreSQL Project**

The PostgreSQL Global Development Group today announced the release of PostgreSQL 17, the latest version of the world's most advanced open source database.

PostgreSQL 17 builds on decades of open source development, improving its performance and scalability while adapting to emergent data access and storage patterns. This release of PostgreSQL adds significant overall performance gains, including an overhauled memory management implementation for vacuum, optimizations to storage access and improvements for high concurrency workloads, speedups in bulk loading and exports, and query execution improvements for indexes. PostgreSQL 17 has features that benefit brand new workloads and critical systems alike, such as additions to the developer experience with the SQL/JSON `JSON_TABLE` command, and enhancements to logical replication that simplify management of high availability workloads and major version upgrades.

"PostgreSQL 17 highlights how the global open source community, which drives the development of PostgreSQL, builds enhancements that help users at all stages of their database journey," said Jonathan Katz, a member of the PostgreSQL core team. "Whether it's improvements for operating databases at scale or new features that build on a delightful developer experience, PostgreSQL 17 will enhance your data management experience."

PostgreSQL, an innovative data management system known for its reliability, robustness, and extensibility, benefits from over 25 years of open source development from a global developer community and has become the preferred open source relational database for organizations of all sizes.

System-wide performance gains

The PostgreSQL vacuum process is critical for healthy operations, requiring server instance resources to operate. PostgreSQL 17 introduces a new internal memory structure for vacuum that consumes up to 20x less memory. This improves vacuum speed and also reduces the use of shared resources, making more available for your workload.

PostgreSQL 17 continues to improve performance of its I/O layer. High concurrency workloads may see up to 2x better write throughput due to improvements with write-ahead log (WAL) processing. Additionally, the new streaming I/O interface speeds up sequential scans (reading all the data from a table) and how quickly `ANALYZE` can update planner statistics.

PostgreSQL 17 also extends its performance gains to query execution. PostgreSQL 17 improves the performance of queries with `IN` clauses that use B-tree indexes, the default index method in PostgreSQL. Additionally, BRIN indexes now support parallel builds. PostgreSQL 17 includes several improvements for query planning, including optimizations for `NOT NULL` constraints, and improvements in processing common table expressions (`WITH` queries). This release adds more SIMD (Single Instruction/Multiple Data) support for accelerating computations, including using AVX-512 for the `bit_count` function.

Further expansion of a robust developer experience

PostgreSQL was the first relational database to add JSON support (2012), and PostgreSQL 17 adds to its implementation of the SQL/JSON standard. `JSON_TABLE` is now available in PostgreSQL 17, letting developers convert JSON data into a standard PostgreSQL table. PostgreSQL 17 now supports SQL/JSON constructors (`JSON`, `JSON_SCALAR`, `JSON_SERIALIZE`) and query functions (`JSON_EXISTS`, `JSON_QUERY`, `JSON_VALUE`), giving developers other ways of interfacing with their JSON data. This release adds more `jsonpath` expressions, with an emphasis of converting JSON data to a native PostgreSQL data type, including numeric, boolean, string, and date/time types.

PostgreSQL 17 adds more features to `MERGE`, which is used for conditional updates, including a `RETURNING` clause and the ability to update views. Additionally, PostgreSQL 17 has new capabilities for bulk loading and data exporting, including up to a 2x performance improvement when exporting large rows using the `COPY` command. `COPY` performance also has improvements when the source and destination encodings match, and includes a new option, `ON_ERROR`, that allows an import to continue even if there is an insert error.

This release expands on functionality both for managing data in partitions and data distributed across remote PostgreSQL instances. PostgreSQL 17 supports using identity columns and exclusion constraints on partitioned tables. The PostgreSQL foreign data wrapper (`postgres_fdw`), used to execute queries on remote PostgreSQL instances, can now push `EXISTS` and `IN` subqueries to the remote server for more efficient processing.

PostgreSQL 17 also includes a built-in, platform independent, immutable collation provider that's guaranteed to be immutable and provides similar sorting semantics to the `C` collation except with `UTF-8` encoding rather than `SQL_ASCII`. Using this new collation provider guarantees that your text-based queries will return the same sorted results regardless of where you run PostgreSQL.

Logical replication enhancements for high availability and major version upgrades

Logical replication is used to stream data in real-time across many use cases. However, prior to this release, users who wanted to perform a major version upgrade would have to drop logical replication slots, which requires resynchronizing data to subscribers after an upgrade. Starting with upgrades from PostgreSQL 17, users don't have to drop logical replication slots, simplifying the upgrade process when using logical replication.

PostgreSQL 17 now includes failover control for logical replication, making it more resilient when deployed in high availability environments. Additionally, PostgreSQL 17 introduces the `pg_createsubscriber` command-line tool for converting a physical replica into a new logical replica.

More options for managing security and operations

PostgreSQL 17 further extends how users can manage the overall lifecycle of their database systems. PostgreSQL has a new TLS option, `sslnegotiation`, that lets users perform a direct TLS handshakes when using ALPN (registered as `postgresql` in the ALPN directory). PostgreSQL 17 also adds the `pg_maintain` predefined role, which gives users permission to perform maintenance operations.

`pg_basebackup`, the backup utility included in PostgreSQL, now supports incremental backups and adds the `pg_combinebackup` utility to reconstruct a full backup. Additionally, `pg_dump` includes a new option called `--filter` that lets you select what objects to include when generating a dump file.

PostgreSQL 17 also includes enhancements to monitoring and analysis features. `EXPLAIN` now shows the time spent for local I/O block reads and writes, and includes two new options: `SERIALIZE` and `MEMORY`, useful for seeing the time spent in data conversion for network transmission, and how much memory was used. PostgreSQL 17 now reports the progress of vacuuming indexes, and adds the `pg_wait_events` system view that, when combined with `pg_stat_activity`, gives more insight into why an active session is waiting.

Additional Features

Many other new features and improvements have been added to PostgreSQL 17 that may also be helpful for your use cases. Please see the release notes for a complete list of new and changed features.

About PostgreSQL

PostgreSQL is the world's most advanced open source database, with a global community of thousands of users, contributors, companies and organizations. Built on over 35 years of engineering, starting at the University of California, Berkeley, PostgreSQL has continued with an unmatched pace of development.

Official E.1. Release 17 Sep 26, 2024

E.1.1. Overview
E.1.2. Migration to Version 17
E.1.3. Changes
Release date: 2024-09-26

E.1.1. Overview

PostgreSQL 17 contains many new features and enhancements, including:

- New memory management system for `VACUUM`, which reduces memory consumption and can improve overall vacuuming performance.
- New SQL/JSON capabilities, including constructors, identity functions, and the `JSON_TABLE()` function, which converts JSON data into a table representation.
- Various query performance improvements, including for sequential reads using streaming I/O, write throughput under high concurrency, and searches over multiple values in a btree index.
- Logical replication enhancements, including:
    - Failover control
    - pg_createsubscriber, a utility that creates logical replicas from physical standbys
    - pg_upgrade now preserves replication slots on both publishers and subscribers
- New client-side connection option, `sslnegotiation=direct`, that performs a direct TLS handshake to avoid a round-trip negotiation.
- pg_basebackup now supports incremental backup.
- `COPY` adds a new option, `ON_ERROR ignore`, that allows a copy operation to continue in the event of an error.

The above items and other new features of PostgreSQL 17 are explained in more detail in the sections below.

E.1.2. Migration to Version 17

A dump/restore using pg_dumpall or use of pg_upgrade or logical replication is required for those wishing to migrate data from any previous release. See Section 18.6 for general information on migrating to new major releases.

Version 17 contains a number of changes that may affect compatibility with previous releases. Observe the following incompatibilities:

- Change functions to use a safe search_path during maintenance operations (Jeff Davis) §
  This prevents maintenance operations (`ANALYZE`, `CLUSTER`, `REFRESH MATERIALIZED VIEW`, `REINDEX`, or `VACUUM`) from performing unsafe access. Functions used by expression indexes and materialized views that need to reference non-default schemas must specify a search path during function creation.
- Restrict `ago` to only appear at the end in `interval` values (Joseph Koshakow) § §
  Also, prevent empty interval units from appearing multiple times.
- Remove server variable old_snapshot_threshold (Thomas Munro) §
  This variable allowed vacuum to remove rows that potentially could be still visible to running transactions, causing "snapshot too old" errors later if accessed. This feature might be re-added to PostgreSQL later if an improved implementation is found.
- Change `SET SESSION AUTHORIZATION` handling of the initial session user's superuser status (Joseph Koshakow) §
  The new behavior is based on the session user's superuser status at the time the `SET SESSION AUTHORIZATION` command is issued, rather than their superuser status at connection time.
- Remove feature which simulated per-database users (Nathan Bossart) §
  The feature, `db_user_namespace`, was rarely used.
- Remove wal_sync_method value `fsync_writethrough` on Windows (Thomas Munro) §
  This value was the same as `fsync` on Windows.
- Change file boundary handling of two WAL file name functions (Kyotaro Horiguchi, Andres Freund, Bruce Momjian) §
  The functions `pg_walfile_name()` and `pg_walfile_name_offset()` used to report the previous LSN segment number when the LSN was on a file segment boundary; it now returns the current LSN segment.
- Remove server variable `trace_recovery_messages` since it is no longer needed (Bharath Rupireddy) §
- Remove information schema column `element_types.domain_default` (Peter Eisentraut) §
- Change pgrowlocks lock mode output labels (Bruce Momjian) §

- Remove `buffers_backend` and `buffers_backend_fsync` from `pg_stat_bgwriter` (Bharath Rupireddy) §
  These fields are considered redundant to similar columns in `pg_stat_io`.
- Rename I/O block read/write timing statistics columns of pg_stat_statements (Nazir Bilal Yavuz) §
  This renames `blk_read_time` to `shared_blk_read_time`, and `blk_write_time` to `shared_blk_write_time`.
- Change `pg_attribute.attstattarget` and `pg_statistic_ext.stxstattarget` to represent the default statistics target as `NULL` (Peter Eisentraut) § §
- Rename `pg_collation.colliculocale` to `colllocale` and `pg_database.daticulocale` to `datlocale` (Jeff Davis) §
- Rename `pg_stat_progress_vacuum` column `max_dead_tuples` to `max_dead_tuple_bytes`, rename `num_dead_tuples` to `num_dead_item_ids`, and add `dead_tuple_bytes` (Masahiko Sawada) § §
- Rename SLRU columns in system view `pg_stat_slru` (Alvaro Herrera) §
  The column names accepted by `pg_stat_reset_slru()` are also changed.

E.1.3. Changes

Below you will find a detailed account of the changes between PostgreSQL 17 and the previous major release.

E.1.3.1. Server

E.1.3.1.1. Optimizer

- Allow the optimizer to improve CTE plans by considering the statistics and sort order of columns referenced in earlier row output clauses (Jian Guo, Richard Guo, Tom Lane) § §
- Improve optimization of `IS NOT NULL` and `IS NULL` query restrictions (David Rowley, Richard Guo, Andy Fan) § §
  Remove `IS NOT NULL` restrictions from queries on `NOT NULL` columns and eliminate scans on `NOT NULL` columns if `IS NULL` is specified.
- Allow partition pruning on boolean columns on `IS [NOT] UNKNOWN` conditionals (David Rowley) §

- Improve optimization of range values when using containment operators <@ and @> (Kim Johan Andersson, Jian He) §
- Allow correlated `IN` subqueries to be transformed into joins (Andy Fan, Tom Lane) §
- Improve optimization of the `LIMIT` clause on partitioned tables, inheritance parents, and `UNION ALL` queries (Andy Fan, David Rowley) §
- Allow query nodes to be run in parallel in more cases (Tom Lane) §
- Allow `GROUP BY` columns to be internally ordered to match `ORDER BY` (Andrei Lepikhov, Teodor Sigaev) §
  This can be disabled using server variable enable_group_by_reordering.
- Allow `UNION` (without `ALL`) to use MergeAppend (David Rowley) §
- Fix MergeAppend plans to more accurately compute the number of rows that need to be sorted (Alexander Kuzmenkov) §
- Allow GiST and SP-GiST indexes to be part of incremental sorts (Miroslav Bendik) §
  This is particularly useful for `ORDER BY` clauses where the first column has a GiST and SP-GiST index, and other columns do not.
- Add columns to `pg_stats` to report range-type histogram information (Egor Rogov, Soumyadeep Chakraborty) §

E.1.3.1.2. Indexes

- Allow btree indexes to more efficiently find a set of values, such as those supplied by `IN` clauses using constants (Peter Geoghegan, Matthias van de Meent) §
- Allow BRIN indexes to be created using parallel workers (Tomas Vondra, Matthias van de Meent) §

E.1.3.1.3. General Performance

- Allow vacuum to more efficiently remove and freeze tuples (Melanie Plageman, Heikki Linnakangas) §
  WAL traffic caused by vacuum is also more compact.
- Allow vacuum to more efficiently store tuple references (Masahiko Sawada, John Naylor) § § § §
  Additionally, vacuum is no longer silently limited to one gigabyte of memory when maintenance_work_mem or autovacuum_work_mem are higher.
- Optimize vacuuming of relations with no indexes (Melanie Plageman) §
- Increase default vacuum_buffer_usage_limit to 2MB (Thomas Munro) §

- Improve performance when checking roles with many memberships (Nathan Bossart) §
- Improve performance of heavily-contended WAL writes (Bharath Rupireddy) §
- Improve performance when transferring large blocks of data to a client (Melih Mutlu) §
- Allow the grouping of file system reads with the new system variable io_combine_limit (Thomas Munro, Andres Freund, Melanie Plageman, Nazir Bilal Yavuz) § § §

E.1.3.1.4. Monitoring

- Create system view `pg_stat_checkpointer` (Bharath Rupireddy, Anton A. Melnikov, Alexander Korotkov) § § §
  Relevant columns have been removed from `pg_stat_bgwriter` and added to this new system view.
- Improve control over resetting statistics (Atsushi Torikoshi, Bharath Rupireddy) § § §
  Allow `pg_stat_reset_shared()` (with no arguments) and pg_stat_reset_shared(`NULL`) to reset all shared statistics. Allow pg_stat_reset_shared('slru') and `pg_stat_reset_slru()` (with no arguments) to reset SLRU statistics, which was already possible with pg_stat_reset_slru(NULL).
- Add log messages related to WAL recovery from backups (Andres Freund) §
- Add log_connections log line for `trust` connections (Jacob Champion) §
- Add log message to report walsender acquisition and release of replication slots (Bharath Rupireddy) §
  This is enabled by the server variable log_replication_commands.
- Add system view `pg_wait_events` that reports wait event types (Bertrand Drouvot) §
  This is useful for adding descriptions to wait events reported in `pg_stat_activity`.
- Add wait events for checkpoint delays (Thomas Munro) §
- Allow vacuum to report the progress of index processing (Sami Imseih) §
  This appears in system view `pg_stat_progress_vacuum` columns `indexes_total` and `indexes_processed`.

E.1.3.1.5. Privileges

- Allow granting the right to perform maintenance operations (Nathan Bossart) §
  The permission can be granted on a per-table basis using the `MAINTAIN` privilege and on a per-role basis via the `pg_maintain` predefined role. Permitted operations are `VACUUM`, `ANALYZE`, `REINDEX`, `REFRESH MATERIALIZED VIEW`, `CLUSTER`, and `LOCK TABLE`.
- Allow roles with `pg_monitor` membership to execute `pg_current_logfile()` (Pavlo Golub, Nathan Bossart) §

E.1.3.1.6. Server Configuration

- Add system variable allow_alter_system to disallow `ALTER SYSTEM` (Jelte Fennema-Nio, Gabriele Bartolini) §
- Allow `ALTER SYSTEM` to set unrecognized custom server variables (Tom Lane) §
  This is also possible with `GRANT ON PARAMETER`.
- Add server variable transaction_timeout to restrict the duration of transactions (Andrey Borodin, Japin Li, Junwang Zhao, Alexander Korotkov) § § §
- Add a builtin platform-independent collation provider (Jeff Davis) § § § §
  This supports `C` and `C.UTF-8` collations.
- Add server variable huge_pages_status to report the use of huge pages by Postgres (Justin Pryzby) §
  This is useful when huge_pages is set to `try`.
- Add server variable to disable event triggers (Daniel Gustafsson) §
  The setting, event_triggers, allows for the temporary disabling of event triggers for debugging.
- Allow the SLRU cache sizes to be configured (Andrey Borodin, Dilip Kumar, Alvaro Herrera) §
  The new server variables are commit_timestamp_buffers, multixact_member_buffers, multixact_offset_buffers, notify_buffers, serializable_buffers, subtransaction_buffers, and transaction_buffers. commit_timestamp_buffers, transaction_buffers, and subtransaction_buffers scale up automatically with shared_buffers.

E.1.3.1.7. Streaming Replication and Recovery

- Add support for incremental file system backup (Robert Haas, Jakub Wartak, Tomas Vondra) § §
  Incremental backups can be created using pg_basebackup's new `--incremental` option. The new application pg_combinebackup allows manipulation of base and incremental file system backups.

- Allow the creation of WAL summarization files (Robert Haas, Nathan Bossart, Hubert Depesz Lubaczewski) § § § §
  These files record the block numbers that have changed within an LSN range and are useful for incremental file system backups. This is controlled by the server variables summarize_wal and wal_summary_keep_time, and introspected with `pg_available_wal_summaries()`, `pg_wal_summary_contents()`, and `pg_get_wal_summarizer_state()`.
- Add the system identifier to file system backup manifest files (Amul Sul) §
  This helps detect invalid WAL usage.
- Allow connection string value `dbname` to be written when pg_basebackup writes connection information to `postgresql.auto.conf` (Vignesh C, Hayato Kuroda) §
- Add column `pg_replication_slots.invalidation_reason` to report the reason for invalid slots (Shveta Malik, Bharath Rupireddy) § §
- Add column `pg_replication_slots.inactive_since` to report slot inactivity duration (Bharath Rupireddy) § § §
- Add function `pg_sync_replication_slots()` to synchronize logical replication slots (Hou Zhijie, Shveta Malik, Ajin Cherian, Peter Eisentraut) § §
- Add the `failover` property to the replication protocol (Hou Zhijie, Shveta Malik) §

E.1.3.1.8. Logical Replication

- Add application pg_createsubscriber to create a logical replica from a physical standby server (Euler Taveira) §
- Have pg_upgrade migrate valid logical slots and subscriptions (Hayato Kuroda, Hou Zhijie, Vignesh C, Julien Rouhaud, Shlok Kyal) § §
  This allows logical replication to continue quickly after the upgrade. This only works for old PostgreSQL clusters that are version 17 or later.
- Enable the failover of logical slots (Hou Zhijie, Shveta Malik, Ajin Cherian) §
  This is controlled by an optional fifth argument to `pg_create_logical_replication_slot()`.
- Add server variable sync_replication_slots to enable failover logical slot synchronization (Shveta Malik, Hou Zhijie, Peter Smith) § §
- Add logical replication failover control to `CREATE/ALTER SUBSCRIPTION` (Shveta Malik, Hou Zhijie, Ajin Cherian) § §

- Allow the application of logical replication changes to use hash indexes on the subscriber (Hayato Kuroda) §
  Previously only btree indexes could be used for this purpose.
- Improve logical decoding performance in cases where there are many subtransactions (Masahiko Sawada) §
- Restart apply workers if subscription owner's superuser privileges are revoked (Vignesh C) §
  This forces reauthentication.
- Add `flush` option to `pg_logical_emit_message()` (Michael Paquier) §
  This makes the message durable.
- Allow specification of physical standbys that must be synchronized before they are visible to subscribers (Hou Zhijie, Shveta Malik) § §
  The new server variable is synchronized_standby_slots.
- Add worker type column to `pg_stat_subscription` (Peter Smith) §

E.1.3.2. Utility Commands

- Add new `COPY` option `ON_ERROR ignore` to discard error rows (Damir Belyalov, Atsushi Torikoshi, Alex Shulgin, Jian He, Yugo Nagata) § § § §
  The default behavior is `ON_ERROR stop`.
- Add new `COPY` option `LOG_VERBOSITY` which reports `COPY FROM` ignored error rows (Bharath Rupireddy) §
- Allow `COPY FROM` to report the number of skipped rows during processing (Atsushi Torikoshi) §
  This appears in system view column `pg_stat_progress_copy.tuples_skipped`.
- In `COPY FROM`, allow easy specification that all columns should be forced null or not null (Zhang Mingli) §
- Allow partitioned tables to have identity columns (Ashutosh Bapat) §
- Allow exclusion constraints on partitioned tables (Paul A. Jungwirth) §
  As long as exclusion constraints compare partition key columns for equality, other columns can use exclusion constraint-specific comparisons.
- Add clearer `ALTER TABLE` method to set a column to the default statistics target (Peter Eisentraut) §
  The new syntax is `ALTER TABLE ... SET STATISTICS DEFAULT`; using `SET STATISTICS -1` is still supported.
- Allow `ALTER TABLE` to change a column's generation expression (Amul Sul) §

The syntax is `ALTER TABLE ... ALTER COLUMN ... SET EXPRESSION`.

- Allow specification of table access methods on partitioned tables (Justin Pryzby, Soumyadeep Chakraborty, Michael Paquier) § §
- Add `DEFAULT` setting for `ALTER TABLE .. SET ACCESS METHOD` (Michael Paquier) §
- Add support for event triggers that fire at connection time (Konstantin Knizhnik, Mikhail Gribkov) §
- Add event trigger support for `REINDEX` (Garrett Thornburg, Jian He) §
- Allow parenthesized syntax for `CLUSTER` options if a table name is not specified (Nathan Bossart) §

E.1.3.2.1. `EXPLAIN`

- Allow `EXPLAIN` to report optimizer memory usage (Ashutosh Bapat) §
  The option is called `MEMORY`.
- Add `EXPLAIN` option `SERIALIZE` to report the cost of converting data for network transmission (Stepan Rutz, Matthias van de Meent) §
- Add local I/O block read/write timing statistics to `EXPLAIN`'s `BUFFERS` output (Nazir Bilal Yavuz) §
- Improve `EXPLAIN`'s display of SubPlan nodes and output parameters (Tom Lane, Dean Rasheed) §
- Add JIT `deform_counter` details to `EXPLAIN` (Dmitry Dolgov) §

E.1.3.3. Data Types

- Allow the `interval` data type to support `+/-infinity` values (Joseph Koshakow, Jian He, Ashutosh Bapat) §
- Allow the use of an `ENUM` added via `ALTER TYPE` if the type was created in the same transaction (Tom Lane) §
  This was previously disallowed.

E.1.3.4. MERGE

- Allow `MERGE` to modify updatable views (Dean Rasheed) §
- Add `WHEN NOT MATCHED BY SOURCE` to `MERGE` (Dean Rasheed) §
  `WHEN NOT MATCHED` on target rows was already supported.
- Allow `MERGE` to use the `RETURNING` clause (Dean Rasheed) §
  The new `RETURNING` function `merge_action()` reports on the DML that generated the row.

E.1.3.5. Functions

- Add function `JSON_TABLE()` to convert `JSON` data to a table representation (Nikita Glukhov, Teodor Sigaev, Oleg Bartunov, Alexander Korotkov, Andrew Dunstan, Amit Langote, Jian He) § §
  This function can be used in the `FROM` clause of `SELECT` queries as a tuple source.
- Add SQL/JSON constructor functions `JSON()`, `JSON_SCALAR()`, and `JSON_SERIALIZE()` (Nikita Glukhov, Teodor Sigaev, Oleg Bartunov, Alexander Korotkov, Andrew Dunstan, Amit Langote) §
- Add SQL/JSON query functions `JSON_EXISTS()`, `JSON_QUERY()`, and `JSON_VALUE()` (Nikita Glukhov, Teodor Sigaev, Oleg Bartunov, Alexander Korotkov, Andrew Dunstan, Amit Langote, Peter Eisentraut, Jian He) § § § § §
- Add jsonpath methods to convert `JSON` values to other `JSON` data types (Jeevan Chalke) §
  The jsonpath methods are `.bigint()`, `.boolean()`, `.date()`, `.decimal([precision [, scale]])`, `.integer()`, `.number()`, `.string()`, `.time()`, `.time_tz()`, `.timestamp()`, and `.timestamp_tz()`.
- Add `to_timestamp()` time zone format specifiers (Tom Lane) §
  `TZ` accepts time zone abbreviations or numeric offsets, while `OF` accepts only numeric offsets.
- Allow the session time zone to be specified by `AS LOCAL` (Vik Fearing) §
  This is useful when converting adding and removing time zones from time stamps values, rather than specifying the literal session time zone.
- Add functions `uuid_extract_timestamp()` and `uuid_extract_version()` to return UUID information (Andrey Borodin) §
- Add functions to generate random numbers in a specified range (Dean Rasheed) §
  The functions are `random(min, max)` and they take values of type `integer`, `bigint`, and `numeric`.
- Add functions to convert integers to binary and octal strings (Eric Radman, Nathan Bossart) §
  The functions are `to_bin()` and `to_oct()`.
- Add Unicode informational functions (Jeff Davis) §
  Function `unicode_version()` returns the Unicode version, `icu_unicode_version()` returns the ICU version, and

`unicode_assigned()` returns if the characters are assigned Unicode codepoints.

- Add function `xmltext()` to convert text to a single `XML` text node (Jim Jones) §
- Add function `to_regtypemod()` to return the type modifier of a type specification (David Wheeler, Erik Wienhold) §
- Add `pg_basetype()` function to return a domain's base type (Steve Chavez) §
- Add function `pg_column_toast_chunk_id()` to return a value's TOAST identifier (Yugo Nagata) §
  This returns `NULL` if the value is not stored in TOAST.

E.1.3.6. PL/pgSQL

- Allow plpgsql `%TYPE` and `%ROWTYPE` specifications to represent arrays of non-array types (Quan Zongliang, Pavel Stehule) §
- Allow plpgsql `%TYPE` specification to reference composite column (Tom Lane) §

E.1.3.7. libpq

- Add libpq function to change role passwords (Joe Conway) §
  The new function, `PQchangePassword()`, hashes the new password before sending it to the server.
- Add libpq functions to close portals and prepared statements (Jelte Fennema-Nio) §
  The functions are `PQclosePrepared()`, `PQclosePortal()`, `PQsendClosePrepared()`, and `PQsendClosePortal()`.
- Add libpq API which allows for blocking and non-blocking cancel requests, with encryption if already in use (Jelte Fennema-Nio) §
  Previously only blocking, unencrypted cancel requests were supported.
- Add libpq function `PQsocketPoll()` to allow polling of network sockets (Tristan Partin, Tom Lane) § §
- Add libpq function `PQsendPipelineSync()` to send a pipeline synchronization point (Anton Kirilov) §
  This is similar to `PQpipelineSync()` but it does not flush to the server unless the size threshold of the output buffer is reached.
- Add libpq function `PQsetChunkedRowsMode()` to allow retrieval of results in chunks (Daniel Vérité) §

- Allow TLS connections without requiring a network round-trip negotiation (Greg Stark, Heikki Linnakangas, Peter Eisentraut, Michael Paquier, Daniel Gustafsson) § § § § § § § §
  This is enabled with the client-side option `sslnegotiation=direct`, requires ALPN, and only works on PostgreSQL 17 and later servers.

E.1.3.8. psql

- Improve psql display of default and empty privileges (Erik Wienhold, Laurenz Albe) §
  Command `\dp` now displays `(none)` for empty privileges; default still displays as empty.
- Have backslash commands honor `\pset null` (Erik Wienhold, Laurenz Albe) §
  Previously `\pset null` was ignored.
- Allow psql's `\watch` to stop after a minimum number of rows returned (Greg Sabino Mullane) §
  The parameter is `min_rows`.
- Allow psql connection attempts to be canceled with control-C (Tristan Partin) §
- Allow psql to honor `FETCH_COUNT` for non-`SELECT` queries (Daniel Vérité) §
- Improve psql tab completion (Dagfinn Ilmari Mannsåker, Gilles Darold, Christoph Heiss, Steve Chavez, Vignesh C, Pavel Borisov, Jian He) § § § § § § § §

E.1.3.9. Server Applications

- Add application pg_walsummary to dump WAL summary files (Robert Haas) §
- Allow pg_dump's large objects to be restorable in batches (Tom Lane) §
  This allows the restoration of many large objects to avoid transaction limits and to be restored in parallel.
- Add pg_dump option `--exclude-extension` (Ayush Vatsa) §
- Allow pg_dump, pg_dumpall, and pg_restore to specify include/exclude objects in a file (Pavel Stehule, Daniel Gustafsson) §
  The option is called `--filter`.
- Add the `--sync-method` parameter to several client applications (Justin Pryzby, Nathan Bossart) §
  The applications are initdb, pg_basebackup, pg_checksums, pg_dump, pg_rewind, and pg_upgrade.

- Add pg_restore option `--transaction-size` to allow object restores in transaction batches (Tom Lane) §
  This allows the performance benefits of transaction batches without the problems of excessively large transaction blocks.
- Change pgbench debug mode option from `-d` to `--debug` (Greg Sabino Mullane) §
  Option `-d` is now used for the database name, and the new `--dbname` option can be used as well.
- Add pgbench option `--exit-on-abort` to exit after any client aborts (Yugo Nagata) §
- Add pgbench command `\syncpipeline` to allow sending of sync messages (Anthonin Bonnefoy) §
- Allow pg_archivecleanup to remove backup history files (Atsushi Torikoshi) §
  The option is `--clean-backup-history`.
- Add some long options to pg_archivecleanup (Atsushi Torikoshi) §
  The long options are `--debug`, `--dry-run`, and `--strip-extension`.
- Allow pg_basebackup and pg_receivewal to use dbname in their connection specification (Jelte Fennema-Nio) §
  This is useful for connection poolers that are sensitive to the database name.
- Add pg_upgrade option `--copy-file-range` (Thomas Munro) §
  This is supported on Linux and FreeBSD.
- Allow reindexdb `--index` to process indexes from different tables in parallel (Maxim Orlov, Svetlana Derevyanko, Alexander Korotkov) §
- Allow reindexdb, vacuumdb, and clusterdb to process objects in all databases matching a pattern (Nathan Bossart) § § §
  The new option `--all` controls this behavior.

E.1.3.10. Source Code

- Remove support for OpenSSL 1.0.1 (Michael Paquier) §
- Allow tests to pass in OpenSSL FIPS mode (Peter Eisentraut) § §
- Use CPU AVX-512 instructions for bit counting (Paul Amonson, Nathan Bossart, Ants Aasma) § §
- Require LLVM version 10 or later (Thomas Munro) §
- Use native CRC instructions on 64-bit LoongArch CPUs (Xudong Yang) §
- Remove AIX support (Heikki Linnakangas) §

- Remove the Microsoft Visual Studio-specific PostgreSQL build option (Michael Paquier) §
  Meson is now the only available method for Visual Studio builds.
- Remove configure option `--disable-thread-safety` (Thomas Munro, Heikki Linnakangas) § §
  We now assume all supported platforms have sufficient thread support.
- Remove configure option `--with-CC` (Heikki Linnakangas) §
  Setting the `CC` environment variable is now the only supported method for specifying the compiler.
- User-defined data type receive functions will no longer receive their data null-terminated (David Rowley) §
- Add incremental `JSON` parser for use with huge `JSON` documents (Andrew Dunstan) §
- Convert top-level `README` file to Markdown (Nathan Bossart) §
- Remove no longer needed top-level `INSTALL` file (Tom Lane) §
- Remove make's `distprep` option (Peter Eisentraut) §
- Add make support for Android shared libraries (Peter Eisentraut) §
- Add backend support for injection points (Michael Paquier) § § § §
  This is used for server debugging and they must be enabled at server compile time.
- Add dynamic shared memory registry (Nathan Bossart) §
  This allows shared libraries which are not initialized at startup to coordinate dynamic shared memory access.
- Fix `emit_log_hook` to use the same time value as other log records for the same query (Kambam Vinay, Michael Paquier) §
- Improve documentation for using `jsonpath` for predicate checks (David Wheeler) §

E.1.3.11. Additional Modules

- Allow joins with non-join qualifications to be pushed down to foreign servers and custom scans (Richard Guo, Etsuro Fujita) §
  Foreign data wrappers and custom scans will need to be modified to handle these cases.
- Allow pushdown of `EXISTS` and `IN` subqueries to postgres_fdw foreign servers (Alexander Pyhalov) §
- Increase the default foreign data wrapper tuple cost (David Rowley, Umair Shahid) § §
  This value is used by the optimizer.
- Allow dblink database operations to be interrupted (Noah Misch) §

- Allow the creation of hash indexes on ltree columns (Tommy Pavlicek) §
  This also enables hash join and hash aggregation on ltree columns.
- Allow unaccent character translation rules to contain whitespace and quotes (Michael Paquier) §
  The syntax for the `unaccent.rules` file has changed.
- Allow amcheck to check for unique constraint violations using new option `--checkunique` (Anastasia Lubennikova, Pavel Borisov, Maxim Orlov) §
- Allow citext tests to pass in OpenSSL FIPS mode (Peter Eisentraut) §
- Allow pgcrypto tests to pass in OpenSSL FIPS mode (Peter Eisentraut) §
- Remove some unused SPI macros (Bharath Rupireddy) §
- Remove adminpack contrib extension (Daniel Gustafsson) §
  This was used by now end-of-life pgAdmin III.
- Allow `ALTER OPERATOR` to set more optimization attributes (Tommy Pavlicek) §
  This is useful for extensions.
- Allow extensions to define custom wait events (Masahiro Ikeda) § § § §
  Custom wait events have been added to postgres_fdw and dblink.
- Add pg_buffercache function `pg_buffercache_evict()` to allow shared buffer eviction (Palak Chaturvedi, Thomas Munro) §
  This is useful for testing.

E.1.3.11.1. pg_stat_statements

- Replace `CALL` parameters in pg_stat_statements with placeholders (Sami Imseih) §
- Replace savepoint names stored in `pg_stat_statements` with placeholders (Greg Sabino Mullane) §
  This greatly reduces the number of entries needed to record `SAVEPOINT`, `RELEASE SAVEPOINT`, and `ROLLBACK TO SAVEPOINT` commands.
- Replace the two-phase commit GIDs stored in `pg_stat_statements` with placeholders (Michael Paquier) §
  This greatly reduces the number of entries needed to record `PREPARE TRANSACTION`, `COMMIT PREPARED`, and `ROLLBACK PREPARED`.
- Track `DEALLOCATE` in `pg_stat_statements` (Dagfinn Ilmari Mannsåker, Michael Paquier) §
  `DEALLOCATE` names are stored in `pg_stat_statements` as placeholders.

- Add local I/O block read/write timing statistics columns of `pg_stat_statements` (Nazir Bilal Yavuz) § §
  The new columns are `local_blk_read_time` and `local_blk_write_time`.
- Add JIT deform_counter details to `pg_stat_statements` (Dmitry Dolgov) §
- Add optional fourth argument (`minmax_only`) to `pg_stat_statements_reset()` to allow for the resetting of only min/max statistics (Andrei Zubkov) §
  This argument defaults to `false`.
- Add `pg_stat_statements` columns `stats_since` and `minmax_stats_since` to track entry creation time and last min/max reset time (Andrei Zubkov) §