PostgreSQL

en.wikibooks.org

July 9, 2016

Contents

1 Fundamentals

PostgreSQL supports 32- and 64-bit hardware. It is available on the following CPU architectures: x86, x86_64, IA64, PowerPC, PowerPC 64, S/390, S/390x, Sparc, Sparc 64, ARM, MIPS, MIPSEL, M68K, PA-RISC [1] and operating systems [2]

- BSD
  - FreeBSD
  - OpenBSD
- Linux
  - Red Hat family Linux (including CentOS/Fedora/Scientific/Oracle variants)
  - Debian GNU/Linux and derivatives
  - Ubuntu Linux and derivatives
  - SuSE and OpenSuSE
- Mac OS X
- Other Unix: AIX, HP/UX, Unixware
- Solaris (Sparc and i386)
- Windows (Win 2000 SP4 +)

1.1 References

Before you download PostgreSQL you must make some crucial decisions. First, there are two principle ways to get the system running: either you pick up the complete source code or the prebuild binaries. If you take the source code you must compile them with a C compiler (at least C89-compliant, in most cases people use GCC) [3] to the binary format of your computer. If you get the binaries directly the compilation step is superfluous. Next, you must know for which operating system you need the software. PostgreSQL supports a lot of UNIX-based systems (including MAC OS X) as well as Windows - both in 32- and 64-bit versions.

1 CPU Architectures `http://www.postgresql.org/docs/9.5/static/supported-platforms.html`
2 Operating Systems `http://www.postgresql.org/download`
3 Requirements for Compilation `http://www.postgresql.org/docs/current/static/install-requirements.html`

1.2 Download

After you have made the above decisions you can download the source code and/or the binaries from this page[4] and its subpages. For some operating systems you will find a graphical installer which leads you through the subsequent installation steps. For the same or other operating systems not only the PostgreSQL DBMS will be downloaded but also the DBA tool `pgAdmin`, which helps you doing your daily work thru its graphical interface.

There are different versions available: the actual release, old releases and the upcomming release.

1.3 Installation

Installation steps vary depending on the choosen operating system. In the simplest case the above mentioned graphical installer hides the details. The PostgreSQL wiki[5] and documentation[6] leads you thru all necessary steps for your operating system.

If you install from source code, details are explained for Unix systems[7] and Windows[8].

After a successful installation you will have

- The PostgreSQL binaries on your disc.
- A first database cluster called 'main' on your disc. The database cluster consists of an empty database called 'postgres' (plus two template databases) and an user/role called 'postgres' as well.
- A set of programs (Unix) repectively a service (Windows) running on your computer. These programs/service handle the database cluster as an entire.

1.4 First steps

You can create your tables, views, functions etc. in this database 'postgres' and work with the user 'postgres'. But this approach is not recommended. The user 'postgres' has very high privileges by default and the database 'postgres' is sometimes used by tools and third party programs as a container for temporary data. You are encouraged to define your own database, one user who acts as the owner of the database and some application users.

As a first step start psql with user 'postgres' and create your own users. Please notice, that 'users' respective 'roles' are global objects which are known by all databases within the cluster, not only within a certain database. But users/roles have specific rights within each database.

```
$ psql
```

4 http://www.postgresql.org/download/
5 https://wiki.postgresql.org/wiki/Detailed_installation_guides
6 http://www.postgresql.org/download/
7 http://www.postgresql.org/docs/current/static/installation.html
8 http://www.postgresql.org/docs/current/static/install-windows.html

```
postgres=#
postgres=# -- the future owner of the new database shall be 'finance_master'
 with DDL and DML rights
postgres=# CREATE ROLE finance_master;
CREATE ROLE
postgres=# ALTER  ROLE finance_master WITH NOSUPERUSER INHERIT CREATEROLE
 CREATEDB LOGIN NOREPLICATION ENCRYPTED PASSWORD 'xxx';
ALTER ROLE
postgres=# -- one user for read/write and one for read-only access (no DDL
 rights)
postgres=# CREATE ROLE rw_user;
CREATE ROLE
postgres=# ALTER  ROLE rw_user WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB
 LOGIN NOREPLICATION ENCRYPTED PASSWORD 'xxx';
ALTER ROLE
postgres=# CREATE ROLE ro_user;
CREATE ROLE
postgres=# ALTER  ROLE ro_user WITH NOSUPERUSER INHERIT NOCREATEROLE NOCREATEDB
 LOGIN NOREPLICATION ENCRYPTED PASSWORD 'xxx';
ALTER ROLE
postgres=#
```

Next, create a new database 'finance_db'. You can do this as user 'postgres' or as the previously created 'finance_master'.

```
postgres=#
postgres=# CREATE DATABASE finance_db
postgres=#    WITH OWNER = finance_master
postgres=#    ENCODING   = 'UTF8'
postgres=#    LC_COLLATE = 'en_US.UTF-8'
postgres=#    LC_CTYPE   = 'en_US.UTF-8';
CREATE DATABASE
postgres=#
```

As the last step you have to delegate the intended rights to the users/roles. This is a little tricky because PostgreSQL uses an elaborated role system where every role inherits rights from the implicit 'public' role.

```
postgres=#
postgres=# \connect finance_db
finance_db=# -- revoke schema creation from role 'public' because all roles
 inherit her rights from 'public'
finance_db=# REVOKE CREATE ON DATABASE finance_db FROM public;
REVOKE
finance_db=# -- same: revoke table creation
finance_db=# REVOKE CREATE ON SCHEMA public FROM public;
REVOKE
finance_db=# -- grant only DML rights to 'rw_user', no DDL rights like 'CREATE
 TABLE'
finance_db=# GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES    IN SCHEMA
 public TO rw_user;
GRANT
finance_db=# GRANT USAGE                          ON ALL SEQUENCES IN SCHEMA
 public TO rw_user;
GRANT
finance_db=# -- grant read rights to the read-only user
finance_db=# GRANT SELECT                         ON ALL TABLES    IN SCHEMA
 public TO ro_user;
GRANT
postgres=#
```

1.5 References

To promote a consistent use and understandig of important terms we list and define them herinafter. In some cases there are some short annotations to give a first introduction to the subject.

1.6 Database Cluster

1.6.1 Overview

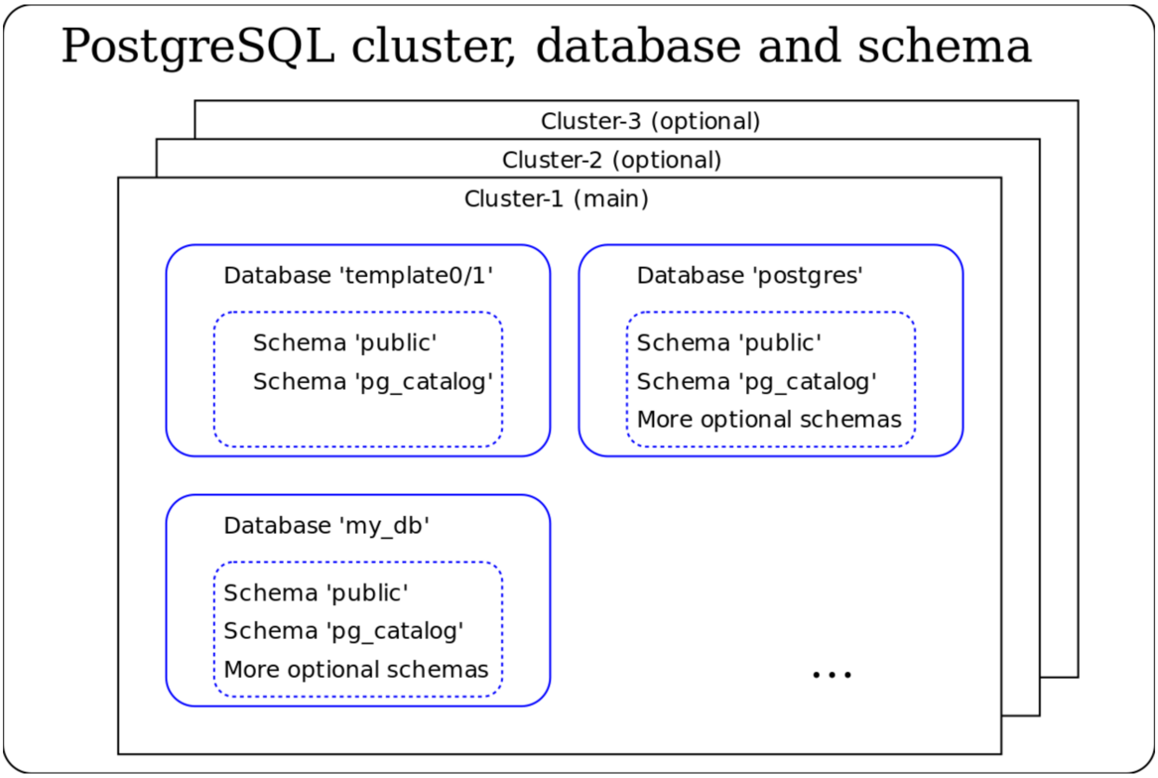

Figure 1

1.6.2 Server (or Node)

A server is some (real or virtual) hardware where PostgreSQL is installed. Please don't exchange it with the term *instance*.

1.6.3 Cluster of Nodes

A set of nodes, which interchange information via replication.

1.6.4 Installation

After you have downloaded and installed PostgreSQL, you have a set of programs, scripts, configuration- and other files on a *server*. This set is called the 'Installation'. It includes all *instance* programs as well as some client programs like `psql`.

1.6.5 Server Database

The term *server database* is often used in the context of client/server connections to refer to an *instance* or a single *database*.

1.6.6 Cluster (or 'Database Cluster')

A cluster is a storage area (directory, subdirectories and files) in the file system, where a collection of databases plus meta-information resides. Within the database cluster there are also the definitions of global objects like users and their rights. They are known across the entire database cluster. (Access rights for an user may be limited to individual objects like a certain table or a certain schema, thus it is defined that the user did not have this access rights to the other objects of the cluster.) Within a database cluster there are at least three databases: 'template0', 'template1', 'postgres' and possibly more.

- 'template0': A template database, which may be used by the command `CREATE DATABASE` (template0 should never be modified)
- 'template1': A template database, which may be used by the command `CREATE DATABASE` (template1 may be modified by DBA)
- 'postgres': An empty database, mainly for maintenance purposes

Most PostgreSQL installations use only one database cluster. Its name is 'main'. But you can create more clusters on the same PostgreSQL installation, see tools `initdb` further down.

1.6.7 Instance (or 'Database Server Instance' or 'Database Server' or 'Backend')

An instance is a group of processes (on a UNIX server) respectively one service (on a Windows server) plus shared memory, which controls and manages exactly one *cluster*. Using IP terminology one can say that one instance occupies one IP/port combination, eg. the combination `http://localhost:5432`. It is possible that on a different port of the same *server* another instance is running. The processes (in a UNIX server), which build an instance, are called: postmaster (creates one 'postgres'-process per client-connection), logger, checkpointer, background writer, WAL writer, autovacuum launcher, archiver, stats collector. The role of each process is explained in the chapter architecture[9].

9 Chapter 1.9.8 on page 15

If you have many *clusters* on your *server*, you can run many instances at the same machine - one per *cluster*.

Hint: Other publications sometimes use the term *server* to refer to an instance. As the term *server* is widely used to refere to real or virtual hardware, we do not use *server* as a synonym for *instance*.

1.6.8 Database

A database is a storage area in the file system, where a collection of objects is stored in files. The objects consist of data, metadata (table definitions, data types, constraints, views, ...) and other data like indices. Those objects are stored in the default database 'postgres' or in a newly created database.

The storage area for one database is organized as one subdirectory tree within the storage area of the database cluster. Thus a database cluster may contain multiple databases.

In a newly created database cluster (see below: initdb) there is an empty database with the name 'postgres'. In most cases this database keeps empty and application data is stored in separate databases like 'finance' or 'engineering'. Nevertheless 'postgres' shall not be droped because some tools try to store temporary data within this database.

1.6.9 Schema

A schema is a namespace within a database: it contains named objects (tables, data types, functions, and operators) whose names can duplicate those of other objects existing in other schemas of this database. Every database contains the default schema 'public' and may contain more schemas. All objects of one schema must reside within the same database. Objects of different schemas within the same database may have the same name.

There is another special schema in each database. The schema 'pg_catalog' contains all system tables, built-in data types, functions, and operators. See also 'Search Path' below.

1.6.10 Search Path (or 'Schema Search Path')

A Search Path is a list of schema names. If applications use unqualified object names (eg.: 'employee_table' for a table name), the search path is used to locate this object in the given sequence of schemas. The schema 'pg_catalog' is always the first part of the search path although it is not explicitely listed in the search path. This behaviour ensures that PostgreSQL finds the system objects.

1.6.11 initdb (OS command)

Despite of its name the utility `initdb` creates a new *cluster*, which contains the 3 *databases* 'template0', 'template1' and 'postgres'.

1.6.12 createdb (OS command)

The utility `createdb` creates a new *database* within the actual *cluster*.

1.6.13 CREATE DATABASE (SQL command)

The SQL command `CREATE DATABASE` creates a new *database* within the actual *cluster*.

1.6.14 Directory Structure

A *cluster* and its *databases* consists of files, which hold data, actual status information, modification information and a lot more. Those files are organized in a fixed way under one directory node.

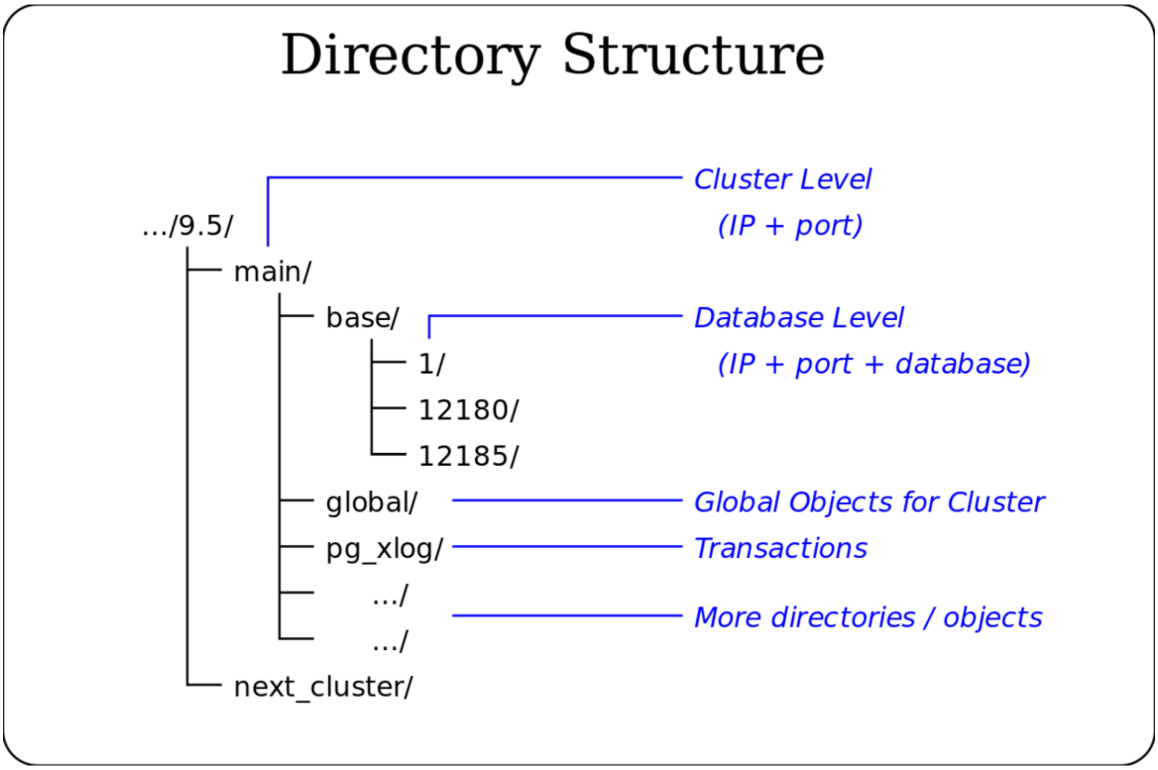

Figure 2

1.7 Consistent Writes

1.7.1 Shared Buffers

Shared bufferes are RAM pages, which mirror pages of data files on disc. They exist due to performance reasons. The term *shared* results from the fact that a lot of processes read and write to that area.

1.7.2 'Dirty' Page

Pages in the *shared bufferes* mirror pages of data files on disc. When clients request changes of data, those pages get changed without - provisionally - a change of the according pages on disc. Until the *background writer* writes those modified pages to disc, they are called 'dirty' pages.

1.7.3 Checkpoint

A checkpoint is a special point in time where it is guarantied that the database files are in a consistent state. At checkpoint time all change records are flushed to the WAL file, all dirty data pages (in shared buffers) are flushed to disc, and at last a special checkpoint record is written to the WAL file.

The instance's checkpointer process automatically triggers checkpoints on a regular basis. Additionally they can be forced by issuing the command `CHECKPOINT` in a client program. For the database system it takes a lot of time to perform a checkpoint - because of the physical writes to disc.

1.7.4 WAL File

WAL files contain the changes which are applied to the data by modifiing commands like `INSERT`, `UPDATE`, `DELETE` or `CREATE TABLE ...`. This is redundant information as it is also recorded in the data files (for better performance at a later time). According to the configuration of the instance there may be more information within WAL files. WAL files reside in the `pg_xlog` directory, has a binary format, and have a fix size of 16MB. When they are no longer needed, they get recycled by renaming and reuseing their already allocated space.

A single information unit within a WAL file is called a *log record*.

Hint: In the PostgreSQL documentation and in related documents there are a lot of other, similar terms which refer to what we denote *WAL file* in our Wikibook: segment, WAL segment, logfile (don't mix it with the term *logfile*, see below), WAL log file, ...

1.7.5 Logfile

The *instance* logs and reports warning and error messages about special situations in readable text files. This logfiles reside at any place in the directory structure of the *server* and are not part of the *cluster*.

Hint: The term 'logfile' does not relate to the other terms of this subchapter. He is mentioned here because the term sometimes is used as a synonym for what we call *WAL file* - see above.

1.7.6 Log Record

A log record is a single information unit within a *WAL file*.

1.7.7 Segment

The term *segment* is sometimes used as a synonym for *WAL file*.

1.7.8 MVCC

Multiversion Concurrency Control[10] (MVCC) is a common database technique to accomplish two goals: First, it allows the management of parallel running transactions on a logical level and second, it ensures high performance for concurrent read and write actions. It is implemented as follows: Whenever values of an existing row changes, PostgreSQL writes a new version of this row to the database without deleting to old one. In such situations the database contains multiple versions of the row. In addition to their regular data the rows contain transaction IDs which allows to decide, which other transactions will see the new or the old row. Hence other transactions sees only those values (of other transactions), which are commited.

Outdated old rows are deleted at a later time by the utility `vacuumdb` respectively the SQL command `vacuum`.

1.8 Backup, Recocery

The term 'cold' as an addition to the backup methode name refers to the fact, that with this methode the instance must be stopped to create a usefull backup. In contrast, the addition 'hot' denotes methodes where the instance MUST run (and hence changes to the data may occur during backup actions).

10 `https://en.wikipedia.org/wiki/Multiversion_concurrency_control`

1.8.1 (Cold) Backup (file system tools)

A cold backup is a consistent copy of all files of the *cluster* with OS tools like cp or tar. During the creation of a cold backup the *instance* must **not** run - otherwise the backup is useless. Hence you need a period of time in which no application use any *database* of the *cluster* - a continious 7x24 operation mode is not possible. And secondly: the cold backup works only on the *cluster* level, not on any finer granularity like database or table.

Hint: A cold backup is sometimes called an "offline backup".

1.8.2 (Hot) Logical Backup (pg_dump utility)

A logical backup is a consistent copy of the data within a *database* or some of its parts. It is created with the utility `pg_dump`. Although `pg_dump` may run in parallel to applications (the *instance* must be up), it creates a consistent snapshot as of the time of its start.

`pg_dump` supports two output formats. The first one is a text format containing SQL commands like CREATE and INSERT. Files created in this format may be used by `psql` to restore the backup-ed data. The second format is a binary format and is called the 'archive format'. Files with this format can be used to restore its data with the tool `pg_restore`.

As mentioned, `pg_dump` works at the *database* level or smaller parts of *databases* like tables. If you want to refer to the *cluster* level, you must use `pg_dumpall`. Please notice, that important objects like users/roles and their rights are always defined at *cluster* level.

Hint: A logical backup is one form of an "online backup".

1.8.3 (Hot) Physical Backup or 'Base Backup'

A physical backup is a **possibly inconsistent** copy of the files of a *cluster*, created with an operating system utility like cp or tar. At first glance such a backup seems to be useless. To understand its purpose, you must know PostgreSQL's recover-from-crash strategy.

> At all times and independent from any backup/recovery action, PostgreSQL maintains *WAL files* - primarily for crash-safety purposes. *WAL files* contain *log records*, which reflects all changes made to the data. In the case of a system crash those *log records* are used to recover the *cluster* to a consistent state. The recover process searches the timestamp of the last *checkpoint* and replays all subsequent *log records* in chronological order against this version of the *cluster*. Through that actions the *cluster* gets recovered to a consistent state and will contain all changes up to the last COMMIT.

The existence of a physical backup plus *WAL files* in combination with this recovery-from-crash technique can be used for backup/recovery purposes. To implement this, you need a physical backup (which may reflect an inconsistent state of the *cluster*) and which acts as the starting point. Additionally you need all *WAL files* since the point in time, when you have created this backup. The recover process uses the described recovery-from-crash technique and replays all *log records* in the *WAL files* against the backup. In the exact same way as before, the *cluster* comes to a consistent state and contains all changes up to the last COMMIT.

Please keep in mind, that physical backups work only on cluster level, not on any finer granularity like database or table.

<u>Hint</u>: A physical backup is one form of an "online backup".

1.8.4 PITR: Point in Time Recovery

If the previously mentioned *physical backup* is used during recovery, the recovery process is not forced to run up to the latest available timestamp. Via a parameter you can stop him at a time in the past. This leads to a state of the *cluster* at this moment. Using this technique you can restore your *cluster* to a time, which is between the time of creating the *physical backup* and the end of the last *WAL file*.

1.8.5 Standalone (Hot) Backup

The standalone backup is a special variant of the physical backup. It offers online backup (the *instance* keeps running) but it lacks the possibility of *PITR*. The recovery process recovers always up to the end of the standalone backup process. *WAL files*, which arise after this point in time, cannot be applyed to this kind of backup. Details are described here[11].

1.8.6 Archiving

Archiving is the process of copying *WAL files* to a failsafe location. When you plan to use *PITR* you must ensure, that the sequence of *WAL files* is saved for a longer period. To support the process of copying *WAL files* at the right moment (when they are completely filled and a switch to the next *WAL file* has taken place), PostgreSQL runs the *archiving process* which is part of the *instance*. This process copies *WAL files* to a configurable destination.

1.8.7 Recovering

Recovering is the process of playing *WAL files* against a *physical backup*. One of the involved steps is the copy of the *WAL files* from the failsafe archive location to its original location in

11 `http://www.postgresql.org/docs/current/static/continuous-archiving.html`

'/pg_xlog'. The aim of recovery is bringing the *cluster* into a consistent state at a defined timestamp.

1.8.8 Archive Recovery Mode

When recovering takes place, the *instance* is in *archive recovery mode.*

1.8.9 Restartpoint

A restartpoint is an action similar to a *checkpoint.* Restartpoints are only performed when the instance is in *archive recovery mode* or in *standby mode.*

1.8.10 Timeline

After a successful recovery PostgreSQL tranfers the *cluster* into a new timeline to avoid problems, which may occur when PITR is reset and *WAL files* reapplied (eg: to a different timestamp). Timeline names are sequential numbers: 1, 2, 3, ...

1.9 Replication

Replication is a technique to send data, which was written within a *master server*, to one or more *standby servers* or even another *master server.*

1.9.1 Master or Primary Server

The master server is an *instance* on a *server* which sends data to other *instances* - additionally to its local processing of data.

1.9.2 Standby or Slave Server

The standby server is an *instance* on a *server* which receives information from a *master server* about changes of its data.

1.9.3 Warm Standby Server

A warm standby server is a running *instance*, which is in *standby_ mode* (recovery.conf file). It continuously reads and processes incoming *WAL files* (in the case of *log-shipping*) or *log records* (in the case of *streaming replication*). It does not accept client connections.

1.9.4 Hot Standby Server

A hot standby server is a warm standby server with the additional flag *hot_standby* in postgres.conf. It accepts client connections and read-only queries.

1.9.5 Synchronous Replication

Replication is called *synchronous*, when the *standby server* processes the received data immediately, sends a confirmation record to the *master server* and the *master server* delays its COMMIT action until he has received the confirmation of the *standby server*.

1.9.6 Asynchronous Replication

Replication is called *asynchronous*, when the *master server* sends data to the *standby server* and does not expect any feedback about this action.

1.9.7 Streaming Replication

The term is used when *log entries* are transfered from *master server* to *standby server* over a TCP connection - in addition to their transfer to the local *WAL file*. Streaming replication is *asynchronous* by default but can also be *synchronous*.

1.9.8 Log-Shipping Replication

Log shipping is the process of transfering *WAL files* from a *master server* to a *standby server*. Log shipping is an asynchronous operation.

The daily work as a PostgreSQL DBA is based on the knowledge of PostgreSQL's architecture: strategy, processes, buffers, files, configration, backup and recovery, replication, and a lot more. The page on hand describes the most basic concepts.

1.10 Introduction

PostgreSQL is a relational database management system[12] with a client-server architecture[13]. At the server side the PostgreSQL's processes and shared memory work together and build an *instance*, which handles the accesses to the data. Client programs connect oneself to the *instance* and request read and write operations.

12 https://en.wikipedia.org/wiki/relational%20database%20management%20system
13 https://en.wikipedia.org/wiki/client-server%20architecture

1.11 The Instance

On Unix systems the instance consists of multiple processes whereas on Windows systems the different tasks run as different threads within one service:

- *postmaster* process
- Multiple *postgres* processes, one for each connection
- *WAL writer* process
- *background writer* process
- *checkpointer* process
- *autovacuum launcher* process (optional)
- *logger* process (optional)
- *archiver* process (optional)
- *stats collector* process (optional)
- *WAL sender* process (if Streaming Replication is active)
- *WAL receiver* process (if Streaming Replication is active)

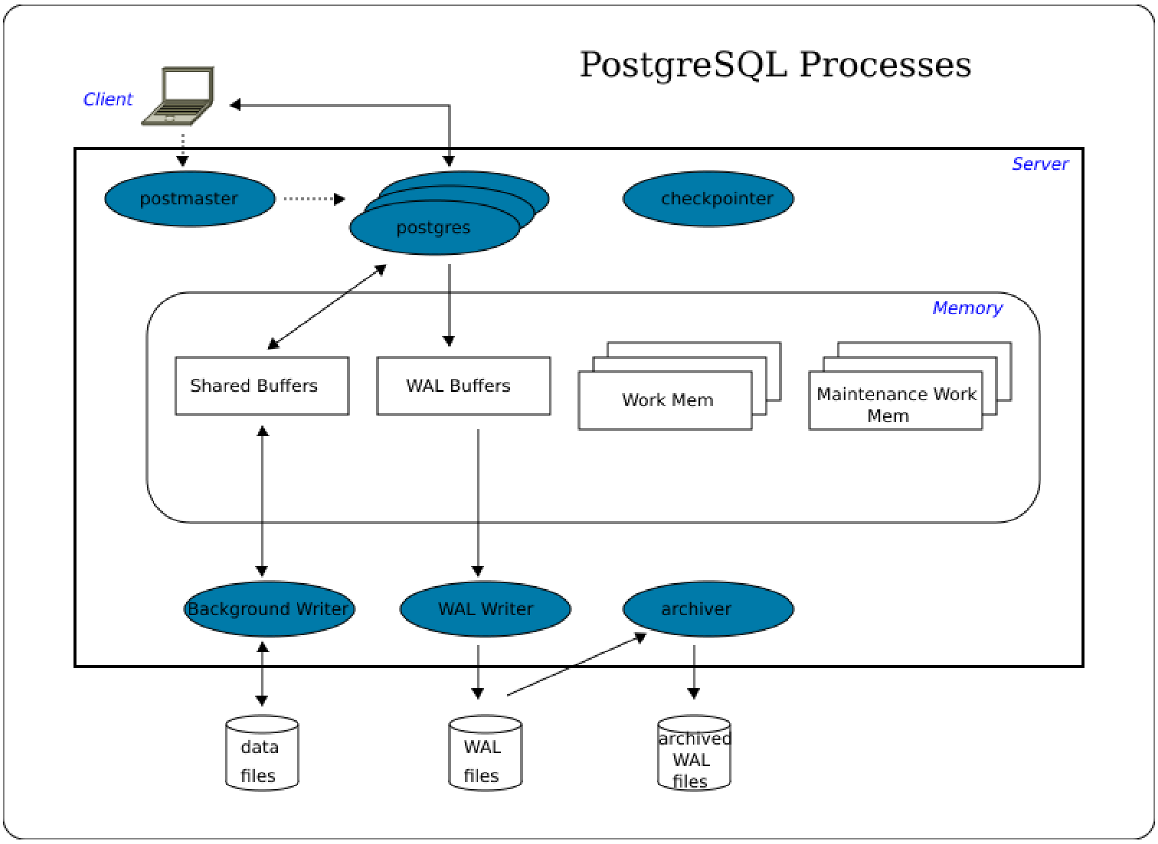

Figure 3

1.12 How Data is processed

1.12.1 Connecting to the Instance

Client applications, which run on a different server than the instance, use the IP protocoll to connect to it. If client application and instance run on the same server, the same connection methode is possible. But it is also possible to use a conncetion via a local socket.

In a first step the application connects to the *postmaster* process. The *postmaster* checks the application's rights and - if successful - starts a new *postgres* process and connects it with the client application.

1.12.2 Accessing Data

Client processes send and request data to and from the instance. For performance reasons, the instance doesn't write or read the requested data directly to or from disk files. First, it buffers them in shared memory and at a later stage it flushes the buffers to disk.

To perform client requests, the corresponding *postgres* process communicates with the shared buffers and WAL buffers and manipulate their contents. When the client requests a `COMMIT`, the *WAL writer* process writes and flushes all WAL records resulting from this transaction to the WAL file. As the WAL file - in contrast to the data files - is written strictly sequentially, this operation is relatively fast. After that, the client gets its `COMMIT` confirmation. At this point, the database is inconsistent, which means that there are differences between shared buffers and the corresponding data files.

Periodically the *background writer* process checks the shared buffers for 'dirty' pages and writes them to the appropriate data files. 'Dirty' pages are those whose content was modified by one of the *postgres* processes after their transfer from disk to memory.

The *checkpointer* process also runs periodically, but less frequently than the *background writer*. When it starts, it prevents further buffer modifications, forces the *background writer* process to write and flush all 'dirty' pages, and forces the *WAL writer* to write and flush a `CHECKPOINT` record to the WAL file after which the database is consistent (i.e. a) the content of the Shared buffers is the same as the data in the files, b) all modifications of WAL buffers are written to WAL files, and c) table data correlates with index data.) This consistency is the purpose of checkpoints.

In essence the instance contains at least the three processes *WAL writer*, *background writer*, and *checkpointer* - and one *postgres* process per connection. In most cases there are some more processes running.

1.12.3 Optional Processes

The *autovacuum launcher* process starts a few number of worker processes, which removes superflous row versions according to the MVCC architecture of PostgreSQL. This work is done in shared memory and the 'dirty' pages are written to disc in the same way as such, which results from write requests of other clients.

The *logger* process writes log, warning, and error messages to a log file (not to the WAL file!).

The *archiver* process copies WAL files, which are completely filled by the *WAL writer*, to a configurable location for mid-term storing.

The *stats collector* process continuously collects information about the number of accesses to tables and indices, total number of rows in tables, and works in coordination with VACUUM/ANALYZE and ANALYZE.

The *WAL sender* and *WAL receiver* processes are part of the Streaming Replication feature. They exchange data about changes in the master server bypassing the WAL files on disc.

1.13 The Directory Structure

Within a cluster there is a fix structure of subdirectories and files. At last all information is stored within these files. Some information contains to the cluster at all, and some belongs to single databases - especially tables and indexes.

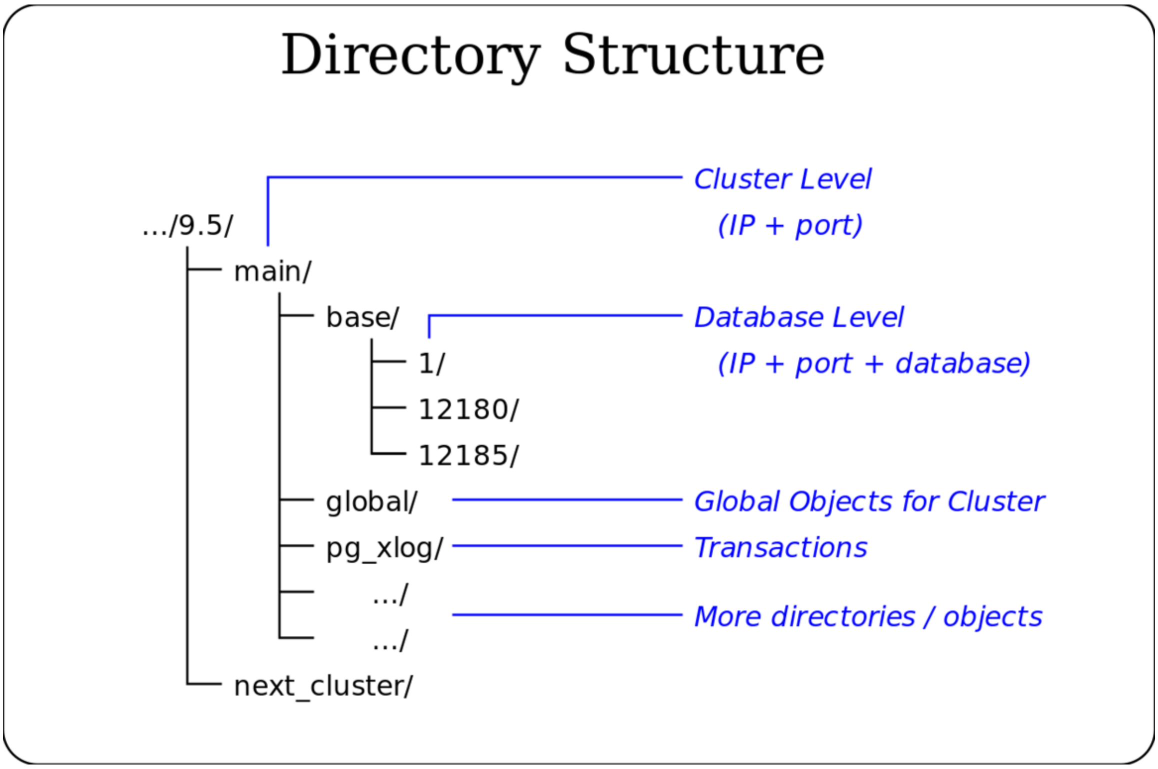

Figure 4

The PostgreSQL instance consists of several processes running on a server platform. They work together in a coordinated manner using common configuration files and a common start/stop procedure. Thus all are running or none of them.

The program `pg_ctl` controls and observes them as a whole. When you are logged in as user *postgres* you can start it from a shell. The simplified syntax is:

```
pg_ctl [ status | start | stop | restart | reload | init ] [-U username] [-P
 password] [--help]
```

1.14 status

When pg_ctl runs in the `status` mode, it lists the actual status of the instance.

```
$ pg_ctl status
pg_ctl: server is running (PID: 864)
/usr/lib/postgresql/9.4/bin/postgres "-D" "/var/lib/postgresql/9.4/main" "-c"
 "config_file=/etc/postgresql/9.4/main/postgresql.conf"
$
```

You can observe, whether the instance is running or not, the process id (PID) of the postmaster, the directory of the cluster and the name of the configuration file.

1.15 start

When pg_ctl runs in the `start` mode, it tries to start the instance.

```
$ pg_ctl start
...
database system is ready to accept connections
$
```

When you see the above message everythink works fine.

1.16 stop

When pg_ctl runs in the `stop` mode, it tries to stop the instance.

```
$ pg_ctl stop
...
database system is shut down
$
```

When you see the above message the instance is shut down, all connections to client applications are closed and no new applications can reach the database. The `stop` mode knows three different modes for shutting down the instance:

- *Smart* mode waits for all active clients to disconnect.
- *Fast* mode (the default) does not wait for clients to disconnect. All active transactions are rolled back and clients are forcibly disconnected.

- *Immediate* mode aborts all server processes immediately, without a clean shutdown.

Syntax: `pg_ctl stop [-m s[mart] | f[ast] | i[mmediate] ]`

1.17 restart

When pg_ctl runs in the `restart` mode, it performs the same actions as in a sequence of `stop` and `start`.

1.18 reload

In the `reload` mode the instance reads and reloads its configuration file.

1.19 init

In the `init` mode the instance creates a complete new cluster with the 3 databases *template0*, *template1*, and *postgres*. This command needs the additional parameter `-D datadir` to know at which place in the file system it shall create the new cluster.

The main configuration file is *postgresql.conf*. He is divided into serval sections according to different tasks. The second important configuration file is *pg_ hba.conf*, where authentication definitions are stored.

Both files reside in the special directory $PGDATA (Debian/Ubuntu) or in the main directory of the cluster (RedHat).

Numerous definitions have a dynamic nature, which means that they take effect with a simple `pg_ctl reload`. Others require a restart of the instance `pg_ctl restart`. The comments within the delivered default configuration files describe which one of the two actions has to be taken.

1.20 postgresql.conf

1.20.1 File Locations

The value of *data_ directory* defines the location of the cluster's main directory. In the same way the value of *hba_file* defines the location and the name of the above mentioned *pg_ hba.conf* file (host based authentication file), where rules for authentication are stored - some more details are shown below[14].

14 Chapter 1.21 on page 22

1.20.2 Connections

In the connections section you define the port number (default: 5432), with which client applications can reach the instance. Furthermore the maximal number of connections is defined as well as some SSL, IP and TCP settings.

1.20.3 Resources

The main definition in the resources section is the size of shared buffers. It determines, how much space is reserved to 'mirrow' the content of data files within PostgeSQL's buffers in RAM. The predefined default value of 128 MB is relative low.

Secondly, there are definitions for the work and the maintenance memory. They determine the RAM sizes for sorts, create index commands, ... This two RAM areas exists per connection and are used individually by them whereas the shared buffers exists only once for the whole instance and are used concurrently by multiple processes.

Additionally there are some definitions concerning *vacuum* and *background writer* processes.

1.20.4 WAL

In the WAL section there are definitions for the behaviour of the WAL mechanism.

First, you define a WAL level out of the four possibilities *minimal*, *achive*, *hot_standby*, and *logical*. Depending on the decision, which kind of archiving or replication you want to use, the WAL mechanism must write only basic information to the WAL files or some more information. *minimal* is the basic methode which is always required for every crash recovery. *archive* is necessary for any archiving action, which includes the point-in-time-recovery (PITR) mechanism. *hot_standby* adds information required to run read-only queries on a standby server. *logical* adds information necessary to support logical decoding.

Additionally and in correlation to the WAL level *archive* there are definitions which describe the archive behaviour. Especially the 'archive_command' is essential. It contains a command which copies WAL files to an archive location.

1.20.5 Replication

If you use replication to a different server, you can define the necessary values for master and standby server in this section. The master reads and pay attention only on the master-definitions and the standby only on the standby-definitions (you can copy this section of 'postgres.conv' directly from master to standby). You must define the WAL level to an appropriate value.

1.20.6 Tuning

The tuning section defines the relative costs of different operations: sequential disc I/O, random disc I/O, process one row, process one index entry, process one function-call or arithmetic operation, size of effective RAM pages (PostgreSQL + OS) per process which will be available at runtime. These values are used by the query planner during its seach for an optimal query execution plan. The values are no real values (in sense of milliseconds or number of CPU cycles), they are only a) a rough guideline for the query planer and b) relative to each other. The real values during later query execution may differ significantly.

There is also a subsection concerning costs for the genetic query optimizer, which - in opposite to the standard query optimizer - implements a heuristic searching for optimal plans.

1.20.7 Error Logging

The error logging section defines the amount, location and format of log messages which are reported in error situations or for debugging purposes.

1.20.8 Statistics

In the statistics section you can defines - among others - the amount of statistic collection for parsing, planing and execution of queries.

1.21 pg_hba.conf

The *pg_hba.conf* file (host based authentication) contains rules for client access to the instance. All connection attempts of clients, which does not satisfy this rules are rejected. The rules restrict the connection type, client IP adress, database within the cluster, user-name, and authentication methode.

There are two main connection types: local connections (*local*) via sockets and connections via TCP/IP (*host*). The term *local* refers to the situation, where a client program resides on the same machine as the instance. But even in such situations the client may enforce the *host* connection type by using the TCP/IP syntax (eg: 'localhost:5432') referring the cluster.

The client IP adress is a single IPv4 or IPv6 adress or a masking of a net-segment via a CIDR mask.

The database and the client user name must be given explicitely or may be abbreviated by the key word "ALL".

There are different authentication methodes

- *trust*: don't ask for any password
- *reject*: don't allow any access
- *password*: ask for a password
- *md5*: same as 'password', but the transfer of the password occurrs MD5-encrypted
- *peer*: trust the client, if he uses the same database username as his operation system username (only applicable for local connections)

Since the pg_hba.conf records are examined sequentially for each connection attempt, the order of the records is significant. The first match between defined criterias and properties of incoming connection requests hits.

Multiversion Concurrency Control[15] (MVCC) is a common database technique to accomplish two goals: first, it allows the management of parallel running transactions on a logical level and second, it ensures high performance for concurrent read and write actions. MVCC implementation is very complex, thus this short introduction is only the tip of an iceberg.

Hint: Please consider the wording on this page: *row* is used in the conventional sense of relational database systems, whereas *record* denotes one or more versions of this *row*.

The idea is, that the change of a column value does not lead to a change of the value in the original record but leads to a complete new record with the changed value. Thus an UPDATE command does not really update values in records. It creates additional records and leaves the values in the old records unchanged.

This behaviour raises the question, how other processes shall distinct between the multiple records of a row. To resolve the situation PostgreSQL adds some additional hidden system columns to every record. The two columns `xmin` and `xmax` contains transaction IDs. `xmin` contains the transaction ID of the transaction, which has created the record. `xmax` contains the transaction ID of the transaction, which has created the next version of this record - or has deleted the record. `xmax` may be 0.

Time	**tmin**	**tmax**	**column value**	**Transaction #25**	**Transaction #33**
t_0	20	0	1		
				UPDATE ...	
t_1	20	25	1		
	25	0	2		
				COMMIT	
					SELECT ...

If the above row is deleted at a later point in time, the ID of this transaction is stored in `xmax` of the latest record - without creating an additional record or changing any column value.

Time	**tmin**	**tmax**	**column value**	**Transaction #101**	**Transaction #120**
t_1	20	25	1		
	25	0	2		
				DELETE ...	
t_2	20	25	1		
	25	101	2		
				COMMIT	
					SELECT ...

Together with additional flags concerning COMMITs plus information about their own state, other transactions can decide, which records are visible to them and which are not - because `tmin` or `tmax` are lower or higher than their own transaction ID.

In the first example transaction #33 cannot see the record with value **1** as `tmax` is lower than its own TrID, but not 0. It can see the record with value **2** as `tmax` is 0 and `tmin` is lower than its TrID. Thus the SELECT returns the second record as the row. In the second example transaction #120 cannot see any of the two records as its TrID is heigher than both `tmax` and no `tmax` is 0. Thus the SELECT returns no row.

15 https://en.wikipedia.org/wiki/Multiversion_concurrency_control

The situation can get much more complicated in situations where the SELECT runs before the COMMIT of the writing transaction, or when the writing transaction use multiple write-operations within its transaction context, or if a transaction aborts, ...

The utility `vacuumdb` respectively the SQL command `vacuum` physically deletes outdated old records at a later time.

WAL files are files, where changed data values are stored in a binary format. This is additional information and in this respect it is redundant to the information in the database files. WAL files are a specific kind of 'diff' files.

Writing to WAL files is very fast as they are written always sequentially. In contrast to WAL files database files are organized in special structures like trees, which possibly must be reorganized during write operations or which points to blocks at far positions. Thus writes to database files are much slower.

When a client requests a write operation like UPDATE, the modifications to the data is not instantly written to database files. For the mentioned performance reasons this is done in a special sequence and - in some parts - asynchroniously to the client requests. First, data is written and flushed to WAL files. Second, it is stored in shared buffers in RAM. And in the end it is written from shared buffers to database files. The client must not wait, until the end of all operations. After the first two very fast actions, he is informed that his request is completed. The third operation is performed asynchroniously at an later (or prior) point in time.

2 Special Topics

There are various tools which supports the DBA in its daily work. Some parts of this work can be done in standard SQL syntax, eg: `CREATE USER ...`, whereas a lot of important tasks like backups or cleanups are out of scope of SQL and are supported only by ventor-specific SQL extentions or utilities, eg: VACUUM. Thus in most cases the DBA tools support standard-SQL syntax as well as PostgreSQL-specific SQL syntax and the spawning of PostgreSQL's utilities.

2.1 psql

psql is a client program which is delivered as an integral part of the PostgreSQL downloads. Similar to a bash shell it is a line-mode program and may run on the server hardware or at a client. *psql* knows two kinds of commands:

- Commands starting with a backslash, eg: `\dt` to list tables. Those commands are interpreted by *psql* itself.
- All other commands are send to the instance and intepreted there, eg: `SELECT * FROM mytable;`.

Thus it is an ideal tool for interactive and batch SQL processing. The whole range of PostgreSQL SQL syntax can be used to perform everythink, what can be expressed in SQL.

```
$ # start psql from a bash shell for database 'postgres' and user 'postgres'
$ psql postgres postgres
postgres=#
postgres=# -- a standard SQL command
postgres=# CREATE TABLE t1 (id integer, col_1 text);
CREATE TABLE
postgres=# -- display information about the new table
postgres=# \dt t1
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+---------
 public | t1   | table | postgres
(1 row)
postgres=#
postgres=# -- perform a PostgreSQL specific task - as an example of a typically
 DBA action
postgres=# SELECT pg_start_backup('pitr');
 pg_start_backup
-----------------
 0/2000028
(1 row)
postgres=#
postgres=# -- terminate psql
postgres=#\q
$
```

Here are some more examples of *psql* 'backslash'-commands

- `\h` lists syntax of SQL commands
- `\h SQL-command` lists syntax of the named SQL-command
- `\?` help to all 'backslash' commands
- `\l` lists all databases in the actual cluster
- `\echo :DBNAME` lists the actual database (consider upper case letters). In most cases the name is part of the psql prompt.
- `\dn` lists all schemas in the actual database
- `\d` lists all tables, views, sequences, materialized views, and foreign tables in the actual schema
- `\dt` lists all tables in the actual schema
- `\d+ TABLENAME` lists all columns and indexes in table TABLENAME
- `\du` lists all users in the actual cluster
- `\dp` lists access rights (priviledges) to tables, ...
- `\o FILENAME` redirects following output to FILENAME
- `\t` changes output to 'pure' data (no header, ...)
- `\! COMMAND` executes COMMAND in a shell (outside psql)
- `\q` terminates *psql*

2.2 pgAdmin

pgAdmin is a tool with a graphical user interface for Unix, Mac OSX and Windows operating systems. In most cases it runs on a different hardware than the instance. For the major operating systems it is an integral part of the download, but it is possible to download the tool separately[1].

pgAdmin extends the functionalities of *psql* by a lot of intuitive, graphical representations of database objects, eg. schemas, tables, columns, users, result lists, query execution plans, dependencies between database objects, and much more. To give you a first impression of the surface, some screenshots[2] are online.

2.3 phpPgAdmin

phpPgAdmin is a graphical tool with a similar feature set as *pgAdmin*. It is written in PHP, therefore you additionally need Apache and PHP packages.

phpPgAdmin is not part of the standard PostgreSQL downloads, it's distributed via sourgeforge[3].

1 `http://www.pgadmin.org/download/`
2 `http://www.pgadmin.org/screenshots/`
3 `http://sourceforge.net/projects/phppgadmin/`

2.4 Other Tools

There is a lot of other tools[4] with a GUI interface. Their functionality variies greatly from pure SQL support up to entity-relationship and UML support. Some of the tools are open/free source, others proprietary.

PostgreSQL supports the concept of `roles` [5] to realize security issues. The concept is independent from operating system user accounts.

This concept of roles subsumes the concepts of "users" and "groups". A role can be thought of as either a database user, or a group of database users, depending on how the role is set up. Roles have certain privileges on database objects like tables or functions and can assign those privileges to other roles. Roles are global across a cluster - and not per individual database.

Often users, which shall have identical privileges, are grouped together to a user group and the privileges are granted to the group.

```
-- the user group
CREATE ROLE group_1 ENCRYPTED PASSWORD 'xyz';
GRANT SELECT ON table_1 TO group_1;
-- the users
CREATE ROLE adam LOGIN ENCRYPTED PASSWORD 'xyz';  -- Default is NOLOGIN
CREATE ROLE anne LOGIN ENCRYPTED PASSWORD 'xyz';
-- the link between user group and users
GRANT group_1 TO adam, anne;
```

Concerning `CREATE ROLE` you can assign the privileges SUPERUSER, CREATEDB, CREATEROLE, REPLICATION and LOGIN. Concerning `GRANT` you can pass on access privileges to database objects like tables - or you define group membership.

Implicitly there is the special role PUBLIC, which can be thought of as a group that always includes all roles. Thus, privileges assigned to PUBLIC are implicitly given to all roles, even when those roles are created at a later stage.

2.5 References

2.6 Protocol

All access to data is done by server (or backend) processes, to which client (or frontend) processes must connect to. In most cases instances of the two process classes reside on different hardware, but it's also possible that they run on the same computer. The communication between them uses a PostgreSQL-specific protocol, which runs over TCP/IP or over UNIX sockets. For every incoming new connection the backend process (sometimes called the *postmaster*- process) creates a new *postgres* backend process. This backend process gets part of the *PostgeSQL instance*, which is responsible for data accesses and database consistence.

4 `https://wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools`

5 Concept of roles `http://www.postgresql.org/docs/current/static/user-manag.html`

The protocol handles the authentication process, client request, server responses, exceptions, special situations like a NOTIFY, and the final regular or irregular termination of the connection.

2.7 Driver

Some client programs like *psql* use this protocol directly and drivers like ODBC, JDBC (type 4), Perl DBI, and those for Python, C, C++, and much more are based on if. The C library *libpq* has a special role: she is the basis for many driver implementations.

You can find an extensive list of drivers at the postgres wiki [6] and some more commercial and open source implemenations at the 'products' site [7].

2.8 Authentication

Clients must authenticate himself before they get access to any data. This process has one or two stages. Optionally, the client may get access to the server by satisfying the operating system hurdles. This is often realized by delivering a public ssh key. The authentification against PostgeSQL is a separate, independent step using a database-username, which may or may not correlate to an operating system username. PostgreSQL stores all rules for this second step in the file *pg_hba.conf*.

pg_hba.conf stores every rule in one line. The lines are evaluated from start to end and the first matching line applies. The main layout of this lines is as follows

```
local   DATABASE  USER            METHOD  [OPTIONS]
host    DATABASE  USER   ADDRESS  METHOD  [OPTIONS]
```

Words in upper case must be replaced by specific values whereas *local* and *host* are key words. They decide, for which kind of connection the rule shall apply: *local* for clients residing at the same computer as the backend (they use UNIX sockets for the communication) and *host* for clients at different computers (they use TCP/IP). There is one noteable exception: In the former case clients can use the usual TCP/IP syntax `--host=localhost --port=5432` to switch over to use TCP/IP. Thus the *host* syntax applies for them.

DATABASE and USER have to be replaced by the name of the database and the name of the database-user, for which the rule will apply. In both cases the key word ALL is possible to define, that the rule shall apply to all databases respectivelly all database-users.

ADDRESS must be replaced by the hostname or the IP adress plus CIDR mask[8] of the client, for which the rule will apply. IPv6 notation is supported.

METHODE is one of the following. The thereby defined rule (=line) applies, if database/user/address combination matches.

6 Driver Wiki `https://wiki.postgresql.org/wiki/List_of_drivers`
7 Commercial and open source driver `http://www.postgresql.org/download/products/2/`
8 `https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing%23CIDR_notation`

- trust: The connection is allowed without a password.
- reject: The connection is rejected.
- password: The client must send a valid user/password combination.
- md5: Same as 'password', but the password is encrypted.
- ldap: It uses LDAP as the password verification method.
- peer: The connection is allowed, if the client is authorized against the operation system with the same username as the given database username. This method is only supported on local connections.

There are some more techniques in respect to the METHODE.

Some examples:

```
# joe cannot connect to mydb - eg. with psql -, when he is logged in to the
 backend.
local  mydb  joe  reject

# bill (and all other persons) can connect to mydb when they are logged in to
 the
# backend  without specifying any further password.  joe will never reach this
 rule, he
# is rejected by the rule in the line before. The rule sequence is important!
local  mydb  all  trust

# joe can connect to mydb from his workstation '192.168.178.10', if he sends
# the valid md5 encrypted password
host  mydb  joe  192.168.178.10/32 md5

# every connection to mydb comeing from the IP range 192.168.178.0 -
 192.168.178.255
# is accepted, if they send the valid md5 encrypted password
host  mydb  all  192.168.178.0/24 md5
```

For the DATABASE specification there is the special keyword REPLICATION. It denotes the streaming replication process. REPLICATION is not part of ALL and must be specified separately.

2.9 References

Creating backups is an essential task for every database administrator. He must ensure that in the case of a hardware or software crash the database can be restored with minimal data loss. PostgreSQL offers different strategies to support the DBA in his effort to achieve this goal.

First off all, backup technology can generally be divided into two classes: A **cold** backup is a backup which is taken during a period of time, where no database file is open. In the case of PostgreSQL this means, that the instance is not running. The second class of backups are called **hot** backups. They are taken during normal working times, which means, that applications can perform read and write actions in parallel to the backup creation.

```
Backups
   |
   +-- Cold Backup
   |
```

```
+-- Hot Backups
     |
     +-- Logical Backup
     |
     +-- Physical Backup
     |
     +-- Physical Backup plus PITR
```

2.10 Cold Backup

A cold backup is a consistent copy of all files which constitutes the cluster with all of its databases. To be 'consistent' this copy cannot be taken at any time. There is only one way to create a consistent, usefull cold backup: You must shut down the PostgreSQL instance `pg_ctl stop`, which disconnects all applications from all databases.

After the instance is shut down, you can use any operating system utility (cp, tar, dd, rsync, ...) to create a copy of all cluster files to a secure location: on the disc of a different server, on any backup system at a SAN or the intranet, a tape system, ...

Which files constitutes the cluster and are therefor necessary to copy?

- All files under the directory node, where the cluster is installed. The logical $PGDATA points to this directory. Its name is somethink like '.../postgresql/9.4/main'.
- If the cluster uses the directory layout as it is used on the Linux derivate Ubuntu, the configuration files are located outside of the above directory structure in a separate directory. In this case you must additionally copy the directory with the configuration files.
- All files, which are used as a tablespace.

One may try to backup only special parts of a cluster, eg. a huge file which represents a table on a separate partition or tablespace - or the opposite: everything without this huge file. Even if the instance is shut down during the generation of such a partial copy, copies of this kind are useless. The restore of a cold backup needs all data files and files with metainformation of the cluster.

Cold backups are sometimes called *offline backups*.

Advantages

- The methode is easy to install.

Disadvantages

- A continuous 7x24 operation mode of any of the databases of this cluster is not possible.
- It is not possible to backup smaller parts of a cluster like a single database or table.
- You cannot restore parts of the backup. Either you restore everythink or nothing.
- After a crash you cannot restore the data to any point in time after the last backup generation. All changes to the data after this moment gets lost.

How to Recover

In the case of a crash you can restore the data from a cold backup by performing the following steps:

- Stop the instance.
- Delete all original files of the crashed cluster: $PGDATA plus, for the Ubuntu way, the configuration files.
- Copy the files of the backup to their original places.
- Start the instance. It shall start in the normal way, without any special message.

2.11 Hot Backups

In opposite to *cold backups* a *hot backup* is taken during the instance is running and applications may change data during this periode of time.

Hot backups are sometimes called *online backups*.

2.11.1 Logical Backup

A logical backup is a consistent copy of the data within a database or some of its parts. It is created with the utility `pg_dump`. Although `pg_dump` may run in parallel to applications (the instance must be up), it creates a consistent snapshot as of the time of its start. If any data changes its value during the creation of the backup, the backup takes the old value whereas the application sees the new one. (Logical backups runs in serializable transactions.) This is possible because of PostgreSQL's MVCC implementation.

`pg_dump` supports two output formats. The first one is a text format containing SQL commands like CREATE and INSERT. Files created in this format may be used by `psql` to restore the backup-ed data. The second format is a binary format and is called the *archive format*. To restore files with this format you must use the tool `pg_restore`.

The following diagram visualise the cooperation of `pg_dump`, `psql` and `pg_restore`.

```
script in SQL syntax  <--  pg_dump text format  <--  database  -->  pg_dump
 binary format  --> binary file
        |
                   |
        +-------------------->  psql  ------------>  database  <---------
 pg_restore  <-----------+
```

`pg_dump` can dump data, schema definitions or both. The parameters `--data-only` and `--schema-only` control, which parts are dumped.

As mentioned, `pg_dump` works at the database level or smaller parts of databases like tables. If you want to refer to the cluster level, you must use `pg_dumpall`. Please notice, that important objects like users/roles and their rights are always defined at cluster level. `pg_dumpall` without detailed parameters will dump everything of the cluster: all data and structures of all databases plus all user definitions plus definitions of their rights. With the parameter `--globals-only` you can restrict its behaviour to dump global objects only.

```
Some Examples:

$ # dump complete database 'finance' in text format to a file
$ pg_dump --dbname=finance --username=boss --file=finance.sql
$
$ # restore database content (to a different or an empty database)
$ psql --dbname=finance_x --username=boss <finance.sql
$
$
$
$ # dump table 'person' of database 'finance' in binary format to a file
$ pg_dump --dbname=finance --username=boss --table=person --format=c
 --file=finance_person.archive
$
$ restore table 'person' from binary archive
$ pg_restore --dbname=finance_x --username=boss --format=c
 <finance_person.archive
$
```

Advantages

- Continuous 7x24 operation mode is possible.
- Small parts of cluster or database may be backup-ed or restored
- Using the text format you can switch from one PostgreSQL version to another or from one hardware platform to another.

Disadvantages

- The text format uses much space, but it compresses well.

How to Recover

As mentioned in the above diagram the recovery process depends on the format of the dump. Text files are in standard SQL syntax. To recreate objects from this format you can use SQL utilities like `psql`. Binary files are in the *archive format*. They can only be used by the utility pg_restore.

2.11.2 Physical Backup

A physical backup is an **inconsistent** copy of the files of a cluster, created with operating system utilities like cp or tar taken at a time whereas applications modify data. At first glance such a backup seems to be useless. To understand its purpose, you must know PostgreSQL's recover-from-crash strategy.

At all times and independent from any backup/recovery action, PostgreSQL maintains *Write Ahead Log (WAL) files* - primarily for crash-safety purposes. Such *WAL files* contain *log records*, which reflects all changes made to the data. Prior to the transfer to the data files of the database the *log records* are stored at disc. In the case of a system crash those *log records* are used to recover the cluster to a consistent state. The recover process searches the timestamp of the last *checkpoint* and replays all subsequent *log records* in chronological order against this version of the cluster. Through that actions the cluster gets recovered to a consistent state and will contain all changes up to the last COMMIT.

The existence of a physical backup with its *WAL files* in combination with this recovery-from-crash technique can be used for backup/recovery purposes. To implement this, you have to restore the physical backup with its *WAL files*. When the instance starts again, it uses the described recovery-from-crash technique and replays all *log records* in the *WAL files* against the restored database files. In the exact same way as before, the cluster comes to a consistent state and contains all changes up to the last COMMIT (of backup-time).

Please keep in mind, that physical backups work only on cluster level, not on any finer granularity like database or table.

Physical backup without PITR sometimes is called *standalone hot physical backup*.

Advantages

- Continuous 7x24 operation mode is possible.

Disadvantages

- Physical backup works only on cluster level, not on any finer granularity like database or table.
- Without PITR (see below) you will lose all data changes between the time, when the physical backup is taken, and the crash.

How to Take the Backup and Recover

To use this technique it is necessary to configure some parameters in the *postgres.conf* file for WAL and archive actions. As the usual technique is *Physical Backup plus PITR* we describe it in the next chapter.

2.11.3 Physical Backup plus PITR

The term *PITR* stands for *Point In Time Recovery* and denotes a technique, where you can restore the cluster to any point in time between the creation of the backup and the crash.

The *Physical Backup plus PITR* strategy takes a physical backup plus all WAL files, which are created and archived since the time of taking this backup. To implement it, three actions must be taken:

- Define all necessary parameters in *postgres.conf*
- Generate a physical backup
- Archive all arising WAL files

If a recovery becomes necessary, you have to delete all files in the cluster, recreate the cluster by copying the physical backup to its original location, create the file *recovery.conf* with some recovery-information and restart the instance. The instance will recreate the cluster according to its parameters in *postgres.conf* and *recovery.conf* to a consistent state including all data changes until last COMMIT.

Advantages

- Continuous 7x24 operation mode is possible.
- Recover with minimal data lose.
- Generating WAL files is the basis for additional features like *replication*.

Disadvantages

- Physical backup works only on cluster level, not on any finer granularity like database or table.
- If your database is very busy and changes a lot of data, many WAL files may arise.

How to Take the Backup

Step 1
You have to define some parameters in *postgres.conf* so that: WAL files are on the level 'archive' or higher, archiving of WAL files is activated and a copy command is defined to transfers WAL files to a failsafe location.

```
# collect enough information in WAL files
wal_level = 'archive'
# activate ARCHIVE mode
archive_mode = on
# supply a command to transfer WAL files to a failsafe location (cp, scp, rsync,
 ...)
# %p is the pathname including filename. %f is the filename only.
archive_command = 'scp %p dba@archive_server:/postgres/wal_archive/%f'
```

After the parameters are defined, you must restart the cluster `pg_ctl restart`. The cluster will continuously generate WAL files in its subdirectory *pg_xlog* in concordance with data changes in the database. When it has filled a WAL file and must switch to the next one, it will move the old one to the defined archive location.

Step 2
You must create a *physical* or *base backup* with an operating system utility during the instance is in a special 'backup' mode. In this mode the instance will perform a checkpoint and create some additional files.

```
$ # start psql and set the instance to 'backup' mode, where it creates a
 checkpoint
$ psql -c "SELECT pg_start_backup('AnyBackupLabel');"
$
```

```
$ # copy the cluster's files
$ scp -r $PGDATA dba@archive_server:/postgres/whole_cluster/
$
$ # start psql again and finish 'backup' mode
$ psql -c "SELECT pg_stop_backup();"
$
```

If you like to do so, you can replace the three steps by a single call to the utility *pg_ basebackup*.

Step 3
That's all. All other activities are taken by the instance, especially the continuous copy of completely filled WAL files to the archive location.

How to Recover

To perform a recovery the original *physical* or *base backup* is copied back and the instance is configured to perform recovery during its start.

- Stop the instance - if it is still running.
- Create a copy of the crashed cluster - if you have enough disc space. Maybe, you will need it in a later stage.
- Delete all files of the crashed cluster.
- Recreate the cluster files from the base backup.
- Create a file *recovery.conf* in $PGDATA with a command similar to: restore_command = 'scp dba@archive_server:/postgres/wal_archive/%f %p'. This copy command is the reverse of the command in *postgres.conf*, which saved the WAL files to the archive location.
- Start the instance. During startup the instance will copy and process all WAL files found in the archive location.

The fact, that *recovery.conf* exists, signals the instance to perform a recovery. After a successful recovery this file is renamed.

If you want to recover to some previous point in time prior to the occurrence of the crash (but behind the creation of the backup), you can do so by specifying this point in time in the *recovery.conf* file. In this case the recovery process will stop before processing all archived WAL files. This feature is the origin of the term *Point-In-Time-Recovery*.

In summary the *recovery.conf* file may look like this:

```
restore_command      = 'scp dba@archive_server:/postgres/wal_archive/%f %p'
recovery_target_time = '2016-01-31 06:00:00 CET'
```

2.12 Additional Tools

There is an open source project *Barman* [9], which simplifies the steps of backup and recovery. The system helps you, if you have to manage a lot of servers and instances and it becomes complicate to configure and remember all the details about your server landscape.

2.13 References

Replication is the process of transfering data changes from one or many databases (master) to one or many other databases (standby) running on one or many other nodes. The purpose of replication is

- High Availability: If one node fails, another node replaces him and applications can work continuously.
- Scaling: The workload demand may be too high for one single node. Therefore it is spread over several nodes.

2.14 Concepts

PostgreSQL offers a bunch of largely mutually independent concepts for use in replication solutions. They can be picked up and combined - with only few restrictions - depending on the use case.

Events

- With *Trigger Based Replication* a trigger (per table) starts the transfer of changed data. This technique is outdated and not used.
- With *Log Based Replication* such information is transfered, which describes data changes and is created and stored in WAL files anyway.

Shipping

- *Log-Shipping Replication* (or *File-based Replication*) denotes the transfer of completely filled WAL files (16 MB) from master to standby. This technique is not very elegant and will be replaced by *Streaming Replication* over time.
- *Streaming Replication* denotes the transfer of log records (single change information) from master to standby over a TCP connection.

Primary parameter: 'primary_conninfo' in recovery.conf on standby server.

Format

- In *Physical Format* the transfered WAL records have the same structure as they are used in WAL files. They reflect the strucure of database files including block numbers, VACUUM information and more.

9 Barman `http://www.pgbarman.org/`

- The *Logical Format* is a decoding of WAL records into an abstract format, which is independent from PostgreSQL versions and hardware platforms.

Primary parameter: 'wal_level=logical' in postgres.conf on master server.

Synchronism

- In *Asynchronous Replication* data is transfered to a different node without waiting for a confirmation of its receiving.
- In *Synchronous Replication* the data transfer waits - in the case of a COMMIT - for a confirmation of its successful processing on the standby.

Primary parameter: 'synchronous_standby_names' in postgres.conf on master server.

Standby Mode

- *Hot*: In *Hot Standby Mode* the standby server runs in 'recovery mode', accepts client connections, and processes their read-only queries.
- *Warm*: In *Warm Standby Mode* the standby server runs in 'recovery mode' and doesn't allow clients to connect.
- *Cold*: Although it is not an offical PostgreSQL term, *Cold Standby Mode* can be associated with a not running standby server with log-shipping technique. The WAL files are transfered to the standby but not processed until the standby starts up.

Primary parameter: 'hot_standby=on/off' in recovery.conf on standby server.

Architecture

In opposite to the above categories, the two different architectures are not strictly distinct to each other, eg: if you focus to atomic replication channels of a *Multi-Master* architecture, you will see a *Master/Standby* replication.

- The *Master/Standby* (or *Primary/Slave*) architecture denotes a situation, where one or many standby nodes receive change data from one master node. In such situations standby nodes may replicate the received data to other nodes, so they are master and standby at the same time.
- The *Multi-Master* architecture denotes a situation, where one or many standby nodes receive change data from many master nodes.

2.15 Configuration

This configuration is done in the 'postgres.conf' file (some on the master site, others on the standby site), whereas security configuration is stored in 'pg_hba.conf' (master site), and some important decisions are derived from the existance of 'recovery.conf' (standby site) and its values. The great number of possible combinations of concepts and their correlation to values within the config files may be confusing at the beginning. Therefore we reduce our explanations to the minimal set of values.

Shipping: WAL vs. Streaming

As they are necessary for recovery after a crash, WAL files are generated anyway. If they are used to shipp information to a standby server, it is necessary to add some more information to the files. This is activated by choosing a higher value for wal_level.

```
# WAL parameters on MASTER's postgres.conf
wal_level='archive' | 'hot_standby' # choose one
archive_mode='on'                   # activate the feature
archive_command='scp ...'           # the transfer-to-standby command
```

When you switch the shipping technique to streaming instead of WAL you must not deactivate WAL generating and transfering. For safety reasons you may want to transfer WAL files anyway (to a platform different from the standby server). Therefore you can retain the above parameters in addition to streaming replication parameters.

The streaming activities are initiated by the standby server. When he finds the file 'recovery.conf' during its start up, he assumes that it's neccessay to perform a recovery. In our case of replication he uses nearly the same techiques as in the recovery-from-crash situation. The parameters in 'recovery.conf' advice him to start a so-called WAL receiver process within its instance. This process connects to the master server and initiates a WAL sender process over there. Both exchange information in an endless loop whereas the standby server keeps in 'recovery mode'.

The authorization at the operating system level shall be done by exchanging ssh keys.

```
#  Parameters in the STANDBY's recovery.conf
standby_mode='on'   # activates standby mode
# How to reach the master:
primary_conninfo='user=<replication_dbuser_at_master>
host=<IP_of_master_server> port=<port_of_master_server>
                 sslmode=prefer sslcompression=1 krbsrvname=...'
# This file can be created by the pg_basebackup utility, see below
```

On the master site there must be a privileged database user with the special role REPLICATION:

```
CREATE ROLE <replication_dbuser_at_master> REPLICATION ...;
```

And the master must accept connections from the standby in general and with a certain number of processes.

```
# Allow connections from standby to master in MASTER's postgres.conf
listen_addresses ='<ip_of_standby_server>'         # what IP address(es) to
listen on
max_wal_senders = 5   # no more replication processes/connections than this
number
```

Additionally, authentication of the replication database user must be possible. Please notice that the key word ALL for the database name does not include the authentication of the replication activities. 'Replication' is a key word of its own and must be noted explicitely.

```
# One additional line in MASTER's pg_hba.conf
# Allow the <replication_dbuser> to connect from standby to master
host  replication   <replication_dbuser>   <IP_of_standby_server>/32    trust
```

Now you are ready to start. First, you must start the master. Second, you must transfer the complete databases from the master to the standby. And at last you can start the standby. Just as the replication, the transfer of the databases is initiated at the standby site.

```
pg_basebackup -h <IP_of_master_server> -D main --xlog-methode=stream
--checkpoint=fast -R
```

The utility *pg_basebackup* transfers everythink to the directory 'main' (shall be empty), in this case it uses the streaming methode, it initiates a checkpoint at the master site to enforce consistency of database files and WAL files, and due to the -R flag it generates previous mentioned recovery.conf file.

Format: Physical vs. Logical

The decoding of WAL records from their physical format to a logical format was introduced in PostgreSQL 9.4. The physical format contains - among others - block numbers, VACUUM information and it depends on the used character encoding of the databases. In contrast, the logical format is independent from all these details - conceptually even from the PostgreSQL version. Decoded records are offered to registered streams for consuming.

This logical format offers some great advantages: transfer to dabases at different major release levels, at different hardware achitectures, and even to other writing master. Thus multi-master-architectures are possible. And additionally it's not necessary to replicate the complete cluster: you can pick single database objects.

In release 9.5 the feature is not delivered with core PostgreSQL. You must install some extentions:

```
CREATE EXTENTION btreee_gist;
CREATE EXTENSION bdr;
```

As the feature is relative new, we don't offer details and refer to the documentation[10]. And there is an important project Bi-Directional Replication[11], which is based on this technique.

10 http://www.postgresql.org/docs/current/static/logicaldecoding.html
11 http://bdr-project.org/docs/stable/index.html

Synchronism: synchron vs. asynchron

The default behaviour is asynchronuous replication. This means that transfered data is processed at the standby server without any synchronization with the master, even in the case of a COMMIT. In opposite to this behaviour the master of a synchronuous replication waits for a successfull processing of COMMIT statements at the standby before he confirms it to its client.

The synchronuous replication is activated by the parameter 'synchronous_standby_names'. Its values identify such standby servers, for which the synchronicity shall take place. A '*' indicates all standby server.

```
# master's postgres.conf file
synchronous_standby_names = '*'
```

Standby Mode: hot vs. warm

As long as the standby server is running, he will continuously handle incomming change information and store it in its databases. If there is no neccessarity to process requests from applications, he runs in warm standby mode. This behaviour is enforced in the recovery.conf file.

```
# recovery.conf on standby server
hot_standby = off
```

If he shall allow client connections, he must start in hot standby mode.

```
# recovery.conf on standby server
hot_standby = on
```

To generate enough information on the master site for the standby's hot standby mode, its WAL level must also be hot_standby.

```
# postgres.conf on master server
wal_level = hot_standby
```

2.16 Typical Use Cases

We offer some typical combinations of the above mentioned concepts and show its advantages and disadvantages.

2.16.1 Warm Standby with Log-Shipping

In this situation a master sends information about changed data to a standby using completely filled WAL files (16 MB). The standby continuously processes the incomming information, which means that the changes made on the master are seen at the standby over time.

To build this scenario, you must perform steps, which are very similar to Backup with PITR[12]:

- Take a physical backup exactly as described in Backup with PITR[13] and transfer it to the standby.
- At the master site `postgres.conf` must specify `wal_level=archive;archive_mode=on` and a copy command to transfer WAL files to the standby site.
- At the standby site the central step is the creation of a `recovery.conf` file with the line `standby_mode='on'`. This is a sign to the standby to perform an 'endless recovery process' after its start.
- `recovery.conf` must contain some more definitions: `restore_command`, `archive_cleanup_command`

With this parametrisation the master will copy its completely filled WAL files to the standby. The standby processes the received WAL files by copying the change information into its database files. This behaviour is nearly the same as a recovery after a crash. The difference is, that the recovery mode is not finish after processing the last WAL file, the standby waits for the arrival of the next WAL file.

You can copy the arrising WAL files to a lot of servers and activate warm standby on each of them. Doing so, you get a lot of standbys.

2.16.2 Hot Standby with Log-Shipping

This variant offers a very valuable feature in comparision with the warm standby scenario: applications can connect to the standby and send read requests to him while he runs in standby mode.

To achive this situation, you must increase `wal_level` to `hot_standby` at the master site. This leads to some additional information in the WAL files. And on the standby site you must add `hot_standby=on` in `postgres.conf`. After its start the standby will not only process the WAL files but also accept and response to read-requests from clients.

The main use case for hot standby is load-balancing. If there is a huge number of read-requests, you can reduce the masters load by delegating them to one or more standby servers. This solution scales very good across a great number of parallel working standby servers.

Both scenarios *cold/hot with log-shipping* have a common shortage: The amount of transfered data is always 16 MB. Depending on the frequency of changes at the master site it

12 Chapter 2.11.3 on page 33
13 Chapter 2.11.3 on page 33

can take a long time until the transfer is started. The next chapter shows a technique which does not have this deficiency.

2.16.3 Hot Standby with Streaming Replication

The use of files to transfer information from one server to another - as it is shown in the above log-shipping scenarios - has a lot of shortages and is therefore a little outdated. Direct communication between programs running on different nodes is more complex but offers significant advantages: the speed of communication is incredible higher and in much cases the size of transfered data is smaller. In order to gain these benefits, PostgreSQL has implemented the streaming replication technique, which connects master and standby servers via TCP. This technique adds two additional processes: the *WAL sender* process at the master site and the *WAL receiver* process at the standby site. They exchange information about data changes in the master's database.

The communication is initiated by the standby site and must run with a database user with REPLICATION privileges. This user must be created at the master site and authorized in the master's pg_hba.conf file. The master must accept connections from the standby in general and with a certain number of processes. The authorization at the operating system level shall be done by exchanging ssh keys.

```
Master site:
============

-- SQL
CREATE ROLE <replication_dbuser_at_master> REPLICATION ...;

# postgres.conf: allow connections from standby to master
listen_addresses ='<ip_of_standby_server>'        # what IP address(es) to
 listen on
max_wal_senders = 5   # no more replication processes/connections than this
 number
# make hot standby possible
wal_level = hot_standby

# pg_conf: one additional line (the 'all' entry doesn't apply to replication)
# Allow the <replication_dbuser> to connect from standby to master
host   replication   <replication_dbuser>   <IP_of_standby_server>/32    trust


Standby site:
=============

# recovery.conf (this file can be created by the pg_basebackup utility, see
 below)
standby_mode='on'   # activates standby mode
# How to reach the master:
primary_conninfo='user=<replication_dbuser_at_master_server>
 host=<IP_of_master_server> port=<port_of_master_server>
                 sslmode=prefer sslcompression=1 krbsrvname=...'

# postgres.conf: activate hot standby
hot_standby = on
```

Now you are ready to start. First, you must start the master. Second, you must transfer the complete databases from the master to the standby. And at last you start the standby.

Just as the replication activities, the transfer of the databases is initiated at the standby site.

```
pg_basebackup -h <IP_of_master_server> -D main --xlog-methode=stream
--checkpoint=fast -R
```

The utility *pg_ basebackup* transfers everythink to the directory 'main' (shall be empty), in this case it uses the streaming methode, it initiates a checkpoint at the master site to enforce consistency of database files and WAL files, and due to the -R flag it generates the previous mentioned recovery.conf file.

The activation of the 'hot' standby is done exactly as in the previous use case.

2.17 An Additional Tool

If you have to manage a complex replication use case, you may want to check the open source project 'repmgr'[14]. It supports you to monitor the cluser of nodes or perform a failover.

If you have a table with a very huge amount of data, it may be helpfull to scatter the data to different physical tables with a common data structure. In use cases, where INSERTs and SELECTs concern only one of those tables and DELETEs concern really all rows of another table, you can get great performance benefits from partitioning. Typically this is the case, if there is any timeline for the rows.

Partitioning uses the INHERIT feature. First, you define a master table.

```
CREATE TABLE log (
  id       int not null,
  logdate  date not null,
  message  varchar(500)
);
```

Next, you create the partitions with the same structure as the master and ensure, that only rows within the expected data range can be stored there.

```
CREATE TABLE log_2015_01 (CHECK (logdate >= DATE '2015-01-01' AND logdate < DATE
 '2015-02-01')) INHERITS (log);
CREATE TABLE log_2015_02 (CHECK (logdate >= DATE '2015-02-01' AND logdate < DATE
 '2015-03-01')) INHERITS (log);
...
CREATE TABLE log_2015_12 (CHECK (logdate >= DATE '2015-12-01' AND logdate < DATE
 '2016-01-01')) INHERITS (log);
CREATE TABLE log_2016_01 (CHECK (logdate >= DATE '2016-01-01' AND logdate < DATE
 '2016-02-01')) INHERITS (log);
...
```

It's a good idea to create an index.

```
CREATE INDEX log_2015_01_idx ON log_2015_01 (logdate);
```

14 http://www.repmgr.org/

```
CREATE INDEX log_2015_02_idx ON log_2015_02 (logdate);
...
```

We need a function, which transfers rows into the appropriate partition.

```
CREATE OR REPLACE FUNCTION log_ins_function() RETURNS TRIGGER AS $$
BEGIN
  IF (NEW.logdate >= DATE '2015-01-01' AND NEW.logdate < DATE '2015-02-01' )
 THEN
        INSERT INTO log_2015_01 VALUES (NEW.*);
  ELSIF (NEW.logdate >= DATE '2015-02-01' AND NEW.logdate < DATE '2015-03-01' )
 THEN
        INSERT INTO log_2015_02 VALUES (NEW.*);
  ELSIF ...
    ...
  END IF;
  RETURN NULL;
END;
$$
LANGUAGE plpgsql;
```

The function is called by a trigger.

```
CREATE TRIGGER log_ins_trigger
  BEFORE INSERT ON log
  FOR EACH ROW EXECUTE PROCEDURE log_ins_function();
```

When this is done, new rows mostly will go to the newest partition. And after some years you can drop old partitions as a whole. This dropping shall be done with the command DROP TABLE - not with a DELETE ... command. The DROP command is much faster than the DELETE command as it removes everything in one single step instead of touching every single row. For SELECT commands the query optimizer has the chance to avoid scanning unnecessary tables.

The default behaviour of PostgreSQL is, that all data, indexes, and management information is stored in subdirectories of a single directory. But this approach does not fit always. In some situation you may want to change the storage area of one or more tables: data grows and may blow up partition limits, you may want to use fast devices like a ssd for heavily used tables, etc. . Therefore you need a technique to become more flexible.

Tablespaces offers the possibility to push data on arbitrary directories within your file system.

```
CREATE TABLESPACE fast LOCATION '/ssd1/postgresql/fastTablespace';
```

After the tablespace is defined it can be used in DDL statements.

```
CREATE TABLE t1(col_1 int) TABLESPACE fast;
```

When upgrading the PostgreSQL software, you must take care of the data in the cluster - depending on the question whether it is an upgrade of a major or a minor version. The PostgreSQL version number consists of two or three groups of digits, divided by colons.

The first two groups denotes the major version and the third group (if present) denotes the minor version.

Upgrades within minor versions are simple. The internal data format does not change, so you only need to install the new software while the instance is down.

Upgrades of major versions may lead to incompatiblities of internal data structures. Therefore special actions may become necessary. There are several strategies to overcome the situation. In many cases upgrades of major versions additionally introduce some user-visible incompatibilities, so application programming changes might be required. You should read the release notes carefully.

2.18 pg_upgrade

`pg_upgrade` is a utility which modifies data files and system catalogs according to the needs of the new version. It has two major behaviors: In --link mode files are modified in place, otherwise the files are copied to a new location.

2.19 pg_dumpall

`pg_dumpall` is a standard utility to generate **logical** backups of the cluster. Files generated by `pg_dumpall` are plain text files and thus independent from all internal structures. When modifications of the data's internal structure become necessary (upgrade, different hardware architecture, different operating system, ...), such logical backups can be used for the data transfer from the old to the new system.

2.20 Replication

The Slony replication system offers the possiblity to transfer data over different major versions. Using this, you can switch a replication slave to the new master within a very short time frame.

PostgreSQL offers replication in *logical streaming* format. With the actual version 9.5 this feature is restricted to the same versions of master and standby server, but it is planned to extend it for use in a heterogenuous server landscape.

3 See also

- Wikibook SQL[1]
- Wikipedia PostgreSQL[2]
- Converting MySQL to PostgreSQL[3]

1 https://en.wikibooks.org/wiki/SQL
2 https://en.wikipedia.org/wiki/PostgreSQL
3 https://en.wikibooks.org/wiki/Converting%20MySQL%20to%20PostgreSQL

4 External links

- PostgreSQL Homepage[1]
- PostgreSQL Documentation[2]
- PostgreSQL Wiki[3]

1 http://www.postgresql.org/about/
2 http://www.postgresql.org/docs/current/static/index.html
3 https://wiki.postgresql.org/wiki/

5 License

1. REDIRECT Wikibooks:GNU Free Documentation License[1]

1 https://en.wikibooks.org/wiki/GNU%20Free%20Documentation%20License

6 Contributors

Edits	User
2	Adrignola[1]
1	Aya[2]
1	Darklama[3]
3	Derbeth[4]
1	Guanaco[5]
34	Kelti[6]
5	Maveric149[7]
1	Robert Horning[8]
1	SB Johnny[9]
4	Whiteknight[10]
2	Withinfocus[11]

1 https://en.wikibooks.org/wiki/User:Adrignola
2 https://en.wikibooks.org/wiki/User:Aya
3 https://en.wikibooks.org/wiki/User:Darklama
4 https://en.wikibooks.org/wiki/User:Derbeth
5 https://en.wikibooks.org/wiki/User:Guanaco
6 https://en.wikibooks.org/wiki/User:Kelti
7 https://en.wikibooks.org/wiki/User:Maveric149
8 https://en.wikibooks.org/wiki/User:Robert_Horning
9 https://en.wikibooks.org/wiki/User:SB_Johnny
10 https://en.wikibooks.org/wiki/User:Whiteknight
11 https://en.wikibooks.org/wiki/User:Withinfocus

List of Figures

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/deed.en
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.
- LFK: Lizenz Freie Kunst. http://artlibre.org/licence/lal/de
- CFR: Copyright free use.

- EPL: Eclipse Public License. http://www.eclipse.org/org/documents/epl-v10.php

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[12]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

12 Chapter 7 on page 59

1	Kelti[13], Kelti[14]	
2	Kelti[15], Kelti[16]	
3	Kelti[17], Kelti[18]	
4	Kelti[19], Kelti[20]	

13 http://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
14 https://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
15 http://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
16 https://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
17 http://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
18 https://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
19 http://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1
20 https://commons.wikimedia.org/w/index.php?title=User:Kelti&action=edit&redlink=1

7 Licenses

7.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

7.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

7.3 GNU Lesser General Public License

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.