\dfS pg_*: Um passeio pelas funções administrativas do PostgreSQL

Características do catálogo

Armazena metadados da instância

Extenso conjunto de informações distribuídas em tabelas e views de sistema

- Definição de objetos (tabelas, atributos, índices, tipos, &c)
- Estatísticas do banco de dados

Organizado e padronizado

Convenção de nomes simples e consistente

sufixo	**descrição**
pg_	objeto de catálogo (reservado apenas para *schema*)
stat_	Informação de estatística
io_	eventos de E/S
ts_	configurações de FTS

- Ex.: *pg_catalog, pg_stat_activity, pg_statio_user*

Características do catálogo (cont.)

Compreensível

Dicionário de simples entendimento e manipulação

Extensível

Com pouco esforço é possível adicionar novas funcionalidades (funções, operadores, &c)

Completo

Além do acesso aos metadados, permite analisar arquivos de dados

Informações sobre o catálogo

2331 funções de sistema

86 funções estatísticas

36 views administrativas

48 tabelas de sistema

71 objetos no *information_schema* (ISO/IEC 9075-11:2008)

Sobre o nome da palestra

```
01.   SELECT routine_schema, routine_name, data_type
02.     FROM information_schema.routines
```

OU

```
postgres=# \dfS+
```

```
  Schema   |         Name                |     Result data
type     |
-----------+-----------------------------+-----------------------
-|
pg_catalog | pg_database_size           |
bigint                    |
```

```
pg_catalog | pg_database_size           |
bigint                    |
pg_catalog | pg_describe_object         |
text                      |
pg_catalog | pg_encoding_max_length     |
integer                   |
pg_catalog | pg_encoding_to_char        |
name                      |
pg_catalog | pg_extension_config_dump   |
void                      |
pg_catalog | pg_function_is_visible     |
boolean                   |
```

Tipos de funções

Funções administrativas

- Obter informações de objetos do banco de dados através do catálogo

Funções estatísticas

- Monitorar estatísticas de atividade do servidor

Funções WAL

- Manipular e obter informações do log de transações (*aka* WAL)

Funções administrativas

Ex.: obtendo informações de um arquivo

```
01.    SELECT current_setting('config_file');
```

```
          current_setting
-----------------------------------------
 /Library/PostgreSQL/8.4/data/postgresql.conf
```

```
01.    SELECT now() - modification AS "última modificação"
02.      FROM pg_stat_file(current_setting('config_file'));
```

```
-[ RECORD 1 ]------+-----------------------
última modificação | 5 days 17:18:21.507422
```

Funções administrativas

Ex.: Tempo de vida do servidor

```
01.   SELECT format('%s, up %s, %s users, cache hit ratio: %%
      %s',
02.      LOCALTIME, (CURRENT_DATE - date_trunc('days',
      pg_postmaster_start_time())),
03.      pg_stat_get_db_numbackends(oid),
04.      round(( pg_stat_get_db_blocks_hit(oid)::float
05.           / (pg_stat_get_db_blocks_fetched(oid)
06.            + pg_stat_get_blocks_hit(oid) + 1) *
      100)::numeric,2)) AS "uptime"
07.    FROM pg_catalog.pg_database
```

```
08.    WHERE datname = current_database();
```

```
-[ RECORD 1 ]-------------------------------------------------
uptime | 17:42:52, up 8 days, 2 users, cache hit ratio: % 81.61
```

Funções administrativas

Parece, mas não é...

```
01.   SELECT pg_relation_size('foobar') AS falso,
02.          pg_table_size('foobar') AS verdadeiro;
```

```
+--------+------------+
| falso  | verdadeiro |
+--------+------------+
| 368640 |     393216 |
+--------+------------+
```

Funções administrativas

Parece, mas não é...

```
01.   SELECT pg_relation_size('foobar','fsm' /* ou vm */)
02.          ,pg_total_relation_size('foobar');
```

```
+------------------+------------------------+
| pg_relation_size | pg_total_relation_size |
+------------------+------------------------+
|            24576 |                 393216 |
+------------------+------------------------+
```

Funções administrativas

Outras funções administrativas

função	*resultado*
SELECT pg_database_size('postgres')	6759224
SELECT pg_indexes_size('foobar')	245760
SELECT pg_size_pretty('100024')	98 kB
SELECT pg_relation_filepath('foobar')	base/12180/16393

Funções administrativas - *WAL*

Ex.: Obtendo o registro atual do XLOG

```
01.    SELECT pg_current_xlog_insert_location();
```

```
pg_current_xlog_insert_location
--------------------------------
0/11570578
```

Funções administrativas - *WAL*

Obtendo *offset* do registro atual do XLOG

```
01.   SELECT CAST(X'570578' AS INTEGER) /* x =
      01010111000010101111000 */
02.       AS file_offset;
```

OU

```
01.   SELECT file_name, file_offset
02.     FROM
      pg_xlogfile_name_offset(pg_current_xlog_location());
```

```
+--------------------------+-------------+
|        file_name         | file_offset |
+--------------------------+-------------+
| 00000001000000000000011 |    5703032  |
+--------------------------+-------------+
```

Funções administrativas - *WAL*

Ex.: Obtendo informações do LOG

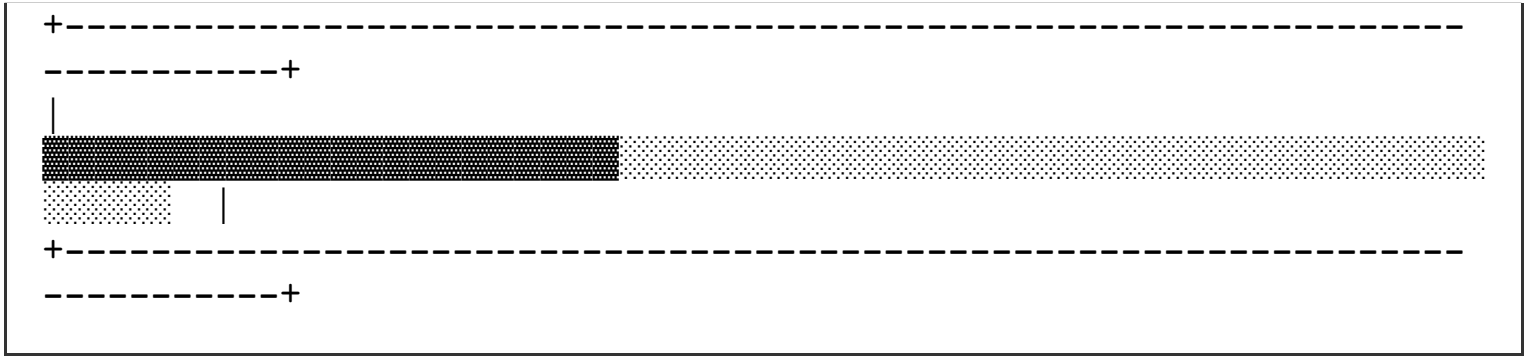
```
01.   WITH RECURSIVE xlog(i,c) AS
02.   (VALUES (0,pg_current_xlog_location())
03.     UNION ALL
04.   SELECT generate_series(1,
05.       (((pg_xlogfile_name_offset(
06.               pg_current_xlog_location())).file_offset /
      16777216.) * 100
07.       )::int), NULL
08.   )
```

Funções administrativas - *WAL*

Ex.: Obtendo informações XLOG (cont)

```
01.   SELECT rpad(array_to_string(array_agg(
02.            regexp_replace(i::varchar,'\d{1,3}',U&'\2593')
03.            ),'','',100,U&'\2591')
04.   AS "Gráfico de ocupação do XLOG"
05.   FROM xlog
```

```
+-------------------------------------------------------------
-----------+
|                        Gráfico de ocupação do
XLOG                           |
```



Funções administrativas

Outras funções do WAL

função	*resultado*
SELECT pg_switch_xlog()	0/F47DFF8
SELECT pg_create_restore_point(CURRENT_DATE::text);	0/10000110
SELECT pg_xlog_replay_pause()	void
SELECT pg_xlog_replay_resume()	void

Funções de estatísticas

Ex.: Obtendo informações E/S

```
01.   SELECT round(( pg_stat_get_db_blocks_hit(oid)::float
02.   / (pg_stat_get_db_blocks_fetched(oid)
03.   + pg_stat_get_blocks_hit(oid) + 1) * 100)::numeric,2) AS
      "hit ratio"
04.   FROM pg_catalog.pg_database;
```

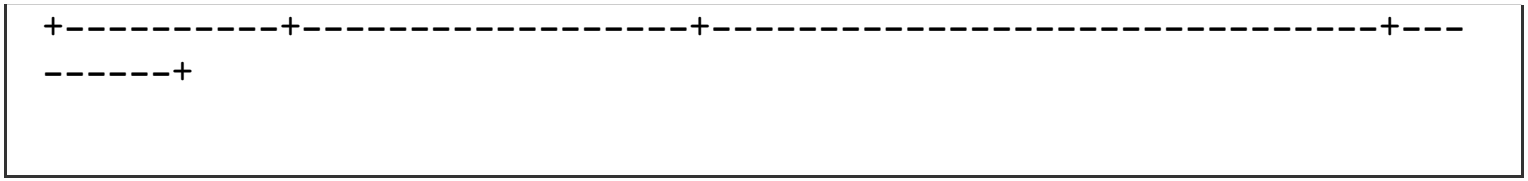
```
+-----------+-----------+
|  datname  | hit ratio |
+-----------+-----------+
| postgres  |     96.67 |
+-----------+-----------+
```

Funções estatísticas

Ex.: Obtendo informações de um *backend*:

```
01.   SELECT usesysid, application_name, backend_start,
      waiting
02.     FROM pg_stat_get_activity(pg_backend_pid())
```

```
+----------+-----------------+-------------------------------+---
------+
| usesysid | application_name |         backend_start         |
waiting |
+----------+-----------------+-------------------------------+---
------+
|       10 | psql             | 2011-11-03 12:14:25.966554-02 |
f       |
```



Funções estatísticas

Outras funções de estatísticas

função	*resultado*
SELECT pg_stat_get_bgwriter_requested_checkpoints();	36
SELECT pg_stat_get_buf_written_backend();	6
SELECT pg_stat_get_backend_waiting(1023);	false

Referências

1. http://www.postgresql.org
2. http://www.postgresql.org/docs/9.1/interactive/functions-admin.html
3. http://www.postgresql.org/docs/9.1/interactive/monitoring-stats.html

```
SELECT
pg_terminate_backend(pg_backend_pid());
```