PostgreSQL extension's development

Dimitri Fontaine

Dec. 7, 2010

Content

Content

Content

Content

Definitions

PostgreSQL extensibility is remarkable but incomplete.

Example (Basic SQL query)

```
SELECT col
  FROM table
 WHERE stamped > date 'today' - interval '1 day'
```

Some extensions example

46 Contribs, Community extensions, Private ones...

- cube
- ltree
- citext
- hstore
- intagg

- adminpack
- pgq
- pg_trgm
- wildspeed
- dblink

- PostGIS
- ip4r
- temporal
- prefix
- pgfincore

- pgcrypto
- pg_stattuple
- pg_freespacemap
- pg_stat_statements
- pg_standby

PostgreSQL extensibility is remarkable but incomplete.

It lacks dump and restore support.

Installing an extension

Example (Installing an extension before 9.1)

```
apt-get install postgresql-contrib-9.0
apt-get install postgresql-9.0-ip4r
psql -f /usr/share/postgresql/9.0/contrib/hstore.sql
```

- so, what did it install? ok, reading the *script*
- Oh, nice, it's all in the `public` schema
- Oh, very nice, no `ALTER OPERATOR SET SCHEMA`

Wait, it gets better!

Installing an extension

Example (Installing an extension before 9.1)

```
apt-get install postgresql-contrib-9.0
apt-get install postgresql-9.0-ip4r
psql -f /usr/share/postgresql/9.0/contrib/hstore.sql
```

- so, what did it install? ok, reading the *script*
- Oh, nice, it's all in the `public` schema
- Oh, very nice, no `ALTER OPERATOR SET SCHEMA`

Wait, it gets better!

Installing an extension

Example (Installing an extension before 9.1)

```
apt-get install postgresql-contrib-9.0
apt-get install postgresql-9.0-ip4r
psql -f /usr/share/postgresql/9.0/contrib/hstore.sql
```

- so, what did it install? ok, reading the *script*
- Oh, nice, it's all in the `public` schema
- Oh, very nice, no `ALTER OPERATOR SET SCHEMA`

Wait, it gets better!

Installing an extension

Example (Installing an extension before 9.1)

```
apt-get install postgresql-contrib-9.0
apt-get install postgresql-9.0-ip4r
psql -f /usr/share/postgresql/9.0/contrib/hstore.sql
```

- so, what did it install? ok, reading the *script*
- Oh, nice, it's all in the `public` schema
- Oh, very nice, no `ALTER OPERATOR SET SCHEMA`

Wait, it gets better!

Installing an extension

Example (Installing an extension before 9.1)

```
apt-get install postgresql-contrib-9.0
apt-get install postgresql-9.0-ip4r
psql -f /usr/share/postgresql/9.0/contrib/hstore.sql
```

- so, what did it install? ok, reading the *script*
- Oh, nice, it's all in the `public` schema
- Oh, very nice, no `ALTER OPERATOR SET SCHEMA`

Wait, it gets better!

backup and restores

```
pg_dump -h remote mydb | psql fresh
```

- extensions objects are an entire part of your database
- but they are maintained elsewhere, that's just a dependency
- pg_dump makes no difference
- what about upgrading systems (system, database, extension)

backup and restores

```
pg_dump -h remote mydb | psql fresh
```

- extensions objects are an entire part of your database
- but they are maintained elsewhere, that's just a dependency
- pg_dump makes no difference
- what about upgrading systems (system, database, extension)

backup and restores

```
pg_dump -h remote mydb | psql fresh
```

- extensions objects are an entire part of your database
- but they are maintained elsewhere, that's just a dependency
- pg_dump makes no difference
- what about upgrading systems (system, database, extension)

What problems are we solving?

It's all about clearing up the mess. No feature is accepted in PostgreSQL without complete support for dump and restore nowadays. And that's good news.

Example (the goal: have pg_dump output this)

```
CREATE EXTENSION hstore WITH NO USER DATA;
```

What problems are we solving?

It's all about clearing up the mess. No feature is accepted in PostgreSQL without complete support for dump and restore nowadays. And that's good news.

Example (the goal: have pg_dump output this)

```
CREATE EXTENSION hstore WITH NO USER DATA;
```

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt pg_dump
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt pg_dump
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt pg_dump
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt `pg_dump`
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt `pg_dump`
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt `pg_dump`
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

How are we solving our problems?

Lots of little things need to happen:

- Rely on the OS to install the *script* and *module*
- Register the extension in the catalogs, to get an *OID*
- Track dependencies at `CREATE EXTENSION` time
- Adapt `pg_dump`
- Offer a `WITH SCHEMA` facility
- Offer `ALTER EXTENSION SET SCHEMA`
- Don't forget `DROP EXTENSION RESTRICT|CASCADE`

Extensions and user data

What if an extension gets modified after install?

- `pg_dump` support is all about *excluding* things from dumps
- some extensions install default data
- and allow users to edit them
- now you want the data in your dumps, right?

Extensions and user data

What if an extension gets modified after install?

- `pg_dump` support is all about *excluding* things from dumps
- some extensions install default data
- and allow users to edit them
- now you want the data in your dumps, right?

Extensions and user data

What if an extension gets modified after install?

- `pg_dump` support is all about *excluding* things from dumps
- some extensions install default data
- and allow users to edit them
- now you want the data in your dumps, right?

The effort in figures

```
git diff master..  | diffstat | tail -1
244 files changed, 4940 insertions(+), 1362
deletions(-)
```

- 5 patches, 7 branches, its own *Commit Fest* section
- about 18 months to get an agreement on what to develop *first*
- 2 *Developer Meeting* interventions, in Ottawa, *PgCon*
- 4 weeks full time, countless evenings

The effort in figures

```
git diff master..  | diffstat | tail -1
244 files changed, 4940 insertions(+), 1362
deletions(-)
```

- 5 patches, 7 branches, its own *Commit Fest* section
- about 18 months to get an agreement on what to develop *first*
- 2 *Developer Meeting* interventions, in Ottawa, *PgCon*
- 4 weeks full time, countless evenings

The effort in figures

```
git diff master..  | diffstat | tail -1
244 files changed, 4940 insertions(+), 1362
deletions(-)
```

- 5 patches, 7 branches, its own *Commit Fest* section
- about 18 months to get an agreement on what to develop *first*
- 2 *Developer Meeting* interventions, in Ottawa, *PgCon*
- 4 weeks full time, countless evenings

The effort in figures

```
git diff master.. | diffstat | tail -1
244 files changed, 4940 insertions(+), 1362
deletions(-)
```

- 5 patches, 7 branches, its own *Commit Fest* section
- about 18 months to get an agreement on what to develop *first*
- 2 *Developer Meeting* interventions, in Ottawa, *PgCon*
- 4 weeks full time, countless evenings

The effort in figures

```
git diff master.. | diffstat | tail -1
244 files changed, 4940 insertions(+), 1362
deletions(-)
```

- 5 patches, 7 branches, its own *Commit Fest* section
- about 18 months to get an agreement on what to develop *first*
- 2 *Developer Meeting* interventions, in Ottawa, *PgCon*
- 4 weeks full time, countless evenings

What's to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore WITH SCHEMA utils;`
- `\dx[+]`
- `select * from pg_extension_objects('unaccent');`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `pg_execute_sql_file()`
- `pg_execute_sql_string()`

What's to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore WITH SCHEMA utils;`
- `\dx[+]`
- `select * from pg_extension_objects('unaccent');`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `pg_execute_sql_file()`
- `pg_execute_sql_string()`

What's to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore WITH SCHEMA utils;`
- `\dx[+]`
- `select * from pg_extension_objects('unaccent');`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `pg_execute_sql_file()`
- `pg_execute_sql_string()`

What's to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore WITH SCHEMA utils;`
- `\dx[+]`
- `select * from pg_extension_objects('unaccent');`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `pg_execute_sql_file()`
- `pg_execute_sql_string()`

What's to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore WITH SCHEMA utils;`
- `\dx[+]`
- `select * from pg_extension_objects('unaccent');`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `pg_execute_sql_file()`
- `pg_execute_sql_string()`

Using PGXS

Simpler way to have your files installed at the right place, using `make install`. But `Makefiles` are hard, right?

Example (citext.control.in)

```
MODULES = citext
DATA_built = citext.sql
DATA = uninstall_citext.sql
REGRESS = citext

EXTENSION = $(MODULES)
EXTVERSION = $(VERSION)
```

Using PGXS

Simpler way to have your files installed at the right place, using `make install`. But `Makefiles` are hard, right?

Example (citext.control.in)

```
MODULES = citext
DATA_built = citext.sql
DATA = uninstall_citext.sql
REGRESS = citext

EXTENSION = $(MODULES)
EXTVERSION = $(VERSION)
```

The control file

It's a very complex file containing the *meta data* that PostgreSQL needs to know about to be able to register your *extension* in its *system catalogs*. It looks like this:

Example (citext.control.in)

```
# citext
comment = 'case-insensitive character string type'
version = 'EXTVERSION'
```

The control file

It's a very complex file containing the *meta data* that PostgreSQL needs to know about to be able to register your *extension* in its *system catalogs*. It looks like this:

Example (citext.control.in)

```
# citext
comment = 'case-insensitive character string type'
version = 'EXTVERSION'
```

custom_variable_classes

The `custom_variable_classes` should be only known by extensions authors.

> *A custom variable is a variable not normally known to PostgreSQL proper but used by some add-on module. Such variables must have names consisting of a class name, a dot, and a variable name. `custom_variable_classes` specifies all the class names in use in a particular installation.*

custom_variable_classes

The `custom_variable_classes` should be only known by extensions authors.

> *A custom variable is a variable not normally known to PostgreSQL proper but used by some add-on module. Such variables must have names consisting of a class name, a dot, and a variable name. `custom_variable_classes` specifies all the class names in use in a particular installation.*

custom_variable_classes, continued

Example (pg_stat_statements.control.in)

```
# pg_stat_statements
comment = 'tracking execution statistics of all SQL statemen
version = 'EXTVERSION'
custom_variable_classes = 'pg_stat_statements'
pg_stat_statements.max = 1000
pg_stat_statements.track = 'top'
pg_stat_statements.track_utility = true
pg_stat_statements.save = true
```

Extension script and user data

Example (Flag your pg_dump worthy objects)

```
DO $$
BEGIN
IF pg_extension_with_user_data() THEN
  create schema foo;
  create table foo.bar(id serial primary key);
  perform pg_extension_flag_dump('foo.bar_id_seq'::regclass
  perform pg_extension_flag_dump('foo.bar::regclass);
END IF;
END;
$$;
```

debian and pg_buildext

Contributed and available in *debian squeeze*, `postgresql-server-dev-all`

Example (debian/pgversions)

```
8.4
9.0
```

debian and pg_buildext

Contributed and available in *debian squeeze*, `postgresql-server-dev-all`

Example (debian/rules)

```
include /usr/share/postgresql-common/pgxs_debian_control.mk

install: build
# build all supported version
pg_buildext build $(SRCDIR) $(TARGET) "$(CFLAGS)"

# then install each of them
for v in `pg_buildext supported-versions $(SRCDIR)`; do \
dh_install -ppostgresql-$$v-pgfincore ;\
done
```

Money

4 week full time at home, thanks to 2ndQuadrant, and to our affiliation with European Research

> *The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 258862*

Any question?

Now is a pretty good time to ask!

If you want to leave feedback, consider visiting
`http://2010.pgday.eu/feedback`