Lorenzo Alberton

Contact me ·

« **Articles**

Extracting META information from PostgreSQL (INFORMATION_SCHEMA)

PHP, Database, PostgreSQL, INFORMATION_SCHEMA · 08 May 2006 09:48 · 56 comments

Abstract: Extracting META information from a PostgreSQL database using the INFORMATION_SCHEMA views and the system catalogs (pg_class, pg_user, pg_view)

Following my tutorial on how to extract meta informations from Firebird SQL, I'm now going to show how to retrieve the same informations from **PostgreSQL**, using the `INFORMATION_SCHEMA` (available since PostgreSQL 7.4) and with system catalogs (`pg_class, pg_user, pg_view`, etc).
NB: as you probably know, you can list tables/indices/sequences/views from the command line with the `\d{t|i|s|v}` command, but here I want to show how to extract these informations using standard SQL queries.

Test data

We need a few sample tables, indices and views to test the following queries, so let's create them. We also create a sample TRIGGER and a function.

```
1 -- sample data to test PostgreSQL INFORMATION_SCHEMA
2
3 -- TABLE TEST
4 CREATE TABLE TEST (
5   TEST_NAME CHAR(30) NOT NULL,
6   TEST_ID INTEGER DEFAULT '0' NOT NULL,
7   TEST_DATE TIMESTAMP NOT NULL
8 );
9 ALTER TABLE TEST ADD CONSTRAINT PK_TEST PRIMARY KEY (TEST_ID);
10
11-- TABLE TEST2 with some CONSTRAINTs and an INDEX
12CREATE TABLE TEST2 (
13  ID INTEGER NOT NULL,
14  FIELD1 INTEGER,
15  FIELD2 CHAR(15),
16  FIELD3 VARCHAR(50),
17  FIELD4 INTEGER,
18  FIELD5 INTEGER,
19  ID2 INTEGER NOT NULL
20);
21ALTER TABLE TEST2 ADD CONSTRAINT PK_TEST2 PRIMARY KEY (ID2);
22ALTER TABLE TEST2 ADD CONSTRAINT TEST2_FIELD1ID_IDX UNIQUE (ID, FIELD1);
23ALTER TABLE TEST2 ADD CONSTRAINT TEST2_FIELD4_IDX UNIQUE (FIELD4);
24CREATE INDEX TEST2_FIELD5_IDX ON TEST2(FIELD5);
25
26-- TABLE NUMBERS
```

```
27CREATE TABLE NUMBERS (
28  NUMBER INTEGER DEFAULT '0' NOT NULL,
29  EN CHAR(100) NOT NULL,
30  FR CHAR(100) NOT NULL
31);
32
33-- TABLE NEWTABLE
34CREATE TABLE NEWTABLE (
35  ID INT DEFAULT 0 NOT NULL,
36  SOMENAME VARCHAR (12),
37  SOMEDATE TIMESTAMP NOT NULL
38);
39ALTER TABLE NEWTABLE ADD CONSTRAINT PKINDEX_IDX PRIMARY KEY (ID);
40CREATE SEQUENCE NEWTABLE_SEQ INCREMENT 1 START 1;
41
42-- VIEW on TEST
43CREATE VIEW "testview"(
44  TEST_NAME,
45  TEST_ID,
46  TEST_DATE
47) AS
48SELECT *
49FROM TEST
50WHERE TEST_NAME LIKE 't%';
51
52-- VIEW on NUMBERS
53CREATE VIEW "numbersview"(
54  NUMBER,
55  TRANS_EN,
56  TRANS_FR
57) AS
58SELECT *
59FROM NUMBERS
60WHERE NUMBER > 100;
61
62-- TRIGGER on NEWTABLE
63CREATE FUNCTION add_stamp() RETURNS OPAQUE AS '
64    BEGIN
65      IF (NEW.somedate IS NULL OR NEW.somedate = 0) THEN
```

```
        NEW.somedate := CURRENT_TIMESTAMP;
        RETURN NEW;
      END IF;
    END;
' LANGUAGE 'plpgsql';

CREATE TRIGGER ADDCURRENTDATE
BEFORE  INSERT OR UPDATE
ON newtable  FOR EACH  ROW
  EXECUTE PROCEDURE add_stamp();

-- TABLEs for testing CONSTRAINTs
CREATE TABLE testconstraints (
  someid  integer NOT NULL,
  somename  character varying(10) NOT NULL,
  CONSTRAINT testconstraints_id_pk  PRIMARY KEY (someid)
);
CREATE TABLE testconstraints2 (
  ext_id  integer NOT NULL,
  modified  date,
  uniquefield  character varying(10) NOT NULL,
  usraction  integer NOT NULL,
  CONSTRAINT testconstraints_id_fk  FOREIGN KEY (ext_id)
      REFERENCES testconstraints (someid) MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT unique_2_fields_idx UNIQUE (modified, usraction),
  CONSTRAINT uniquefld_idx UNIQUE (uniquefield)
);
```

List TABLEs

Here's the query that will return the names of the tables defined in the current database:

```
SELECT relname
  FROM pg_class
 WHERE relname !~ '^(pg_|sql_)'
   AND relkind = 'r';
<!--
SELECT c.relname AS "Name"
```

```
  FROM pg_class c, pg_user u
 WHERE c.relowner = u.usesysid
   AND c.relkind = 'r'
   AND NOT EXISTS (
       SELECT 1
         FROM pg_views
        WHERE viewname = c.relname
       )
   AND c.relname !~ '^(pg_|sql_)'
UNION
SELECT c.relname AS "Name"
  FROM pg_class c
 WHERE c.relkind = 'r'
   AND NOT EXISTS (
       SELECT 1
         FROM pg_views
        WHERE viewname = c.relname
       )
   AND NOT EXISTS (
       SELECT 1
         FROM pg_user
        WHERE usesysid = c.relowner
       )
   AND c.relname !~ '^pg_';
-->
-- using INFORMATION_SCHEMA:

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

List VIEWs

Here's the query that will return the names of the VIEWs defined in the current database:

```
-- 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_';
<!--
# show only the VIEWs referencing a given table

      SELECT viewname
        FROM pg_views
NATURAL  JOIN pg_tables
       WHERE tablename ='test';
-->
```

List users

```
SELECT usename
```

```
2  FROM pg_user;
```

List table fields

Here's the query that will return the names of the fields of the TEST2 table:

```
1  SELECT a.attname
2    FROM pg_class c, pg_attribute a, pg_type t
3   WHERE c.relname = 'test2'
4     AND a.attnum > 0
5     AND a.attrelid = c.oid
6     AND a.atttypid = t.oid
7
8  -- with INFORMATION_SCHEMA:
9
10 SELECT column_name
11   FROM information_schema.columns
12  WHERE table_name = 'test2';
```

Detailed table field info

If you want some more info about the field definitions, you can retrieve a larger subset of the fields available in the schema:

```
1  SELECT a.attnum AS ordinal_position,
2           a.attname AS column_name,
3           t.typname AS data_type,
4           a.attlen AS character_maximum_length,
5           a.atttypmod AS modifier,
6           a.attnotnull AS notnull,
7           a.atthasdef AS hasdefault
8      FROM pg_class c,
9           pg_attribute a,
10          pg_type t
11    WHERE c.relname = 'test2'
12      AND a.attnum > 0
13      AND a.attrelid = c.oid
14      AND a.atttypid = t.oid
15 ORDER BY a.attnum;
16
```

```
-- with 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;
```

List INDICES

Here's the query that will return the names of the INDICES defined in the TEST2 table. Unfortunately I have no idea how to extract them from the INFORMATION_SCHEMA. If you do, please let me know.
NB: the CONSTRAINTs are not listed

```
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'
       );

-- Alternative using JOINs (thanks to William Stevenson):

SELECT
    c.relname AS index_name
FROM
    pg_class AS a
    JOIN pg_index AS b ON (a.oid = b.indrelid)
    JOIN pg_class AS c ON (c.oid = b.indexrelid)
WHERE
```

```
21    a.relname = 'test2';
```

Detailed INDEX info

If you want to know which table columns are referenced by an index, you can do it in two steps: first you get the table name and field(s) position with this query:

```
1  SELECT relname, indkey
2    FROM pg_class, pg_index
3   WHERE pg_class.oid = pg_index.indexrelid
4     AND pg_class.oid IN (
5      SELECT indexrelid
6        FROM pg_index, pg_class
7       WHERE pg_class.relname='test2'
8         AND pg_class.oid=pg_index.indrelid
9         AND indisunique != 't'
10        AND indisprimary != 't'
11);
```

Then, using your favorite language, you explode the indkey (the key separator is a space), and for each key you run this query:

```
1SELECT t.relname, a.attname, a.attnum
2     FROM pg_index c
3LEFT JOIN pg_class t
4       ON c.indrelid  = t.oid
5LEFT JOIN pg_attribute a
6       ON a.attrelid = t.oid
7      AND a.attnum = ANY(indkey)
8    WHERE t.relname = 'test2'
9      AND a.attnum = 6; -- this is the index key
```

List CONSTRAINTs

Here's the query that will return the names of the CONSTRAINTs defined in the TEST2 table:

```
1  SELECT relname
2    FROM pg_class
3   WHERE oid IN (
4     SELECT indexrelid
5       FROM pg_index, pg_class
```

```
      WHERE pg_class.relname='test2'
        AND pg_class.oid=pg_index.indrelid
        AND (   indisunique = 't'
             OR indisprimary = 't'
        )
);

-- with INFORMATION_SCHEMA:

SELECT constraint_name, constraint_type
  FROM information_schema.table_constraints
 WHERE table_name = 'test2';
<!--
SELECT *
  FROM information_schema.constraint_column_usage
-->
```

Detailed CONSTRAINT info

If you want to retrieve detailed info from any constraint (fields, type, rules, referenced table and fields for FOREIGN KEYs, etc.) given its name and table, here's the query to do so:

```
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';
```

The "`constraint_key`" and "`fk_constraint_key`" fields returned by the first query are space-separated strings containing the position of the fields involved (in the `FOREIGN KEY` constraint and those referenced by it), so you may need to retrieve them with another query on the respective tables. Since the field positions are stored as arrays, you can't (to the best of my knowledge) get all the field names with an unique query (well, you could with a stored procedure).
The second query, the one using the `INFORMATION_SCHEMA`, is certainly more straightforward, albeit slower.

List sequences

A **SEQUENCE** is an object that automatically generate sequence numbers. A `SEQUENCE` is often used to ensure a unique value in a `PRIMARY KEY` that must uniquely identify the associated row.

```
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'
);
```

List 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');
```

List only the triggers for a given table:

```
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');
```

Detailed TRIGGER info

Show more informations about the trigger definitions:

```
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,
      CASE trg.tgtype & cast(1 as int2)
        WHEN 0 THEN 'STATEMENT'
        ELSE 'ROW'
      END AS action_orientation
 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');
```

List FUNCTIONs

```
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';
```

Albe Laurenz sent me the following function that is even more informative: for a function name and schema, it selects the position in the argument list, the direction, the name and the data-type of each argument. This procedure requires PostgreSQL 8.1 or later.

```
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[pos];
  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$$;
```

Show PROCEDURE definition

```
SELECT p.proname AS procedure_name,
           p.pronargs AS num_args,
           t1.typname AS return_type,
           a.rolname AS procedure_owner,
           l.lanname AS language_type,
           p.proargtypes AS argument_types_oids,
           prosrc AS body
      FROM pg_proc p
 LEFT JOIN pg_type t1 ON p.prorettype=t1.oid
LEFT JOIN pg_authid a ON p.proowner=a.oid
LEFT JOIN pg_language l ON p.prolang=l.oid
    WHERE proname = :PROCEDURE_NAME;
```

pg_proc.proargtypes contains an array of oids pointing to pg_type.oid. You can use unnest(), generate_procedure() or the function in the previous paragraph to retrieve the data type of each parameter.

What else?

If you'd like to see some other example queries, or have some comments and/or suggestions, just drop me a mail (you can find my address in the footer of this page) and I'll add them to this list.
HTH.

Back

---

56 responses to "Extracting META information from PostgreSQL (INFORMATION_SCHEMA)"

james, 23 June 2009 04:44

hi,
reading through your queries, really good stuff. One slight but critical part is missing and I am wondering if you could add it.
Under the section 'Detailed CONSTRAINT info' you have 2 queries that retrieve index information, I am interested in getting 4 columns for foreign keys, src_table, src_column, dst_table, dst_column, your 2nd information_schema sql can do this fine but it is slowww, your longer query above that one is much faster but is missing just one column, the dst_column (foreign table column name) and I have tried but cant find how to reliable retrieve it as I dont have an ERD for the pg_* tables.
Any help is much appreciated, thanks!

Lorenzo Alberton, 27 June 2009 23:23

Hi, what you want is a not easy with one single query, because to get the name of the fields you have to do a JOIN on pg_attribute, but the pg_constraint conkey and confkey fields are arrays (listing the column numbers within the tables in the constraint). If you know your FOREIGN KEY constraint is on a single field, you can get what you want with this query:

```
1    SELECT c.conname AS constraint_name,
2           t.relname AS table_name,
3           a.attname AS field_name,
4           t2.relname AS references_table,
5           a2.attname AS references_field
6      FROM pg_constraint c
7 LEFT JOIN pg_class t  ON c.conrelid  = t.oid
8 LEFT JOIN pg_class t2 ON c.confrelid = t2.oid
9 LEFT JOIN pg_attribute a ON t.oid = a.attrelid AND a.attnum = c.conkey[1]
10LEFT JOIN pg_attribute a2 ON t2.oid = a2.attrelid AND a2.attnum = c.confkey[1]
11    WHERE t.relname = 'testconstraints2'
12     AND c.conname = 'testconstraints_id_fk'
13     AND c.contype = 'f';
```

If the FK is on multiple columns, then either you parse the conkey/confkeys arrays and do another query to select the names of the matching columns, or you try something like generate_subscripts(): http://www.postgresql.org/docs/8.4/static/arrays.html

Grant Paton-Simpson, 24 August 2009 03:52

Thank you so much for this information. I am building a postgresql module for my open source statistics package SOFA Statistics and I found it very hard to get the required information in a useful form.

pepy, 10 September 2009 08:05

Lorenzo hi! Thank you for sharing your knowledge with us. I found your notes very useful. Have you got anything similar concerning how to retrieve users queries on a postgres database? I 've tried pg_stat_activity but it only shows current queries and only select queries (and not update, insert, delete). I 've tried pg_log directory by changing in the conf file the log_line_prefix and log_statement= 'all' but it has implications on the load (creates extreme big log file). Is there any easy way to track users queries? Thank you Pepy

Jignesh, 02 October 2009 13:37

Hi, I am new to PostgreSQL. I have started learning and written triggers and they are working fine. Currently I am using spi_exe_query to insert/update my log table when primary table updated. Could you please give me an example of spi_prepare and spi_execute_plan usage in PL/Perl that I can use in my trigger. I couldn\'t find any example in documentation.
Basically, I want to prepare a plan and want to store it in %_SHARED hash so that I can use whenever I want. Moreover, if table definition changed then I should purge the cache.
Any suggestion would be much appreciated.
Thanks, Jignesh

newbie, 13 October 2009 09:53

Thanks Lorenzo.
Could you provied an example for conkey/confkeys arrays parsing in the case of multiple columns in fk ?

Jordi, 14 October 2009 10:23

When you talk about to List Indices you said you don't have no idea to find using information schema. I found this way:

```
1SELECT * FROM information_schema.table_constraints  WHERE table_name = 'test'
```

In this table you have 'constraint_type' to know the index type.

And, in addition, I found this select if someone are interested to know primary key(s) of one table:

```
SELECT * FROM information_schema.key_column_usage WHERE table_name = 'test'
```

Lorenzo Alberton, 14 October 2009 20:24

Jordi, the first query you mention only lists constraints, not indices. The second one is not entirely correct either, since it will list all the columns having some constraint. A better way of listing the PRIMARY KEYs is:

```
SELECT constraint_name
FROM information_schema.table_constraints
WHERE table_name = 'test' and constraint_type = 'PRIMARY KEY';
```

Or, if you are interested in the full details for the PK, this one:

```
    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 = 'test'
```

```
26      AND tc.constraint_type = 'PRIMARY KEY';
```

Lorenzo Alberton, 14 October 2009 20:33

Jignesh, have a look at my article about table auditing: http://www.alberton.info/postgresql_table_audit.html. It's not something I would recommend on a live system, but it's OK if you only want to play around.

jjp, 21 October 2009 23:03

i'm getting some strange results for both information schema queries for detailed pk's and detailed constraint's. looks almost like a cartesian product.
how portable is information_schema, i hear its a standard but is this like browser standards or real standards?

jjp, 21 October 2009 23:25

re: my last comment, i neglected to notice or mention that for detailed constraint info i left off the constraint name in where clause.
if i modify the above PK query to look for tc.constraint_type = 'FOREIGN KEY' i get odd results as well.
any idea how to query the information schema to get full FK details on a given table without specifying the constraint name?

Pugazendhi Asaimuthu, 01 December 2009 20:59

Thanks Lorenzo. You have an excellent resource for us here. In spite of all this I'm unable to solve my problem. Can you please help with this. I need to query column statistics to get tablename, rowcount, columnname, no. of nulls, null percentage and no. of distinct values. Thanks for your help.

Nakh, 26 January 2010 13:57

Thanks for sharing it, a great post !

KJ, 05 February 2010 09:05

Hi,I am new to PostgreSql.I found your material very helpful.I am trying to write a generic function for auditing databases using pl/pgsql.I tried to use the same logic as you have used in your function using pl/tcl (http://www.alberton.info/postgresql_table_audit.html) but the use of new and old is giving errors.Is it not possible to convert new and old into arrays and then use them in pl/pgsql?If yes,then can you please help me out in this? Thank You

Lorenzo Alberton, 05 February 2010 09:36

@KJ: unfortunately not, the reason why I used tcl instead of pl/pgsql is exactly that one: you can't use NEW and OLD with variable identifiers for the field names.

KJ, 08 February 2010 11:56

I am really Thankful to you for such a quick response.....I guess using another language would be a better option....

Bruno Faria, 09 February 2010 13:54

Hi! I need get the function body. How do I can get it? Thks

Lorenzo Alberton, 10 February 2010 21:56

@Bruno: I updated the article with the query you need, to fetch the stored procedure body. HTH.

Bruno, 12 February 2010 15:49

Hi Alberton, Thk to you for a quick solution. it works!

Sameer, 26 February 2010 07:12

Do you know how to extract more information about the sequences in Postgres ??
Basically I need the sequence name (which I got), its start value, increment value and max value (I tried the information_schema.sequences but dont see any inofrmation except names)

Lorenzo Alberton, 28 February 2010 15:21

@Sameer Unfortunately I don't know. The INFORMATION_SCHEMA.SEQUENCE view is incomplete (as you can see from the definition [1], it's returning NULL for those values), probably waiting for the relevant info to be exposed. Looking at the sequence.c file [2], it looks like it shouldn't be that hard to add a function returning the sequence info.
[1] http://git.postgresql.org/gitweb?p=postgresql.git;a=blob_plain;f=src/backend/catalog/information_schema.sql;hb=HEAD
[2] http://git.postgresql.org/gitweb?p=postgresql.git;a=blob_plain;f=src/backend/commands/sequence.c;hb=HEAD

rbn, 10 March 2010 20:50

In regards to constraint meta data. I've been attempting to extract information about constraints as well, mostly in regard to foreign keys. I started with a query from stack overflow that I found (http://stackoverflow.com/questions/1567051/introspect-postgresql-8-3-to-find-foreign-keys) and modified it somewhat. This isn't quite what I need, but it is a start, and maybe it will help others too. What I'm interested in doing with this, eventually, is also including field counts from tables with those foreign references. I'm still simply trying learn about the pg_catalogs and meta data and how to best extract meaningful information from them.

Here's a query that gives the four column output mentioned in the conversation about constraints, albeit, only for foreign keys. I didn't write the original version, and I've changed it a little bit, possibly for the worse, but I've learned a lot toying with it. Any help or comments or corrections would be appreciated.

```
SELECT conname,
       conrelid as "table",
       pgc.relname as "tabname",
       a.attname as columns,
       confrelid as "foreign table",
       pgf.relname as ftabname,
       af.attname as "fcolumn"
  FROM pg_attribute AS af,
       pg_attribute AS a,
       pg_class pgc,
       pg_class pgf,
       ( SELECT conname,
                conrelid,
                confrelid,
                conkey[i] AS conkey,
                confkey[i] as confkey
           FROM ( SELECT conname,
                         conrelid,
                         confrelid,
                         conkey,
                         confkey,
                         generate_series(1, array_upper(conkey, 1)) AS i
                    FROM pg_constraint
                   WHERE contype = 'f' ) AS ss
          ) AS ss2
 WHERE af.attnum = confkey
   AND af.attrelid = confrelid
   AND a.attnum = conkey
   AND a.attrelid = conrelid
   AND pgf.relfilenode = confrelid
   AND pgc.relfilenode = conrelid
ORDER BY ftabname, fcolumn, tabname, columns
```

Glen Jarvis, 24 March 2010 00:39

This page has been an incredible resource. I have it bookmarked and I been using it for two weeks as a reference.
I noticed that you use information_schema in many places. This is incredibly trivial compared to the system table. However, for completeness, I noticed you didn\'t have the information_schema for enabled_rules (instead of pg_user);
select * from information_schema.enabled_roles ;

Axeia, 02 April 2010 23:13

Thank you for sharing, some of these are really hard to find especially if you're accustomed to other databases that list things somewhat more readable for us poor humans.

Greg Lindstrom, 07 April 2010 01:55

Thanks for the postgres tutorial. How can I pull the comments on the database, tables and column? I've looked at the database dump but can't figure it out.
Thanks, --greg

Sara, 07 April 2010 06:21

Thank you so much for this information

Lorenzo Alberton, 13 April 2010 11:07

@Greg: Given this sample table:

```
1CREATE TABLE test_comments (
2  id integer NOT NULL DEFAULT 0,
3  testfield character varying(10),
4  CONSTRAINT test_comments_pk PRIMARY KEY (id)
5);
6COMMENT ON SCHEMA public IS 'For  testing comments (schema).';
7COMMENT ON TABLE test_comments IS 'For testing comments (table).';
8COMMENT ON COLUMN test_comments.testfield IS 'For testing comments (field).';
```

Here's how you can retrieve schema, table and column comments with one query:

```
1   SELECT d.nspname AS schema_name,
2          pg_catalog.obj_description(d.oid) AS schema_comment,
3          c.relname AS table_name,
4          pg_catalog.obj_description(c.oid) AS table_comment,
5          a.attnum AS ordinal_position,
```

```
6           a.attname AS column_name,
7           t.typname AS data_type,
8           a.attlen AS character_maximum_length,
9           pg_catalog.col_description(c.oid, a.attnum) AS field_comment,
10          a.atttypmod AS modifier,
11          a.attnotnull AS notnull,
12          a.atthasdef AS hasdefault
13     FROM pg_class c
14LEFT JOIN pg_attribute a ON a.attrelid = c.oid
15LEFT JOIN pg_type t ON a.atttypid = t.oid
16LEFT JOIN pg_namespace d ON d.oid = c.relnamespace
17    WHERE c.relname = 'test_comments'
18      AND a.attnum > 0
19ORDER BY a.attnum;
```

HTH

masc, 21 April 2010 16:22

here's how to retrieve index information for a specific table in one shot.

```
1  SELECT a.index_name, b.attname
2    FROM (
3      SELECT a.indrelid,
4             c.relname index_name,
5             unnest(a.indkey) index_num
6        FROM pg_index a,
7             pg_class b,
8             pg_class c
9       WHERE b.relname='tablename'
10        AND b.oid=a.indrelid
11        AND a.indisprimary != 't'
12        AND a.indexrelid=c.oid
13        ) a,
14        pg_attribute b
15 WHERE a.indrelid = b.attrelid
16    AND a.index_num = b.attnum
17 ORDER BY a.index_name, a.index_num
```

Craigbert, 13 May 2010 19:32

Lorenzo, WOW! Thank you so much for putting this together! The sample database you have at the top is great, as is the rest of the stuff you have here! I do have one question. I am trying to gather index information and I see that you have a two step query for that. I would like to know how to determine whether the index is unique or not and the position of the fields within the index. Can that be done? THANKS!!!!

Lorenzo Alberton, 14 May 2010 09:52

@masc: thanks! You the man.
@Craigbert: starting from masc's query, here's how you can get the info you need:

```
SELECT a.index_name,
       b.attname,
       b.attnum,
       a.indisunique,
       a.indisprimary
  FROM ( SELECT a.indrelid,
                a.indisunique,
                a.indisprimary,
                c.relname index_name,
                unnest(a.indkey) index_num
           FROM pg_index a,
                pg_class b,
                pg_class c
          WHERE b.relname='test2'
            AND b.oid=a.indrelid
            AND a.indexrelid=c.oid
       ) a,
       pg_attribute b
 WHERE a.indrelid = b.attrelid
   AND a.index_num = b.attnum
ORDER BY a.index_name,
       a.index_num
```

Macgayver, 27 May 2010 01:05

Oh Great, very good code. Tanks for Contibuition. I From Brazil.

Jason, 02 June 2010 15:55

Thanks a lot! Very useful!

Chuck, 21 June 2010 20:37

I need to retrieve a functions "CONTEXT". Basically, I want to retrieve what functions called a sub-function. I know this information is available because it displays in the pgsql log file. Great resource you have here.

Wassim, 23 June 2010 17:40

Hi Lorenzo, is there a query to display the actual trigger create statement:
(CREATE TRIGGER some_trigger BEFORE INSERT OR UPDATE ON mytable FOR EACH ROW EXECUTE PROCEDURE some_function();) ,
or we have to construct it based on the information from this query "SELECT * FROM information_schema.triggers WHERE trigger_schema NOT IN ('pg_catalog', 'information_schema');"
Thanks

Lucas, 13 September 2010 06:12

Lorenzo,
this is great :). Thanks, you made my day, today :P
thx Lucas.

chrelad, 28 September 2010 22:34

This is a great reference, thanks!

VIlius, 15 November 2010 19:06

YOu are GOD :) this reference is so amazing!

Mario Majcica, 10 December 2010 14:25

Great contrib, thanks for whole series of articles about schema extrapolation!

Tony, 27 January 2011 21:23

how i obtain the ddl of objets in postgres? for example any table
please!!!

Zatman, 22 March 2011 09:51

If I have one table Foo with column row_id that is being referenced by table FooBar.foo_id and also another table FooBee.foo_id, could you give an example on how to find if row_id X in Foo is being referenced by another table?
Thanks

Amit, 29 April 2011 12:39

Can you get a query that show the List of dependent object for a given table/view.

Lorenzo Alberton, 29 April 2011 20:56

@Amit: let's add some context to your question: supposing we have a view dependent on another view, like in this case:

```
CREATE VIEW test2_view1 AS (
   SELECT id, field1, field2
     FROM test2
    WHERE id2 > 10
 );
CREATE VIEW test2_view2 AS (
   SELECT id, field1, field2
     FROM test2_view1
    WHERE id > 5
);
```

if we try to drop the first view, then we get an error:

```
DROP VIEW test2_view1;
-- ERROR:  cannot drop view test2_view1 because other objects depend on it
-- DETAIL:  view test2_view2 depends on view test2_view1
-- HINT:  Use DROP ... CASCADE to drop the dependent objects too.
```

To show the dependent objects (either another view or a table) for the first view we can run this query:

```
   SELECT v.relname AS "dependent_view",
          t.relname AS "referenced_relation"
     FROM pg_depend dv
LEFT JOIN pg_class v ON v.oid = dv.refobjid
LEFT JOIN pg_namespace nv ON v.relnamespace = nv.oid
LEFT JOIN pg_depend dt
       ON dv.classid = dt.classid
```

```
8       AND dv.objid = dt.objid
9       AND dv.refobjid <> dt.refobjid
10      AND dv.refclassid = dt.refclassid
11      AND dv.classid = 'pg_catalog.pg_rewrite'::regclass
12      AND dv.refclassid = 'pg_catalog.pg_class'::regclass
13LEFT JOIN pg_class t ON t.oid = dt.refobjid
14LEFT JOIN pg_namespace nt
15       ON t.relnamespace = nt.oid
16      AND nv.nspname = 'public'
17      AND nt.nspname = 'public'
18    WHERE dv.deptype = 'i'
19      AND v.relkind = 'v'
20      AND t.relkind IN ('r', 'v')
21      AND v.relname = 'test2_view2' -- VIEW NAME
22 GROUP BY v.relname, t.relname;
```

HTH

Amit Jain, 30 April 2011 19:45

In a two word You are GREAT and GENIOUS !!!!

Jacek Wysocki, 25 May 2011 09:18

Hi, you save My day :) thanks!

Tim Holahan, 26 June 2011 15:04

This page has been very helpful to me several times in the past several years in which I've been using postgres. I'm grateful to the author.
Several times I've used this information to figure out how to grant permissions to all objects in a schema to a particular user.
I thought it might be helpful to mention that, as of 9.0, postgres does have the syntax to grant privileges on all tables (as well as other objects) in a schema:

```
1GRANT SELECT ON ALL TABLES  IN SCHEMA public TO user;
2GRANT EXECUTE ON ALL FUNCTIONS  IN SCHEMA public TO user;
```

Here's the link:
http://www.postgresql.org/docs/current/interactive/sql-grant.html

Hendri Soni, 06 September 2011 09:53

thank you, this is what I was looking.

Les, 03 October 2011 08:06

Thanks - I\'d never have worked it out myself

vee, 31 October 2011 10:31

Thank you Lorenzo... this post is very helpfull for me... thank you very much... :D but, 1 question, is there a query to display the source script of the procedure?

```
1(BEGIN
2IF (NEW.somedate IS NULL OR NEW.somedate = 0) THEN
3NEW.somedate := CURRENT_TIMESTAMP;
4RETURN NEW;
5END IF;
6END;)
```

Nirmal, 10 November 2011 06:31

How to find the dependencies in procedure and function like Tables, View etc

Aditi, 11 November 2011 09:42

Hello, I find this article very useful... I am working on query optimization in PostgreSql. Can you tell me how to make changes in PostgreSql Metadata.How should I go about it as I am making small changes in the queries... Plz reply asap. Regards, Aditi

Nirmal, 11 November 2011 11:40

Please tell me how to get the name of tables which is in procedures.

Ron, 27 November 2011 14:36

Change all references to "unique_constraint_*" to "constraint_*" in your joins. The way it is now, it will only work for unique constraints. By removing the "unique_" prefix, it should work for all constrains, including non-unique constraints. I'm using it for foreign keys without the "unique_"; prefix, and it seems to be working fine.

Pankaj Shinde, 02 December 2011 07:02

Hi Lorenzo, we are migrating our application from oracle DB to Postgres DB. in which one of piece of code requires metadata. So i am converting all oracle queries to Postgres. But here i am stucked can you please help me in this regard. The Oracle query is

```
SELECT CONSTRAINT_NAME,
       CONSTRAINT_TYPE,
       TABLE_NAME,
       R_OWNER,
       R_CONSTRAINT_NAME,
       INDEX_OWNER,
       INDEX_NAME
 FROM user_constraints
WHERE STATUS='ENABLED'
ORDER BY constraint_type, TABLE_NAME
```

I have written equivalent postgres Query as:

```
SELECT a.constraint_name,
       b.constraint_type,
       b.table_name,
       a.constraint_schema,
       a.unique_constraint_name
FROM information_schema.referential_constraints a,
       information_schema.table_constraints b
WHERE a.constraint_name = b.constraint_name
ORDER BY b.constraint_type, b.table_name
```

but its not returning the index_name and index_owner how to get it? please help, its urgent!

Lorenzo Alberton, 04 December 2011 22:47

@Pankaj, I'm not sure this is what you are looking for, give it a go:

```
SELECT i.indkey AS "index_pos",
       co.conname AS "constraint_name",
       CASE co.contype
          WHEN 'p' THEN 'PRIMARY KEY'
          WHEN 'u' THEN 'UNIQUE'
       END AS "constraint_type",
```

```
7         oc.rolname AS "constraint_owner",
8         clt.relname AS "table_name",
9         ot.rolname AS "table_owner",
10        cli.relname AS "index_name",
11        oi.rolname AS "index_owner"
12   FROM pg_index i
13   LEFT JOIN pg_constraint co ON co.conindid = i.indexrelid
14   LEFT JOIN pg_class cli ON i.indexrelid = cli.oid
15   LEFT JOIN pg_class clt ON i.indrelid = clt.oid
16   LEFT JOIN pg_class clc ON co.conrelid = clc.oid
17   LEFT JOIN pg_authid oi ON oi.oid = cli.relowner
18   LEFT JOIN pg_authid ot ON ot.oid = clt.relowner
19   LEFT JOIN pg_authid oc ON oc.oid = clc.relowner
20  WHERE (i.indisprimary OR i.indisunique)
21    AND clt.relname = 'test2';
```

Pankaj Shinde, 05 December 2011 09:43

Hi Lorenzo, it helped me to find out solution. i got the desired output with following query:

```
1  SELECT pc.oid,
2              conname as constraint_name,
3              contype as constraint_type,
4              table_name,
5              rc.unique_constraint_schema as r_owner,
6              rc.unique_constraint_name as r_constraint_name,
7              index_owner,
8              indexname
9  from pg_constraint pc
10 INNER JOIN (SELECT * FROM
11                  (SELECT oid as tableId,
12                              relname as table_name
13                     FROM pg_class
14                     where relkind='r') A,
15                  (SELECT pg_class.oid as indexId,
16                              pg_class.relname as indexname,
17                              information_schema.schemata.schema_owner as index_owner
18                     FROM information_schema.schemata, pg_class
19                     where relkind='i') B
```

```
20                  ) AB
21          ON conrelid=tableId and conindid=indexId
22left join information_schema.referential_constraints rc on rc.constraint_name = pc.conname
23order by constraint_type, table_name
```

thanx a lot.

Bruno, 01 February 2012 16:57

Hi,
I am looking for the grants that one user(role) has. For example, if I give the grant bellow:
grant select on tablektx to bruno;
Now, I want to know which grants the user \"bruno\" has.
Thanks

Related articles

- Extracting META information from Interbase/Firebird SQL (INFORMATION_SCHEMA)
- Extracting META information from Oracle (INFORMATION_SCHEMA)
- Extracting META information from SQL Server (INFORMATION_SCHEMA)
- Updated articles to extract META informations from databases

Latest articles

- On batching vs. latency, and jobqueue models
- Updated Kafka PHP client library
- Musings on some technical papers I read this weekend: Google Dremel, NoSQL comparison, Gossip Protocols
- Historical Twitter access - A journey into optimising Hadoop jobs
- Kafka proposed as Apache incubator project
- NoSQL Databases: What, When and Why (PHPUK2011)
- PHPNW10 slides and new job!

Filter articles by topic

AJAX, Apache, Book Review, Charset, Cheat Sheet, Data structures, Database, Firebird SQL, Hadoop, Imagick, INFORMATION_SCHEMA, JavaScript, Kafka, Linux, Message Queues, mod_rewrite, Monitoring, MySQL, NoSQL, Oracle, PDO, PEAR, Performance, PHP, PostgreSQL, Profiling, Scalability, Security, SPL, SQL Server, SQLite, Testing, Tutorial, TYPO3, Windows, Zend Framework

---

Search

Follow @lorenzoalberton

Back