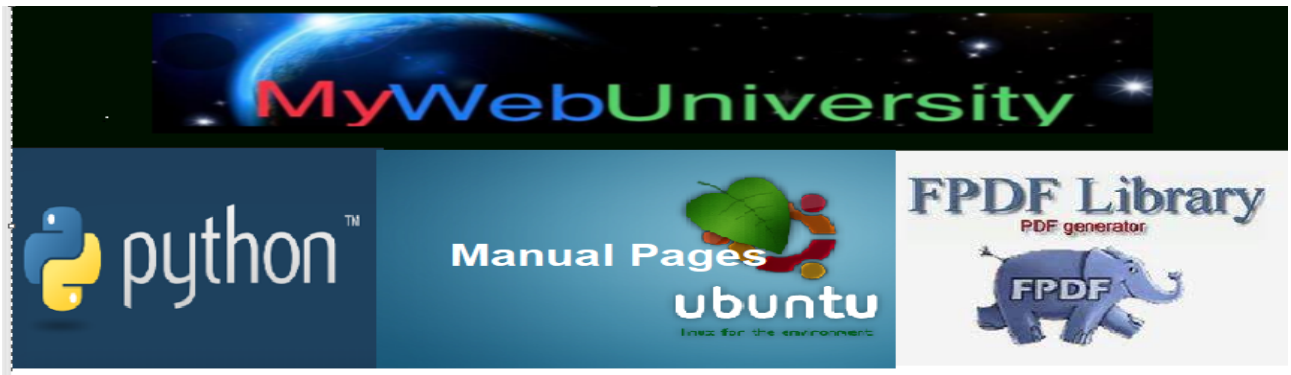

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'pg_dump.1'

$ man pg_dump.1

PG_DUMP(1) PostgreSQL 18.4 Documentation PG_DUMP(1)

NAME

pg_dump - export a PostgreSQL database as an SQL script or to other formats

SYNOPSIS

pg_dump [connection-option...] [option...] [dbname]

DESCRIPTION

pg_dump is a utility for exporting a PostgreSQL database. It makes consistent exports even if the database is being used concurrently. pg_dump does not block other users accessing the database (readers or writers). Note, however, that except in simple cases, pg_dump is generally not the right choice for taking regular backups of production databases. See Chapter 25 for further discussion.

pg_dump only dumps a single database. To export an entire cluster, or to export global objects that are common to all databases in a cluster (such as roles and tablespaces), use pg_dumpall(1).

Dumps can be output in script or archive file formats. Script dumps are plain-text files containing the SQL commands required to reconstruct the database to the state it was in at the time it was saved. To restore from such a script, feed it to psql(1). Script files can be used to reconstruct the database even on other machines and other architectures; with some modifications, even on other SQL database products.

The alternative archive file formats must be used with pg_restore(1) to rebuild the database. They allow pg_restore to be selective about what is restored, or even to reorder the items prior to being restored. The archive file formats are designed to be portable across architectures.

When used with one of the archive file formats and combined with pg_restore, pg_dump provides a flexible archival and transfer mechanism. pg_dump can be used to export an entire database, then pg_restore can be used to examine the archive and/or select which parts of the database are to be restored. The most flexible output file formats are the ?custom? format (-Fc) and the ?directory? format (-Fd). They allow for selection and reordering of all archived items, support parallel restoration, and are compressed by default. The ?directory? format is the only format that supports parallel dumps.

While running pg_dump, one should examine the output for any warnings (printed on standard error), especially in light of the limitations listed below.

Warning

Restoring a dump causes the destination to execute arbitrary code of the source superusers' choice. Partial dumps and partial restores do not limit that. If the source superusers are not trusted, the dumped SQL statements must be inspected before restoring. Non-plain-text dumps can be inspected by using pg_restore's --file option. Note that the client running the dump and restore need not trust the source or destination superusers.

OPTIONS

The following command-line options control the content and format of the output.

dbname

Specifies the name of the database to be dumped. If this is not specified, the environment variable PGDATABASE is used. If that is not set, the user name specified for the connection is used.

-a

--data-only

Dump only the data, not the schema (data definitions) or statistics. Table data, large objects, and sequence values are dumped.

This option is similar to, but for historical reasons not identical to, specifying --section=data.

-b

--large-objects

--blobs (deprecated)

Include large objects in the dump. This is the default behavior except when --schema, --table, --schema-only, --statistics-only, or --no-data is specified. The -b switch is

therefore only useful to add large objects to dumps where a specific schema or table has been requested. Note that large objects are considered data and therefore will be included when --data-only is used, but not when --schema-only or --statistics-only is.

-B

--no-large-objects

--no-blobs (deprecated)

Exclude large objects in the dump.

When both -b and -B are given, the behavior is to output large objects, when data is being dumped, see the -b documentation.

-c

--clean

Output commands to DROP all the dumped database objects prior to outputting the commands for creating them. This option is useful when the restore is to overwrite an existing database. If any of the objects do not exist in the destination database, ignorable error messages will be reported during restore, unless --if-exists is also specified.

This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

-C

--create

Begin the output with a command to create the database itself and reconnect to the created database. (With a script of this form, it doesn't matter which database in the destination installation you connect to before running the script.) If --clean is also specified, the script drops and recreates the target database before reconnecting to it.

With --create, the output also includes the database's comment if any, and any configuration variable settings that are specific to this database, that is, any ALTER DATABASE ... SET ... and ALTER ROLE ... IN DATABASE ... SET ... commands that mention this database. Access privileges for the database itself are also dumped, unless --no-acl is specified.

This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

-e pattern

--extension=pattern

Dump only extensions matching pattern. When this option is not specified, all non-system

extensions in the target database will be dumped. Multiple extensions can be selected by writing multiple -e switches. The pattern parameter is interpreted as a pattern according to the same rules used by psql's \d commands (see Patterns), so multiple extensions can also be selected by writing wildcard characters in the pattern. When using wildcards, be careful to quote the pattern if needed to prevent the shell from expanding the wildcards.

Any configuration relation registered by pg_extension_config_dump is included in the dump if its extension is specified by --extension.

Note

When -e is specified, pg_dump makes no attempt to dump any other database objects that the selected extension(s) might depend upon. Therefore, there is no guarantee that the results of a specific-extension dump can be successfully restored by themselves into a clean database.

-E encoding

--encoding=encoding

Create the dump in the specified character set encoding. By default, the dump is created in the database encoding. (Another way to get the same result is to set the PGCLIENTENCODING environment variable to the desired dump encoding.) The supported encodings are described in Section 23.3.1.

-f file

--file=file

Send output to the specified file. This parameter can be omitted for file based output formats, in which case the standard output is used. It must be given for the directory output format however, where it specifies the target directory instead of a file. In this case the directory is created by pg_dump and must not exist before.

-F format

--format=format

Selects the format of the output. format can be one of the following:

p

plain

Output a plain-text SQL script file (the default).

c

custom

Output a custom-format archive suitable for input into pg_restore. Together with the directory output format, this is the most flexible output format in that it allows manual selection and reordering of archived items during restore. This format is also compressed by default.

d

directory

Output a directory-format archive suitable for input into pg_restore. This will create a directory with one file for each table and large object being dumped, plus a so-called Table of Contents file describing the dumped objects in a machine-readable format that pg_restore can read. A directory format archive can be manipulated with standard Unix tools; for example, files in an uncompressed archive can be compressed with the gzip, lz4, or zstd tools. This format is compressed by default using gzip and also supports parallel dumps.

t

tar

Output a tar-format archive suitable for input into pg_restore. The tar format is compatible with the directory format: extracting a tar-format archive produces a valid directory-format archive. However, the tar format does not support compression. Also, when using tar format the relative order of table data items cannot be changed during restore.

-j njobs

--jobs=njobs

Run the dump in parallel by dumping njobs tables simultaneously. This option may reduce the time needed to perform the dump but it also increases the load on the database server. You can only use this option with the directory output format because this is the only output format where multiple processes can write their data at the same time.

pg_dump will open njobs + 1 connections to the database, so make sure your max_connections setting is high enough to accommodate all connections.

Requesting exclusive locks on database objects while running a parallel dump could cause the dump to fail. The reason is that the pg_dump leader process requests shared locks (ACCESS SHARE) on the objects that the worker processes are going to dump later in order to make sure that nobody deletes them and makes them go away while the dump is running. If another client then requests an exclusive lock on a table, that lock will not be

granted but will be queued waiting for the shared lock of the leader process to be released. Consequently any other access to the table will not be granted either and will queue after the exclusive lock request. This includes the worker process trying to dump the table. Without any precautions this would be a classic deadlock situation. To detect this conflict, the pg_dump worker process requests another shared lock using the NOWAIT option. If the worker process is not granted this shared lock, somebody else must have requested an exclusive lock in the meantime and there is no way to continue with the dump, so pg_dump has no choice but to abort the dump.

To perform a parallel dump, the database server needs to support synchronized snapshots, a feature that was introduced in PostgreSQL 9.2 for primary servers and 10 for standbys. With this feature, database clients can ensure they see the same data set even though they use different connections. pg_dump -j uses multiple database connections; it connects to the database once with the leader process and once again for each worker job. Without the synchronized snapshot feature, the different worker jobs wouldn't be guaranteed to see the same data in each connection, which could lead to an inconsistent backup.

-n pattern

--schema=pattern

Dump only schemas matching pattern; this selects both the schema itself, and all its contained objects. When this option is not specified, all non-system schemas in the target database will be dumped. Multiple schemas can be selected by writing multiple -n switches. The pattern parameter is interpreted as a pattern according to the same rules used by psql's \d commands (see Patterns), so multiple schemas can also be selected by writing wildcard characters in the pattern. When using wildcards, be careful to quote the pattern if needed to prevent the shell from expanding the wildcards; see Examples below.

Note

When -n is specified, pg_dump makes no attempt to dump any other database objects that the selected schema(s) might depend upon. Therefore, there is no guarantee that the results of a specific-schema dump can be successfully restored by themselves into a clean database.

Note

Non-schema objects such as large objects are not dumped when -n is specified. You

can add large objects back to the dump with the --large-objects switch.

-N pattern

--exclude-schema=pattern

Do not dump any schemas matching pattern. The pattern is interpreted according to the same rules as for -n. -N can be given more than once to exclude schemas matching any of several patterns.

When both -n and -N are given, the behavior is to dump just the schemas that match at least one -n switch but no -N switches. If -N appears without -n, then schemas matching -N are excluded from what is otherwise a normal dump.

-O

--no-owner

Do not output commands to set ownership of objects to match the original database. By default, pg_dump issues ALTER OWNER or SET SESSION AUTHORIZATION statements to set ownership of created database objects. These statements will fail when the script is run unless it is started by a superuser (or the same user that owns all of the objects in the script). To make a script that can be restored by any user, but will give that user ownership of all the objects, specify -O.

This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

-R

--no-reconnect

This option is obsolete but still accepted for backwards compatibility.

-s

--schema-only

Dump only the object definitions (schema), not data or statistics.

This option cannot be used with --data-only or --statistics-only. It is similar to, but for historical reasons not identical to, specifying --section=pre-data --section=post-data.

(Do not confuse this with the --schema option, which uses the word ?schema? in a different meaning.)

To exclude table data for only a subset of tables in the database, see --exclude-table-data.

-S username

--superuser=username

Specify the superuser user name to use when disabling triggers. This is relevant only if --disable-triggers is used. (Usually, it's better to leave this out, and instead start the resulting script as superuser.)

-t pattern

--table=pattern

Dump only tables with names matching pattern. Multiple tables can be selected by writing multiple -t switches. The pattern parameter is interpreted as a pattern according to the same rules used by psql's \d commands (see Patterns), so multiple tables can also be selected by writing wildcard characters in the pattern. When using wildcards, be careful to quote the pattern if needed to prevent the shell from expanding the wildcards; see Examples below.

As well as tables, this option can be used to dump the definition of matching views, materialized views, foreign tables, and sequences. It will not dump the contents of views or materialized views, and the contents of foreign tables will only be dumped if the corresponding foreign server is specified with --include-foreign-data.

The -n and -N switches have no effect when -t is used, because tables selected by -t will be dumped regardless of those switches, and non-table objects will not be dumped.

Note

When -t is specified, pg_dump makes no attempt to dump any other database objects that the selected table(s) might depend upon. Therefore, there is no guarantee that the results of a specific-table dump can be successfully restored by themselves into a clean database.

-T pattern

--exclude-table=pattern

Do not dump any tables matching pattern. The pattern is interpreted according to the same rules as for -t. -T can be given more than once to exclude tables matching any of several patterns.

When both -t and -T are given, the behavior is to dump just the tables that match at least one -t switch but no -T switches. If -T appears without -t, then tables matching -T are excluded from what is otherwise a normal dump.

-v

--verbose

Specifies verbose mode. This will cause pg_dump to output detailed object comments and start/stop times to the dump file, and progress messages to standard error. Repeating the option causes additional debug-level messages to appear on standard error.

-V

--version

Print the pg_dump version and exit.

-x

--no-privileges

--no-acl

Prevent dumping of access privileges (grant/revoke commands).

-Z level

-Z method[:detail]

--compress=level

--compress=method[:detail]

Specify the compression method and/or the compression level to use. The compression method can be set to gzip, lz4, zstd, or none for no compression. A compression detail string can optionally be specified. If the detail string is an integer, it specifies the compression level. Otherwise, it should be a comma-separated list of items, each of the form keyword or keyword=value. Currently, the supported keywords are level and long.

If no compression level is specified, the default compression level will be used. If only a level is specified without mentioning an algorithm, gzip compression will be used if the level is greater than 0, and no compression will be used if the level is 0.

For the custom and directory archive formats, this specifies compression of individual table-data segments, and the default is to compress using gzip at a moderate level. For plain text output, setting a nonzero compression level causes the entire output file to be compressed, as though it had been fed through gzip, lz4, or zstd; but the default is not to compress. With zstd compression, long mode may improve the compression ratio, at the cost of increased memory use.

The tar archive format currently does not support compression at all.

--binary-upgrade

This option is for use by in-place upgrade utilities. Its use for other purposes is not recommended or supported. The behavior of the option may change in future releases without notice.

--column-inserts

--attribute-inserts

Dump data as INSERT commands with explicit column names (INSERT INTO table (column, ...) VALUES ...). This will make restoration very slow; it is mainly useful for making dumps that can be loaded into non-PostgreSQL databases. Any error during restoring will cause only rows that are part of the problematic INSERT to be lost, rather than the entire table contents.

--disable-dollar-quoting

This option disables the use of dollar quoting for function bodies, and forces them to be quoted using SQL standard string syntax.

--disable-triggers

This option is relevant only when creating a dump that includes data but does not include schema. It instructs pg_dump to include commands to temporarily disable triggers on the target tables while the data is restored. Use this if you have referential integrity checks or other triggers on the tables that you do not want to invoke during data restore.

Presently, the commands emitted for --disable-triggers must be done as superuser. So, you should also specify a superuser name with -S, or preferably be careful to start the resulting script as a superuser.

This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

--enable-row-security

This option is relevant only when dumping the contents of a table which has row security. By default, pg_dump will set row_security to off, to ensure that all data is dumped from the table. If the user does not have sufficient privileges to bypass row security, then an error is thrown. This parameter instructs pg_dump to set row_security to on instead, allowing the user to dump the parts of the contents of the table that they have access to.

Note that if you use this option currently, you probably also want the dump be in INSERT format, as the COPY FROM during restore does not support row security.

--exclude-extension=pattern

Do not dump any extensions matching pattern. The pattern is interpreted according to the same rules as for -e. --exclude-extension can be given more than once to exclude

extensions matching any of several patterns.

When both -e and --exclude-extension are given, the behavior is to dump just the extensions that match at least one -e switch but no --exclude-extension switches. If --exclude-extension appears without -e, then extensions matching --exclude-extension are excluded from what is otherwise a normal dump.

--exclude-table-and-children=pattern

This is the same as the -T/--exclude-table option, except that it also excludes any partitions or inheritance child tables of the table(s) matching the pattern.

--exclude-table-data=pattern

Do not dump data for any tables matching pattern. The pattern is interpreted according to the same rules as for -t. --exclude-table-data can be given more than once to exclude tables matching any of several patterns. This option is useful when you need the definition of a particular table even though you do not need the data in it.

To exclude data for all tables in the database, see --schema-only or --statistics-only.

--exclude-table-data-and-children=pattern

This is the same as the --exclude-table-data option, except that it also excludes data of any partitions or inheritance child tables of the table(s) matching the pattern.

--extra-float-digits=ndigits

Use the specified value of extra_float_digits when dumping floating-point data, instead of the maximum available precision. Routine dumps made for backup purposes should not use this option.

--filter=filename

Specify a filename from which to read patterns for objects to include or exclude from the dump. The patterns are interpreted according to the same rules as the corresponding options: -t/--table, --table-and-children, -T/--exclude-table, and --exclude-table-and-children for tables, -n/--schema and -N/--exclude-schema for schemas, --include-foreign-data for data on foreign servers, --exclude-table-data and --exclude-table-data-and-children for table data, and -e/--extension and --exclude-extension for extensions. To read from STDIN, use - as the filename. The --filter option can be specified in conjunction with the above listed options for including or excluding objects, and can also be specified more than once for multiple filter files.

The file lists one object pattern per row, with the following format:

{ include | exclude } { extension | foreign_data | table | table_and_children | table_data | table_data_and_children | schema } PATTERN

The first keyword specifies whether the objects matched by the pattern are to be included or excluded. The second keyword specifies the type of object to be filtered using the pattern:

- extension: extensions. This works like the -e/--extension or --exclude-extension option.
- foreign_data: data on foreign servers. This works like the --include-foreign-data option. This keyword can only be used with the include keyword.
- table: tables. This works like the -t/--table or -T/--exclude-table option.
- table_and_children: tables including any partitions or inheritance child tables. This works like the --table-and-children or --exclude-table-and-children option.
- table_data: table data of any tables matching pattern. This works like the --exclude-table-data option. This keyword can only be used with the exclude keyword.
- table_data_and_children: table data of any tables matching pattern as well as any partitions or inheritance children of the table(s). This works like the --exclude-table-data-and-children option. This keyword can only be used with the exclude keyword.
- schema: schemas. This works like the -n/--schema or -N/--exclude-schema option.

Lines starting with # are considered comments and ignored. Comments can be placed after an object pattern row as well. Blank lines are also ignored. See Patterns for how to perform quoting in patterns.

Example files are listed below in the Examples section.

--if-exists

Use DROP ... IF EXISTS commands to drop objects in --clean mode. This suppresses ?does not exist? errors that might otherwise be reported. This option is not valid unless --clean is also specified.

--include-foreign-data=foreignserver

Dump the data for any foreign table with a foreign server matching foreignserver pattern. Multiple foreign servers can be selected by writing multiple --include-foreign-data switches. Also, the foreignserver parameter is interpreted as a pattern according to the same rules used by psql's \d commands (see Patterns), so multiple foreign servers can also be selected by writing wildcard characters in the

pattern. When using wildcards, be careful to quote the pattern if needed to prevent the shell from expanding the wildcards; see Examples below. The only exception is that an empty pattern is disallowed.

Note

Using wildcards in --include-foreign-data may result in access to unexpected foreign servers. Also, to use this option securely, make sure that the named server must have a trusted owner.

Note

When --include-foreign-data is specified, pg_dump does not check that the foreign table is writable. Therefore, there is no guarantee that the results of a foreign table dump can be successfully restored.

--inserts

Dump data as INSERT commands (rather than COPY). This will make restoration very slow; it is mainly useful for making dumps that can be loaded into non-PostgreSQL databases. Any error during restoring will cause only rows that are part of the problematic INSERT to be lost, rather than the entire table contents. Note that the restore might fail altogether if you have rearranged column order. The --column-inserts option is safe against column order changes, though even slower.

--load-via-partition-root

When dumping data for a table partition, make the COPY or INSERT statements target the root of the partitioning hierarchy that contains it, rather than the partition itself. This causes the appropriate partition to be re-determined for each row when the data is loaded. This may be useful when restoring data on a server where rows do not always fall into the same partitions as they did on the original server. That could happen, for example, if the partitioning column is of type text and the two systems have different definitions of the collation used to sort the partitioning column.

--lock-wait-timeout=timeout

Do not wait forever to acquire shared table locks at the beginning of the dump. Instead fail if unable to lock a table within the specified timeout. The timeout may be specified in any of the formats accepted by SET statement_timeout. (Allowed formats vary depending on the server version you are dumping from, but an integer number of milliseconds is accepted by all versions.)

--no-comments

Do not dump COMMENT commands.

--no-data

Do not dump data.

--no-policies

Do not dump row security policies.

--no-publications

Do not dump publications.

--no-schema

Do not dump schema (data definitions).

--no-security-labels

Do not dump security labels.

--no-statistics

Do not dump statistics. This is the default.

--no-subscriptions

Do not dump subscriptions.

--no-sync

By default, pg_dump will wait for all files to be written safely to disk. This option causes pg_dump to return without waiting, which is faster, but means that a subsequent operating system crash can leave the dump corrupt. Generally, this option is useful for testing but should not be used when dumping data from production installation.

--no-table-access-method

Do not output commands to select table access methods. With this option, all objects will be created with whichever table access method is the default during restore.
This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

--no-tablespaces

Do not output commands to select tablespaces. With this option, all objects will be created in whichever tablespace is the default during restore.
This option is ignored when emitting an archive (non-text) output file. For the archive formats, you can specify the option when you call pg_restore.

--no-toast-compression

Do not output commands to set TOAST compression methods. With this option, all columns will be restored with the default compression setting.

--no-unlogged-table-data

Do not dump the contents of unlogged tables and sequences. This option has no effect on whether or not the table and sequence definitions (schema) are dumped; it only suppresses dumping the table and sequence data. Data in unlogged tables and sequences is always excluded when dumping from a standby server.

--on-conflict-do-nothing

Add ON CONFLICT DO NOTHING to INSERT commands. This option is not valid unless --inserts, --column-inserts or --rows-per-insert is also specified.

--quote-all-identifiers

Force quoting of all identifiers. This option is recommended when dumping a database from a server whose PostgreSQL major version is different from pg_dump's, or when the output is intended to be loaded into a server of a different major version. By default, pg_dump quotes only identifiers that are reserved words in its own major version. This sometimes results in compatibility issues when dealing with servers of other versions that may have slightly different sets of reserved words. Using --quote-all-identifiers prevents such issues, at the price of a harder-to-read dump script.

--restrict-key=restrict_key

Use the provided string as the psql \restrict key in the dump output. This can only be specified for plain-text dumps, i.e., when --format is set to plain or the --format option is omitted. If no restrict key is specified, pg_dump will generate a random one as needed. Keys may contain only alphanumeric characters.

This option is primarily intended for testing purposes and other scenarios that require repeatable output (e.g., comparing dump files). It is not recommended for general use, as a malicious server with advance knowledge of the key may be able to inject arbitrary code that will be executed on the machine that runs psql with the dump output.

--rows-per-insert=nrows

Dump data as INSERT commands (rather than COPY). Controls the maximum number of rows per INSERT command. The value specified must be a number greater than zero. Any error during restoring will cause only rows that are part of the problematic INSERT to be lost, rather than the entire table contents.

--section=sectionname

Only dump the named section. The section name can be pre-data, data, or post-data. This option can be specified more than once to select multiple sections. The default is to

dump all sections.

The data section contains actual table data, large-object contents, sequence values, and statistics for tables, materialized views, and foreign tables. Post-data items include definitions of indexes, triggers, rules, statistics for indexes, and constraints other than validated check and not-null constraints. Pre-data items include all other data definition items.

--sequence-data

Include sequence data in the dump. This is the default behavior except when --no-data, --schema-only, or --statistics-only is specified.

--serializable-deferrable

Use a serializable transaction for the dump, to ensure that the snapshot used is consistent with later database states; but do this by waiting for a point in the transaction stream at which no anomalies can be present, so that there isn't a risk of the dump failing or causing other transactions to roll back with a serialization_failure. See Chapter 13 for more information about transaction isolation and concurrency control.

This option is not beneficial for a dump which is intended only for disaster recovery. It could be useful for a dump used to load a copy of the database for reporting or other read-only load sharing while the original database continues to be updated. Without it the dump may reflect a state which is not consistent with any serial execution of the transactions eventually committed. For example, if batch processing techniques are used, a batch may show as closed in the dump without all of the items which are in the batch appearing.

This option will make no difference if there are no read-write transactions active when pg_dump is started. If read-write transactions are active, the start of the dump may be delayed for an indeterminate length of time. Once running, performance with or without the switch is the same.

--snapshot=snapshotname

Use the specified synchronized snapshot when making a dump of the database (see Table 9.100 for more details).

This option is useful when needing to synchronize the dump with a logical replication slot (see Chapter 47) or with a concurrent session.

In the case of a parallel dump, the snapshot name defined by this option is used rather

than taking a new snapshot.

--statistics

Dump optimizer statistics.

--statistics-only

Dump only the statistics, not the schema (data definitions) or data. Optimizer statistics for tables, materialized views, foreign tables, and indexes are dumped.

--strict-names

Require that each extension (-e/--extension), schema (-n/--schema) and table (-t/--table) pattern match at least one extension/schema/table in the database to be dumped. This also applies to filters used with --filter. Note that if none of the extension/schema/table patterns find matches, pg_dump will generate an error even without --strict-names.

This option has no effect on --exclude-extension, -N/--exclude-schema, -T/--exclude-table, or --exclude-table-data. An exclude pattern failing to match any objects is not considered an error.

--sync-method=method

When set to fsync, which is the default, pg_dump --format=directory will recursively open and synchronize all files in the archive directory.

On Linux, syncfs may be used instead to ask the operating system to synchronize the whole file system that contains the archive directory. See recovery_init_sync_method for information about the caveats to be aware of when using syncfs.

This option has no effect when --no-sync is used or --format is not set to directory.

--table-and-children=pattern

This is the same as the -t/--table option, except that it also includes any partitions or inheritance child tables of the table(s) matching the pattern.

--use-set-session-authorization

Output SQL-standard SET SESSION AUTHORIZATION commands instead of ALTER OWNER commands to determine object ownership. This makes the dump more standards-compatible, but depending on the history of the objects in the dump, might not restore properly. Also, a dump using SET SESSION AUTHORIZATION will certainly require superuser privileges to restore correctly, whereas ALTER OWNER requires lesser privileges.

-?

--help

Show help about pg_dump command line arguments, and exit.

The following command-line options control the database connection parameters.

-d dbname

--dbname=dbname

Specifies the name of the database to connect to. This is equivalent to specifying dbname as the first non-option argument on the command line. The dbname can be a connection string. If so, connection string parameters will override any conflicting command line options.

-h host

--host=host

Specifies the host name of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the Unix domain socket. The default is taken from the PGHOST environment variable, if set, else a Unix domain socket connection is attempted.

-p port

--port=port

Specifies the TCP port or local Unix domain socket file extension on which the server is listening for connections. Defaults to the PGPORT environment variable, if set, or a compiled-in default.

-U username

--username=username

User name to connect as.

-w

--no-password

Never issue a password prompt. If the server requires password authentication and a password is not available by other means such as a .pgpass file, the connection attempt will fail. This option can be useful in batch jobs and scripts where no user is present to enter a password.

-W

--password

Force pg_dump to prompt for a password before connecting to a database.

This option is never essential, since pg_dump will automatically prompt for a password if the server demands password authentication. However, pg_dump will waste a connection

attempt finding out that the server wants a password. In some cases it is worth typing -W to avoid the extra connection attempt.

--role=rolename

Specifies a role name to be used to create the dump. This option causes pg_dump to issue a SET ROLE rolename command after connecting to the database. It is useful when the authenticated user (specified by -U) lacks privileges needed by pg_dump, but can switch to a role with the required rights. Some installations have a policy against logging in directly as a superuser, and use of this option allows dumps to be made without violating the policy.

ENVIRONMENT

PGDATABASE

PGHOST

PGOPTIONS

PGPORT

PGUSER

Default connection parameters.

PG_COLOR

Specifies whether to use color in diagnostic messages. Possible values are always, auto and never.

This utility, like most other PostgreSQL utilities, also uses the environment variables supported by libpq (see Section 32.15).

DIAGNOSTICS

pg_dump internally executes SELECT statements. If you have problems running pg_dump, make sure you are able to select information from the database using, for example, psql(1). Also, any default connection settings and environment variables used by the libpq front-end library will apply.

The database activity of pg_dump is normally collected by the cumulative statistics system. If this is undesirable, you can set parameter track_counts to false via PGOPTIONS or the ALTER USER command.

NOTES

If your database cluster has any local additions to the template1 database, be careful to restore the output of pg_dump into a truly empty database; otherwise you are likely to get errors due to duplicate definitions of the added objects. To make an empty database without

any local additions, copy from template0 not template1, for example:

```
CREATE DATABASE foo WITH TEMPLATE template0;
```

When a dump without schema is chosen and the option --disable-triggers is used, pg_dump emits commands to disable triggers on user tables before inserting the data, and then commands to re-enable them after the data has been inserted. If the restore is stopped in the middle, the system catalogs might be left in the wrong state.

When --statistics is specified, pg_dump will include most optimizer statistics in the resulting dump file. This does not include all statistics, such as those created explicitly with CREATE STATISTICS (CREATE_STATISTICS(7)), custom statistics added by an extension, or statistics collected by the cumulative statistics system. Therefore, it may still be useful to run ANALYZE after restoring from a dump file to ensure optimal performance; see Section 24.1.3 and Section 24.1.6 for more information.

Because pg_dump is used to transfer data to newer versions of PostgreSQL, the output of pg_dump can be expected to load into PostgreSQL server versions newer than pg_dump's version. pg_dump can also dump from PostgreSQL servers older than its own version. (Currently, servers back to version 9.2 are supported.) However, pg_dump cannot dump from PostgreSQL servers newer than its own major version; it will refuse to even try, rather than risk making an invalid dump. Also, it is not guaranteed that pg_dump's output can be loaded into a server of an older major version ? not even if the dump was taken from a server of that version. Loading a dump file into an older server may require manual editing of the dump file to remove syntax not understood by the older server. Use of the --quote-all-identifiers option is recommended in cross-version cases, as it can prevent problems arising from varying reserved-word lists in different PostgreSQL versions.

When dumping logical replication subscriptions, pg_dump will generate CREATE SUBSCRIPTION commands that use the connect = false option, so that restoring the subscription does not make remote connections for creating a replication slot or for initial table copy. That way, the dump can be restored without requiring network access to the remote servers. It is then up to the user to reactivate the subscriptions in a suitable way. If the involved hosts have changed, the connection information might have to be changed. It might also be appropriate to truncate the target tables before initiating a new full table copy. If users intend to copy initial data during refresh they must create the slot with two_phase = false. After the initial sync, the two_phase option will be automatically enabled by the subscriber if the subscription had been originally created with two_phase = true option.

It is generally recommended to use the -X (--no-psqlrc) option when restoring a database from a plain-text pg_dump script to ensure a clean restore process and prevent potential conflicts with non-default psql configurations.

EXAMPLES

To dump a database called mydb into an SQL-script file:

```
$ pg_dump mydb > db.sql
```

To reload such a script into a (freshly created) database named newdb:

```
$ psql -X -d newdb -f db.sql
```

To dump a database into a custom-format archive file:

```
$ pg_dump -Fc mydb > db.dump
```

To dump a database into a directory-format archive:

```
$ pg_dump -Fd mydb -f dumpdir
```

To dump a database into a directory-format archive in parallel with 5 worker jobs:

```
$ pg_dump -Fd mydb -j 5 -f dumpdir
```

To reload an archive file into a (freshly created) database named newdb:

```
$ pg_restore -d newdb db.dump
```

To reload an archive file into the same database it was dumped from, discarding the current contents of that database:

```
$ pg_restore -d postgres --clean --create db.dump
```

To dump a single table named mytab:

```
$ pg_dump -t mytab mydb > db.sql
```

To dump all tables whose names start with emp in the detroit schema, except for the table named employee_log:

```
$ pg_dump -t 'detroit.emp*' -T detroit.employee_log mydb > db.sql
```

To dump all schemas whose names start with east or west and end in gsm, excluding any schemas whose names contain the word test:

```
$ pg_dump -n 'east*gsm' -n 'west*gsm' -N '*test*' mydb > db.sql
```

The same, using regular expression notation to consolidate the switches:

```
$ pg_dump -n '(east|west)*gsm' -N '*test*' mydb > db.sql
```

To dump all database objects except for tables whose names begin with ts_:

```
$ pg_dump -T 'ts_*' mydb > db.sql
```

To specify an upper-case or mixed-case name in -t and related switches, you need to double-quote the name; else it will be folded to lower case (see Patterns). But double

quotes are special to the shell, so in turn they must be quoted. Thus, to dump a single table with a mixed-case name, you need something like

```
$ pg_dump -t "\"MixedCaseName\"" mydb > mytab.sql
```

To dump all tables whose names start with mytable, except for table mytable2, specify a filter file filter.txt like:

```
include table mytable*
exclude table mytable2
$ pg_dump --filter=filter.txt mydb > db.sql
```

SEE ALSO

pg_dumpall(1), pg_restore(1), psql(1)