PostgreSQL extension’s development

Dimitri Fontaine

May 20, 2011
What’s an Extension?
- Before 9.1 and CREATE EXTENSION

The extension specs & scope
- Scope
- Specs
- Implementation details...

Extension for their authors: YOU.
- PGXS and the control file
- Extensions Upgrades
- Extensions and packaging

Conclusion
- Sponsoring
- Any question?
1. What’s an Extension?
   - Before 9.1 and CREATE EXTENSION

2. The extension specs & scope
   - Scope
   - Specs
   - Implementation details...

3. Extension for their authors: YOU.
   - PGXS and the control file
   - Extensions Upgrades
   - Extensions and packaging

4. Conclusion
   - Sponsoring
   - Any question?
Content

1. What’s an Extension?
   - Before 9.1 and CREATE EXTENSION

2. The extension specs & scope
   - Scope
   - Specs
   - Implementation details...

3. Extension for their authors: YOU.
   - PGXS and the control file
   - Extensions Upgrades
   - Extensions and packaging

4. Conclusion
   - Sponsoring
   - Any question?
1. What’s an Extension?
   - Before 9.1 and CREATE EXTENSION

2. The extension specs & scope
   - Scope
   - Specs
   - Implementation details...

3. Extension for their authors: YOU.
   - PGXS and the control file
   - Extensions Upgrades
   - Extensions and packaging

4. Conclusion
   - Sponsoring
   - Any question?
What is a “Extension”?
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

- 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

46 Contribs, Community extensions, Private ones...
PostgreSQL extensibility is remarkable but incomplete.

It lacks dump and restore support.
Before 9.1 and CREATE EXTENSION
Installing an extension

Example (Installing an extension before 9.1)

```bash
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)

```bash
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)
The extension specs & scope
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)

```sql
CREATE EXTENSION IF NOT EXISTS hstore WITH SCHEMA public;
```
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 IF NOT EXISTS hstore WITH SCHEMA public;
```
Agenda
What’s an Extension?
The extension specs & scope
Extension for their authors: YOU.
Conclusion

Scope
Specs
Implementation details...

Specs
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
- Manage upgrading ALTER EXTENSION UPDATE
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
- Manage upgrading ALTER EXTENSION UPDATE
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`
- Manage upgrading `ALTER EXTENSION UPDATE`
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`
- Manage upgrading `ALTER EXTENSION UPDATE`
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
- Manage upgrading ALTER EXTENSION UPDATE
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
- Manage upgrading ALTER EXTENSION UPDATE
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
- Manage upgrading ALTER EXTENSION UPDATE
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
- Manage upgrading ALTER EXTENSION UPDATE
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?
Implementation
The effort in figures

```
git diff -stat master..extension | tail -1
260 files changed, 4202 insertions(+), 2073 deletions(-)
git -no-pager diff -stat extension..upgrade | tail -1
125 files changed, 1976 insertions(+), 81 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, 3 months of refining
The effort in figures

```bash
git diff -stat master..extension | tail -1
260 files changed, 4202 insertions(+), 2073 deletions(-)

```

```bash
git -no-pager diff -stat extension..upgrade | tail -1
125 files changed, 1976 insertions(+), 81 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, 3 months of refining
The effort in figures

```
$ git diff -stat master..extension | tail -1
  260 files changed, 4202 insertions(+), 2073 deletions(-)
$ git -no-pager diff -stat extension..upgrade | tail -1
  125 files changed, 1976 insertions(+), 81 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, 3 months of refining
The effort in figures

```
git diff -stat master..extension | tail -1
260 files changed, 4202 insertions(+), 2073 deletions(-)
git -no-pager diff -stat extension..upgrade | tail -1
125 files changed, 1976 insertions(+), 81 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, 3 months of refining
The effort in figures

git diff -stat master..extension | tail -1
260 files changed, 4202 insertions(+), 2073 deletions(-)
git -no-pager diff -stat extension..upgrade | tail -1
125 files changed, 1976 insertions(+), 81 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, 3 months of refining
What’s to know, now

Some new commands and catalogs:

- `CREATE EXTENSION hstore SCHEMA utils;`
- `CREATE EXTENSION hstore VERSION 1.1;`
- `\dx`
- `ALTER EXTENSION hstore SET SCHEMA addons;`
- `DROP EXTENSION hstore CASCADE;`
- `ALTER EXTENSION hstore UPDATE TO version;`
- `CREATE EXTENSION hstore FROM unpackaged;`
Some new commands and catalogs:

- CREATE EXTENSION hstore SCHEMA utils;
- CREATE EXTENSION hstore VERSION 1.1;
- \dx
- ALTER EXTENSION hstore SET SCHEMA addons;
- DROP EXTENSION hstore CASCADE;
- ALTER EXTENSION hstore UPDATE TO version;
- CREATE EXTENSION hstore FROM unpackaged;
Some new commands and catalogs:

- CREATE EXTENSION hstore SCHEMA utils;
- CREATE EXTENSION hstore VERSION 1.1;
- \dx
- ALTER EXTENSION hstore SET SCHEMA addons;
- DROP EXTENSION hstore CASCADE;
- ALTER EXTENSION hstore UPDATE TO version;
- CREATE EXTENSION hstore FROM unpackaged;
What’s to know, now

Some new commands and catalogs:

- CREATE EXTENSION hstore SCHEMA utils;
- CREATE EXTENSION hstore VERSION 1.1;
- \dx
- ALTER EXTENSION hstore SET SCHEMA addons;
- DROP EXTENSION hstore CASCADE;
- ALTER EXTENSION hstore UPDATE TO version;
- CREATE EXTENSION hstore FROM unpackaged;
PGXS and the control file
Simpler way to have your files installed at the right place, using `make install`. But Makefiles are hard, right?

### Example (citext/Makefile)

- **MODULES** = citext
- **EXTENSION** = citext
- **DATA** = citext--1.0.sql citext--unpackaged--1.0.sql
- **REGRESS** = citext
Using PGXS

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

Example (citext/Makefile)

```
MODULES = citext
EXTENSION = citext
DATA = citext--1.0.sql citext--unpackaged--1.0.sql
REGRESS = citext
```
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)

# citext extension
comment = 'data type for case-insensitive character strings'
default_version = '1.0'
module_pathname = '$libdir/citext'
relocatable = true
```
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)

# citext extension
comment = 'data type for case-insensitive character strings'
default_version = '1.0'
module_pathname = '$libdir/citext'
relocatable = true
```
A relocatable extension installs all its object into the first schema of the search_path. It’s then possible to `ALTER EXTENSION SET SCHEMA`. 
An extension that needs to know where some of its objects are installed is not relocatable. The extension installation script is then required to use the `@extschema@` placeholder as the schema to work with.

**Example (tsearch2/tsearch2-unpackaged-1.0.sql)**

```sql
ALTER EXTENSION tsearch2 ADD type @extschema@.tsvector;
ALTER EXTENSION tsearch2 ADD type @extschema@.tsquery;
```
Example (Flag your pg_dump worthy objects)

```
CREATE TABLE my_config (key text, value text);
SELECT pg_catalog.pg_extension_config_dump('my_config', '');

CREATE TABLE my_config (key text, value text, standard_entry boolean);
SELECT pg_catalog.pg_extension_config_dump('my_config', 'WHERE NOT standard_entry');
```
Extension Upgrades
ALTER EXTENSION ... UPDATE;

- Versions “numbers” are just strings
- Provide scripts `extension-old-new.sql`
- Updates only refer to changes in the SQL script
- Secondary control files `extension-new.control`
ALTER EXTENSION ... UPDATE;

- Versions “numbers” are just strings
- Provide scripts `extension-old-new.sql`
- Updates only refer to changes in the SQL script
- Secondary control files `extension-new.control`
ALTER EXTENSION ... UPDATE;

- Versions “numbers” are just strings
- Provide scripts extension-old-new.sql
- Updates only refer to changes in the SQL script
- Secondary control files extension-new.control
Versions “numbers” are just strings

Provide scripts extension-old-new.sql

Updates only refer to changes in the SQL script

Secondary control files extension-new.control
ALTER EXTENSION ... UPDATE [TO VERSION];

- system view pg_available_extensions
- system view pg_available_extension_versions
- CREATE EXTENSION ... FORM old_versions

Example (hstore-unpackaged-1.0.sql)

ALTER EXTENSION hstore ADD type hstore;
ALTER EXTENSION hstore ADD function hstore_in(cstring);
ALTER EXTENSION hstore ADD function hstore_out(hstore);
...

ALTER EXTENSION ... UPDATE [TO VERSION];
Upgrade Scripts

Extension author has to provide scripts for all supported upgrades

- PostgreSQL handles upgrade paths
- 1.0-1.1 then 1.1-1.2
- system view pg_available_extension_versions
- Be careful about downgrade paths!
Extension and packaging
Contributed and available in Debian squeeze, postgresql-server-dev-all

Example (debian/pgversions)

8.4
9.0
 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_builddext build $(SRCDIR) $(TARGET) "$(CFLAGS)"

# then install each of them
for v in `pg_builddext supported-versions $(SRCDIR)`; do
  dh_install -ppostgresql-$$v-pgfincore 
  done
```
Conclusion
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
Now is a pretty good time to ask!