NAME

pg_restore − restore a PostgreSQL database from an archive file created by pg_dump

SYNOPSIS

pg_restore [*connection−option...*] [*option...*] [*filename*]

DESCRIPTION

pg_restore is a utility for restoring a PostgreSQL database from an archive created by **pg_dump**(1) in one of the non−plain−text formats. It will issue the commands necessary to reconstruct the database to the state it was in at the time it was saved. The archive files also allow pg_restore to be selective about what is restored, or even to reorder the items prior to being restored. The archive files are designed to be portable across architectures.

pg_restore can operate in two modes. If a database name is specified, pg_restore connects to that database and restores archive contents directly into the database. Otherwise, a script containing the SQL commands necessary to rebuild the database is created and written to a file or standard output. This script output is equivalent to the plain text output format of pg_dump. Some of the options controlling the output are therefore analogous to pg_dump options.

Obviously, pg_restore cannot restore information that is not present in the archive file. For instance, if the archive was made using the "dump data as **INSERT** commands" option, pg_restore will not be able to load the data using **COPY** statements.

OPTIONS

pg_restore accepts the following command line arguments.

filename
: Specifies the location of the archive file (or directory, for a directory−format archive) to be restored. If not specified, the standard input is used.

−a
−−data−only
: Restore only the data, not the schema (data definitions). Table data, large objects, and sequence values are restored, if present in the archive.

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

−c
−−clean
: Clean (drop) database objects before recreating them. (Unless **−−if−exists** is used, this might generate some harmless error messages, if any objects were not present in the destination database.)

−C
−−create
: Create the database before restoring into it. If **−−clean** is also specified, drop and recreate the target database before connecting to it.

  When this option is used, the database named with **−d** is used only to issue the initial **DROP DATABASE** and **CREATE DATABASE** commands. All data is restored into the database name that appears in the archive.

−d *dbname*
−−dbname=*dbname*
: Connect to database *dbname* and restore directly into the database.

−e
−−exit−on−error
: Exit if an error is encountered while sending SQL commands to the database. The default is to continue and to display a count of errors at the end of the restoration.

−f *filename*
−−file=*filename*

Specify output file for generated script, or for the listing when used with **–l**. Use – for the standard output, which is also the default.

–F *format*
––format=*format*
Specify format of the archive. It is not necessary to specify the format, since pg_restore will determine the format automatically. If specified, it can be one of the following:

c
custom
The archive is in the custom format of pg_dump.

d
directory
The archive is a directory archive.

t
tar
The archive is a **tar** archive.

–I *index*
––index=*index*
Restore definition of named index only. Multiple indexes may be specified with multiple **–I** switches.

–j *number–of–jobs*
––jobs=*number–of–jobs*
Run the most time–consuming parts of pg_restore — those which load data, create indexes, or create constraints — using multiple concurrent jobs. This option can dramatically reduce the time to restore a large database to a server running on a multiprocessor machine.

Each job is one process or one thread, depending on the operating system, and uses a separate connection to the server.

The optimal value for this option depends on the hardware setup of the server, of the client, and of the network. Factors include the number of CPU cores and the disk setup. A good place to start is the number of CPU cores on the server, but values larger than that can also lead to faster restore times in many cases. Of course, values that are too high will lead to decreased performance because of thrashing.

Only the custom and directory archive formats are supported with this option. The input must be a regular file or directory (not, for example, a pipe). This option is ignored when emitting a script rather than connecting directly to a database server. Also, multiple jobs cannot be used together with the option **––single–transaction**.

–l
––list
List the table of contents of the archive. The output of this operation can be used as input to the **–L** option. Note that if filtering switches such as **–n** or **–t** are used with **–l**, they will restrict the items listed.

–L *list–file*
––use–list=*list–file*
Restore only those archive elements that are listed in *list–file*, and restore them in the order they appear in the file. Note that if filtering switches such as **–n** or **–t** are used with **–L**, they will further restrict the items restored.

list–file is normally created by editing the output of a previous **–l** operation. Lines can be moved or removed, and can also be commented out by placing a semicolon (;) at the start of the line. See below for examples.

−n *schema*
−−schema=*schema*
Restore only objects that are in the named schema. Multiple schemas may be specified with multiple **−n** switches. This can be combined with the **−t** option to restore just a specific table.

−N *schema*
−−exclude−schema=*schema*
Do not restore objects that are in the named schema. Multiple schemas to be excluded may be specified with multiple **−N** switches.

When both **−n** and **−N** are given for the same schema name, the **−N** switch wins and the schema is excluded.

−O
−−no−owner
Do not output commands to set ownership of objects to match the original database. By default, pg_restore issues **ALTER OWNER** or **SET SESSION AUTHORIZATION** statements to set ownership of created schema elements. These statements will fail unless the initial connection to the database is made by a superuser (or the same user that owns all of the objects in the script). With **−O**, any user name can be used for the initial connection, and this user will own all the created objects.

−P *function−name(argtype [, ...])*
−−function=*function−name(argtype [, ...])*
Restore the named function only. Be careful to spell the function name and arguments exactly as they appear in the dump file's table of contents. Multiple functions may be specified with multiple **−P** switches.

−R
−−no−reconnect
This option is obsolete but still accepted for backwards compatibility.

−s
−−schema−only
Restore only the schema (data definitions), not data, to the extent that schema entries are present in the archive.

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.)

−S *username*
−−superuser=*username*
Specify the superuser user name to use when disabling triggers. This is relevant only if **−−disable−triggers** is used.

−t *table*
−−table=*table*
Restore definition and/or data of only the named table. For this purpose, “table” includes views, materialized views, sequences, and foreign tables. Multiple tables can be selected by writing multiple **−t** switches. This option can be combined with the **−n** option to specify table(s) in a particular schema.

Note
When **−t** is specified, pg_restore makes no attempt to restore any other database objects that the selected table(s) might depend upon. Therefore, there is no guarantee that a specific−table restore into a clean database will succeed.

Note
This flag does not behave identically to the **−t** flag of pg_dump. There is not currently any provision for

wild−card matching in pg_restore, nor can you include a schema name within its **−t**.

Note
In versions prior to PostgreSQL 9.6, this flag matched only tables, not any other type of relation.

−T *trigger*
−−trigger=*trigger*
Restore named trigger only. Multiple triggers may be specified with multiple **−T** switches.

−v
−−verbose
Specifies verbose mode.

−V
−−version
Print the pg_restore version and exit.

−x
−−no−privileges
−−no−acl
Prevent restoration of access privileges (grant/revoke commands).

−1
−−single−transaction
Execute the restore as a single transaction (that is, wrap the emitted commands in **BEGIN/COMMIT**). This ensures that either all the commands complete successfully, or no changes are applied. This option implies **−−exit−on−error**.

−−disable−triggers
This option is relevant only when performing a data−only restore. It instructs pg_restore to execute 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, run pg_restore as a PostgreSQL superuser.

−−enable−row−security
This option is relevant only when restoring the contents of a table which has row security. By default, pg_restore will set row_security to off, to ensure that all data is restored in to the table. If the user does not have sufficient privileges to bypass row security, then an error is thrown. This parameter instructs pg_restore to set row_security to on instead, allowing the user to attempt to restore the contents of the table with row security enabled. This might still fail if the user does not have the right to insert the rows from the dump into the table.

Note that this option currently also requires the dump be in **INSERT** format, as **COPY FROM** does not support row security.

−−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.

−−no−data−for−failed−tables
By default, table data is restored even if the creation command for the table failed (e.g., because it already exists). With this option, data for such a table is skipped. This behavior is useful if the target database already contains the desired table contents. For example, auxiliary tables for PostgreSQL extensions such as PostGIS might already be loaded in the target database; specifying this option prevents duplicate or obsolete data from being loaded into them.

This option is effective only when restoring directly into a database, not when producing SQL script

output.

−−no−publications
Do not output commands to restore publications, even if the archive contains them.

−−no−security−labels
Do not output commands to restore security labels, even if the archive contains them.

−−no−subscriptions
Do not output commands to restore subscriptions, even if the archive contains them.

−−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.

−−section=*sectionname*
Only restore 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 restore all sections.

The data section contains actual table data as well as large−object definitions. Post−data items consist of definitions of indexes, triggers, rules and constraints other than validated check constraints. Pre−data items consist of all other data definition items.

−−strict−names
Require that each schema (**−n/−−schema**) and table (**−t/−−table**) qualifier match at least one schema/table in the backup file.

−−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.

−?
−−help
Show help about pg_restore command line arguments, and exit.

pg_restore also accepts the following command line arguments for connection parameters:

−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_restore to prompt for a password before connecting to a database.

This option is never essential, since pg_restore will automatically prompt for a password if the server demands password authentication. However, pg_restore 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 perform the restore. This option causes pg_restore 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_restore, 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 restores to be performed without violating the policy.

ENVIRONMENT

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). However, it does not read **PGDATABASE** when a database name is not supplied.

DIAGNOSTICS

When a direct database connection is specified using the **−d** option, pg_restore internally executes SQL statements. If you have problems running pg_restore, 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.

NOTES

If your installation has any local additions to the template1 database, be careful to load the output of pg_restore 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;
```

The limitations of pg_restore are detailed below.

- When restoring data to a pre−existing table and the option **−−disable−triggers** is used, pg_restore emits commands to disable triggers on user tables before inserting the data, then emits 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.
- pg_restore cannot restore large objects selectively; for instance, only those for a specific table. If an archive contains large objects, then all large objects will be restored, or none of them if they are excluded via **−L**, **−t**, or other options.

See also the **pg_dump**(1) documentation for details on limitations of pg_dump.

Once restored, it is wise to run **ANALYZE** on each restored table so the optimizer has useful statistics; see Section 24.1.3 and Section 24.1.6 for more information.

EXAMPLES

Assume we have dumped a database called mydb into a custom−format dump file:

```
$ pg_dump −Fc mydb > db.dump
```

To drop the database and recreate it from the dump:

```
$ dropdb mydb
$ pg_restore −C −d postgres db.dump
```

The database named in the **–d** switch can be any database existing in the cluster; pg_restore only uses it to issue the **CREATE DATABASE** command for mydb. With **–C**, data is always restored into the database name that appears in the dump file.

To reload the dump into a new database called newdb:

$ **createdb –T template0 newdb**
$ **pg_restore –d newdb db.dump**

Notice we don't use **–C**, and instead connect directly to the database to be restored into. Also note that we clone the new database from template0 not template1, to ensure it is initially empty.

To reorder database items, it is first necessary to dump the table of contents of the archive:

$ **pg_restore –l db.dump > db.list**

The listing file consists of a header and one line for each item, e.g.:

```
;
; Archive created at Mon Sep 14 13:55:39 2009
;     dbname: DBDEMOS
;     TOC Entries: 81
;     Compression: 9
;     Dump Version: 1.10–0
;     Format: CUSTOM
;     Integer: 4 bytes
;     Offset: 8 bytes
;     Dumped from database version: 8.3.5
;     Dumped by pg_dump version: 8.3.8
;
;
; Selected TOC Entries:
;
3; 2615 2200 SCHEMA – public pasha
1861; 0 0 COMMENT – SCHEMA public pasha
1862; 0 0 ACL – public pasha
317; 1247 17715 TYPE public composite pasha
319; 1247 25899 DOMAIN public domain0 pasha
```

Semicolons start a comment, and the numbers at the start of lines refer to the internal archive ID assigned to each item.

Lines in the file can be commented out, deleted, and reordered. For example:

```
10; 145433 TABLE map_resolutions postgres
;2; 145344 TABLE species postgres
;4; 145359 TABLE nt_header postgres
6; 145402 TABLE species_records postgres
;8; 145416 TABLE ss_old postgres
```

could be used as input to pg_restore and would only restore items 10 and 6, in that order:

$ **pg_restore –L db.list db.dump**

SEE ALSO

pg_dump(1), **pg_dumpall**(1), **psql**(1)