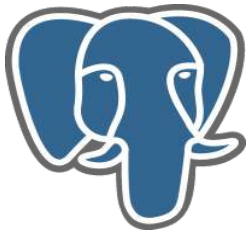
PostgreSQL, en driftssikker og fullverdig databasemotor

Oktober 2005 – NUUG - Høyskolen i Oslo, HiO

Rafael Martinez, USIT, UiO
r.m.guerrero@usit.uio.no



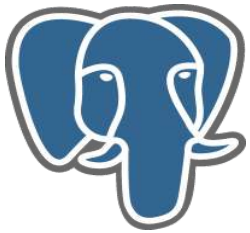
PostgreSQL

- History
- Features
- Administration
- Tuning
- Replication

Requirements

- Unix system
- Database in a production system
- Dedicated database server

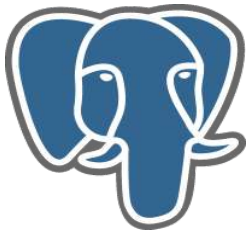
History



PostgreSQL

Ingres 1977-1985 – *The beginning*

- **IN**teractive **G**raphics **RE**trieval **S**ystem
- Proof of concept for relational databases.
- Michael Stonebraker, professor at Berkeley, California.
- Established the company RTI in 1982.
- Ingres was bought by Computer Associates in 1994
- Ingres -> Informix, NonStop SQL, Sybase -> Microsoft SQL server



PostgreSQL

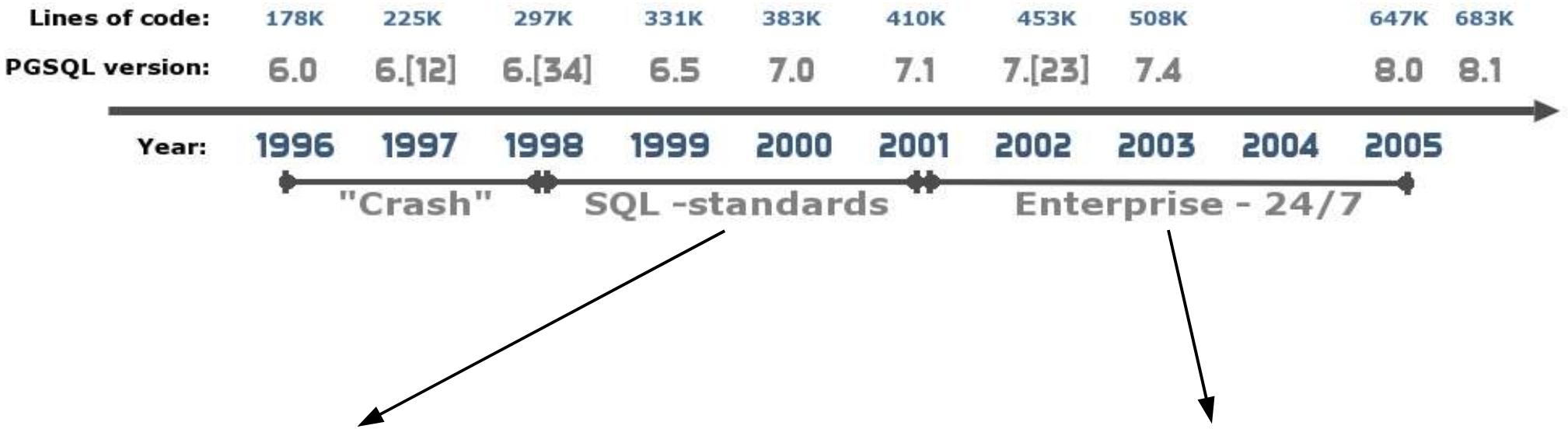
Postgres 1986-1994 – A*s in "after Ingres"*

- A project meant to break new ground in database concepts.
- “Objects relational” technologies.
- POSTQUEL query language.
- Rules, procedures, extensible types with indices and object-relational concepts are introduced.

- Code base of Ingres not used as a basis for Postgres.
- Commercialized to become Illustra.
- Bought by Informix.
- Informix was bought by IBM in 2001.
- Informix <-> DB2?

Postgres95 1994-1995 – New life in the OpenSource world

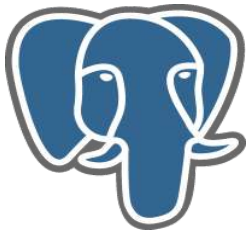
- Two Ph.D. students from Stonebraker's lab, Andrew Yu and Jolly Chen started Postgres95.
- Postgres' POSTQUEL query language replaced with with an extended subset of SQL.
- Departed from academia to a new life in the open source world with a group of dedicated developers outside of Berkeley.
- Establishment of the PostgreSQL Global Development Team.
- Released as PostgreSQL 6.0 in 1996.



- Multiversion Concurrency Control (MVCC)
- Important SQL features
- Improved build-in types
- Speed

- Improved performance
- Improved administration & maintenance
- 24/7 ready

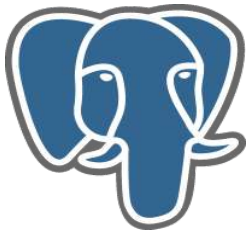
Features



PostgreSQL

Overall features

- ORDBMS
- Minimal administration
- Stability
- Excellent performance
- Data integrity (ACID)
- Portable
- Extensible
- BSD license

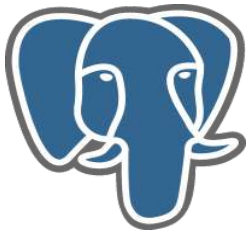


PostgreSQL

General features

- Fully ACID compliance (Atomic, Consistent, Isolated, Durable)
- ANSI SQL 92/99/2003 compliance
- Foreign keys (referential integrity)
- Multi-version concurrency control (MVCC)
- Point-in-time recovery PITR
- Tablespaces
- Savepoints
- Functional and partial indices
- Native SSL support
- Native Kerberos support

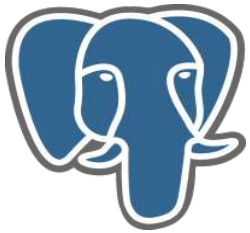
- Linux, UNIX (AIX, BSD, HP-UX, SGI, IRIX, Mac OS X, Solaris, SunOS, Tru64), BeOS, Windows.



PostgreSQL

Development features

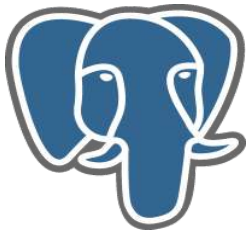
- Stored procedures, PL/pgSQL, PL/Tcl, PL/Perl, PL/Python
- Native interfaces for ODBC, JDBC, C, C++, PHP, Perl, TCL, ECPG, Python and Ruby
- User defined data types, functions and operators
- Open and documented API.



PostgreSQL

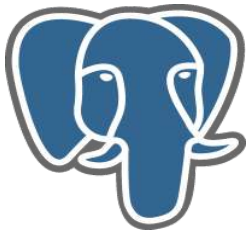
SQL features

- Rules
- Views
- Triggers
- Cursors
- Sequences
- Inheritance
- Outer joins
- Sub-selects
- Support for UNION (ALL/EXCEPT)
- Unicode



PostgreSQL

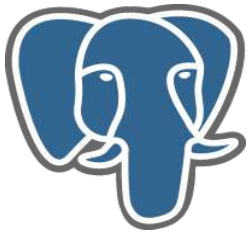
Administration



PostgreSQL

Things we are not going to talk about

- PostgreSQL installation
- PostgreSQL cluster initialization
- Create users
- Create databases
- Create tables, indexes, etc
- Programming



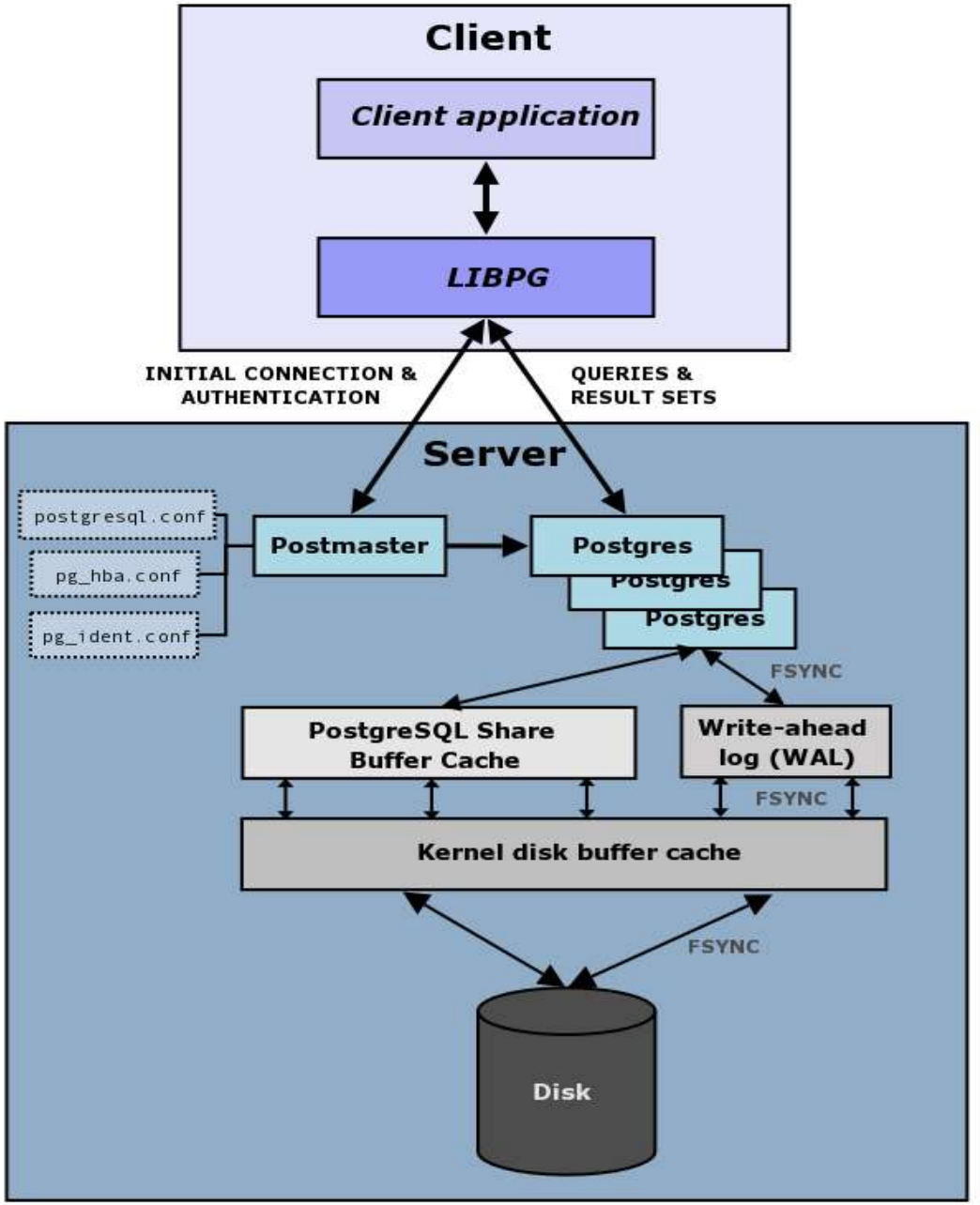
PostgreSQL

Things we are going to talk about

- PostgreSQL overview
- Data directory layout
- pg_hba.conf
- postgresql.conf
- psql ++
- Tablespaces
- Backup / PITR
- Vacuum / Analyze
- System tables

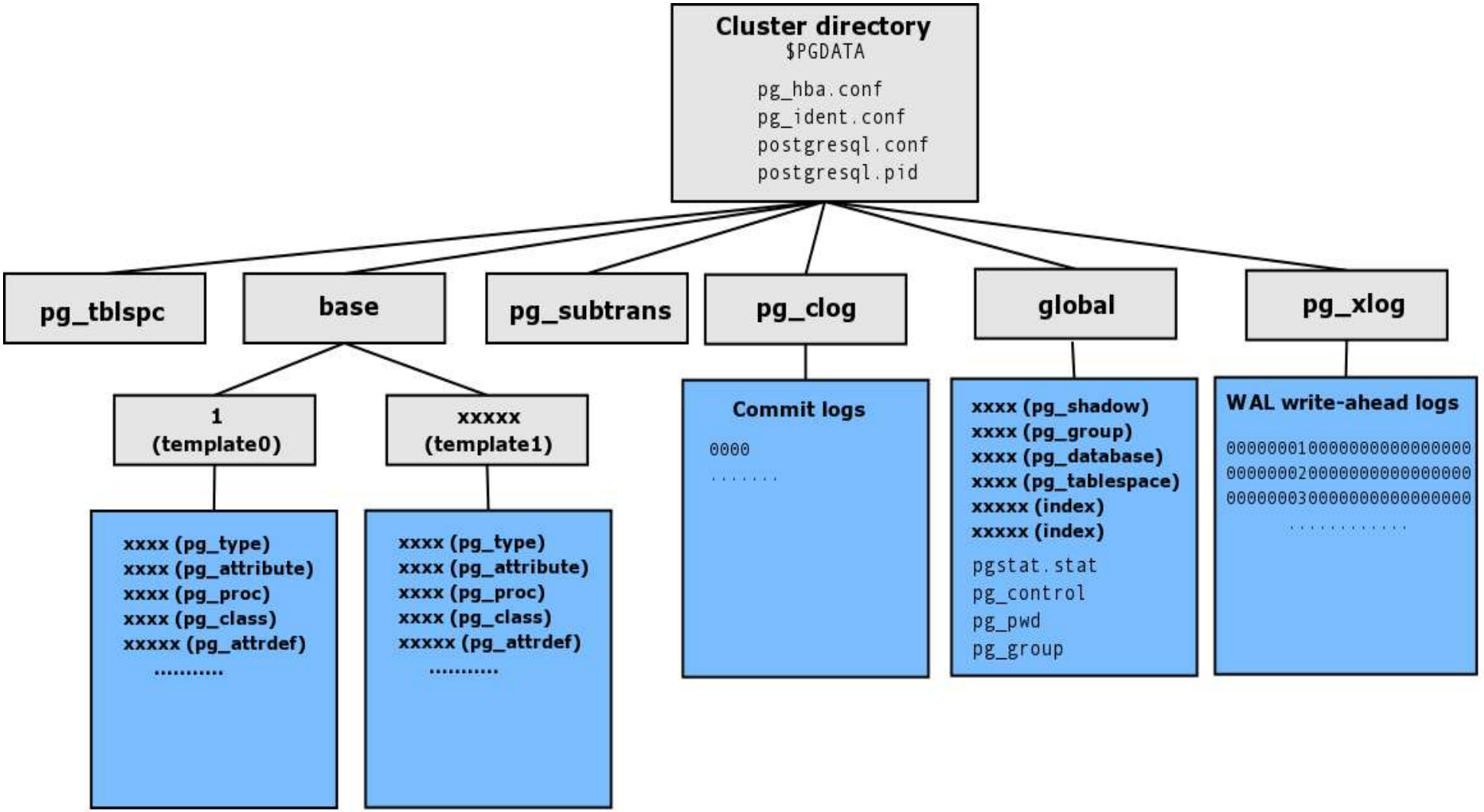
PostgreSQL

- It uses a multi-process model
- It does not use multi-threading.



PostgreSQL

Data directory layout



$PGDATA/pg_hba.conf

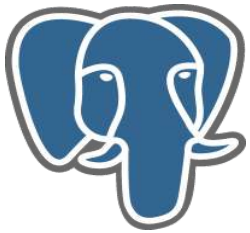
The PostgreSQL Client Authentication Configuration file controls:

- Which hosts are allowed to connect
- How clients are authenticated
- Which PostgreSQL user names they can use
- Which databases they can access

A record may have one of these seven formats:

```
local       database  user  authentication-method   [authentication-option]

host        database  user  CIDR-address  authentication-method  [authentication-option]
hostssl     database  user  CIDR-address  authentication-method  [authentication-option]
hostnossl   database  user  CIDR-address  authentication-method  [authentication-option]

host        database  user  IP-address  IP-mask  authentication-method  [authentication-option]
hostssl     database  user  IP-address  IP-mask  authentication-method  [authentication-option]
hostnossl   database  user  IP-address  IP-mask  authentication-method  [authentication-option]
```

authentication-method: trust, reject, md5, crypt, password, krb4, krb5, ident, or pam

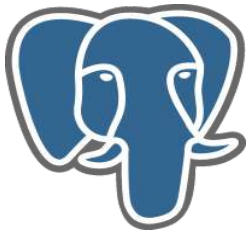


PostgreSQL

$PGDATA/postgresql.conf (I)

The postgreSQL configuration file defines configuration parameters

- Connection and authentication settings
- Resource consumption
- Write Ahead Log (WAL)
- Query planning
- Error reporting and logging
- Runtime statistics
- Client connection defaults
- Lock management
- Version and platform compatibility

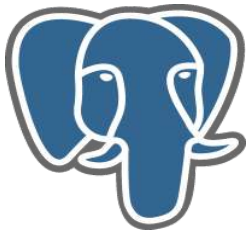


PostgreSQL

$PGDATA/postgresql.conf (II)

- Many configuration parameters with full documentation
- Default values are not good for a production system
- Minimum list of parameters that should be activated or changed

```
listen_addresses
max_connections
superuser_reserved_connections

share_buffers
work_mem
maintenance_work_mem

wal_buffers
checkpoint_segments

max_fsm_pages
effective_cache_size

log_directory
log_filename

stats_start_collector
stats_command_string
stats_block_level
stats_row_level
stats_reset_on_server_start
```



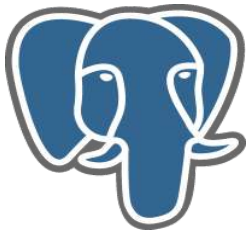
PostgreSQL

psql – PostgreSQL interactive terminal (I)

```
Usage:
  psql [OPTIONS]... [DBNAME [USERNAME]]

General options:
  -d DBNAME       specify database name to connect to (default: "postgres")
  -c COMMAND      run only single command (SQL or internal) and exit
  -f FILENAME     execute commands from file, then exit
  -l              list available databases, then exit
  -v NAME=VALUE   set psql variable NAME to VALUE
  -X              do not read startup file (~/.psqlrc)
  --help          show this help, then exit
  --version       output version information, then exit

Input and output options:
  -a              echo all input from script
  -e              echo commands sent to server
  -E              display queries that internal commands generate
  -q              run quietly (no messages, only query output)
  -o FILENAME     send query results to file (or |pipe)
  -n              disable enhanced command line editing (readline)
  -s              single-step mode (confirm each query)
  -S              single-line mode (end of line terminates SQL command)

Output format options:
  -A              unaligned table output mode (-P format=unaligned)
  -H              HTML table output mode (-P format=html)
  -t              print rows only (-P tuples_only)
  -T TEXT         set HTML table tag attributes (width, border) (-P tableattr=)
  -x              turn on expanded table output (-P expanded)
  -P VAR[=ARG]    set printing option VAR to ARG (see \pset command)
  -F STRING       set field separator (default: "|") (-P fieldsep=)
  -R STRING       set record separator (default: newline) (-P recordsep=)

Connection options:
  -h HOSTNAME     database server host or socket directory (default: "local socket")
  -p PORT         database server port (default: "5432")
  -U NAME         database user name (default: "postgres")
  -W              prompt for password (should happen automatically)
```

psql – PostgreSQL interactive terminal (II)

```
-bash-2.05b$ psql template1
Welcome to psql 8.0.4, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

template1=#
```



PostgreSQL

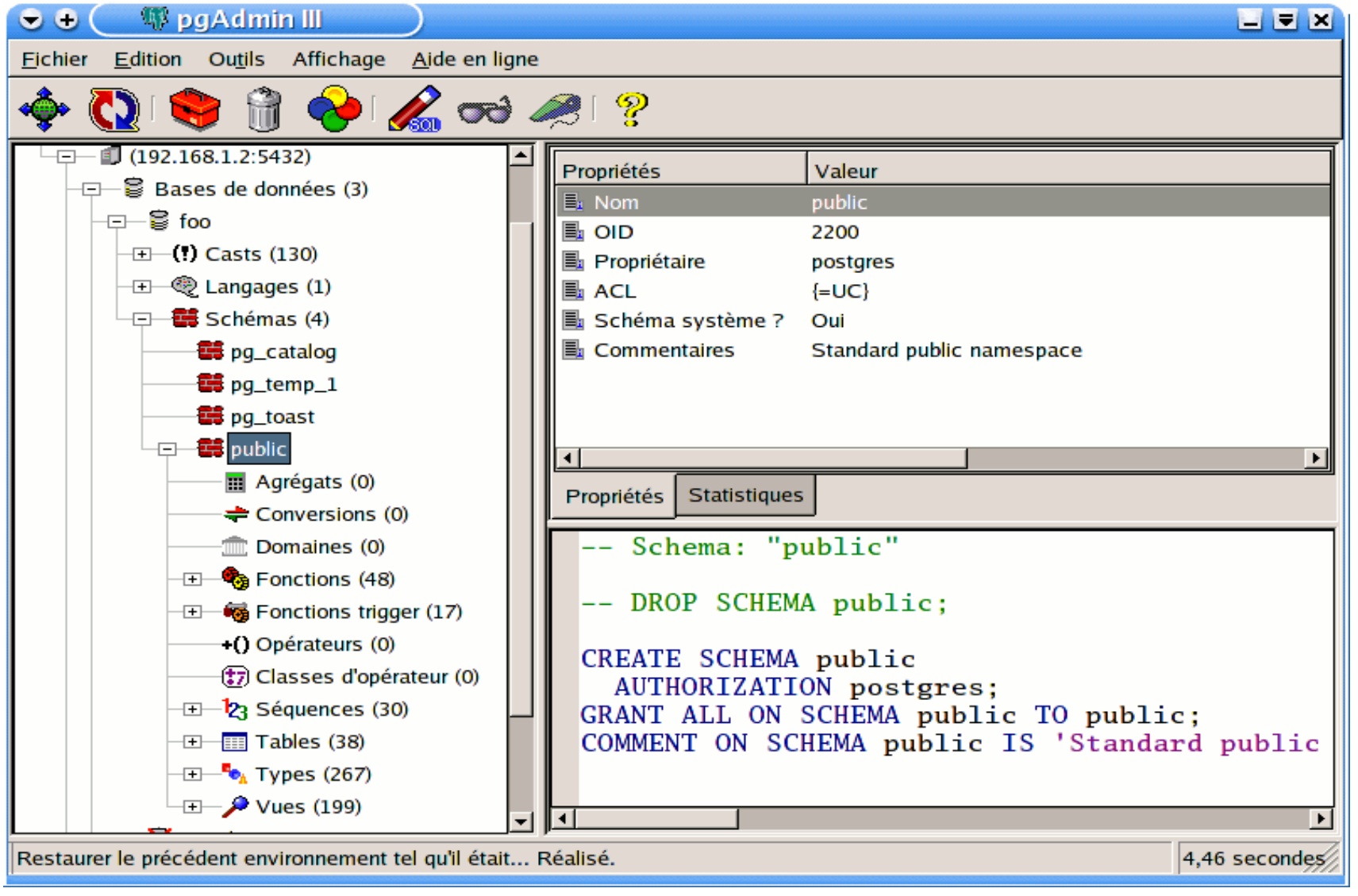
psql – PostgreSQL interactive terminal (III)

```
template1=# \?
```

```
General
  \c[onnect] [DBNAME|- [USER]]
                 connect to new database (currently "template1")
  \cd [DIR]      change the current working directory
  \copyright     show PostgreSQL usage and distribution terms
  \encoding [ENCODING]
                 show or set client encoding
  \h [NAME]      help on syntax of SQL commands, * for all commands
  \q             quit psql
  \set [NAME [VALUE]]
                 set internal variable, or list all if no parameters
  \timing        toggle timing of commands (currently off)
  \unset NAME    unset (delete) internal variable
  \! [COMMAND]   execute command in shell or start interactive shell

Query Buffer
  \e [FILE]      edit the query buffer (or file) with external editor
  \g [FILE]      send query buffer to server (and results to
                 file or |pipe)
  \p             show the contents of the query buffer
  \r             reset (clear) the query buffer
  \s [FILE]      display history or save it to file
  \w FILE        write query buffer to file

Input/Output
  \echo [STRING] write string to standard output
  \i FILE        execute commands from file
  \o [FILE]      send all query results to file or |pipe
  \qecho [STRING]
                 write string to query output stream (see \o)

Informational
  \d [NAME]      describe table, index, sequence, or view
  \d{t|i|s|v|S} [PATTERN] (add "+" for more detail)
                 list tables/indexes/sequences/views/system tables
  \da [PATTERN]  list aggregate functions
  \db [PATTERN]  list tablespaces (add "+" for more detail)
  \dc [PATTERN]  list conversions
  \dC            list casts
  \dd [PATTERN]  show comment for object
  \dD [PATTERN]  list domains
  \df [PATTERN]  list functions (add "+" for more detail)
  \dg [PATTERN]  list groups
  \dn [PATTERN]  list schemas (add "+" for more detail)
  \do [NAME]     list operators
  \dl            list large objects, same as \lo_list
  \dp [PATTERN]  list table, view, and sequence access privileges
  \dT [PATTERN]  list data types (add "+" for more detail)
  \du [PATTERN]  list users
  \l             list all databases (add "+" for more detail)
  \z [PATTERN]   list table, view, and sequence access privileges
                 (same as \dp)

Formatting
  \a             toggle between unaligned and aligned output mode
  \C [STRING]    set table title, or unset if none
  \f [STRING]    show or set field separator for unaligned query output
  \H             toggle HTML output mode (currently off)
  \pset NAME [VALUE]
                 set table output option
                 (NAME := {format|border|expanded|fieldsep|footer|null|
                 recordsep|tuples_only|title|tableattr|pager})
  \t             show only rows (currently off)
  \T [STRING]    set HTML <table> tag attributes, or unset if none
  \x             toggle expanded output (currently on)

Copy, Large Object
  \copy ...      perform SQL COPY with data stream to the client host
  \lo_export LOBOID FILE
  \lo_import FILE [COMMENT]
  \lo_list
  \lo_unlink LOBOID    large object operations
```

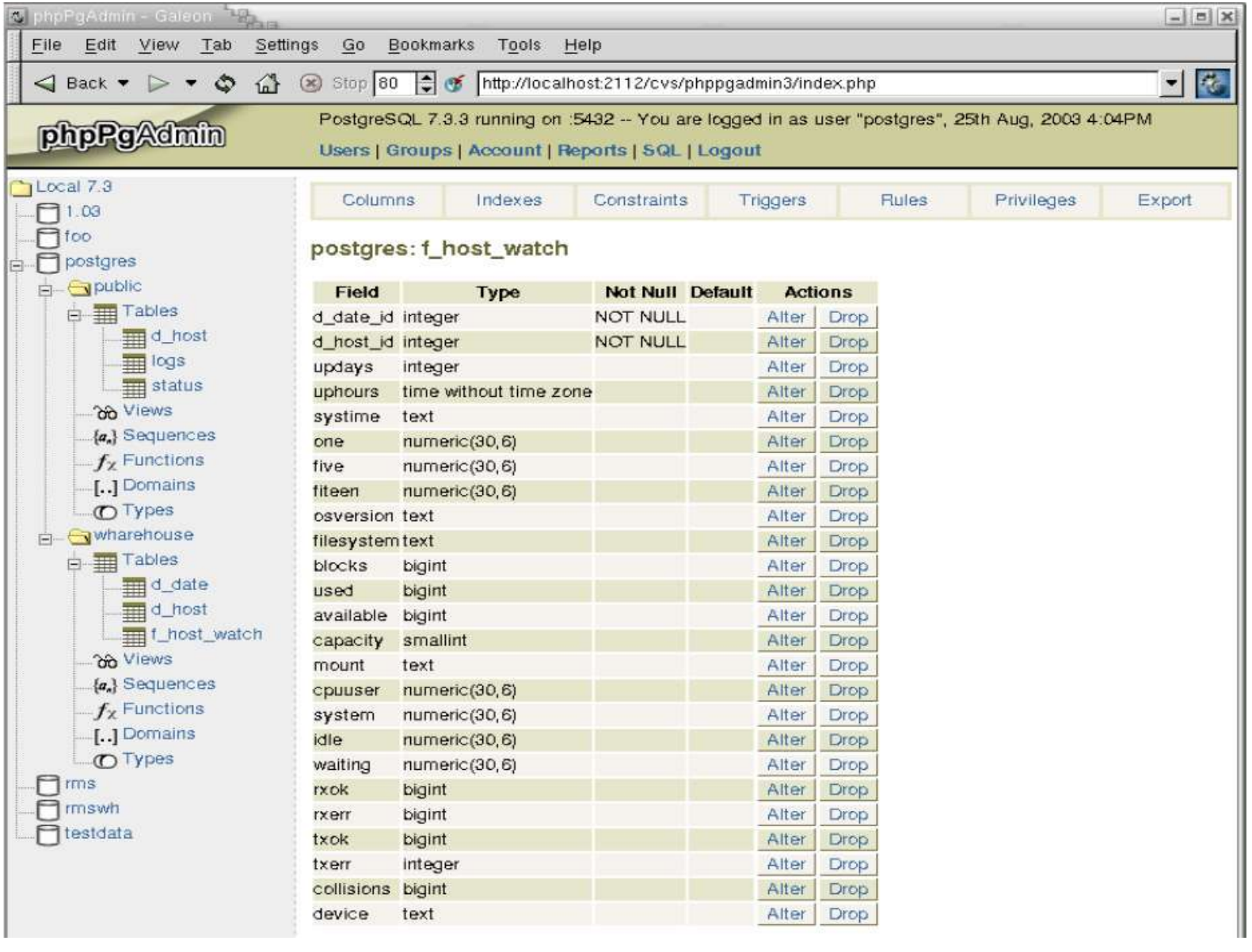
PostgreSQL

pgAdmin III



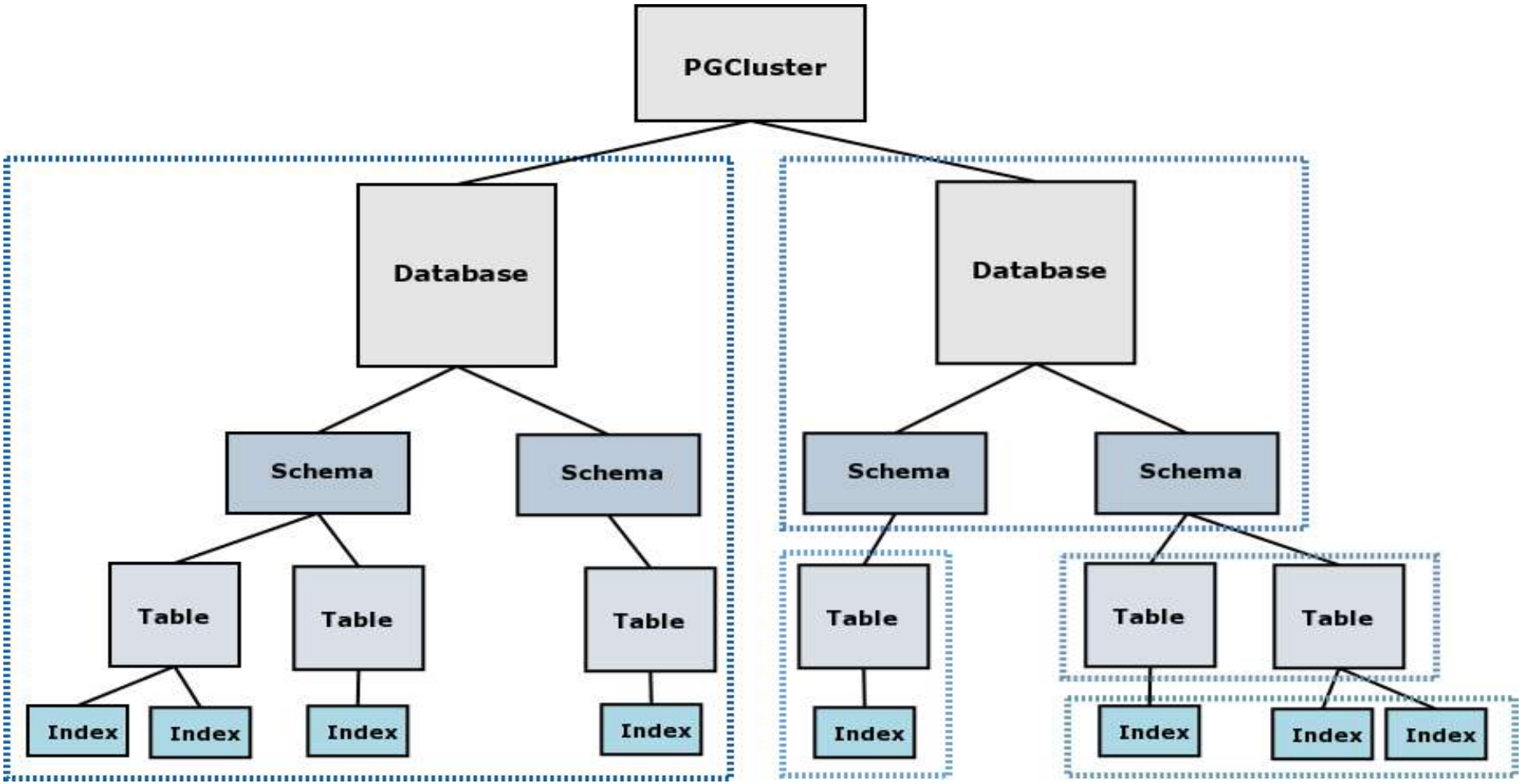
PostgreSQL

phpPgAdmin



PostgreSQL

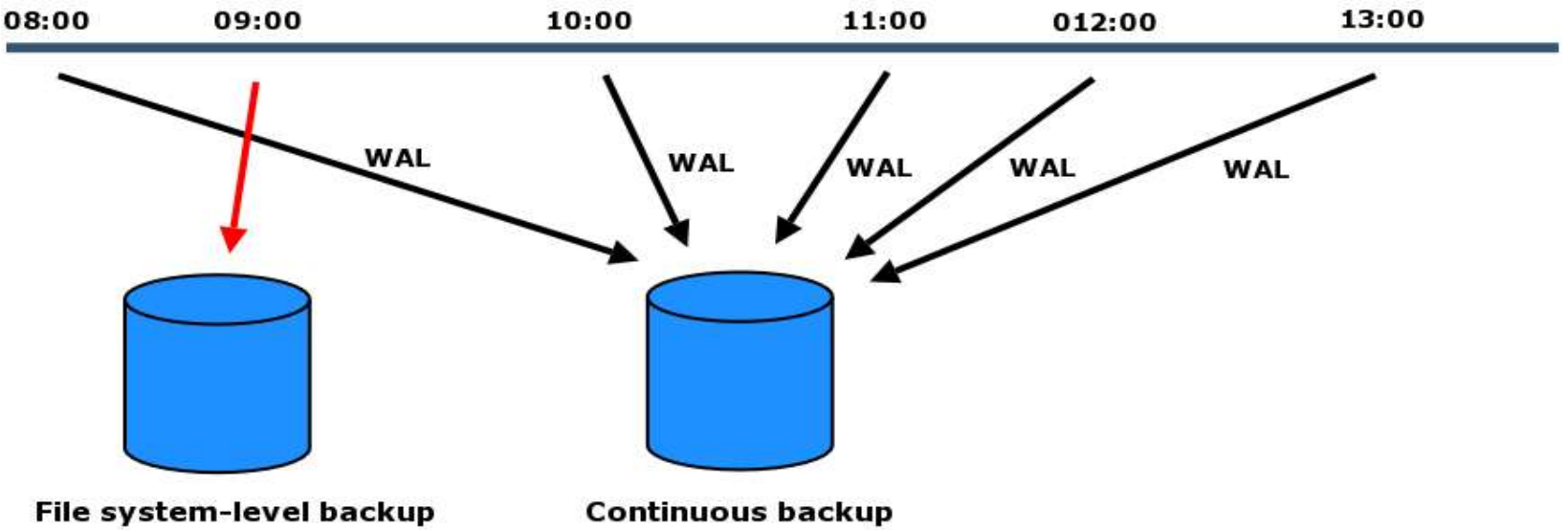
Tablespaces - \db+



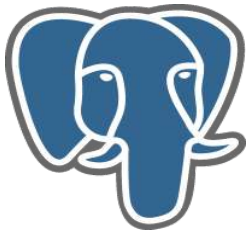
- Define locations in the file system where Databases, tables and indexes can be stored
- Control the disk layout of a PostgreSQL installation
- Can be use to optimize performance

PostgreSQL

PITR – Point In Time Recovery



- Hot backup
- Combines a file-system-level backup with backup of WAL files
- The file-system-level backup can be inconsistent
- Only restoration of an entire database cluster can be done
- Enables recover to the time of crash or an arbitrary chosen point in time since last file-system-level backup
- More difficult to administrate



PostgreSQL

Cold / Hot Backup

- **File system-level**

  - Cold backup
  - Tar, cpio while shutdown
  - File system snapshot (inconsistent?)
  - rsync -> shutdown -> rsync -> start

- **pg_dump/pg_dumpall**

  - Hot Backup
  - Extract a schema/data/database or DB cluster into a script/archive file
  - Consistent backup (MVCC)
  - Non blocking job (read/write)

$PGDATA and backup files should be in different disk systems to avoid loss of data

PostgreSQL

Vacuum / Analyze (non blocking)

Original heap
with expired
rows identified

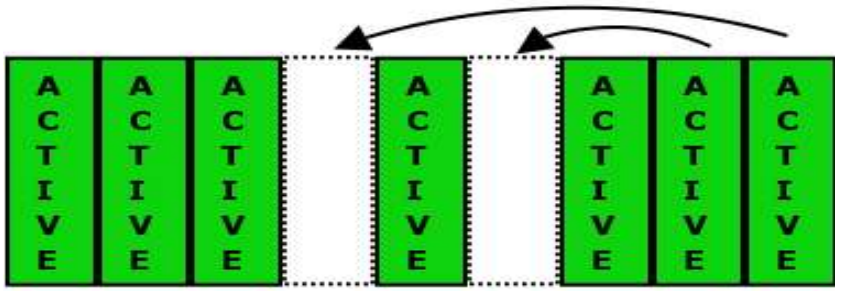
Space reclaimed
for reuse without
truncating the file

- **analyze** updates the data statistics used by the PostgreSQL query planner
- It can be executed alone or together with vacuum (**vacuum analyze**)

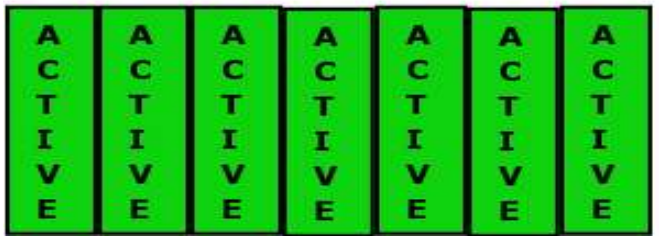
PostgreSQL

Vacuum full (blocking)

Original heap with expired rows identified

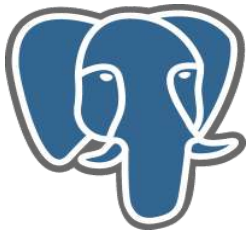


Move trailing rows into expired slots



Truncate file

Vacuum prevents also *transaction ID wraparound* failures after 4 billions (4×10^9) transactions



PostgreSQL

System tables - \dS

- pg_shadow
- pg_settings
- pg_locks
- pg_tablespace
- pg_stat_activity
- pg_stat_*
- pg_statio_*
- pg_class
- .....

Example 1:

```
test001=# SELECT * from pg_shadow ;

 usename  | usesysid | usecreatedb | usesuper | usecatupd |              passwd                | valuntil | useconfig
----------+----------+-------------+----------+-----------+------------------------------------+----------+-----------
 postgres |        1 | t           | t        | t         |                                    |          |
 pgadmin  |      100 | f           | f        | f         | md55cd31c25de000c28135d138df5690e21 |          |
 rafael   |      101 | f           | f        | f         | md55cd31c76f9470c2abcd8636df5cc6381 |          |
 ola      |      102 | f           | f        | f         | md55cd31c76f94753746bbbbbaa54870e21 |          |
 tomas    |      103 | f           | f        | f         | md55cd31c792637a34bd3234aaadb720e21 |          |
(5 rows)
```

Example 2:

```
ps auxww | grep ^postgres
```

```
postgres 18260  0.0  0.7 370440 14428 pts/1  S    15:30   0:00 /usr/local/bin/postmaster
postgres 18261  0.0  0.0   6036  2036 pts/1  S    15:30   0:00 postgres: logger process
postgres 18263  0.0  0.8 371044 18204 pts/1  S    15:30   0:00 postgres: writer process
postgres 18264  0.0  0.1   7036  2472 pts/1  S    15:30   0:00 postgres: stats buffer process
postgres 18265  0.0  0.1   6748  2712 pts/1  S    15:30   0:00 postgres: stats collector process
postgres 17515  0.0  0.7 371240 10244 pts/1  S    20:39   0:00 postgres: postgres template1 [local] idle
```

master server process

logging

background buffer writer

statistics collector

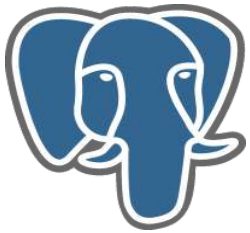
client connection

postgres: *user database host activity*

```
template1=# SELECT * from pg_stat_activity ;
 datid |  datname  | procpid | usesysid | usename  | current_query |          query_start
-------+-----------+---------+----------+----------+---------------+-------------------------------
     1 | template1 |   17515 |        1 | postgres | <IDLE>        | 2005-10-08 21:15:04.245929+02
```

```
template1=# SELECT * from pg_locks ;
 relation | database | transaction |  pid  |      mode       | granted
----------+----------+-------------+-------+-----------------+---------
    16839 |        1 |             | 17515 | AccessShareLock | t
          |          |     9339816 | 17515 | ExclusiveLock   | t
```

```
template1=# SELECT oid,relname from pg_class where oid ='16839';
  oid  | relname
-------+----------
 16839 | pg_locks
```



PostgreSQL

Tuning

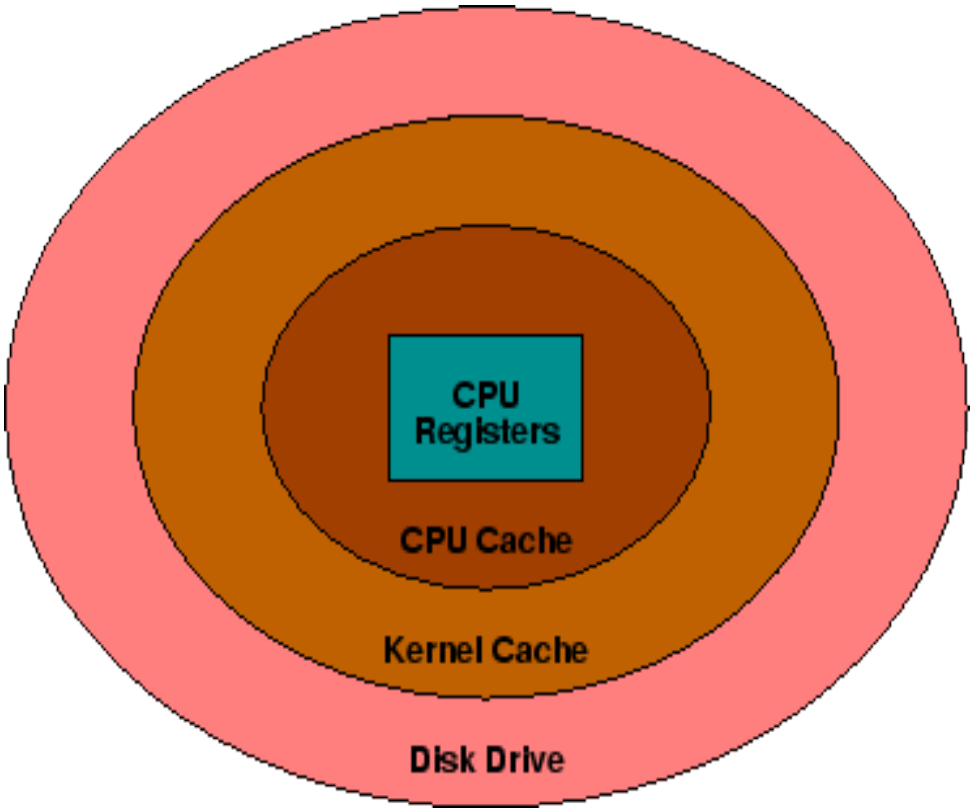
Database performance tuning

We can do two things to improve performance

Improve the use of the CPU memory and disk drives

Optimize the queries sent to the database. Use EXPLAIN to obtain information about a query.

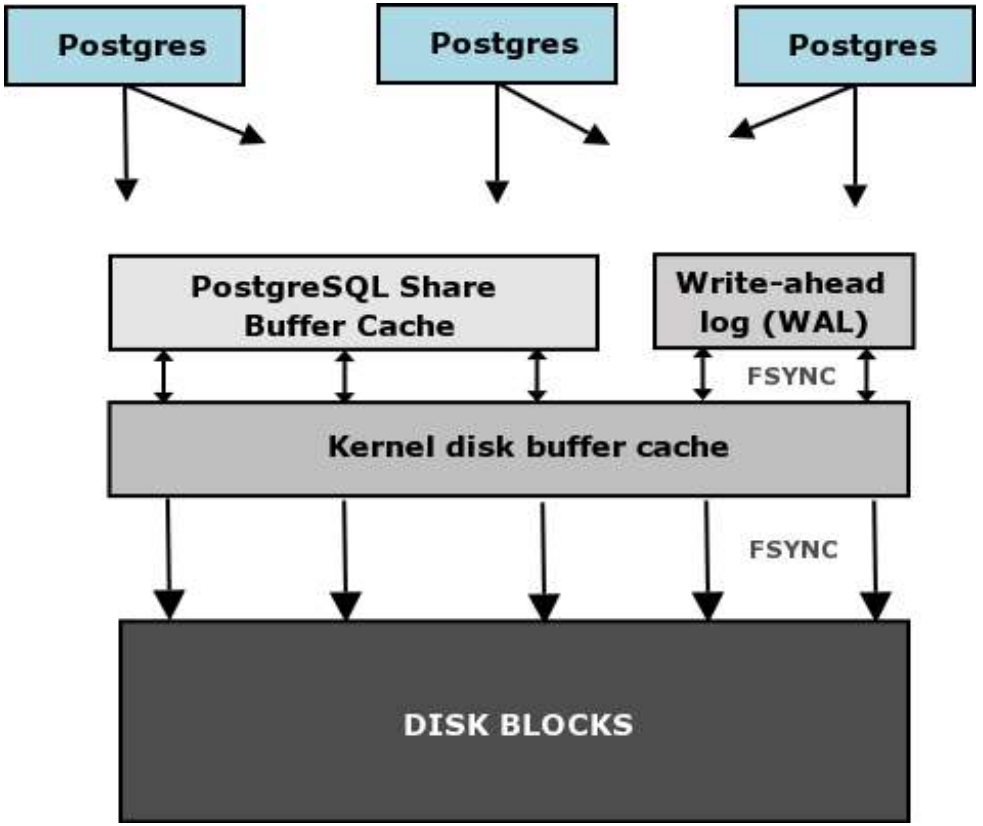
PostgreSQL



- The most frequently used information is stored next to the CPU
- Less frequently accessed information is stored farther away and brought to the CPU as needed

PostgreSQL

Shared Buffer Cache (`shared_buffers`)



- Large enough to hold most commonly accessed tables
- Small enough to avoid swap pagein activity
- Complex calculation of total RAM, database size, number of connections, and query complexity

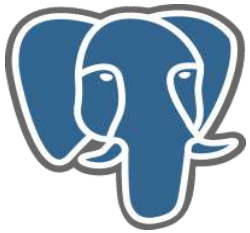
- Quick rule --> between 1 000 and 50 000 buffers (8Kb each -> ca.8-400Mb)
- My default is 15% of available RAM
- Never more than 1/3 of available RAM

```
Shared memory and semaphores values (example: linux):

kernel.shmmax = ((250 + (8*shared_buffers) + (14*max_connections))*2*1024)
kernel.shmall = ((250 + (8*shared_buffers) + (14*max_connections))*2*1024)
```

Sort memory batch size (`work_mem`)

- Allocated per operation (non-shared) (ORDER BY, DISTINCT, merge joins, hash joins, IN subqueries)
- Defines a celling on the amount of memory to use before using disk
- Can be adjusted upwards/downwards depending on amount of available RAM, query size and number of connections
- Can be set per connection at query time
- Monitor the PostgreSQL temp-files in `$PGDATA/base/<DB_OID>/pgsql_tmp`

- 2-4% of available RAM if we have just a few big sessions.
- My default is 8192 (size in Kb)

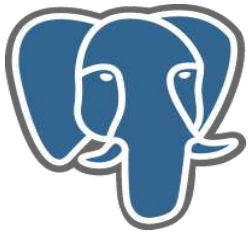


PostgreSQL

Maintenance operation's memory (`maintenance_work_mem`)

- Maximum amount of memory to be used in maintenance operations (VACUUM, ANALIZE, CREATE INDEX, ALTER TABLE, ADD FOREIGN KEY)
- Raise it with large databases and enough RAM
- Can be allocated at runtime so we can increase it temporarily.

- 50-75% of on-size disk of your larger table or index.
- 32-256Mb if this can not be determined
- My default is 131072 (size in Kb)



PostgreSQL

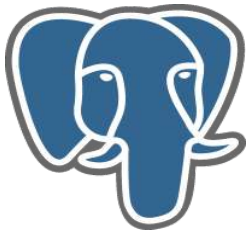
Free space map (`max_fsm_pages`)

- Sizes the register which tracks partially empty data pages for population with new data
- If set right, makes VACUUM faster and removes the need for VACUUM FULL or REINDEX
- Should be slightly more than the total number of data pages which will be touched by updates and deletes between vacuums

- From VACUUM VERBOSE ANALYZE, example:

```
[....]
INFO:  free space map: 197 relations, 30363 pages stored; 33568 total pages needed
DETAIL:  Allocated FSM size: 2000 relations + 40000 pages = 354 kB shared memory.
```

This is saying that we need 33568 fsm_pages slots to remember every single page that has a useful amount of free space.

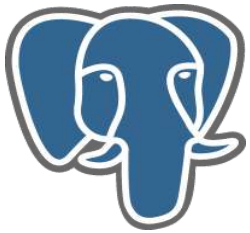


PostgreSQL

Planner cost constants (`effective_cache_size`)

- Tells the query planner the largest possible database object that could be expected to be cached
- Used by the optimizer to estimate the size of the kernel's disk buffer cache used by PostgreSQL

- Around 2/3 in a dedicated server
- My default is 50% of available RAM (8Kb each)



PostgreSQL

Write Ahead Log (`wal_buffers / checkpoints_segments`)

- **`wal_buffers`** defines the number of disk-page buffers allocated in shared memory for WAL data
- Needs only to be large enough to hold the amount of WAL data generated by one typical transaction
- Between 16-64 buffers to be sure

- **`checkpoints_segments`** defines the maximum distance between automatic WAL checkpoints, in log file segments (each segment is normally 16 megabytes)
- The most effective setting for dealing with large updates, data loading, and heavy OLTP activity
- Heavy loads --> 8-16
- Very large write loads (several Gb of data) --> up to 128-256
- My default is 32
- Check logfile for warnings.
- It requires a significant amount of disk space for the `$PGDATA/pg_xlog` directory:
  `( 2 x checkpoint_segments + 1 ) x 16MB`

Putting the database transaction log `$PGDATA/pg_xlog` on its own dedicated disk resource, will make a big difference in performance on databases with high write activity.

pg_stat* system tables to obtain information

EXAMPLE DATABASE: webmail_stats
PERIOD: 30 days
Statistics for webmail at UiO

```
webmail_stats=# SELECT * from pg_stat_user_tables where schemaname = 'public';
```

relid	schemaname	relname	seq_scan	seq_tup_read	idx_scan	idx_tup_fetch	n_tup_ins	n_tup_upd	n_tup_del
22516149	public	users_stats	593	28 962	0	0	96	0	93
22516147	public	login_stats	983	32 294	0	0	64	128	62
371003	public	login_hist	6 205	48 517 780 671	729 879	36 563 060 420	978 195	0	445 161

```
webmail_stats=# SELECT * from pg_statio_user_tables where schemaname = 'public';
```

relid	schemaname	relname	heap_blks_read	heap_blks_hit	idx_blks_read	idx_blks_hit
22516147	public	login_stats	442	1 142	3 500	490
371003	public	login_hist	3 046 174 877	34 505 800 539	123 731 810	71 110 729
22516149	public	users_stats	336	632	1 580	310

From disk:

$3\ 046\ 174\ 877$ blocks x 8Kb/block = $2.436939902 \times 10^{10}$ / 1024 / 1024 = 23 240 Gb --> 32 Gb/hour

$123\ 731\ 810$ blocks x 8Kb/block = 9.89854480×10^{8} / 1024 / 1024 = 944 Gb --> 1.3 Gb/hour

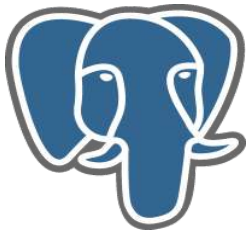
33.3 Gb/hour

From RAM:

$34\ 505\ 800\ 539$ blocks x 8Kb/block = $2.760464042 \times 10^{11}$ / 1024 / 1024 = 263 258 Gb --> 365 Gb/hour

$71\ 110\ 729$ blocks x 8Kb/block = 5.68885832×10^{8} / 1024 / 1024 = 542 Gb --> 0.7 Gb/hour

365.7 Gb/hour



PostgreSQL

General performance tips

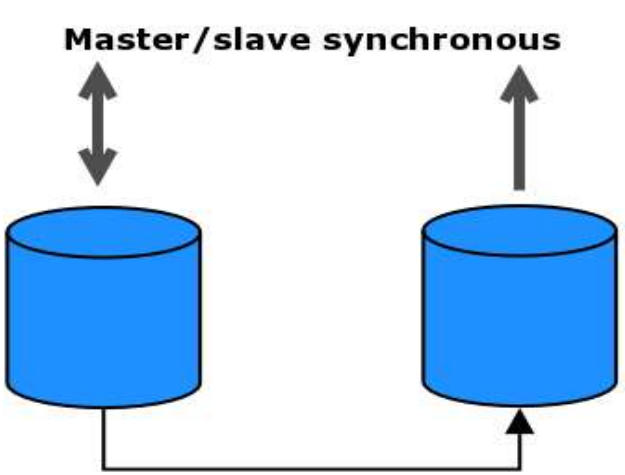
- Run ANALYZE / VACUUM ANALYZE often
- High-performance disk arrays > RAM > CPU
- More disks == better --> Use tablespaces
- RAID 1+0 / 0+1 > RAID 5
- Separate the Transaction Log from the Database - dedicated disk resources
- SCSI is preferred for heavily-used database servers
- Multiple CPUs help to spread multiple database connections among the available CPUs
- Use CLUSTER (or similar method) in heavily-updated tables

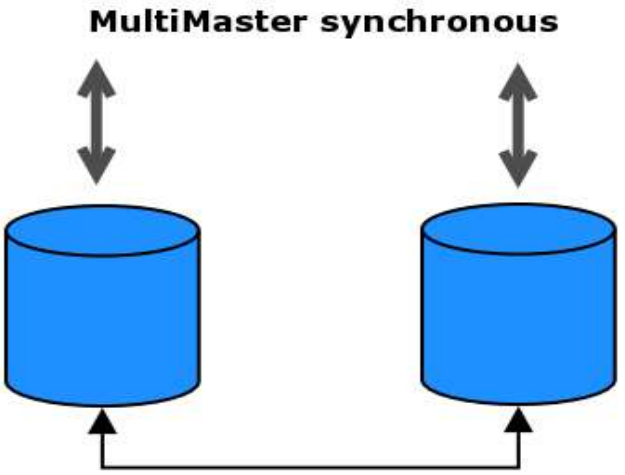
- Populating a Database with a large amount of data:
  - Use copy instead of inserts
  - Remove indexes during population
  - Increase `maintenance_work_mem`
  - Increase `checkpoint_segments`
  - fsync = false / **do not forget to change this to true afterwards**
  - Run ANALYZE afterwards

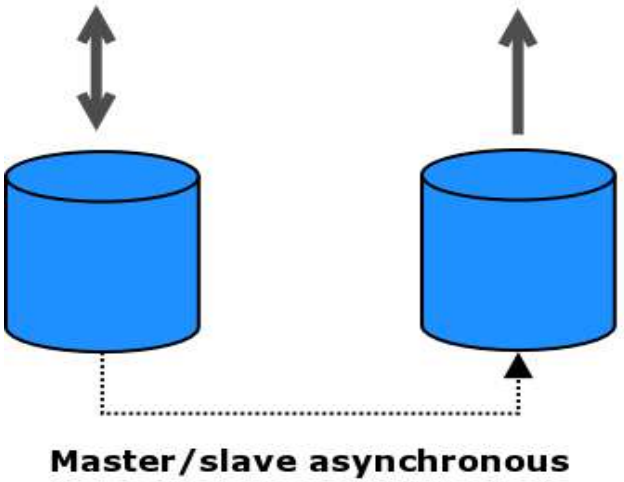
- Use LVM / journal-based file systems
- Data and backups on different disk resources
- Run the database in a dedicated server

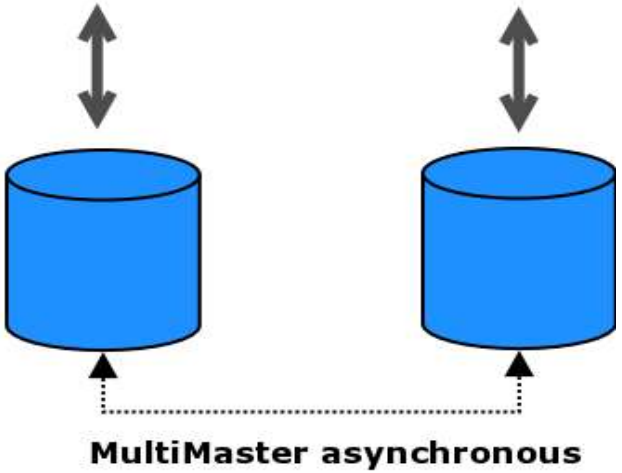
Replication


PostgreSQL


Master/slave synchronous

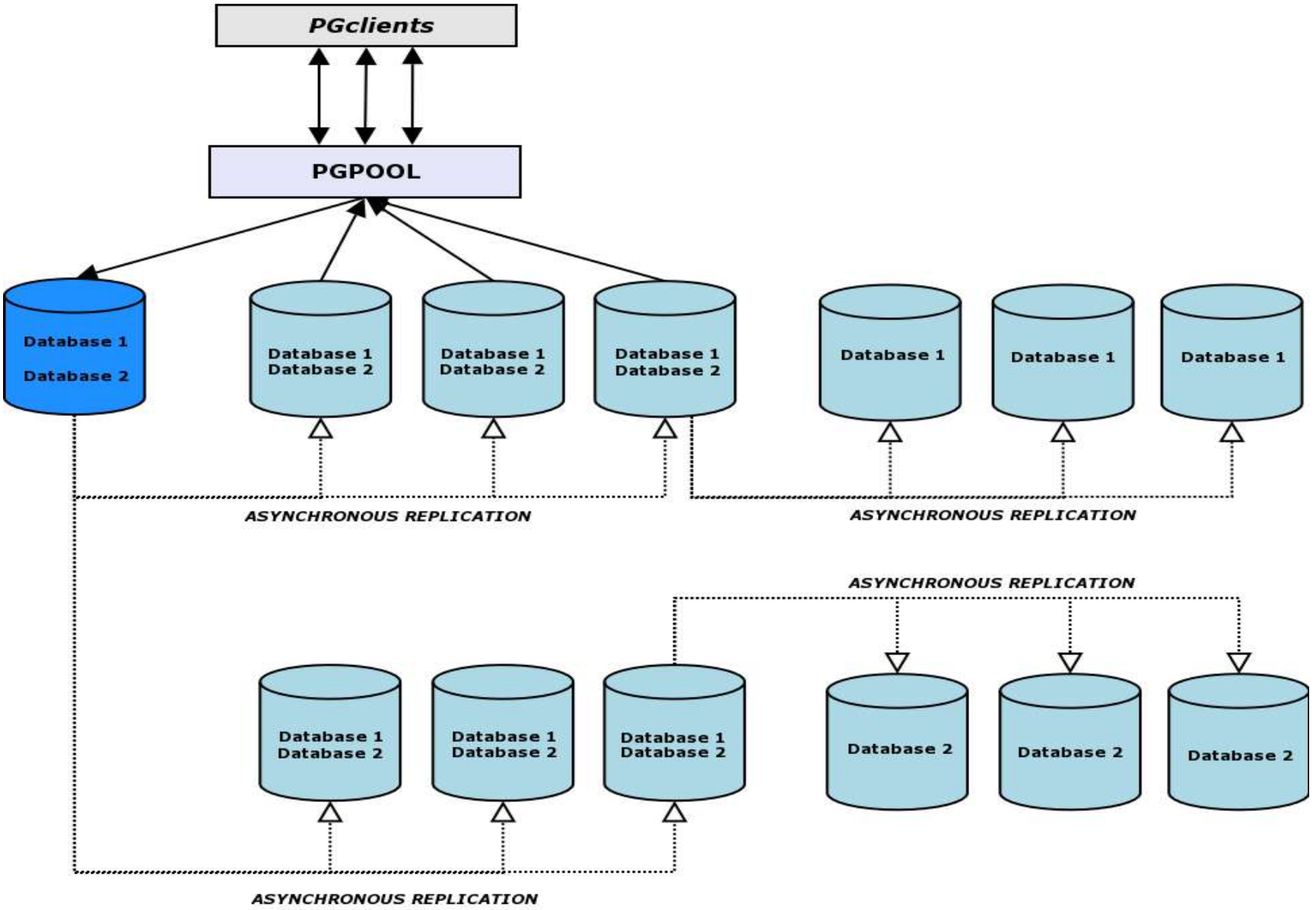

MultiMaster synchronous


Master/slave asynchronous


MultiMaster asynchronous

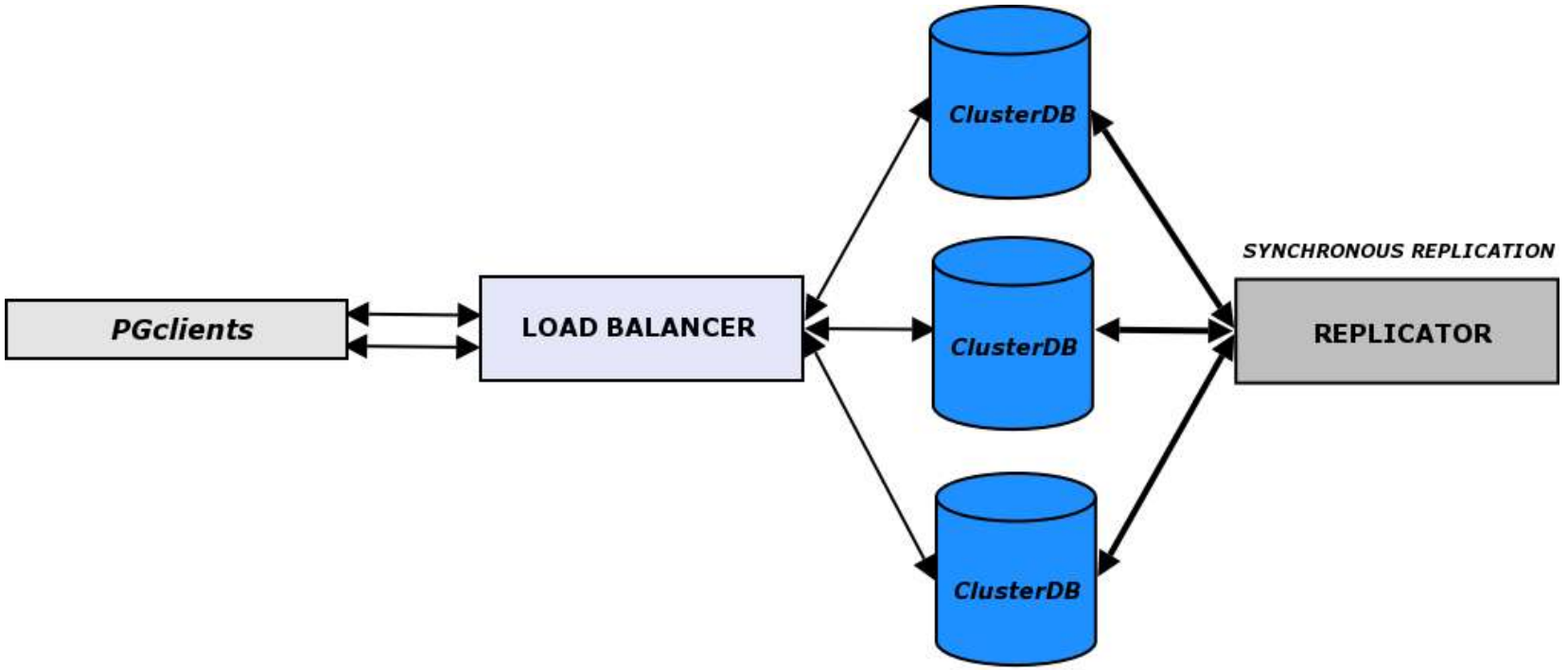
PostgreSQL

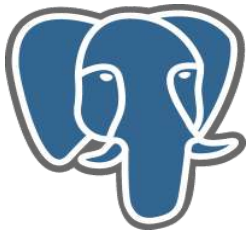
Pgpool / Slony -I



PostgreSQL

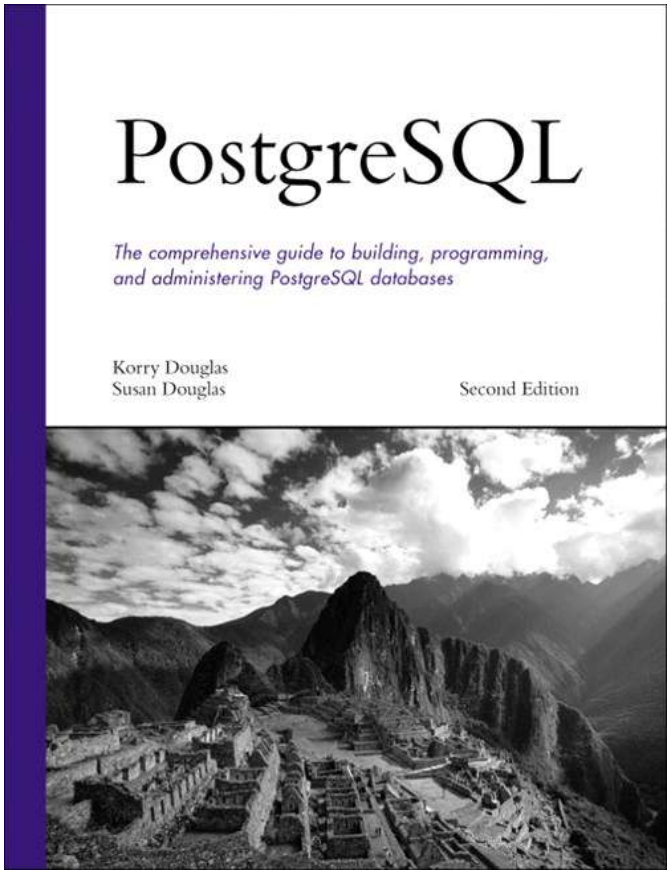
PGCluster



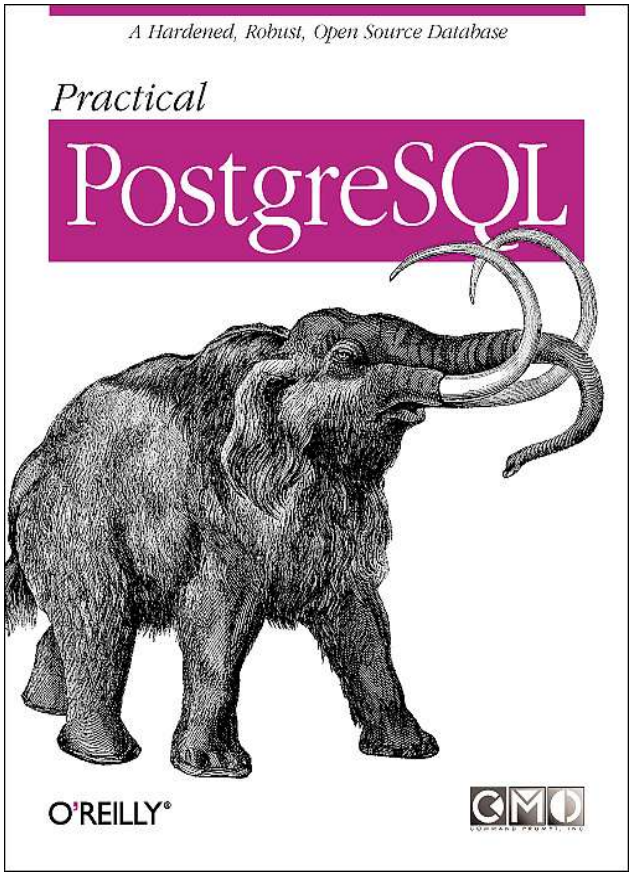


PostgreSQL

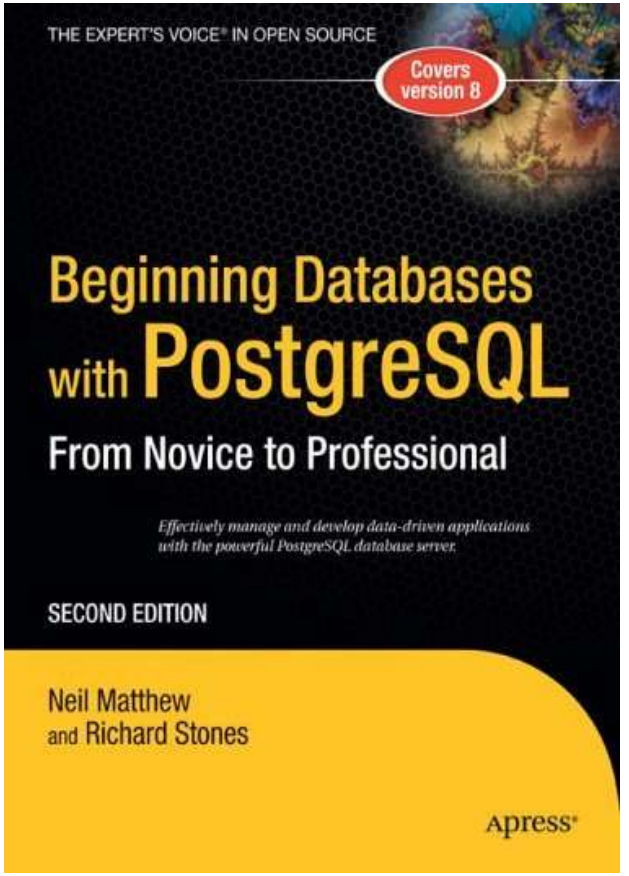
Books



[1]



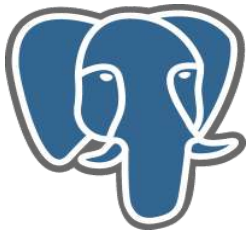
[2]



[3]

[1] PostgreSQL (second edition), *Korry Douglas & Susan Douglas* – ISBN: 0-672-32756-2
[2] Practical PostgreSQL, *Command Prompt, Joshua Drake & John Worsley* – ISBN: 1-565-92846-6
[3] Beginning databases with PostgreSQL (second edition), *Richard Stones & Neil Matthew* – ISBN: 1-590-59478-9

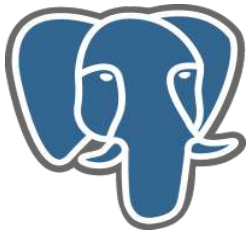
[1][2]



PostgreSQL

Resources

- **PostgreSQL**: Official webside. - `http://www.postgresql.org/`
- **pgFoundry**: PG Project side - `http://pgfoundry.org/`
- **Mailing lists**: 20. Must lists: *pgsql-admin*
  pgsql-general
  pgsql-performance
  pgsql-hackers
- **IRC**: `irc.freenode.net/#postgresql`



PostgreSQL

References

[1] PostgreSQL documentation, 8.0.x online manual - `http://www.postgresql.org/docs/`

[2] PostgreSQL (second edition), *Korry Douglas / Susan Douglas*, Developer's library.

[3] Beginning databases with PostgreSQL (second edition), *Richard Stones & Neil Matthew*

[4] PostgreSQL mailing lists, `http://www.postgresql.org/community/lists/`

[5] History of PostgreSQL – presentation, *Bruce Momjian* - *`http://candle.pha.pa.us/`*

[6] PostgreSQL Performance tunning – presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[7] Mastering PostgreSQL administration – presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[8] Data processing inside postgresql– presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[9] PostgreSQL internals through pictures – presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[10] PostgreSQL replication solutions – presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[11] PostgreSQL: Past, Present, and Future – presentation, *Bruce Momjian* - `http://candle.pha.pa.us/`

[12] PostgreSQL hardware performance tunning – article, *Bruce Momjian* - `http://candle.pha.pa.us/`

[13] Get to know PostgreSQL – presentation, *Oddbjørn Steffensen* - `http://www.tricknology.org/foilware/`

[14] Power PostgreSQL - `http://www.powerpostgresql.com/`