7) Catálogo do Sistema

Todo SGBD precisa ter seu catálogo de sistema (metadados, dicionário de dados), onde armazena pelo menos:

- Nomes das tabelas
- Nomes dos campos
- Tipos de dados de cada campo
- As constraints
- Informações sobre os índices
- Privilégios de acesso dos elementos

O Catálogo de sistema está disponível no PostgreSQL desde a versão 7.4.

7.1) Tabelas de Sistema do PostgreSQL 8.3

Catalog Name	Purpose
pg_aggregate	aggregate functions
pg_am	index access methods
pg_amop	access method operators
pg_amproc	access method support procedures
pg_attrdef	column default values
pg_attribute	table columns ("attributes")
pg_authid	authorization identifiers (roles)
pg_auth_members	authorization identifier membership relationships
pg_autovacuum	per-relation autovacuum configuration parameters
pg_cast	casts (data type conversions)
pg_class	tables, indexes, sequences, views ("relations")
pg_constraint	check constraints, unique constraints, primary key constraints, foreign key constraints
pg_conversion	encoding conversion information
pg_database	databases within this database cluster
pg_depend	dependencies between database objects
pg_description	descriptions or comments on database objects

Catalog Name	Purpose
pg_enum	enum label and value definitions
pg_index	additional index information
pg_inherits	table inheritance hierarchy
pg_language	languages for writing functions
pg_largeobject	large objects
pg_listener	asynchronous notification support
pg_namespace	schemas
pg_opclass	access method operator classes
pg_operator	operators
pg_opfamily	access method operator families
pg_pltemplate	template data for procedural languages
pg_proc	functions and procedures
pg_rewrite	query rewrite rules
pg_shdepend	dependencies on shared objects
pg_shdescription	comments on shared objects
pg_statistic	planner statistics
pg_tablespace	tablespaces within this database cluster
pg_trigger	triggers
pg_ts_config	text search configurations
pg_ts_config_map	text search configurations' token mappings
pg_ts_dict	text search dictionaries
pg_ts_parser	text search parsers
pg_ts_template	text search templates
pg_type	data types

Mais detalhes em:
http://www.postgresql.org/docs/8.3/interactive/catalogs.html
http://pgdocptbr.sourceforge.net/pg80/catalogs.html
http://www.cs.umu.se/kurser/TDBC86/H06/Slides/16_OH_catalog.pdf (trazendo informações sobre o catálogo do PostgreSQL, Oracle e ODBC).

7.2) Informações sobre as tabelas de sistema

Para visualizar a estrutura de cada uma destas relações, podemos usar o psql. Exemplo:

\d pg_database

Observar que existe um esquema ()

Para visualizar o catálogo completo, que está no esquema 'pg_catalog', usar:

\dS

Um total de 74 na versão 8.3 do PostgreSQL.

Para visualizar a estrutura de uma das relações do esquema pg_catalog:

\d pg_catalog.pg_table

serão listados os campos, seus tipos e abaixo uma definição de view.

Retornando todas as tabelas de sistema de um usuário:

select * from pg_catalog.pg_tables where tableowner='dba1';

select * from pg_catalog.pg_tables where tableowner='postgres';

7.3) Exemplos práticos e úteis

Uso do disco pela Tabela

\c dba_projeto

vacuum analyze clientes;

SELECT relfilenode, relpages FROM pg_class WHERE relname = 'clientes';

relfilenode é o arquivo, com nome usando números num diretório do 'base'.

relpages é o número de páginas ocupado pela relação (tabela). Lembrar que cada página ocupa 8KB. relpages somente é atualizado por VACUUM, ANALYZE e uns poucos comandos de DDL como CREATE INDEX). O valor de relfilenode possui interesse caso se deseje examinar diretamente o arquivo em disco da tabela.

Podemos então saber algo com mais detalhes assim:

\c dba_projeto

vacuum analyza clientes;

SELECT relfilenode AS arquivo, relpages*8 AS tamanho_em_kb FROM pg_class WHERE relname = 'clientes';

No diretório do tutorial existe o arquivo syscat.sql contendo várias consultas interessantes aos catálogos do sistema:

http://pgdocptbr.sourceforge.net/pg80/syscat.sql

```
-- syscat.sql-
--    Exemplos de consultas aos catálogos do sistema

-- Portions Copyright (c) 1996-2003, PostgreSQL Global Development Group
-- Portions Copyright (c) 1994, Regents of the University of California

-- Primeiro definir o caminho de procura do esquema como pg_catalog,
-- para não ser necessário qualificar todo objeto do sistema.
--
SET SEARCH_PATH TO pg_catalog;

--
-- Listar o nome de todos os administradores de banco de dados e o seus bancos de dados.
--
SELECT usename, datname
  FROM pg_user, pg_database
  WHERE usesysid = datdba
  ORDER BY usename, datname;
```

```
--
-- Listar todas as classes definidas pelo usuário
--
SELECT n.nspname, c.relname
  FROM pg_class c, pg_namespace n
  WHERE c.relnamespace=n.oid
    AND c.relkind = 'r'                -- sem índices, visões, etc
    AND n.nspname not like 'pg\\_%'       -- sem catálogos
    AND n.nspname != 'information_schema' -- sem information_schema
  ORDER BY nspname, relname;

-- Listar todos os índices simples (ou seja, àqueles que são definidos
-- sobre uma referência de coluna simples)
--
SELECT n.nspname AS schema_name,
       bc.relname AS class_name,
       ic.relname AS index_name,
       a.attname
  FROM pg_namespace n,
       pg_class bc,            -- classe base
       pg_class ic,            -- classe índice
       pg_index i,
       pg_attribute a          -- atributo na base
  WHERE bc.relnamespace = n.oid
    AND i.indrelid = bc.oid
    AND i.indexrelid = ic.oid
    AND i.indkey[0] = a.attnum
    AND i.indnatts = 1
    AND a.attrelid = bc.oid
  ORDER BY schema_name, class_name, index_name, attname;

--
-- Listar os atributos definidos pelo usuário e seus tipos
-- para todas as classes definidas pelo usuário
--
SELECT n.nspname, c.relname, a.attname, format_type(t.oid, null) AS typname
  FROM pg_namespace n, pg_class c,
       pg_attribute a, pg_type t
  WHERE n.oid = c.relnamespace
    AND c.relkind = 'r'                -- sem índices
    AND n.nspname not like 'pg\\_%'       -- sem catálogos
    AND n.nspname != 'information_schema' -- sem information_schema
    AND a.attnum > 0                   -- sem atributo de sistema
    AND not a.attisdropped             -- sem colunas removidas
    AND a.attrelid = c.oid
    AND a.atttypid = t.oid
  ORDER BY nspname, relname, attname;
```

```
--
-- Listar todos os tipos base definidos pelo usuário (sem incluir os tipos matriz)
--
SELECT n.nspname, u.usename, format_type(t.oid, null) AS typname
  FROM pg_type t, pg_user u, pg_namespace n
  WHERE u.usesysid = t.typowner
    AND t.typnamespace = n.oid
    AND t.typrelid = '0'::oid            -- sem tipos complexos
    AND t.typelem = '0'::oid             -- sem matrizes
    AND n.nspname not like 'pg\\_%'      -- sem catálogos
    AND n.nspname != 'information_schema' -- sem information_schema
  ORDER BY nspname, usename, typname;

--
-- Listar todas as funções de agregação e os tipos em que podem ser aplicadas
--
SELECT n.nspname, p.proname, format_type(t.oid, null) AS typname
  FROM pg_namespace n, pg_aggregate a,
       pg_proc p, pg_type t
  WHERE p.pronamespace = n.oid
    AND a.aggfnoid = p.oid
    AND p.proargtypes[0] = t.oid
  ORDER BY nspname, proname, typname;

-- Restaurar o caminho de procura
--
RESET SEARCH_PATH;
```

Retornar o número de usuários conectados

```
select count(*) from pg_stat_activity;
select count(*) from pg_stat_database;
```

pg_stat_database que apresenta para cada banco de dados o número de conexões.
Fica mais fácil de visualizar do que o pg_stat_activity quando se tem muitas conexões.

Mostrar uso dos índices:

```
select * from pg_statio_user_indexes;
select * from pg_stat_user_indexes;
```

Mostra estatística de uso de todas as tabelas e manutenção:

```
select * from pg_stat_all_tables;
```

Mostra todas as tabelas e informações do atual esquema do atual banco:

```
select * from pg_stat_user_tables;
```

Veja só o retorno: relid | schemaname | relname | seq_scan | seq_tup_read | idx_scan | idx_tup_fetch | n_tup_ins | n_tup_upd | n_tup_del | last_vacuum | last_autovacuum | last_analyze | last_autoanalyze

A função pg_stat_get_backend_idset provê uma conveniente maneira de gerar um registro/linha para cada processo ativo no servidor. Por exemplo, para **exibir os PIDs e as atuais consultas de todos os processos do servidor**:

```
SELECT pg_stat_get_backend_pid(s.backendid) AS procpid,
      pg_stat_get_backend_activity(s.backendid) AS current_query
   FROM (SELECT pg_stat_get_backend_idset() AS backendid) AS s;
```

Visualizar os processos do portgresql num UNIX:

ps auxww | grep ^post

Mostrar todas as tabelas (inclusive de sistema) a quantidade de registros:

select relpages*8192 from pg_class;

Funções para Administração do Sistema

Definindo configurações

A função current_setting retorna o valor corrente da definição nome_da_definição. Corresponde ao comando SQL SHOW. Por exemplo:

```
SELECT current_setting('datestyle');
```

A função set_config define o parâmetro nome_da_configuração como novo_valor. Se o parâmetro é_local for true, então o novo valor se aplica somente à transação corrente. Se for desejado que o novo valor seja aplicado à sessão corrente, deve ser utilizado false. Esta função corresponde ao comando SQL SET. Por exemplo:

```
SELECT set_config('log_statement_stats', 'off', false);
```

Funções de Sinais para o Servidor

pg_cancel_backend(pid)
pg_reload_conf()
pg_rotate_logfile()

Se for bem-sucedida a função retorna 1, caso contrário retorna 0. O ID do processo (pid) de um servidor ativo pode ser encontrado a partir da coluna procpid da visão pg_stat_activity, ou listando os processos do postgres no servidor através do comando do Unix ps.

Funções que Retornam o Tamanho de Objetos

Name	Return Type	Description
`pg_column_size(any)`	`int`	Number of bytes used to store a particular value (possibly compressed)
`pg_tablespace_size(oid)`	`bigint`	Disk space used by the tablespace with the specified OID
`pg_tablespace_size(name)`	`bigint`	Disk space used by the tablespace with the specified name
`pg_database_size(oid)`	`bigint`	Disk space used by the database with the specified OID
`pg_database_size(name)`	`bigint`	Disk space used by the database with the specified name
`pg_relation_size(oid)`	`bigint`	Disk space used by the table or index with the specified OID
`pg_relation_size(text)`	`bigint`	Disk space used by the table or index with the specified name. The table name may be qualified with a schema name
`pg_total_relation_size(oid)`	`bigint`	Total disk space used by the table with the specified OID, including indexes and toasted data
`pg_total_relation_size(text)`	`bigint`	Total disk space used by the table with the specified name, including indexes and toasted data. The table name may be qualified with a schema name
`pg_size_pretty(bigint)`	`text`	Converts a size in bytes into a human-readable format with size units

`pg_column_size` shows the space used to store any individual data value.

`pg_tablespace_size` and `pg_database_size` accept the OID or name of a tablespace or database, and return the total disk space used therein.

`pg_relation_size` accepts the OID or name of a table, index or toast table, and returns the size in bytes.

`pg_total_relation_size` accepts the OID or name of a table or toast table, and returns the size in bytes of the data and all associated indexes and toast tables.

`pg_size_pretty` can be used to format the result of one of the other functions in a human-readable way, using kB, MB, GB or TB as appropriate.

Funções de Acesso a Arquivos

Name	Return Type	Description
`pg_ls_dir(dirname text)`	`setof text`	List the contents of a directory
`pg_read_file(filename text, offset bigint, length bigint)`	`text`	Return the contents of a text file
`pg_stat_file(filename text)`	`record`	Return information about a file

`pg_ls_dir` returns all the names in the specified directory, except the special entries "`.`" and "`..`".

`pg_read_file` returns part of a text file, starting at the given `offset`, returning at most `length` bytes (less if the end of file is reached first). If `offset` is negative, it is relative to the end of the file.

`pg_stat_file` returns a record containing the file size, last accessed time stamp, last modified time stamp, last file status change time stamp (Unix platforms only), file creation timestamp (Windows only), and a `boolean` indicating if it is a directory. Typical usages include:

```
SELECT * FROM pg_stat_file('filename');
SELECT (pg_stat_file('filename')).modification;
```

Mais detalhes em: http://www.postgresql.org/docs/8.3/interactive/functions-admin.html

Consultando a Estrutura de uma Tabela através do Catálogo

```
SELECT
        rel.nspname, rel.relname, attrs.attname, "Type", "Default",   attrs.attnotnull
FROM (
        SELECT c.oid, n.nspname, c.relname
        FROM pg_catalog.pg_class   c
        LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
        WHERE pg_catalog.pg_table_is_visible(c.oid)   ) rel
JOIN (
        SELECT a.attname, a.attrelid, pg_catalog.format_type(a.atttypid,   a.atttypmod)
as "Type",
        (SELECT substring(d.adsrc for 128) FROM pg_catalog.pg_attrdef   d
        WHERE d.adrelid = a.attrelid AND d.adnum = a.attnum AND a.atthasdef)
as "Default",
```

a.attnotnull, a.attnum

FROM pg_catalog.pg_attribute a WHERE a.attnum > 0 AND NOT a.attisdropped)

attrs

ON (attrs.attrelid = rel.oid)

WHERE relname = 'clientes' ORDER BY attrs.attnum;

Função em PlPgSQL:

```
CREATE OR REPLACE FUNCTION Dados_Tabela(varchar(30))
RETURNS SETOF tabela_estrutura AS '
 DECLARE
 r tabela_estrutura%ROWTYPE;
 rec RECORD;
 vTabela alias for $1;
 eSql TEXT;

 BEGIN
 eSql := ''SELECT
       CAST(rel.nspname as TEXT), CAST(rel.relname AS TEXT) ,
CAST(attrs.attname AS TEXT), CAST("Type" AS TEXT),
CAST("Default" AS TEXT), attrs.attnotnull
        FROM
               (SELECT c.oid, n.nspname, c.relname
               FROM pg_catalog.pg_class c
               LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.relnamespace
               WHERE pg_catalog.pg_table_is_visible(c.oid) ) rel
        JOIN
               (SELECT a.attname, a.attrelid,
               pg_catalog.format_type(a.atttypid, a.atttypmod) as "Type",
                      (SELECT substring(d.adsrc for 128) FROM
pg_catalog.pg_attrdef d
                      WHERE d.adrelid = a.attrelid AND d.adnum = a.attnum AND
a.atthasdef)
               as "Default", a.attnotnull, a.attnum
               FROM pg_catalog.pg_attribute a
               WHERE a.attnum > 0 AND NOT a.attisdropped ) attrs
        ON (attrs.attrelid = rel.oid )
        WHERE relname LIKE ''''%'' || vTabela || ''%''''
        ORDER BY attrs.attnum'';
FOR r IN EXECUTE eSql
 LOOP
 RETURN NEXT r;
END LOOP;
 IF NOT FOUND THEN
        RAISE EXCEPTION ''Tabela % não encontrada'', vTabela;
 END IF;
 RETURN;
 END
 '
 LANGUAGE 'plpgsql';
```

Usando:

```
SELECT * FROM Dados_Tabela('clientes');
```

Fonte: http://imasters.uol.com.br/artigo/2137/postgresql/consultando_a_estrutura_de_uma_tabela/

Retornando Informações sobre uma Tabela

```
SELECT pg_attribute.attnum AS index,
attname AS field,
typname AS type,
atttypmod-4 as length,
NOT attnotnull AS "null",
adsrc AS def
FROM pg_attribute,
pg_class,
pg_type,
pg_attrdef
WHERE pg_class.oid=attrelid
AND pg_type.oid=atttypid
AND attnum>0
AND pg_class.oid=adrelid
AND adnum=attnum
AND atthasdef='t'
AND lower(relname)='clientes'
UNION
SELECT pg_attribute.attnum AS index,
attname AS field,
typname AS type,
atttypmod-4 as length,
NOT attnotnull AS "null",
'' AS def
FROM pg_attribute,
pg_class,
pg_type
WHERE pg_class.oid=attrelid
AND pg_type.oid=atttypid
```

AND attnum>0

AND atthasdef='f'

AND lower(relname)='clientes';

Use o comando \x no psql antes de executar a consulta caso queira exibir os registros em sequência, \x novamente para voltar ao normal.

Fonte:
http://imasters.uol.com.br/artigo/1283/postgresql/informacoes_atraves_do_catalogo_do_sistema/

Podemos listar tabelas, índices, sequências e vies usando os comandos do psql:

\d{t|i|s|v}

Abaixo veremos mais algumas funções que, ao contrário, usarão o SQL para receber essas informações.

Execute o trecho do script dba_projeto para criação do banco 'catalogo' da Aula 8.

Listando Tabelas do Banco Atual

```
SELECT relname
  FROM pg_class
 WHERE relname !~ '^(pg_|sql_)'
   AND relkind = 'r';

-- using INFORMATION_SCHEMA:

SELECT table_name
  FROM information_schema.tables
 WHERE table_type = 'BASE TABLE'
   AND table_schema NOT IN
       ('pg_catalog', 'information_schema');
```

Listando Views do Banco Atual

```
-- with postgresql 7.2:

SELECT viewname
  FROM pg_views
 WHERE viewname !~ '^pg_';

-- with postgresql 7.4 and later:

SELECT viewname
  FROM pg_views
 WHERE schemaname NOT IN
       ('pg_catalog', 'information_schema')
   AND viewname !~ '^pg_';

-- using INFORMATION_SCHEMA:
```

```
SELECT table_name
  FROM information_schema.tables
 WHERE table_type = 'VIEW'
   AND table_schema NOT IN
       ('pg_catalog', 'information_schema')
   AND table_name !~ '^pg_';

-- or

SELECT table_name
  FROM information_schema.views
 WHERE table_schema NOT IN ('pg_catalog', 'information_schema')
   AND table_name !~ '^pg_';
```

Listando Todos os Usuários

```
SELECT usename
  FROM pg_user;
```

Retornando os nomes dos Campos de uma Tabela

```
SELECT a.attname
  FROM pg_class c, pg_attribute a, pg_type t
 WHERE c.relname = 'test2'
   AND a.attnum > 0
   AND a.attrelid = c.oid
   AND a.atttypid = t.oid;

-- Usando INFORMATION_SCHEMA:

SELECT column_name
  FROM information_schema.columns
 WHERE table_name = 'test2';
```

Informações Detalhadas sobre os Campos de uma Tabela

```
SELECT a.attnum AS ordinal_position,
         a.attname AS column_name,
         t.typname AS data_type,
         a.attlen AS character_maximum_length,
         a.atttypmod AS modifier,
         a.attnotnull AS notnull,
         a.atthasdef AS hasdefault,
         col_description(a.attrelid, a.attnum) as field_comment
    FROM pg_class c,
         pg_attribute a,
         pg_type t
   WHERE c.relname = 'test2'
     AND a.attnum > 0
     AND a.attrelid = c.oid
     AND a.atttypid = t.oid
ORDER BY a.attnum;

-- Usando INFORMATION_SCHEMA:
```

```
  SELECT ordinal_position,
         column_name,
         data_type,
         column_default,
         is_nullable,
         character_maximum_length,
         numeric_precision
    FROM information_schema.columns
   WHERE table_name = 'test2'
ORDER BY ordinal_position;
```

Retornando os Nomes de Índices de uma Tabela

```
SELECT relname
  FROM pg_class
 WHERE oid IN (
    SELECT indexrelid
      FROM pg_index, pg_class
     WHERE pg_class.relname='test2'
       AND pg_class.oid=pg_index.indrelid
       AND indisunique != 't'
       AND indisprimary != 't'
);
```

Retornando as Constraints de uma Tabela

```
SELECT conname
  FROM pg_constraint, pg_class
 WHERE pg_constraint.conrelid = pg_class.oid
   AND relname = 'test2';


-- with INFORMATION_SCHEMA:

SELECT constraint_name, constraint_type
  FROM information_schema.table_constraints
 WHERE table_name = 'test2';
```

Recebendo Informações Detalhadas sobre as Constraints de uma Tabela

```
SELECT c.conname AS constraint_name,
          CASE c.contype
            WHEN 'c' THEN 'CHECK'
            WHEN 'f' THEN 'FOREIGN KEY'
            WHEN 'p' THEN 'PRIMARY KEY'
            WHEN 'u' THEN 'UNIQUE'
          END AS "constraint_type",
          CASE WHEN c.condeferrable = 'f' THEN 0 ELSE 1 END AS is_deferrable,
          CASE WHEN c.condeferred = 'f' THEN 0 ELSE 1 END AS is_deferred,
          t.relname AS table_name,
          array_to_string(c.conkey, ' ') AS constraint_key,
          CASE confupdtype
            WHEN 'a' THEN 'NO ACTION'
            WHEN 'r' THEN 'RESTRICT'
            WHEN 'c' THEN 'CASCADE'
```

```
            WHEN 'n' THEN 'SET NULL'
            WHEN 'd' THEN 'SET DEFAULT'
          END AS on_update,
          CASE confdeltype
            WHEN 'a' THEN 'NO ACTION'
            WHEN 'r' THEN 'RESTRICT'
            WHEN 'c' THEN 'CASCADE'
            WHEN 'n' THEN 'SET NULL'
            WHEN 'd' THEN 'SET DEFAULT'
          END AS on_delete,
          CASE confmatchtype
            WHEN 'u' THEN 'UNSPECIFIED'
            WHEN 'f' THEN 'FULL'
            WHEN 'p' THEN 'PARTIAL'
          END AS match_type,
          t2.relname AS references_table,
          array_to_string(c.confkey, ' ') AS fk_constraint_key
     FROM pg_constraint c
LEFT JOIN pg_class t  ON c.conrelid  = t.oid
LEFT JOIN pg_class t2 ON c.confrelid = t2.oid
    WHERE t.relname = 'testconstraints2'
     AND c.conname = 'testconstraints_id_fk';

-- with INFORMATION_SCHEMA:

   SELECT tc.constraint_name,
          tc.constraint_type,
          tc.table_name,
          kcu.column_name,
          tc.is_deferrable,
          tc.initially_deferred,
          rc.match_option AS match_type,
          rc.update_rule AS on_update,
          rc.delete_rule AS on_delete,
          ccu.table_name AS references_table,
          ccu.column_name AS references_field
     FROM information_schema.table_constraints tc
LEFT JOIN information_schema.key_column_usage kcu
       ON tc.constraint_catalog = kcu.constraint_catalog
      AND tc.constraint_schema = kcu.constraint_schema
      AND tc.constraint_name = kcu.constraint_name
LEFT JOIN information_schema.referential_constraints rc
       ON tc.constraint_catalog = rc.constraint_catalog
      AND tc.constraint_schema = rc.constraint_schema
      AND tc.constraint_name = rc.constraint_name
LEFT JOIN information_schema.constraint_column_usage ccu
       ON rc.unique_constraint_catalog = ccu.constraint_catalog
      AND rc.unique_constraint_schema = ccu.constraint_schema
      AND rc.unique_constraint_name = ccu.constraint_name
    WHERE tc.table_name = 'testconstraints2'
      AND tc.constraint_name = 'testconstraints_id_fk';
```

Listando as Sequências

```
SELECT relname
  FROM pg_class
 WHERE relkind = 'S'
   AND relnamespace IN (
```

```
        SELECT oid
          FROM pg_namespace
         WHERE nspname NOT LIKE 'pg_%'
           AND nspname != 'information_schema'
);
```

Listando Todas as Triggers

```
SELECT trg.tgname AS trigger_name
  FROM pg_trigger trg, pg_class tbl
 WHERE trg.tgrelid = tbl.oid
   AND tbl.relname !~ '^pg_';
-- or
SELECT tgname AS trigger_name
  FROM pg_trigger
 WHERE tgname !~ '^pg_';

-- with INFORMATION_SCHEMA:

SELECT DISTINCT trigger_name
  FROM information_schema.triggers
 WHERE trigger_schema NOT IN
       ('pg_catalog', 'information_schema');
```

Listando Somente as Triggers de uma Tabela

```
SELECT trg.tgname AS trigger_name
  FROM pg_trigger trg, pg_class tbl
 WHERE trg.tgrelid = tbl.oid
   AND tbl.relname = 'newtable';

-- with INFORMATION_SCHEMA:

SELECT DISTINCT trigger_name
  FROM information_schema.triggers
 WHERE event_object_table = 'newtable'
   AND trigger_schema NOT IN
       ('pg_catalog', 'information_schema');
```

Recebendo Informações Detalhadas sobre as Triggers

```
SELECT trg.tgname AS trigger_name,
       tbl.relname AS table_name,
       p.proname AS function_name,
       CASE trg.tgtype & cast(2 as int2)
         WHEN 0 THEN 'AFTER'
         ELSE 'BEFORE'
       END AS trigger_type,
       CASE trg.tgtype & cast(28 as int2)
         WHEN 16 THEN 'UPDATE'
         WHEN  8 THEN 'DELETE'
         WHEN  4 THEN 'INSERT'
         WHEN 20 THEN 'INSERT, UPDATE'
```

```
         WHEN 28 THEN 'INSERT, UPDATE, DELETE'
         WHEN 24 THEN 'UPDATE, DELETE'
         WHEN 12 THEN 'INSERT, DELETE'
       END AS trigger_event
  FROM pg_trigger trg,
       pg_class tbl,
       pg_proc p
 WHERE trg.tgrelid = tbl.oid
   AND trg.tgfoid = p.oid
   AND tbl.relname !~ '^pg_';

-- with INFORMATION_SCHEMA:

SELECT *
  FROM information_schema.triggers
 WHERE trigger_schema NOT IN
       ('pg_catalog', 'information_schema');
```

Listar as Funções

```
SELECT proname
  FROM pg_proc pr,
       pg_type tp
 WHERE tp.oid = pr.prorettype
   AND pr.proisagg = FALSE
   AND tp.typname <> 'trigger'
   AND pr.pronamespace IN (
       SELECT oid
         FROM pg_namespace
        WHERE nspname NOT LIKE 'pg_%'
          AND nspname != 'information_schema'
);

-- with INFORMATION_SCHEMA:

SELECT routine_name
  FROM information_schema.routines
 WHERE specific_schema NOT IN
       ('pg_catalog', 'information_schema')
   AND type_udt_name != 'trigger';
```

Outra mais detalhada:

```
CREATE OR REPLACE FUNCTION public.function_args(
  IN funcname character varying,
  IN schema character varying,
  OUT pos integer,
  OUT direction character,
  OUT argname character varying,
  OUT datatype character varying)
RETURNS SETOF RECORD AS $$DECLARE
  rettype character varying;
  argtypes oidvector;
  allargtypes oid[];
  argmodes "char"[];
```

```
  argnames text[];
  mini integer;
  maxi integer;
BEGIN
  /* get object ID of function */
  SELECT INTO rettype, argtypes, allargtypes, argmodes, argnames
         CASE
         WHEN pg_proc.proretset
         THEN 'setof ' || pg_catalog.format_type(pg_proc.prorettype, NULL)
         ELSE pg_catalog.format_type(pg_proc.prorettype, NULL) END,
         pg_proc.proargtypes,
         pg_proc.proallargtypes,
         pg_proc.proargmodes,
         pg_proc.proargnames
    FROM pg_catalog.pg_proc
         JOIN pg_catalog.pg_namespace
         ON (pg_proc.pronamespace = pg_namespace.oid)
   WHERE pg_proc.prorettype <> 'pg_catalog.cstring'::pg_catalog.regtype
     AND (pg_proc.proargtypes[0] IS NULL
      OR pg_proc.proargtypes[0] <> 'pg_catalog.cstring'::pg_catalog.regtype)
     AND NOT pg_proc.proisagg
     AND pg_proc.proname = funcname
     AND pg_namespace.nspname = schema
     AND pg_catalog.pg_function_is_visible(pg_proc.oid);

  /* bail out if not found */
  IF NOT FOUND THEN
    RETURN;
  END IF;

  /* return a row for the return value */
  pos = 0;
  direction = 'o'::char;
  argname = 'RETURN VALUE';
  datatype = rettype;
  RETURN NEXT;

  /* unfortunately allargtypes is NULL if there are no OUT parameters */
  IF allargtypes IS NULL THEN
    mini = array_lower(argtypes, 1); maxi = array_upper(argtypes, 1);
  ELSE
    mini = array_lower(allargtypes, 1); maxi = array_upper(allargtypes, 1);
  END IF;
  IF maxi < mini THEN RETURN; END IF;

  /* loop all the arguments */
  FOR i IN mini .. maxi LOOP
    pos = i - mini + 1;
    IF argnames IS NULL THEN
      argname = NULL;
    ELSE
      argname = argnames[i];
    END IF;
    IF allargtypes IS NULL THEN
      direction = 'i'::char;
      datatype = pg_catalog.format_type(argtypes[i], NULL);
    ELSE
      direction = argmodes[i];
      datatype = pg_catalog.format_type(allargtypes[i], NULL);
```

```
    END IF;
    RETURN NEXT;
  END LOOP;

  RETURN;
END;$$ LANGUAGE plpgsql STABLE STRICT SECURITY INVOKER;
COMMENT ON FUNCTION public.function_args(character varying, character
varying)
IS $$For a function name and schema, this procedure selects for each
argument the following data:
- position in the argument list (0 for the return value)
- direction 'i', 'o', or 'b'
- name (NULL if not defined)
- data type$$;
```

Do artigo de Lorenzo Alberton. Extracting metadata from PostgreSQL using INFORMATION_SCHEMA :
http://www.alberton.info/postgresql_meta_info.html
Que também traz artigos sobre os catálogos do Firebird, Oracle e SQL Server.

Saber quantidade de registros no banco inteiro:

SELECT sum(C.reltuples)::int FROM pg_class C WHERE c.relkind = 'r'::"char";

Fonte: Blog da Kenia Milene
http://keniamilene.wordpress.com/2007/09/18/contar-registros-em-banco-postgresql/

Listando os bancos e sua codificação
SELECT datname, pg_encoding_to_char(encoding) FROM pg_database;

Exibindo codificações e versões

SELECT name, setting FROM pg_settings
  WHERE name ~ 'encoding|^lc_|version';

Mostrar data da última execução do Vacuum e do Autovacuum

select relname, last_vacuum, last_autovacuum from
pg_stat_all_tables where schemaname like 'public';

Vide o capítulo "Table Statistics" do Livro PostgreSQL The Comprehensive Guide, da Sam's em:
http://www.iphelp.ru/faq/15/ch04lev1sec4.html

Exibir Todas as Tabelas e seus Registros

```
select
        n.nspname as esquema,
        c.relname as tabela, c.reltuples::int as registros
from pg_class c
        left join pg_namespace n on n.oid = c.relnamespace
        left join pg_tablespace t on t.oid = c.reltablespace
where c.relkind = 'r'::char
        and nspname not in('information_schema','pg_catalog', 'pg_toast')
        order by n.nspname,registros;
```

Outra:

```
select relname as tabelas, reltuples
from pg_class
where relkind = 'r'::char and relname not like 'pg_%' and relname not like 'sql_%' order by reltuples;
```

Outra solução:

VACUUM ANALYZE (em todo o banco)

```
SELECT sum(reltuples) as qtd_linhas
FROM pg_class
WHERE relkind = 'r';
```

criar uma view com a instrução SQL

Rodrigo Hjort - http://icewall.org/~hjort – na lista pgbr-geral

Boa fonte de leitura sobre o Catálogo do Sistema do PostgreSQL:
System Tables no capítulo 6 do livro: PostgreSQL Developer's Handbook de Ewald Geschwinde, Hans-Jürgen Schönig.