The pg_catalog was always there, Use it!

Boriss Mejias
Holistic System Software Engineer
Air Guitar Player and Headbanger

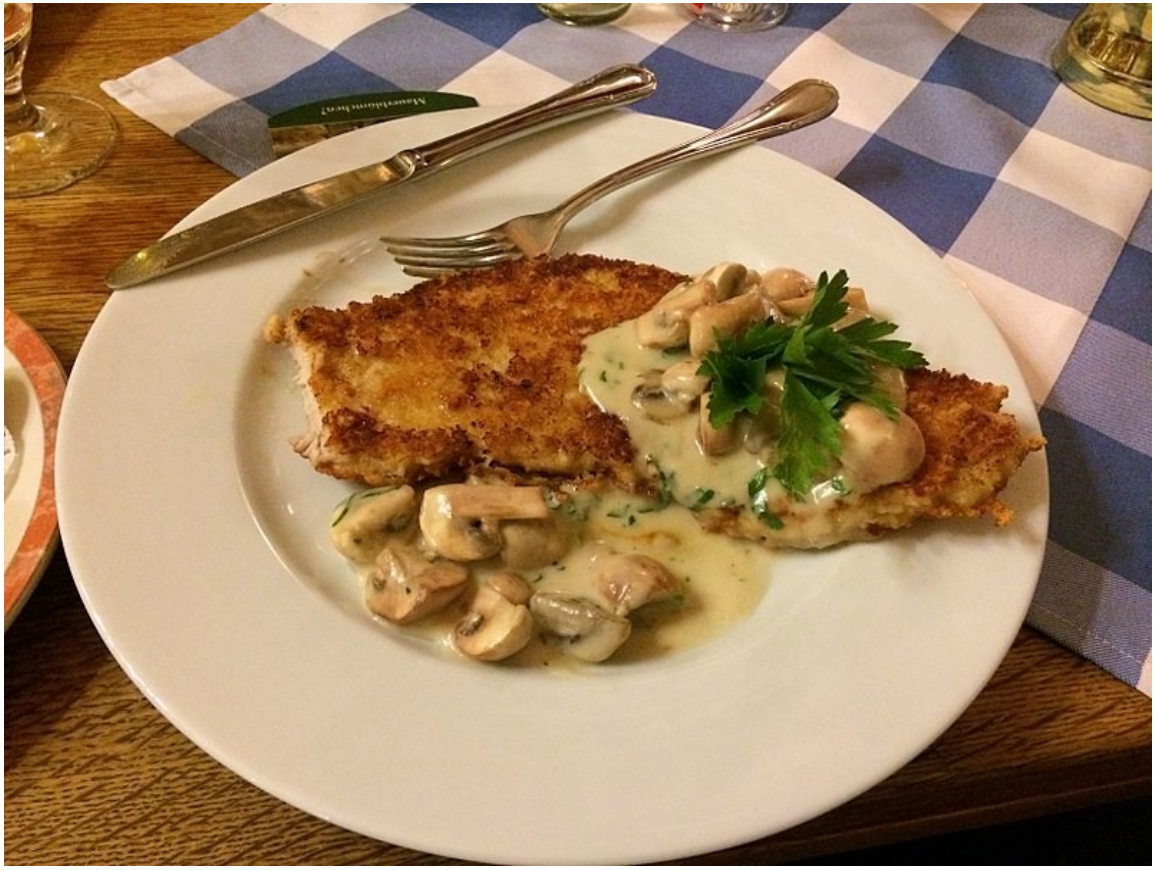
EDB

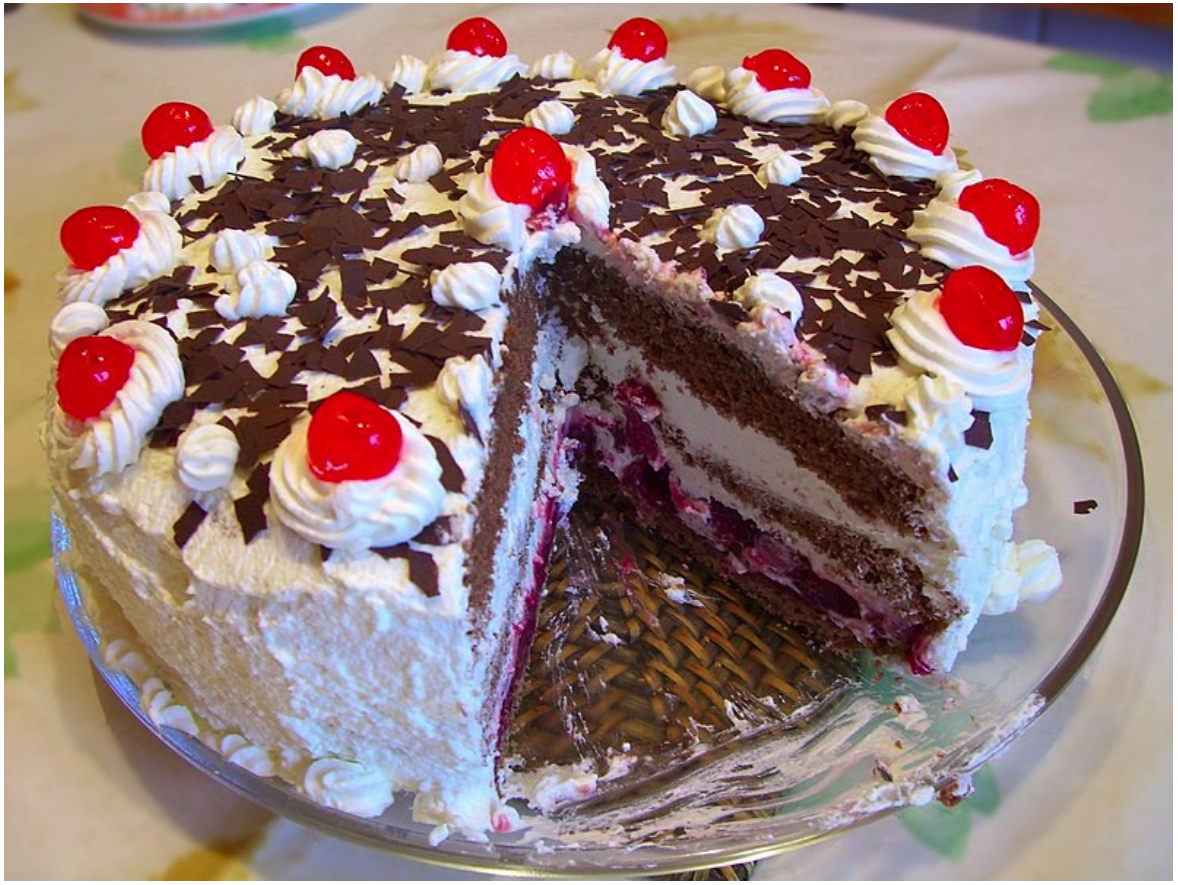
The pg_catalog was always there, Use it!

Boriss Mejias
Solution Architect
Air Guitar Player and Headbanger

EDB


Chex
Biz
Kix
Fresh Step
Whiskas
Friskies





What is the size of my database?

```
SELECT datname
, pg_database_size(datname)
AS size
FROM pg_database ORDER BY size DESC;
   datname    |    size
--------------+------------
 playground   | 2112909983
 leipzig      |  505283075
 postgres     |    8098463
 template1    |    8098463
 template0    |    7954947
(5 rows)
```

```
SELECT datname
, pg_size_pretty(pg_database_size(datname))
AS size
FROM pg_database ORDER BY size DESC;
   datname    |    size
--------------+------------
 playground   |    2015 MB
 leipzig      |     482 MB
 postgres     |    7909 kB
 template1    |    7909 kB
 template0    |    7769 kB
(5 rows)
```

Using the pg_catalog

```
pg_catalog.pg_database
pg_catalog.pg_database_size(name)
pg_catalog.pg_size_pretty(bigint)
```

Connection problem

Connection Problem

User faust can’t connect to database leipzig getting the following error:

```
FATAL:  too many connections for database "leipzig"
```

Configuration and current connections

postgresql.conf

```
    max_connections = 99
```

```
SELECT count(*)
  FROM pg_catalog.pg_stat_activity;
 count
-------
    78
(1 row)
```

Configuration and current connections

```
   SELECT datname, datconnlimit
     FROM pg_database
ORDER BY datname;
    datname     | datconnlimit
----------------+--------------
 leipzig        |           42
 playground     |           -1
 postgres       |           -1
 template0      |           -1
 template1      |           -1
(5 rows)
```

Configuration and current connections

```
   SELECT datname, count(*)
     FROM pg_stat_activity
GROUP BY datname
ORDER BY datname;
 datname    | count
------------+-------
 leipzig    |    42
 playground |    28
 postgres   |     1
            |     7
(4 rows)
```

And by the way

postgresql.conf

```
max_connections = 99
```

And by the way

```
SELECT name, setting, unit
  FROM pg_settings
 WHERE name = 'max_connections';
      name        | setting | unit
------------------+---------+------
 max_connections | 99      |
(1 row)
```

Shared Buffers

```
SELECT name, setting, unit
  FROM pg_settings
 WHERE name = 'shared_buffers';
      name         | setting  | unit
-------------------+----------+------
  shared_buffers | 1048576  | 8kB

SHOW shared_buffers;
 shared_buffers
----------------

 8GB
```

What is the PostgreSQL System Catalog?

EDB

SQL System Catalog

It is the place where a relational database management system stores schema metadata

`information_schema`

SQL System Catalog

It is the place where a relational database management system stores schema metadata

`information_schema`

PostgreSQL's system catalogs are regular tables

`pg_catalog`

System wide catalogs, and per database

Table "pg_catalog.**pg_database**"

Column	Type
datname	name
datdba	oid
encoding	integer
datcollate	name
datctype	name
datistemplate	boolean
datallowconn	boolean
datconnlimit	integer
datlastsysoid	oid
datfrozenxid	xid
datminmxid	xid
dattablespace	oid
datacl	aclitem[]

View "pg_catalog.**pg_stat_activity**"

Column	Type	Column	Type
datid	oid	backend_start	timestampz
datname	name	xact_start	timestampz
pid	integer	query_start	timestampz
leader_pid	integer	state_change	timestampz
usesysid	oid	wait_event_type	text
usename	name	wait_event	text
application_name	text	state	text
client_addr	inet	backend_xid	xid
client_hostname	text	backend_xmin	xid
client_port	integer	query_id	bigint
		query	text
		backend_type	text

View "pg_catalog.**pg_stat_activity**"

Column	Type	Column	Type
datid	oid	backend_start	timestampz
datname	name	xact_start	timestampz
pid	integer	query_start	timestampz
leader_pid	integer	state_change	timestampz
usesysid	oid	wait_event_type	text
usename	name	wait_event	text
application_name	text	state	text
client_addr	inet	backend_xid	xid
client_hostname	text	backend_xmin	xid
client_port	integer	query_id	bigint
		query	text
		backend_type	text

View "pg_catalog.**pg_stat_activity**"

Column	Type	Column	Type
datid	oid	backend_start	timestampz
datname	name	xact_start	timestampz
pid	integer	query_start	timestampz
leader_pid	integer	**state_change**	timestampz
usesysid	oid	wait_event_type	text
username	name	wait_event	text
application_name	text	**state**	text
client_addr	inet	backend_xid	xid
client_hostname	text	backend_xmin	xid
client_port	integer	query_id	bigint
		query	text
		backend_type	text

Spot ‘idle in transaction’

```
SELECT count(*)
  FROM pg_stat_activity
 WHERE state = 'idle in transaction';

SELECT pid
     , datname
     , usename
     , clock_timestamp() - state_change
  FROM pg_stat_activity
 WHERE state = 'idle in transaction';
```

Spot ‘idle in transaction’

```
SELECT count(*)
  FROM pg_stat_activity
 WHERE state = 'idle in transaction';

SELECT pg_terminate_backend(pid)
     , datname
     , usename
     , current_timestamp - state_change
  FROM pg_stat_activity
 WHERE state = 'idle in transaction'
   AND current_timestamp – state_change
                         > '1 min'::interval;
```

Is work_mem large enough?

Check Temporary Files

```
SELECT datname
     , temp_files
     , pg_size_pretty(temp_bytes)
  FROM pg_stat_database;
   datname    | temp_files | pg_size_pretty
--------------+------------+----------------
 postgres     |          0 | 0 bytes
 template1    |          0 | 0 bytes
 template0    |          0 | 0 bytes
 leipzig      |         16 | 7920 kB
 playground   |          4 | 164 MB
(5 rows)
```

The database does not scale

Support Stories

Running our service with one application node it was super fast. With two application nodes the database is super slow.

FIX IT!

Support Stories

Running our service with one application node it was super fast. With two application nodes the database is super slow.

I do not understand what is happening. Would you mind having a look at the database server?

Check Conflicts

```
SELECT datname
     , conflicts
     , deadlocks
  FROM pg_stat_database;
   datname    | conflicts  | deadlocks
--------------+------------+-----------
 postgres     |          0 |         0
 template1    |          0 |         0
 template0    |          0 |         0
 Leipzig      |    1380872 |    866737
 playground   |          0 |         0
(5 rows)
```

Are my indexes being used?

Index Statistics

```
SELECT schemaname, relname, indexrelname
  FROM pg_stat_user_indexes
 WHERE idx_scan=0;
 schemaname |    relname      |  indexrelname
------------+-----------------+-----------------------
 public     | reviews         | idx_beer_timestamp
 public     | many_secrets    | pub_id_pkey
 public     | humans          | human_sin_gin_lower_idx
 test_la    | humans          | human_sin_la_idx
 public     | dummy           | id_pkey
(5 rows)
```

Table Statistics

```
SELECT schemaname
     , relname
     , seq_scan
     , idx_scan
  FROM pg_stat_user_tables;
```

More Statistics

```
pg_stat_all_indexes
pg_stat_all_tables

pg_stat_sys_indexes
pg_stat_sys_tables

pg_stat_user_indexes
pg_stat_user_tables
```

More Statistics

pg_stat_all_indexes	pg_statio_all_indexes
pg_stat_all_tables	pg_statio_all_tables
pg_stat_sys_indexes	pg_statio_sys_indexes
pg_stat_sys_tables	pg_statio_sys_tables
pg_stat_user_indexes	pg_statio_user_indexes
pg_stat_user_tables	pg_statio_user_tables

How many rows in the table?

Classic Query

```
SELECT COUNT(*) FROM addresses;
  count
---------
 1001220
(1 row)

Time: 150.647 ms
```

Classic Query

```
SELECT COUNT(*) FROM addresses;
  count
---------
 1001220
(1 row)

Time: 150.647 ms
```

Using the Catalog

```
SELECT n_live_tup FROM pg_stat_user_tables
WHERE relname='addresses';
 n_live_tup
------------
    1001220
(1 row)

Time: 10.536 ms
```

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	**n_live_tup**	bigint
schemaname	name	**n_dead_tup**	bigint
relname	name	**n_mod_since_analyze**	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	**last_autovacuum**	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	**last_autoanalyze**	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	**autovacuum_count**	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	**autoanalyze_count**	bigint

View "pg_catalog.**pg_stat_user_tables**"

Column	Type	Column	Type
relid	oid	n_live_tup	bigint
schemaname	name	n_dead_tup	bigint
relname	name	n_mod_since_analyze	bigint
seq_scan	bigint	last_vacuum	timestampz
seq_tup_read	bigint	last_autovacuum	timestampz
idx_scan	bigint	last_analyze	timestampz
idx_tup_fetch	bigint	last_autoanalyze	timestampz
n_tup_ins	bigint	vacuum_count	bigint
n_tup_upd	bigint	autovacuum_count	bigint
n_tup_del	bigint	analyze_count	bigint
n_tup_hot_upd	bigint	autoanalyze_count	bigint
		n_ins_since_vacuum	bigint

Replication

Is replication working?

Is replication working?
Is there a lag?

Is replication working?
Is there a lag?
Should we blame Magnus?

EDB

Replication Statisticts

View "pg_catalog.**pg_stat_replication**"

Column	Type	Column	Type
pid	integer	sent_lsn	pg_lsn
usesysid	oid	write_lsn	pg_lsn
usename	name	flush_lsn	pg_lsn
application_name	text	replay_lsn	pg_lsn
client_addr	inet	write_lag	interval
client_hostname	text	flush_lag	interval
client_port	integer	replay_lag	interval
backend_start	timestampz	sync_priority	integer
backend_xmin	xid	sync_state	text
state	text	reply_time	timestampz

Replication Statisticts

View "pg_catalog.**pg_stat_replication**"

Column	Type	Column	Type
pid	integer	sent_lsn	pg_lsn
usesysid	oid	write_lsn	pg_lsn
usename	name	flush_lsn	pg_lsn
application_name	text	replay_lsn	pg_lsn
client_addr	inet	write_lag	interval
client_hostname	text	flush_lag	interval
client_port	integer	replay_lag	interval
backend_start	timestampz	sync_priority	integer
backend_xmin	xid	**sync_state**	text
state	text	**reply_time**	timestampz

Replication Statisticts

View "pg_catalog.**pg_stat_replication**"

Column	Type	Column	Type
pid	integer	**sent_lsn**	pg_lsn
usesysid	oid	**write_lsn**	pg_lsn
usename	name	**flush_lsn**	pg_lsn
application_name	text	**replay_lsn**	pg_lsn
client_addr	inet	write_lag	interval
client_hostname	text	flush_lag	interval
client_port	integer	replay_lag	interval
backend_start	timestampz	sync_priority	integer
backend_xmin	xid	sync_state	text
state	text	reply_time	timestampz

Replication Statisticts

View "pg_catalog.**pg_stat_replication**"

Column	Type	Column	Type
pid	integer	sent_lsn	pg_lsn
usesysid	oid	write_lsn	pg_lsn
usename	name	flush_lsn	pg_lsn
application_name	text	replay_lsn	pg_lsn
client_addr	inet	**write_lag**	interval
client_hostname	text	**flush_lag**	interval
client_port	integer	**replay_lag**	interval
backend_start	timestampz	sync_priority	integer
backend_xmin	xid	sync_state	text
state	text	reply_time	timestampz

More on Replication

```
pg_stat_replication
pg_stat_replication_slots
pg_stat_wal_receiver
pg_stat_subscription
pg_replication_slots
```

Learn by Yourself

Use psql -E

```
$ psql -E -d leipzig
```

Use psql -E

```
$ psql -E -d leipzig
psql (14.3 (Debian 14.3-1.pgdg100+1))
Type "help" for help.

leipzig=> \du
********* QUERY **********
SELECT r.rolname, r.rolsuper, r.rolinherit,
  r.rolcreaterole, r.rolcreatedb, r.rolcanlogin,
  r.rolconnlimit, r.rolvaliduntil,
  ARRAY(SELECT b.rolname
        FROM pg_catalog.pg_auth_members m
        JOIN pg_catalog.pg_roles b ON (m.roleid = b.oid)
        WHERE m.member = r.oid) as memberof
, r.rolreplication
, r.rolbypassrls
FROM pg_catalog.pg_roles r
WHERE r.rolname !~ '^pg_'
ORDER BY 1;
**************************
```

Closing Words

EDB

The PostgreSQL Catalog

It’s AWESOME!

The PostgreSQL Catalog

Not too bad...

The PostgreSQL Catalog

Contains plenty of useful metadata

All relationship between objects

Statistics about the data

Statistics about the sessions

Statistics about replication

Very useful for monitoring

The PostgreSQL Catalog

Contains plenty of useful metadata

All relationship between objects

Statistics about the data

Statistics about the sessions

Statistics about replication

Extremely useful for monitoring

Did Faust really sell his soul to Mephistopheles?

Thank You!

Boriss Mejias
@tchorix
boriss.mejias@enterprisedb.com

EDB™