EDB Postgres™ Advanced Server

Release Notes

Version 12

December 5, 2019

EDB Postgres Advanced Server Release Notes, Version 12
by EnterpriseDB® Corporation

EnterpriseDB Corporation, 34 Crosby Drive Suite 100, Bedford, MA 01730, USA
T +1 781 357 3390 **F** +1 978 589 5701 **E** info@enterprisedb.com **www**.enterprisedb.com

Table of Contents

1 Introduction

With the release of EDB Postgres Advanced Server 12.0, EnterpriseDB continues to lead as the only worldwide company to deliver innovative and low cost open-source-derived database solutions with commercial quality, ease of use, compatibility, scalability, and performance for small or large-scale enterprises.

EDB Postgres Advanced Server 12.0 is built on open-source PostgreSQL 12.0, which introduces a myriad of enhancements that enable databases to scale up and scale out in more efficient ways.

EDB Postgres Advanced Server 12.0 adds a number of new outstanding features, including:

- Interval Partition
- Logical decoding on standby
- `COMPOUND TRIGGER`
- `MEDIAN` and `LISTAGG`
- `CAST_MULTISET`

For more information about EnterpriseDB software, visit:

https://www.enterprisedb.com/enterprise-postgres/edb-postgres-platform

1.1 Installers and Documentation

EDB Postgres Advanced Server v12 is packaged and delivered as interactive installers for Windows; visit the EnterpriseDB website:

https://www.enterprisedb.com/advanced-downloads

RPM Packages are available for Linux from:

https://yum.enterprisedb.com/

Debian/Ubuntu Packages are available for download from:

https://apt.enterprisedb.com/

If you need to request repository access, visit:

https://www.enterprisedb.com/repository-access-request

Please note: the EnterpriseDB repository structure has changed; for detailed installation instructions, please see the EDB Postgres Advanced Server Linux Installation Guide.

Documentation is provided on the EnterpriseDB website. Visit:

https://www.enterprisedb.com/edb-docs/p/edb-postgres-advanced-server

1.1.1 Supported Platforms

EDB Postgres Advanced Server v12 supports 64 bit Linux and Windows server platforms. The Advanced Server 12 RPM packages are supported on the following 64-bit Linux platforms:

- Red Hat Enterprise Linux (x86_64) 6.x and 7.x
- CentOS (x86_64) 6.x and 7.x
- OEL (x86_64) 6.x and 7.x
- PPC-LE 8 running RHEL 7.x

The Advanced Server 12 native packages are supported on the following 64-bit Linux platforms:

- Debian 9
- Ubuntu 18.04 LTS

Graphical installers are supported on the following 64-bit Windows platforms:

- Windows 2019
- Windows Server 2016
- Windows Server 2012 R2 Server

Additional information about supported platforms is available on the EnterpriseDB website:

https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms

1.1.2 Component Certification

The following components are included in the EBD Postgres Advanced Server v12 release:

- Procedural Language Packs – PL/Perl 5.26, PL/Python 3.7, PL/TCL 8.6
- CloneSchema 1.10
- Parallel Clone 1.5
- pgAgent 4.15
- Slony 2.2.8
- Connectors JDBC 42.2.8, ODBC 12.00.0000 .NET 4.0.10.1, OCI 11.0.3.1
- pgAdmin 4.15
- pgBouncer 1.12.0
- pgPool-II & pgPool-IIExtensions 4.0.6
- MTK 53.0.0
- EDBPlus 38.0.0
- PostGIS-JDBC 2.2.1

2 EDB Postgres Advanced Server v12 Features

The major highlights of this release are :

- Advanced Server now supports interval partitioning. Interval partitioning is an extension to range partitioning where the system automatically creates the partition if a new tuple doesn't fit into existing partitions.

- For better Redwood compatibility, Advanced Server now treats the `SELECT UNIQUE` clause synonymous to `SELECT DISTINCT` clause (i.e. they can be used interchangeably in all places).

- Advanced Server now supports `COMPOUND TRIGGERS`. This provides a way to define a single trigger on a table that enables the user to specify actions for different DML event points. It allows defining `BEFORE STATEMENT`, `AFTER STATEMENT`, `BEFORE EACH ROW`, `AFTER EACH ROW` timing events in a single trigger.

- Advanced Server allows logical decoding on a standby server. With this feature, the user can now create a logical replication slot on standby server.

- Advanced Server now provides `CAST(MULTISET..)`. `CAST(MULTISET)` is an extension to the redwood style casting function `CAST(.. AS ..)` where the subquery result is like a table output `CAST` to a collection type.

- Advanced Server now supports two aggregate functions: `LISTAGG` and `MEDIAN`

- The `LISTAGG` function supports string aggregation that concatenates data from multiple rows into a single row in an ordered fashion.

- The `MEDIAN` function calculates a median value from the set of provided values. The aggregate function can be used with numeric, date/time and interval types and much like other aggregate functions, it can be used with window function as well.

- In addition to supporting `%type` and `%rowtype` variables for functions and procedures, Advanced Server now supports the same inside a package as well.

- Advanced Server now supports a Redwood compatible `to_timestamp` implementation, which facilitates stricter input string parsing for timestamp columns for EDB Loader.

- Advanced Server has re-implemented ROWIDs using identity data type after PG v12 PostgreSQL community removed support for OIDs (to which ROWIDs were mapped).

- Advanced Server has added the `SYS_GUID` function to generate and return a globally unique identifier in the form of 16-bytes of RAW data.

- Advanced Server now offers new view that provide information which is compatible with the Oracle data dictionary views:

  `{USER|ALL|DBA}_TAB_PRIVS`

  `{USER|ALL|DBA}_COL_PRIVS`

  `{USER|ALL|DBA}_TAB_DEPENDENCIES`

For information about Advanced Server features that are compatible with Oracle databases, see the following guides:

- *Database Compatibility for Oracle Developer's Guide*

- *Database Compatibility for Oracle Developer's Reference Guide*

- *Database Compatibility for Oracle Developer's Tools and Utilities Guide*

- *Database Compatibility for Oracle Developer's Built-in Package Guide*

2.1 Community PostgreSQL 12 Updates

Advanced Server 12 integrates all of the community PostgreSQL 12 features. To review a complete list of changes to the community PostgreSQL project and the contributors names, see the PostgreSQL 12 Release Notes at:

https://www.postgresql.org/docs/12/release-12.html

The following updates are available in PostgreSQL 12:

Partitioning Updates

Improve performance of many operations on partitioned tables.

Allow tables with thousands of child partitions to be processed efficiently by operations that only affect a small number of partitions.

Allow foreign keys to reference partitioned tables.

Improve speed of COPY into partitioned tables.

Allow partition bounds to be any expression.

Such expressions are evaluated at partitioned-table creation time. Previously, only simple constants were allowed as partition bounds.

Allow CREATE TABLE's tablespace specification for a partitioned table to affect the tablespace of its children.

Avoid sorting when partitions are already being scanned in the necessary order.

ALTER TABLE ATTACH PARTITION is now performed with reduced locking requirements.

Add partition introspection functions.

The new function pg_partition_root() returns the top-most parent of a partition tree, pg_partition_ancestors() reports all ancestors of a partition, and pg_partition_tree() displays information about partitions.

Include partitioned indexes in the system view pg_indexes .

Add psql command \dP to list partitioned tables and indexes.

Improve psql \d and \z display of partitioned tables.

Fix bugs that could cause ALTER TABLE DETACH PARTITION to leave behind incorrect dependency state, allowing subsequent operations to misbehave, for example by not dropping a former partition child index when its table is dropped.

Indexes

Improve performance and space utilization of btree indexes with many duplicates.

Previously, duplicate index entries were stored unordered within their duplicate groups. This caused overhead during index inserts, wasted space due to excessive page splits, and it reduced VACUUM's ability to recycle entire pages. Duplicate index entries are now sorted in heap-storage order.

Indexes pg_upgrade'd from previous releases will not have these benefits.

Allow multi-column btree indexes to be smaller.

Internal pages and min/max leaf page indicators now only store index keys until the change key, rather than all indexed keys. This also improves the locality of index access.

Indexes pg_upgrade'd from previous releases will not have these benefits.

Improve speed of btree index insertions by reducing locking overhead.

Support INCLUDE columns in GiST indexes.

Add support for nearest-neighbor (KNN) searches of SP-GiST indexes.

Reduce the WAL write overhead of GiST, GIN, and SP-GiST index creation.

Allow index-only scans to be more efficient on indexes with many columns.

Improve the performance of vacuum scans of GiST indexes.

Delete empty leaf pages during GiST VACUUM.

Reduce locking requirements for index renaming.

Optimizer

Allow CREATE STATISTICS to create most-common-value statistics for multiple columns.

This improves optimization for queries that test several columns, requiring an estimate of the combined effect of several WHERE clauses. If the columns are correlated and have non-uniform distributions then multi-column statistics will allow much better estimates.

Allow common table expressions (CTEs) to be inlined into the outer query.

Specifically, CTEs are automatically inlined if they have no side-effects, are not recursive, and are referenced only once in the query. Inlining can be prevented by specifying MATERIALIZED, or forced for multiply-referenced CTEs by specifying NOT MATERIALIZED. Previously, CTEs were never inlined and were always evaluated before the rest of the query.

Allow control over when generic plans are used for prepared statements.

This is controlled by the plan_cache_mode server parameter.

Improve optimization of partition and UNION ALL queries that have only a single child.

Improve processing of domains that have no check constraints.

Domains that are being used purely as type aliases no longer cause optimization difficulties.

Pre-evaluate calls of LEAST and GREATEST when their arguments are constants.

Improve optimizer's ability to verify that partial indexes with IS NOT NULL conditions are usable in queries.

Usability can now be recognized in more cases where the calling query involves casts or large x IN (array) clauses.

Compute ANALYZE statistics using the collation defined for each column.

Previously, the database's default collation was used for all statistics. This potentially gives better optimizer behavior for columns with non-default collations.

Improve selectivity estimates for inequality comparisons on ctid columns.

Improve optimization of joins on columns of type tid.

These changes primarily improve the efficiency of self-joins on ctid columns.

Fix the leakproofness designations of some btree comparison operators and support functions.

This allows some optimizations that previously would not have been applied in the presence of security barrier views or row-level security.

General Performance

Enable Just-in-Time (JIT) compilation by default, if the server has been built with support for it.

Note that this support is not built by default, but has to be selected explicitly while configuring the build.

Speed up keyword lookup.

Improve search performance for multi-byte characters in position() and related functions.

Allow toasted values to be minimally decompressed.

This is useful for routines that only need to examine the initial portion of a toasted field.

Allow ALTER TABLE ... SET NOT NULL to avoid unnecessary table scans.

This can be optimized when the table's column constraints can be recognized as disallowing nulls.

Allow ALTER TABLE ... SET DATA TYPE changing between timestamp and timestamptz to avoid a table rewrite when the session time zone is UTC.

In the UTC time zone, these two data types are binary compatible.

Improve speed in converting strings to int2 or int4 integers.

Allow parallelized queries when in SERIALIZABLE isolation mode.

Previously, parallelism was disabled when in this mode.

Use pread() and pwrite() for random I/O.

This reduces the number of system calls required for I/O.

Improve the speed of setting the process title on FreeBSD.

Monitoring

Allow logging of statements from only a percentage of transactions.

The parameter log_transaction_sample_rate controls this.

Add progress reporting to CREATE INDEX and REINDEX operations.

Progress is reported in the pg_stat_progress_create_index system view.

Add progress reporting to CLUSTER and VACUUM FULL.

Progress is reported in the pg_stat_progress_cluster system view.

Add progress reporting to pg_checksums.

This is enabled with the option --progress.

Add counter of checksum failures to pg_stat_database.

Add tracking of global objects in system view pg_stat_database.

Global objects are shown with a pg_stat_database.datid value of zero.

Add the ability to list the contents of the archive directory.

The function is pg_ls_archive_statusdir().

Add the ability to list the contents of temporary directories.

The function, pg_ls_tmpdir(), optionally allows specification of a tablespace.

Add information about the client certificate to the system view pg_stat_ssl.

The new columns are client_serial and issuer_dn. Column clientdn has been renamed to client_dn for clarity.

Restrict visibility of rows in pg_stat_ssl for unprivileged users.

At server start, emit a log message including the server version number.

Prevent logging “incomplete startup packet” if a new connection is immediately closed.

This avoids log spam from certain forms of monitoring.

Include the application_name, if set, in log_connections log messages.

Make the walreceiver set its application name to the cluster name, if set.

Add the timestamp of the last received standby message to pg_stat_replication.

Add a wait event for fsync of WAL segments.

Authentication

Add GSSAPI encryption support.

This feature allows TCP/IP connections to be encrypted when using GSSAPI authentication, without having to set up a separate encryption facility such as SSL. In support of this, add hostgssenc and hostnogssenc record types in pg_hba.conf for selecting connections that do or do not use GSSAPI encryption, corresponding to the existing hostssl and hostnossl record types. There is also a new gssencmode libpq option, and a pg_stat_gssapi system view.

Allow the clientcert pg_hba.conf option to check that the database user name matches the client certificate's common name.

This new check is enabled with clientcert=verify-full.

Allow discovery of an LDAP server using DNS SRV records.

This avoids the requirement of specifying ldapserver. It is only supported if PostgreSQL is compiled with OpenLDAP.

Server Configuration

Add ability to enable/disable cluster checksums using pg_checksums.

The cluster must be shut down for these operations.

Reduce the default value of autovacuum_vacuum_cost_delay to 2ms.

This allows autovacuum operations to proceed faster by default.

Allow vacuum_cost_delay to specify sub-millisecond delays, by accepting fractional values.

Allow time-based server parameters to use units of microseconds (us).

Allow fractional input for integer server parameters.

For example, SET work_mem = '30.1GB' is now allowed, even though work_mem is an integer parameter. The value will be rounded to an integer after any required units conversion.

Allow units to be defined for floating-point server parameters.

Add wal_recycle and wal_init_zero server parameters to control WAL file recycling.

Avoiding file recycling can be beneficial on copy-on-write file systems like ZFS.

Add server parameter tcp_user_timeout to control the server's TCP timeout.

Allow control of the minimum and maximum SSL protocol versions.

The server parameters are ssl_min_protocol_version and ssl_max_protocol_version.

Add server parameter ssl_library to report the SSL library version used by the server.

Add server parameter shared_memory_type to control the type of shared memory to use.

This allows selection of System V shared memory, if desired.

Streaming Replication And Recovery

Allow some recovery parameters to be changed with reload.

These parameters are archive_cleanup_command, promote_trigger_file, recovery_end_command, and recovery_min_apply_delay.

Allow the streaming replication timeout (wal_sender_timeout) to be set per connection.

Previously, this could only be set cluster-wide.

Add function pg_promote() to promote standbys to primaries.

Previously, this operation was only possible by using pg_ctl or creating a trigger file.

Allow replication slots to be copied.

The functions for this are pg_copy_physical_replication_slot() and pg_copy_logical_replication_slot().

Make max_wal_senders not count as part of max_connections.

Add an explicit value of current for recovery_target_timeline.

Make recovery fail if a two-phase transaction status file is corrupt.

Previously, a warning was logged and recovery continued, allowing the transaction to be lost.

Utility Commands

Add REINDEX CONCURRENTLY option to allow reindexing without locking out writes.

This is also controlled by the reindexdb application's --concurrently option.

Add support for generated columns.

The content of generated columns are computed from expressions (including references to other columns in the same table) rather than being specified by INSERT or UPDATE commands.

Add a WHERE clause to COPY FROM to control which rows are accepted.

This provides a simple way to filter incoming data.

Allow enumerated values to be added more flexibly.

Previously, ALTER TYPE ... ADD VALUE could not be called in a transaction block, unless it was part of the same transaction that created the enumerated type. Now it can be called in a later transaction, so long as the new enumerated value is not referenced until after it is committed.

Add commands to end a transaction and start a new one.

The commands are COMMIT AND CHAIN and ROLLBACK AND CHAIN.

Add VACUUM and CREATE TABLE options to prevent VACUUM from truncating trailing empty pages.

These options are vacuum_truncate and toast.vacuum_truncate. Use of these options reduces VACUUM's locking requirements, but prevents returning disk space to the operating system.

Allow VACUUM to skip index cleanup.

This change adds a VACUUM command option INDEX_CLEANUP as well as a table storage option vacuum_index_cleanup. Use of this option reduces the ability to reclaim space and can lead to index bloat, but it is helpful when the main goal is to freeze old tuples.

Add the ability to skip VACUUM and ANALYZE operations on tables that cannot be locked immediately.

This option is called SKIP_LOCKED.

Allow VACUUM and ANALYZE to take optional Boolean argument specifications.

Prevent TRUNCATE, VACUUM and ANALYZE from requesting a lock on tables for which the user lacks permission.

This prevents unauthorized locking, which could interfere with user queries.

Add EXPLAIN option SETTINGS to output non-default optimizer settings.

This output can also be obtained when using auto_explain by setting auto_explain.log_settings.

Add OR REPLACE option to CREATE AGGREGATE.

Allow modifications of system catalogs' options using ALTER TABLE.

Modifications of catalogs' reloptions and autovacuum settings are now supported. (Setting allow_system_table_mods is still required.)

Use all key columns' names when selecting default constraint names for foreign keys.

Previously, only the first column name was included in the constraint name, resulting in ambiguity for multi-column foreign keys.

Data Types

Update assorted knowledge about Unicode to match Unicode 12.1.0.

This fixes, for example, cases where psql would misformat output involving combining characters.

Update Snowball stemmer dictionaries with support for new languages.

This adds word stemming support for Arabic, Indonesian, Irish, Lithuanian, Nepali, and Tamil to full text search.

Allow creation of collations that report string equality for strings that are not bit-wise equal.

This feature supports “nondeterministic” collations that can define case- and accent-agnostic equality comparisons. Thus, for example, a case-insensitive uniqueness constraint on a text column can be made more easily than before. This is only supported for ICU collations.

Add support for ICU collation attributes on older ICU versions.

This allows customization of the collation rules in a consistent way across all ICU versions.

Allow data type name to more seamlessly be compared to other text types.

Type name now behaves much like a domain over type text that has default collation “C”. This allows cross-type comparisons to be processed more efficiently.

Functions

Add support for the SQL/JSON path language.

This allows execution of complex queries on JSON values using an SQL-standard language.

Add support for hyperbolic functions.

Also add log10() as an alias for log(), for standards compliance.

Improve the accuracy of statistical aggregates like variance() by using more precise algorithms.

Allow date_trunc() to have an additional argument to control the time zone.

This is faster and simpler than using the AT TIME ZONE clause.

Adjust to_timestamp()/to_date() functions to be more forgiving of template mismatches.

This new behavior more closely matches the Oracle functions of the same name.

Fix assorted bugs in XML functions.

Specifically, in XMLTABLE, xpath(), and xmlexists(), fix some cases where nothing was output for a node, or an unexpected error was thrown, or necessary escaping of XML special characters was omitted.

Allow the BY VALUE clause in XMLEXISTS and XMLTABLE.

This SQL-standard clause has no effect in PostgreSQL's implementation, but it was unnecessarily being rejected.

Prevent current_schema() and current_schemas() from being run by parallel workers, as they are not parallel-safe.

Allow RECORD and RECORD[] to be used as column types in a query's column definition list for a table function that is declared to return RECORD.

PL/PgSQL

Allow SQL commands and variables with the same names as those commands to be used in the same PL/pgSQL function.

For example, allow a variable called comment to exist in a function that calls the COMMENT SQL command. Previously this combination caused a parse error.

Add new optional warning and error checks to PL/pgSQL.

The new checks allow for run-time validation of INTO column counts and single-row results.

Client Interfaces

Add connection parameter tcp_user_timeout to control libpq's TCP timeout.

Allow libpq (and thus psql) to report only the SQLSTATE value in error messages.

Add libpq function PQresultMemorySize() to report the memory used by a query result.

Remove the no-display/debug flag from libpq's options connection parameter.

This allows this parameter to be set by postgres_fdw.

Allow ecpg to create variables of data type bytea.

This allows ECPG clients to interact with bytea data directly, rather than using an encoded form.

Add PREPARE AS support to ECPG.

Client Applications

Allow vacuumdb to select tables for vacuum based on their wraparound horizon.

The options are --min-xid-age and --min-mxid-age.

Allow vacuumdb to disable waiting for locks or skipping all-visible pages.

The options are --skip-locked and --disable-page-skipping.

Add colorization to the output of command-line utilities.

This is enabled by setting the environment variable PG_COLOR to always or auto. The specific colors used can be adjusted by setting the environment variable PG_COLORS, using ANSI escape codes for colors. For example, the default behavior is equivalent to PG_COLORS="error=01;31:warning=01;35:locus=01".

psql

Add CSV table output mode in psql.

This is controlled by \pset format csv or the command-line --csv option.

Show the manual page URL in psql's \help output for a SQL command.

Display the IP address in psql's \conninfo (Fabien Coelho)

Improve tab completion of CREATE TABLE, CREATE TRIGGER, CREATE EVENT TRIGGER, ANALYZE, EXPLAIN, VACUUM, ALTER TABLE, ALTER INDEX, ALTER DATABASE, and ALTER INDEX ALTER COLUMN.

pgbench

Allow values produced by queries to be assigned to pgbench variables.

The command for this is \gset.

Improve precision of pgbench's --rate option.

Improve pgbench's error reporting with clearer messages and return codes.

Server Applications

Allow control of log file rotation via pg_ctl.

Previously, this was only possible via an SQL function or a process signal.

Properly detach the new server process during pg_ctl start.

This prevents the server from being shut down if the shell script that invoked pg_ctl is interrupted later.

Allow pg_upgrade to use the file system's cloning feature, if there is one.

The --clone option has the advantages of --link, while preventing the old cluster from being changed after the new cluster has started.

Allow specification of the socket directory to use in pg_upgrade.

This is controlled by --socketdir; the default is the current directory.

Allow pg_checksums to disable fsync operations.

This is controlled by the --no-sync option.

Allow pg_rewind to disable fsync operations.

Fix pg_test_fsync to report accurate open_datasync durations on Windows.

pg_dump, pg_dumpall, pg_restore

When pg_dump emits data with INSERT commands rather than COPY, allow more than one data row to be included in each INSERT.

The option controlling this is --rows-per-insert.

Allow pg_dump to emit INSERT ... ON CONFLICT DO NOTHING.

This avoids conflict failures during restore. The option is --on-conflict-do-nothing.

Decouple the order of operations in a parallel pg_dump from the order used by a subsequent parallel pg_restore.

This allows pg_restore to perform more-fully-parallelized parallel restores, especially in cases where the original dump was not done in parallel. Scheduling of a parallel pg_dump is also somewhat improved.

Allow the extra_float_digits setting to be specified for pg_dump and pg_dumpall.

This is primarily useful for making dumps that are exactly comparable across different source server versions. It is not recommended for normal use, as it may result in loss of precision when the dump is restored.

Add --exclude-database option to pg_dumpall.

Source Code

Add CREATE ACCESS METHOD command to create new table types.

This enables the development of new table access methods, which can optimize storage for different use cases. The existing heap access method remains the default.

Add planner support function interfaces to improve optimizer estimates, inlining, and indexing for functions.

This allows extensions to create planner support functions that can provide function-specific selectivity, cost, and row-count estimates that can depend on the function's arguments. Support functions can also supply simplified representations and index conditions, greatly expanding optimization possibilities.

Simplify renumbering manually-assigned OIDs, and establish a new project policy for management of such OIDs.

Patches that manually assign OIDs for new built-in objects (such as new functions) should now randomly choose OIDs in the range 8000—9999. At the end of a development cycle, the OIDs used by committed patches will be renumbered down to lower numbers, currently somewhere in the 4xxx range, using the new renumber_oids.pl script. This approach should greatly reduce the odds of OID collisions between different in-process patches.

While there is no specific policy reserving any OIDs for external use, it is recommended that forks and other projects needing private manually-assigned OIDs use numbers in the high 7xxx range. This will avoid conflicts with recently-merged patches, and it should be a long time before the core project reaches that range.

Build Cygwin binaries using dynamic instead of static libraries.

Remove configure switch --disable-strong-random.

A strong random-number source is now required.

printf-family functions, as well as strerror and strerror_r, now behave uniformly across platforms within Postgres code.

Notably, printf understands %m everywhere; on Windows, strerror copes with Winsock error codes (it used to do so in backend but not frontend code); and strerror_r always follows the GNU return convention.

Require a C99-compliant compiler, and MSVC 2013 or later on Windows.

Use pandoc, not lynx, for generating plain-text documentation output files.

This affects only the INSTALL file generated during make dist and the seldom-used plain-text postgres.txt output file. Pandoc produces better output than lynx and avoids some locale/encoding issues. Pandoc version 1.13 or later is required.

Support use of images in the PostgreSQL documentation.

Additional Modules

Allow ORDER BY sorts and LIMIT clauses to be pushed to postgres_fdw foreign servers in more cases.

Improve optimizer cost accounting for postgres_fdw queries.

Properly honor WITH CHECK OPTION on views that reference postgres_fdw tables.

While CHECK OPTIONs on postgres_fdw tables are ignored (because the reference is foreign), views on such tables are considered local, so this change enforces CHECK OPTIONs on them. Previously, only INSERTs and UPDATEs with RE

2.2 How to Report Problems

To report any issues you are having please contact EnterpriseDB's technical support staff:

- Email: support@enterprisedb.com
- Phone: +1-732-331-1320 or 1-800-235-5891 (US Only)