PostgreSQL, the world's most advanced open-source database

October 2005 – NUUG – Høyskolen i Oslo, HiO
Rafael Martinez, USIT, UiO
r.m.guerrero@usit.uio.no
- History
- Features
- Administration
- Tuning
- Replication
History
Ingres 1977-1985 – The beginning

• Proof of concept for relational databases.
• Michael Stonebraker, professor at Berkeley, California.
• Established the company Ingres in 1980.
• Ingres was bought by Computer Associates in 1994
Postgres 1986-1994 – As 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.
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
Overall features

- Minimal administration
- Stability
- Excellent performance
- Data integrity (ACID)
- Portable
- Extensible
- BSD license
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.
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.
SQL features

- Rules
- Views
- Triggers
- Cursors
- Sequences
- Inheritance
- Outer joins
- Sub-selects
- Support for UNION (ALL/EXCEPT)
- Unicode
Administration
Things we are not going to talk about

- PostgreSQL installation
- PostgreSQL cluster initialization
- Create users
- Create databases
- Create tables, indexes, etc
- Programming
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
- It uses a multi-process model
- It does not use multi-threading.
Data directory layout

Cluster directory

```
$PGDATA

pg_hba.conf
pg_ident.conf
postgresql.conf
postgresql.pid
```

pg_tblspc

```
1 (template0)
```

```
xxxx (pg_type)
xxxx (pg_attribute)
xxxx (pg_proc)
xxxx (pg_class)
xxxxx (pg_attrdef)
...........
```

pg_subtrans

```
xxxxx (template1)
```

```
xxxx (pg_type)
xxxx (pg_attribute)
xxxx (pg_proc)
xxxx (pg_class)
xxxxx (pg_attrdef)
...........
```

pg_clog

```
Commit logs

0000 0000
...........
```

pg_xlog

```
WAL write-ahead logs

00000000
00000000
00000000
...........
```

global

```
xxxx (pg_shadow)
xxxx (pg_group)
xxxx (pg_database)
xxxx (pg_tablespace)
xxxxx (index)
xxxxx (index)
pgstat.stat
pg_control
pg_pwd
pg_group
```

base
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
$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
$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

```sql
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
```
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=#
```
psql – PostgreSQL interactive terminal (III)

General
\c [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
\f [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] list tables/indexes/sequences/views/system tables
\da [PATTERN] list aggregate functions
\db [PATTERN] list tables/indexes/sequences/views/system tables
\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
(set 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
(set 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
pgAdmin III

-- Schema: "public"

-- DROP SCHEMA public;

CREATE SCHEMA public
    AUTHORIZATION postgres;

GRANT ALL ON SCHEMA public TO public;

COMMENT ON SCHEMA public IS 'Standard public namespace';
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
- 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
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
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**)
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 ($4 \times 10^9$) transactions
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;
```

<table>
<thead>
<tr>
<th>username</th>
<th>usessysid</th>
<th>usecreatedb</th>
<th>usesuper</th>
<th>usecatupd</th>
<th>passwd</th>
<th>valuntil</th>
<th>useconfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>postgres</td>
<td>1</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>md55cd31c25de000c28135d138df5690e21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgadmin</td>
<td>100</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>md55cd31c76f9470c2abc8636df5cc6381</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rafael</td>
<td>101</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>md55cd31c76f94753746bbbbb8a543e70e21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ola</td>
<td>102</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>md55cd31c76f94753746bbbbb8a543e70e21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tomas</td>
<td>103</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>md55cd31c792637a34bd3234aadb720e21</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

(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
```

```
postgres: user  database  host  activity
```

```
<table>
<thead>
<tr>
<th>datid</th>
<th>datname</th>
<th>procpid</th>
<th>usesysid</th>
<th>usename</th>
<th>current_query</th>
<th>query_start</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>template1</td>
<td>17515</td>
<td>1</td>
<td>postgres</td>
<td>&lt;IDLE&gt;</td>
<td>2005-10-08 21:15:04.245929+02</td>
</tr>
</tbody>
</table>
```

```
template1=# SELECT * from pg_stat_activity ;
```

```
<table>
<thead>
<tr>
<th>relation</th>
<th>database</th>
<th>transaction</th>
<th>pid</th>
<th>mode</th>
<th>granted</th>
</tr>
</thead>
<tbody>
<tr>
<td>16839</td>
<td>1</td>
<td>17515</td>
<td>t</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
template1=# SELECT oid,relname from pg_class where oid = '16839';
```

```
<table>
<thead>
<tr>
<th>oid</th>
<th>relname</th>
</tr>
</thead>
<tbody>
<tr>
<td>16839</td>
<td>pg_locks</td>
</tr>
</tbody>
</table>
```

```
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.
• 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
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 25% of available RAM
- Never more than 1/3 of available RAM

Shared memory values (ie. Linux):

```
kernell.shmmax = ((250 + (8*shared_buffers) + (14*max_connections))*2*1024)
kernell.shmall = (MEMTOTAL/4096)
```
Sort memory batch size \texttt{(work\_mem)}

- Allocated per operation (non-shared) (ORDER BY, DISTINCT, merge joins, hash joins, IN subqueries)
- Defines a ceiling 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 $\text{PGDATA/base/}<\text{DB\_OID>/pgsql\_tmp}$

- 2-4\% of available RAM if we have just a few big sessions.
- My default is 8192 (size in Kb)
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)
Free space map \((\text{max\_fsm\_pages})\)

- 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.
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)
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 --> 16-32
- Very large write loads (several Gb of data) --> up to 128-256
- My default is 64
- Check logfile for warnings.
- It requires a significant amount of disk space for the $PGDATA/pg_xlog directory:
  \[(2 \times \text{checkpoints_segments} + 1) \times 16\text{MB}\]

  **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';
```

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

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

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

**From disk:**

3 046 174 877 blocks x 8Kb/block = 2.436939902 x 10^{10} / 1024 / 1024 = 23 240 Gb

123 731 810 blocks x 8Kb/block = 9.89854480 x 10^8 / 1024 / 1024 = 944 Gb

**From RAM:**

34 505 800 539 blocks x 8Kb/block = 2.760464042 x 10^{11} / 1024 / 1024 = 263 258 Gb

71 110 729 blocks x 8Kb/block = 5.68885832 x 10^9 / 1024 / 1024 = 542 Gb
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
Pgpool / Slony -I
PGCluster

Diagram: A network diagram showing the components of PGCluster, including PGclients, LOAD BALANCER, ClusterDB, and REPLICATOR, with arrows indicating synchronous replication.
Books

Resources

- **pgFoundry**: PG Project side - [http://pgfoundry.org/](http://pgfoundry.org/)
- **Mailing lists**: 20+. Must lists: *pgsql-admin*
  *pgsql-general*
  *pgsql-performance*
  *pgsql-hackers*
- **IRC**: irc.freenode.net/#postgresql
References

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