NAME

pg_dump − extract a PostgreSQL database into a script file or other archive file

SYNOPSIS

pg_dump [*connection−option...*] [*option...*] [*dbname*]

DESCRIPTION

pg_dump is a utility for backing up a PostgreSQL database. It makes consistent backups even if the database is being used concurrently. pg_dump does not block other users accessing the database (readers or writers).

pg_dump only dumps a single database. To backup 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 backup 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.

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). 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
−−blobs

Include large objects in the dump. This is the default behavior except when **−−schema**, **−−table**, or **−−schema−only** 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 blobs are considered data and therefore will be included when −−data−only is used, but not when −−schema−only is.

−B
−−no−blobs

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 clean (drop) database objects prior to outputting the commands for creating them. (Unless **−−if−exists** is also specified, restore might generate some harmless error messages, if any objects were not present in the destination database.)

This option is only meaningful for the plain−text format. 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.

This option is only meaningful for the plain−text format. For the archive formats, you can specify the option when you call **pg_restore**.

−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.)

−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 blob 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 tool. This format is compressed by default 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 reduces the time of 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 master process requests shared locks 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 master 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.

For a consistent backup, 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 master 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.

If you want to run a parallel dump of a pre−9.2 server, you need to make sure that the database content doesn't change from between the time the master connects to the database until the last worker job has connected to the database. The easiest way to do this is to halt any data modifying processes (DDL and DML) accessing the database before starting the backup. You also need to specify the **−−no−synchronized−snapshots** parameter when running **pg_dump −j** against a pre−9.2 PostgreSQL server.

−n *schema*
−−schema=*schema*

Dump only schemas matching *schema*; 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. Also, the *schema* 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.

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 blobs are not dumped when **−n** is specified. You can add blobs back to the dump with the **−−blobs** switch.

−N *schema*
−−exclude−schema=*schema*
Do not dump any schemas matching the *schema* 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
−−oids
Dump object identifiers (OIDs) as part of the data for every table. Use this option if your application references the OID columns in some way (e.g., in a foreign key constraint). Otherwise, this option should not be used.

−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 only meaningful for the plain−text format. 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.

This option is the inverse of **−−data−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 *table*
−−table=*table*
Dump only tables with names matching *table*. For this purpose, “table” includes views, materialized views, sequences, and foreign tables. Multiple tables can be selected by writing multiple **−t** switches. Also, the *table* 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.

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.

> **Note**
>
> The behavior of the **−t** switch is not entirely upward compatible with pre−8.2 PostgreSQL versions. Formerly, writing −t tab would dump all tables named tab, but now it just dumps whichever one is visible in your default search path. To get the old behavior you can write −t '*.tab'. Also, you must write something like −t sch.tab to select a table in a particular schema, rather than the old locution of −n sch −t tab.

−T *table*
−−exclude−table=*table*

Do not dump any tables matching the *table* 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.

−V
−−version

Print the pg_dump version and exit.

−x
−−no−privileges
−−no−acl

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

−Z *0..9*
−−compress=*0..9*

Specify the compression level to use. Zero means no compression. For the custom archive format, this specifies compression of individual table−data segments, and the default is to compress 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; but the default is not to compress. 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. However, since this option generates a separate command for each row, an error in reloading a row causes only that row 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 data−only dump. It instructs pg_dump to include commands to temporarily disable triggers on the target tables while the data is reloaded. Use this if you have referential integrity checks or other triggers on the tables that you do not want to invoke during data reload.

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 only meaningful for the plain−text format. 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−table−data=*table*
Do not dump data for any tables matching the *table* 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**.

−−if−exists
Use conditional commands (i.e. add an IF EXISTS clause) when cleaning database objects. This option is not valid unless **−−clean** is also specified.

−−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. However, since this option generates a separate command for each row, an error in reloading a row causes only that row 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.

−−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−publications
Do not dump publications.

−−no−security−labels
Do not dump security labels.

−−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−synchronized−snapshots
This option allows running **pg_dump −j** against a pre−9.2 server, see the documentation of the **−j** parameter for more details.

−−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 only meaningful for the plain−text format. For the archive formats, you can specify the option when you call **pg_restore**.

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

−−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.

−−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, and sequence values. Post−data items include definitions of indexes, triggers, rules, and constraints other than validated check constraints. Pre−data items include all other data definition items.

−−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 9.82 for more details).

This option is useful when needing to synchronize the dump with a logical replication slot (see

Chapter 48) 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.

−−strict−names
Require that each schema (**−n/−−schema**) and table (**−t/−−table**) qualifier match at least one schema/table in the database to be dumped. Note that if none of the schema/table qualifiers find matches, pg_dump will generate an error even without **−−strict−names**.

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

−−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.

If this parameter contains an = sign or starts with a valid URI prefix (postgresql:// or postgres://), it is treated as a *conninfo* string. See Section 33.1 for more information.

−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.

This utility, like most other PostgreSQL utilities, also uses the environment variables supported by libpq (see Section 33.14).

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 statistics collector. 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 data−only dump 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.

The dump file produced by pg_dump does not contain the statistics used by the optimizer to make query planning decisions. Therefore, it is wise 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. The dump file also does not contain any **ALTER DATABASE ... SET** commands; these settings are dumped by **pg_dumpall**(1), along with database users and other installation−wide settings.

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 8.0 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.

EXAMPLES

To dump a database called mydb into a SQL−script file:

$ **pg_dump mydb > db.sql**

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

$ **psql −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 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**

SEE ALSO

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