Integrating PostgreSQL Documentation in 3rd Party Applications

An example with MediaWiki

Charles Clavadetscher

Swiss PostgreSQL Users Group

Punto d'incontro, Zurich, Switzerland, 19.01.2017

Outline

1. Introduction
2. Documenting Resources
3. The Old Way
4. Moving On
5. The Magic of Integration
6. The Magic Explained
7. Conclusion

Charles Clavadetscher

- Senior DB Engineer at KOF ETH Zurich
  - KOF is the Center of Economic Research of the
  - ETHZ the Swiss Institute of Technology in Zurich, Switzerland
  - Independent economic research on business cycle tendencies for almost all sectors
  - Maintenance of all databases at KOF: PostgreSQL, Oracle, MySQL and MSSQL Server. Focus on migrating to PostgreSQL
  - Support in business process re-engineering
- Co-founder and treasurer of the SwissPUG, the Swiss PostgreSQL Users Group
- Member of the OK of the Swiss PGDay

Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

Why bother ?

Documenting resources such as software or database objects is one of those activities that suffer most:

- Companies are often unwilling to pay for a decent documentation.
- Developers find it cumbersome and boring to document their objects.

But without a good documentation:

- New developers spend a huge amount of time figuring out from the code how they need to proceed in order to make changes.
- Code is for the authors themselves something new after a few weeks without working on it, i.e. the developer has no big advantage compared to somebody new.

The bottom line is that a good documentation increases the efficiency of IT human resources leading to a reduction of development and maintenance costs. It is in the best interest of a company and its developers not only to support it financially, but also require a commitment to it from all people involved.

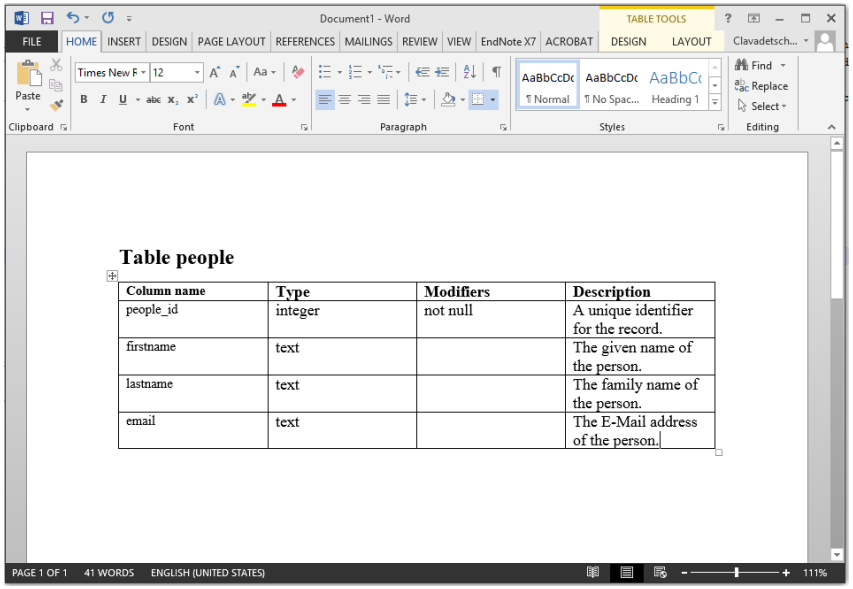
Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

From...

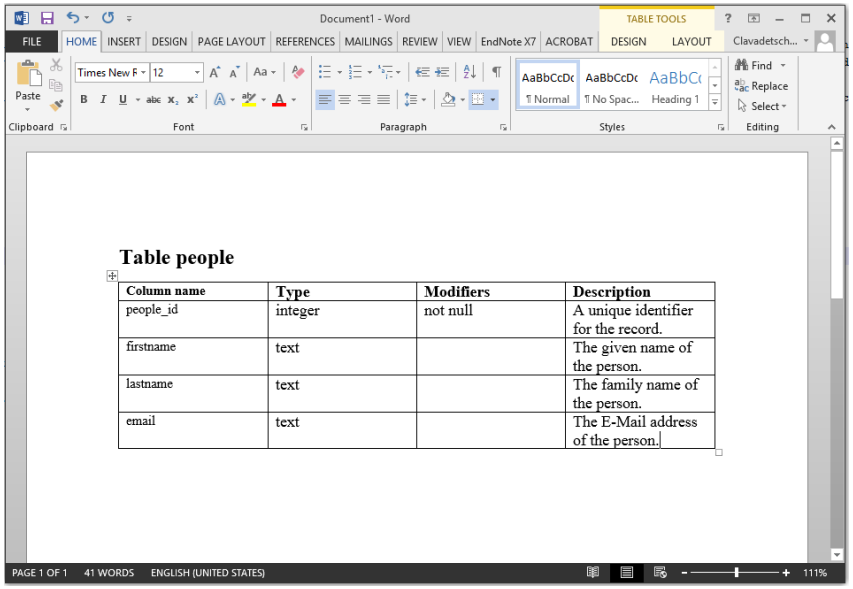
```
db=> CREATE SCHEMA test;

db=> CREATE TABLE test.people
     (
       people_id INTEGER,
       firstname TEXT,
       lastname TEXT,
       email TEXT
     );

db=> CREATE UNIQUE INDEX people_id_idx ON test.people(people_id);

db=> ALTER TABLE test.people ADD PRIMARY KEY USING INDEX people_id_idx;

db=> \d test.people
       Table "test.people"
  Column   |  Type   | Modifiers
-----------+---------+-----------
 people_id | integer | not null
 firstname | text    |
 lastname  | text    |
 email     | text    |
Indexes:
    "people_id_idx" PRIMARY KEY, btree (people_id)
```

... to



Where do the descriptions come from ?

... to



Where do the descriptions come from ? They were entered manually - the good old way.

Fair enough ?

Fair enough ?

Well...

```
db=> ALTER TABLE test.people ADD COLUMN phone TEXT;

db=> \d test.people
      Table "test.people"
  Column   |  Type   | Modifiers
-----------+---------+-----------
 people_id | integer | not null
 firstname | text    |
 lastname  | text    |
 email     | text    |
 phone     | text    |
Indexes:
    "people_id_idx" PRIMARY KEY, btree (people_id)
```

Fair enough ?

Well...

```
db=> ALTER TABLE test.people ADD COLUMN phone TEXT;

db=> \d test.people
      Table "test.people"
  Column   |  Type   | Modifiers
-----------+---------+-----------
 people_id | integer | not null
 firstname | text    |
 lastname  | text    |
 email     | text    |
 phone     | text    |
Indexes:
    "people_id_idx" PRIMARY KEY, btree (people_id
```

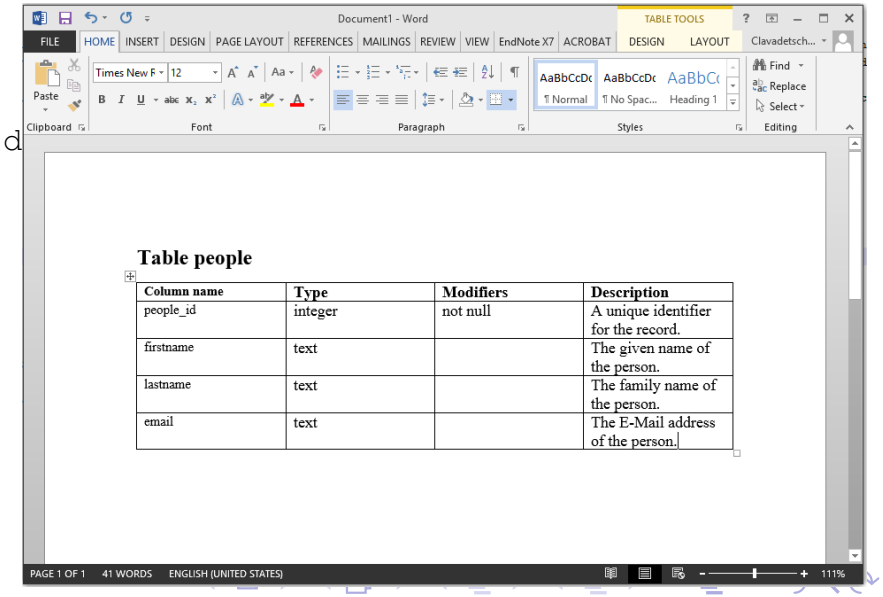


Table people

Column name	Type	Modifiers	Description
people_id	integer	not null	A unique identifier for the record.
firstname	text		The given name of the person.
lastname	text		The family name of the person.
email	text		The E-Mail address of the person.

Didn’t you forget something ?

Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

Document In the Database

```
db=> COMMENT ON TABLE test.people IS E'A table with people';
db=> COMMENT ON COLUMN test.people_id IS E'The unique ID of a person.';
db=> COMMENT ON COLUMN test.firstname IS E'The given name of a person.';
db=> COMMENT ON COLUMN test.lastname IS E'The family name of a person.';
db=> COMMENT ON COLUMN test.email IS E'The E-Mail address of a person.';
db=> COMMENT ON COLUMN test.phone IS E'The phone number of a person.';
db=> COMMENT ON INDEX people_id_idx IS E'The primary key identifier of the table.';

db=> \d+ people
                    Table "test.people"
  Column   |  Type   | Modifiers |           Description
-----------+---------+-----------+---------------------------------
 people_id | integer | not null  | The unique ID of a person.
 firstname | text    |           | The given name of a person.
 lastname  | text    |           | The family name of a person.
 email     | text    |           | The E-Mail address of a person.
 phone     | text    |           | The phone number of a person.
Indexes:
    "people_id_idx" PRIMARY KEY, btree (people_id)

db=> \di+ people_id_idx
     Name      | Type  |  Owner   | Table  |                Description
---------------+-------+----------+--------+-----------------------------------------
 people_id_idx | index | charles | people | The primary key identifier of the table.
```

Why In the Database ?

Why In the Database ?

Because your documentation in the database is as safe as your data itself.

Why In the Database ?

Because your documentation in the database is as safe as your data itself.

- Texts written with anything a bit more complex than plain text (e.g. Word and similar strange programs) are not likely to be readable with new versions of the software after some time.
- If you have copies on paper they will likely to be out-of-date soon.
- If you published electronic copies on a server your documents can get lost in a crash or suffer from the same problems as printed items.

So the question is: How can you synchronize your documentation with a different medium as quick as possible and spending as little effort as possible ?

Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

A small demo...

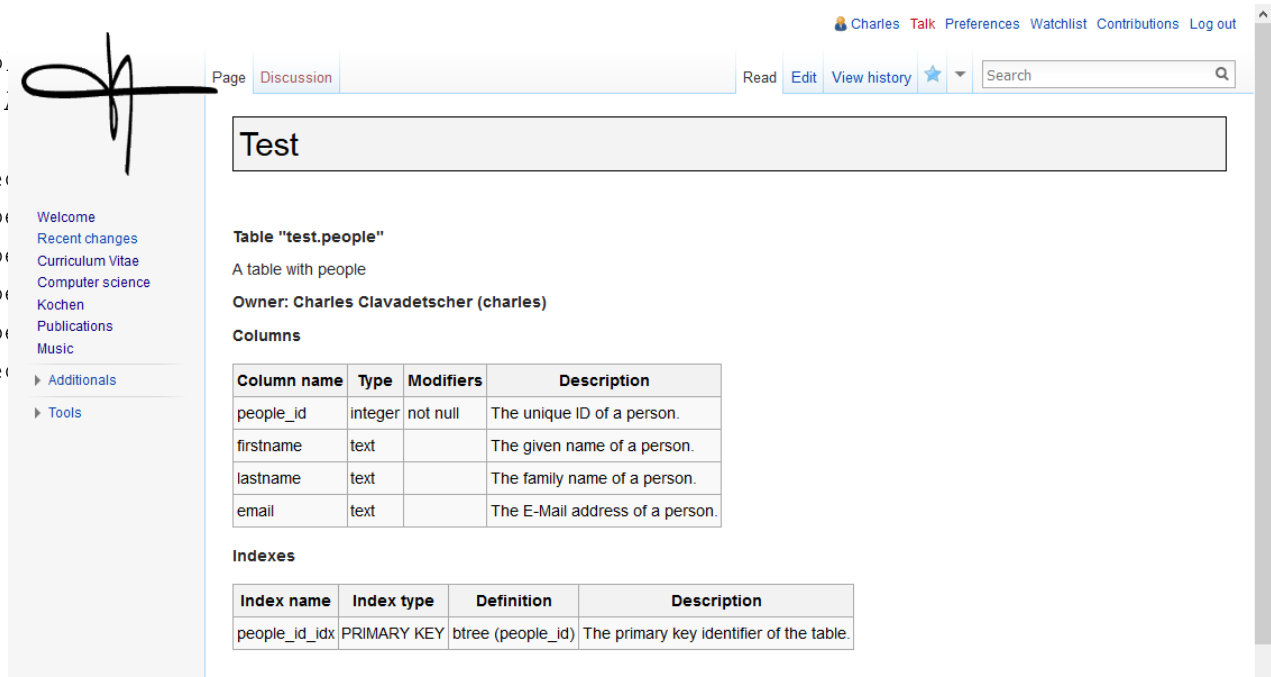
At the beginning: An empty page.

Create a Table

```
db=> CREATE TABLE test.people
     (
       people_id INTEGER,
       firstname TEXT,
       lastname TEXT,
       email TEXT
     );
db=> CREATE UNIQUE INDEX people_id_idx ON test.people(people_id);
db=> ALTER TABLE test.people ADD PRIMARY KEY USING INDEX people_id_idx;

db=> COMMENT ON TABLE test.people IS E'A table with people';
db=> COMMENT ON COLUMN test.people.people_id IS E'The unique ID of a person.';
db=> COMMENT ON COLUMN test.people.firstname IS E'The given name of a person.';
db=> COMMENT ON COLUMN test.people.lastname IS E'The family name of a person.';
db=> COMMENT ON COLUMN test.people.email IS E'The E-Mail address of a person.';
db=> COMMENT ON INDEX test.people_id_idx IS E'The primary key identifier of the table.';
```

Create a Table

```
db=> CREATE TABLE test.people
     (
       people_id INTEGER,
       firstname TEXT,
       lastname TEXT,
       email TEXT
     );
db=> CREATE UNIQUE INDEX peop
db=> ALTER TABLE test.people

db=> COMMENT ON TABLE test.pe
db=> COMMENT ON COLUMN test.p
db=> COMMENT ON COLUMN test.p
db=> COMMENT ON COLUMN test.p
db=> COMMENT ON COLUMN test.p
db=> COMMENT ON INDEX test.pe
```



Charles Talk Preferences Watchlist Contributions Log out

Page Discussion Read Edit View history Search

Test

Welcome
Recent changes
Curriculum Vitae
Computer science
Kochen
Publications
Music
Additionals
Tools

Table "test.people"

A table with people

Owner: Charles Clavadetscher (charles)

Columns

Column name	Type	Modifiers	Description
people_id	integer	not null	The unique ID of a person.
firstname	text		The given name of a person.
lastname	text		The family name of a person.
email	text		The E-Mail address of a person.

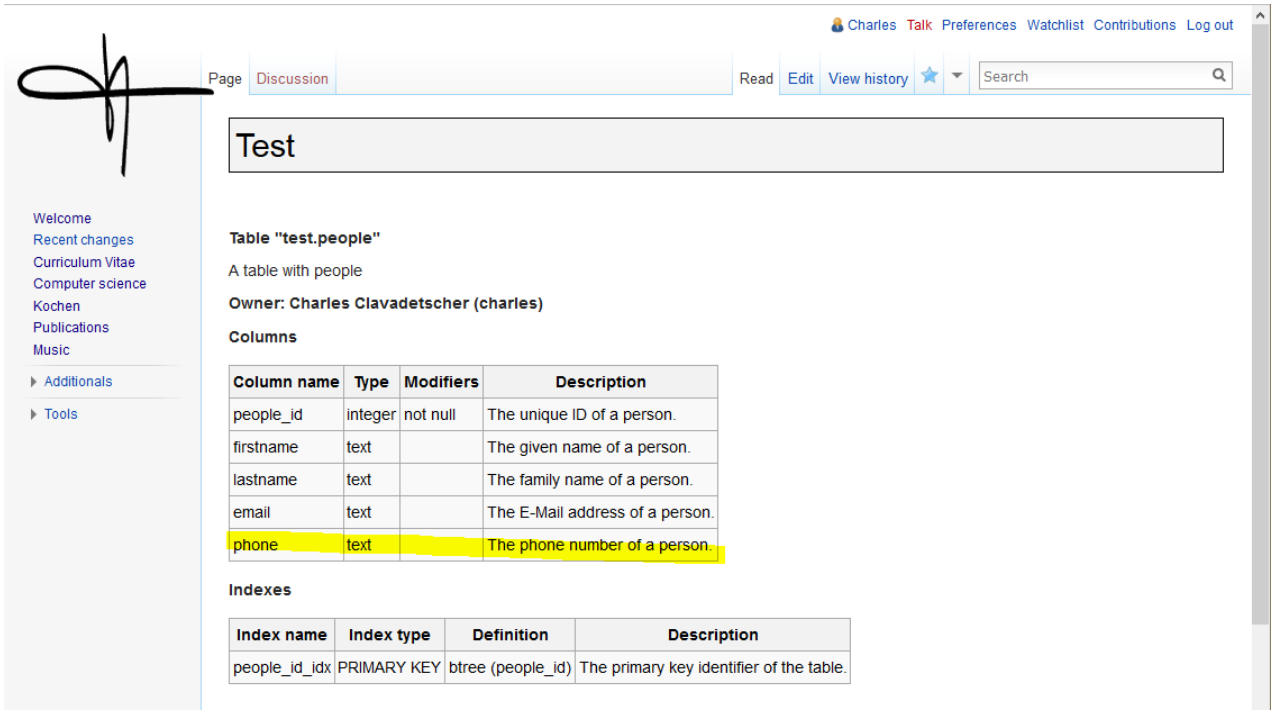
Indexes

Index name	Index type	Definition	Description
people_id_idx	PRIMARY KEY	btree (people_id)	The primary key identifier of the table.

The table appears in the Wiki page.

Modify the table

```
db=> ALTER TABLE test.people ADD COLUMN phone TEXT;
db=> COMMENT ON COLUMN test.people.phone IS E'The phone number of a person.';
```



Charles Talk Preferences Watchlist Contributions Log out

Page Discussion Read Edit View history Search

Test

Welcome
Recent changes
Curriculum Vitae
Computer science
Kochen
Publications
Music
Additionals
Tools

Table "test.people"

A table with people

Owner: Charles Clavadetscher (charles)

Columns

Column name	Type	Modifiers	Description
people_id	integer	not null	The unique ID of a person.
firstname	text		The given name of a person.
lastname	text		The family name of a person.
email	text		The E-Mail address of a person.
phone	text		The phone number of a person.

Indexes

Index name	Index type	Definition	Description
people_id_idx	PRIMARY KEY	btree (people_id)	The primary key identifier of the table.

Add a row security policy

```
db=> CREATE POLICY working_days ON test.people
db=> FOR ALL TO PUBLIC
db=> USING (CURRENT_TIME BETWEEN '08:00:00' AND '17:00:00'
db=> AND date_part('dow',CURRENT_TIMESTAMP) NOT IN (0,6));

db=> ALTER TABLE test.people ENABLE ROW LEVEL SECURITY;

db=> COMMENT ON POLICY working_days ON test.people IS
db=>   'Only allow access to the table during working days (Mo-Fr 08:00-17:00).';
```

Add a row security policy

```
db=> CREATE POLICY working_da
db=> FOR ALL TO PUBLIC
db=> USING (CURRENT_TIME BETW
db=> AND date_part('dow',CURR

db=> ALTER TABLE test.people

db=> COMMENT ON POLICY workin
db=>   'Only allow access to
```

Log in

Page Discussion Read View source View history Search

Welcome
Recent changes
Curriculum Vitae
Computer science
Kochen
Publications
Music

Additionals
Recent changes
Random page
Help

Tools

Test

Table "test.people"

A table with people

Owner: Charles Clavadetscher (charles)

Columns

Column name	Type	Modifiers	Description
people_id	integer	not null	The unique ID of a person.
firstname	text		The given name of a person.
lastname	text		The family name of a person.
email	text		The E-Mail address of a person.
phone	text		The phone number of a person.

Indexes

Index name	Index type	Definition	Description
people_id_idx	PRIMARY KEY	btree (people_id)	The primary key identifier of the table.

Policies

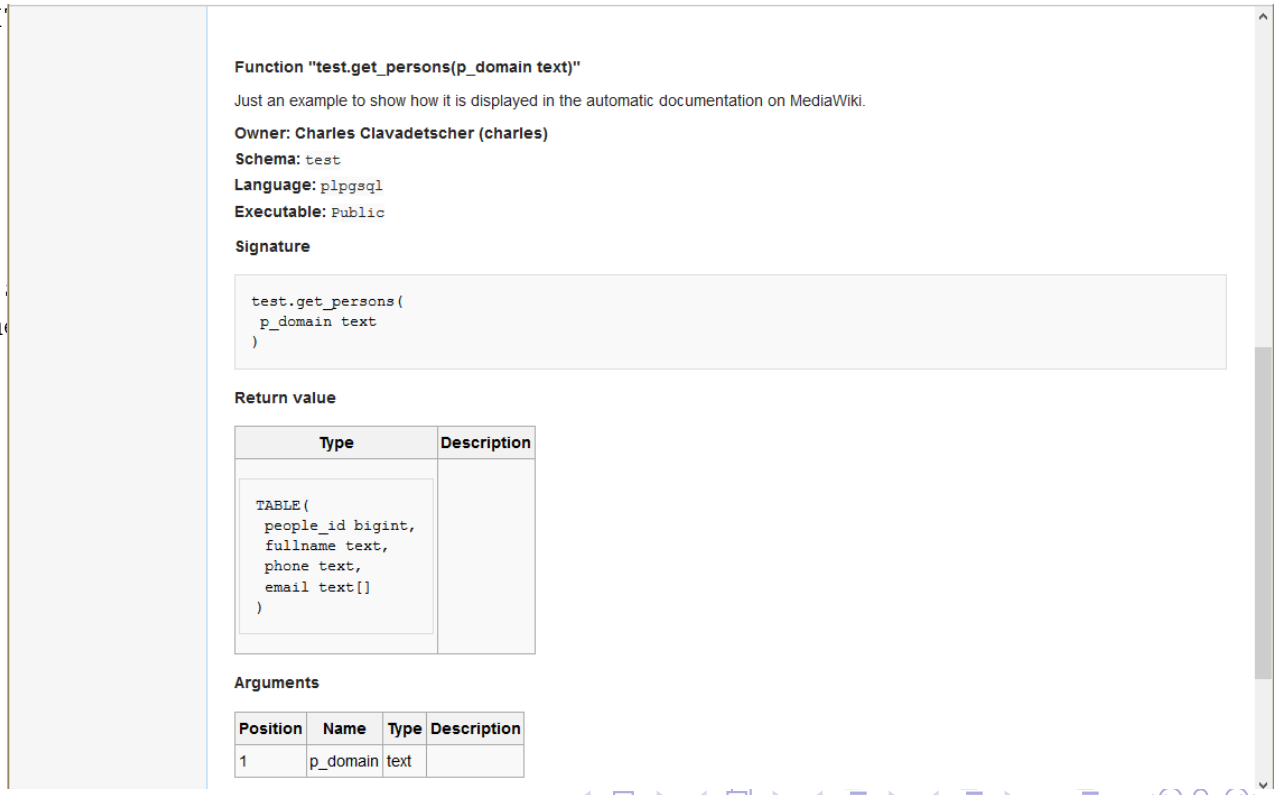
Policy name	For	To	Using	With check	Description
working_days	ALL	PUBLIC	((('now'::text)::time with time zone >= '08:00:00+02'::time with time zone) AND (('now'::text)::time with time zone <= '17:00:00+02'::time with time zone)) AND (date_part('dow'::text, now()) <> ALL (ARRAY[(0)::double precision, (6)::double precision]))		Only allow access to the table during working days (Mo-Fr 08:00-17:00).

Create a Function

```
db=> CREATE OR REPLACE FUNCTION test.get_persons(p_domain TEXT)
     RETURNS TABLE (people_id BIGINT,
                    fullname TEXT,
                    phone TEXT,
                    email TEXT[])
     AS $body$
     BEGIN
     END;
     $body$ LANGUAGE plpgsql;
\p
db=> COMMENT ON FUNCTION test.get_persons(p_domain TEXT)
     IS E'Just an example to show how it is displayed in '||
        E'the automatic documentation on MediaWiki.';
```

Create a Function

```
db=> CREATE OR REPLACE FUNCTION test.get_persons(p_domain TEXT)
     RETURNS TABLE (people_id BIGINT,
                    fullname TEXT,
                    phone TEXT,
                    email TEX
     AS $body$
     BEGIN
     END;
     $body$ LANGUAGE plpgsql;
\p
db=> COMMENT ON FUNCTION test
     IS E'Just an example to
        E'the automatic docum
```



Function "test.get_persons(p_domain text)"

Just an example to show how it is displayed in the automatic documentation on MediaWiki.

Owner: Charles Clavadetscher (charles)
Schema: test
Language: plpgsql
Executable: Public

Signature

```
test.get_persons(
 p_domain text
)
```

Return value

Type	Description
TABLE(people_id bigint, fullname text, phone text, email text[])	

Arguments

Position	Name	Type	Description
1	p_domain	text	

Add Comments to Function Arguments

```
db=> SELECT public.comment_on_function_argument('test.get_persons(TEXT)',
     -1,
     E'A set of entries matching the query domain:\n'||
     E'people_id: The unique identifier of the entry.\n'||
     E'fullname: Given and familiy name of the person.\n...');
db=> SELECT public.comment_on_function_argument('test.get_persons(TEXT)',
     1,
     E'A mail domain in the format <code>@domain</code>.');
```

Add Comments to Function Arguments

```
db=> SELECT public.comment_on_function_argument('test.get_persons(TEXT)',
     -1,
     E'A set of entries matching the query domain:\n'||
     E'people_id: The unique identifier of the entry.\n'||
     E'fullname: Given and familiy name of the person.\n...');
db=> SELECT public.comment_on_function_argument('test.get_persons(TEXT)',
     1,
     E'A mail domain in the f
```

Function "test.get_persons(p_domain text)"

Just an example to show how it is displayed in the automatic documentation on MediaWiki.

Owner: Charles Clavadetscher (charles)
Schema: test
Language: plpgsql
Executable: Public

Signature

```
test.get_persons(
 p_domain text
)
```

Return value

Type	Description
TABLE(people_id bigint, fullname text, phone text, email text[])	A set of entries matching the query domain: people_id: The unique identifier of the entry. fullname: Given and familiy name of the person. ...

Arguments

Position	Name	Type	Description
1	p_domain	text	A mail domain in the format @domain.

Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

Custom Function Delivering Wiki Markup

Function "public.create_wiki_doc(p_title integer, p_type text, p_schema_names text[], p_object_names text[])"

Create a set of wiki markup texts from a set of PostgreSQL objects. Notice that you must qualify or cast at least one of the p_schema_names or p_object_names to TEXT[] in order to enable PostgreSQL to pick this function instead of the base one.
Example: use array['schema_name1','schema_name2',...] or '{schema_name1,schema_name2}'::TEXT[] instead of a pure string representation of the array like '{schema_name1,schema_name2}'.

Owner: Charles Clavadetscher (charles)
Schema: `public`
Language: `plpgsql`
Executable: `Public`

Signature

```
public.create_wiki_doc(
 p_title integer,
 p_type text,
 p_schema_names text[],
 p_object_names text[]
)
```

Return value

Type	Description
`SETOF text`	The wiki markup formatted TEXT with the description of the object in p_object_name.

Arguments

Position	Name	Type	Description
1	p_title	integer	The heading number for the title: NULL or 0=Simple bold text. 1-4=The corresponding HTML header (in wiki markup this is the number of "=" the line is put between).
2	p_type	text	The type of object. Currently supported: table view function
3	p_schema_names	text[]	The name of the schemas where the objects are to be found. Use NULL for all schemas.
4	p_object_names	text[]	The name of the objects to be documented. Use NULL for all objects of this type.

Custom Functions for Arguments - 1

Function "public.comment_on_function_argument(p_proname text, p_position integer, p_comment text)"

A function to create comments on function arguments and return value.

Owner: Charles Clavadetscher (charles)
Schema: `public`
Language: `plpgsql`
Executable: `Public`

Signature

```
public.comment_on_function_argument(
 p_proname text,
 p_position integer,
 p_comment text
)
```

Return value

Type	Description
`boolean`	A boolean value to indicate if the description entry could be done.

Arguments

Position	Name	Type	Description
1	p_proname	text	The function signature: function_name(comma separated list of data types).
2	p_position	integer	The 1-based position of the paramenter for the comment. Use -1 for the return value.
3	p_comment	text	The comment itself.

Custom Functions for Arguments - 2

Function "public.param_description(objoid oid, posnum integer)"

A function to retrieve the comment on a function argument or the return value.

Owner: Charles Clavadetscher (charles)

Schema: `public`

Language: `sql`

Executable: `Public`

Signature

```
public.param_description(
 objoid oid,
 posnum integer
)
```

Return value

Type	Description
`text`	The description of the argument or the return value.

Arguments

Position	Name	Type	Description
1	objoid	oid	The oid of the function: (function_name(comma separated list of data types).oid
2	posnum	integer	The position of the paramenter or -1 for the return value.

Custom Functions Target Table

```
                    Table "pg_catalog.pg_description"
   Column    |  Type   | Modifiers | Storage  | Stats target | Description
-------------+---------+-----------+----------+--------------+-------------
 objoid      | oid     | not null  | plain    |              |
 classoid    | oid     | not null  | plain    |              |
 objsubid    | integer | not null  | plain    |              |
 description | text    |           | extended |              |
Indexes:
    "pg_description_o_c_o_index" UNIQUE, btree (objoid, classoid, objsubid)

db=> SELECT * FROKM pg_catalog.pg_description
     WHERE objoid = 'get_persons(TEXT)'::regprocedure;

 objoid | classoid | objsubid |                    description
--------+----------+----------+-----------------------------------------------------
  89277 |     1255 |       -1 | A set of entries matching the query domain:          +
        |          |          | people_id: The unique identifier of the entry.       +
        |          |          | fullname: Given and familiy name of the person.      +
        |          |          | ...
  89277 |     1255 |        0 | Just an example to show how it is displayed in the   +
        |          |          | automatic documentation on MediaWiki.
  89277 |     1255 |        1 | A mail domain in the format <code>@domain</code>.
(3 rows)
```

ExternalData And MagicNoCache MediaWiki Extensions

```
__NOCACHE__
{{#get_db_data:
db=wikireader
|from=public.create_wiki_doc(0,ARRAY['test'],NULL)
|data=documentation=create_wiki_doc
}}
{{#for_external_table:<nowiki />
{{{documentation}}}
}}
{{#clear_external_data:}}

{{#get_db_data:
db=wikireader
|from=public.create_wiki_doc(0,'function',ARRAY['test'],ARRAY['get_persons(text)'])
|data=documentation=create_wiki_doc
}}
{{#for_external_table:<nowiki />
{{{documentation}}}
}}
{{#clear_external_data:}}
```

What you need

Detailed instructions:
http://www.schmiedewerkstatt.ch/wiki/index.php/Database_self-documenting_Wiki

- An installation of MediaWiki (version 1.19+) on a web server.
- The ExternalData MediaWiki extension.
  - A dedicated database user for this component with only execute privileges on the Wiki markup creating functions.
  - The configuration for the database connection with this user.
- The MagicNoCache MediaWiki extension.
  - This component is not a strict requirements. If you don't use it, you will need to force a cache bypass in MediaWiki using ?action=purge after the page link.
- Install custom functions.
  - Download both scripts and execute them in a psql shell.

Outline

1 Introduction
2 Documenting Resources
3 The Old Way
4 Moving On
5 The Magic of Integration
6 The Magic Explained
7 Conclusion

Resources

- These slides: http://www.schmiedewerkstatt.ch/documents/04-publications/integrating_doc_with_mediawiki_handout_pdfa.pdf
- Working examples: http://www.schmiedewerkstatt.ch/wiki/index.php/Database_self-documenting_Wiki
- MediaWiki: https://www.mediawiki.org
- ExternalData: https://www.mediawiki.org/wiki/Extension:External_Data
- MagicNoCache: https://www.mediawiki.org/wiki/Extension:MagicNoCache
- PostgreSQL COMMENT Command: https://www.postgresql.org/docs/9.5/static/sql-comment.html

Contact

- Work: clavadetscher@kof.ethz.ch
  http://www.kof.ethz.ch
- SwissPUG: clavadetscher@swisspug.org
  http://www.swisspug.org
- Private: charles@schmiedewerkstatt.ch
  http://www.schmiedewerkstatt.ch

Thank you

Thank you very much for your attention !

Feedback

Q&A