Status report on modules

Except we call them extensions now

Dimitri Fontaine

May, 19 2010

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- PGXS and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- PGXS and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- PGXS and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- PGXS and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- PGXS and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- `PGXS` and platform support
- upgrading facilities (*callback*)

Set the goal

What we're talking about:

- dump & reload support
- any source language (C, SQL, PL...)
- procedural language as an extension
- custom variables
- versions & dependencies
- `PGXS` and platform support
- upgrading facilities (*callback*)

Set the goal

What we're *NOT* talking about:

- user defined schema where to install (`pg_extension`)
- source level packaging
- `ACLs`
- `PGAN`
- OS level packaging & distribution

Set the goal

What we're *NOT* talking about:

- user defined schema where to install (pg_extension)
- source level packaging
- ACLs
- PGAN
- OS level packaging & distribution

Set the goal

What we're *NOT* talking about:

- user defined schema where to install (`pg_extension`)
- source level packaging
- `ACLs`
- `PGAN`
- OS level packaging & distribution

Set the goal

What we're *NOT* talking about:

- user defined schema where to install (`pg_extension`)
- source level packaging
- `ACLs`
- `PGAN`
- OS level packaging & distribution

Set the goal

What we're *NOT* talking about:

- user defined schema where to install (`pg_extension`)
- source level packaging
- `ACLs`
- `PGAN`
- OS level packaging & distribution

Step 1: make install

Support files are in `$PGDATA/extensions/foo`, and are `control`, `install.sql` and `uninstall.sql`, `foo.conf`.

Example (extensions/foo/control)

```
name = foo
version = 1.0
custom_variable_classes = 'foo'
depends  = bar (>= 1.1), baz
conflicts = bla (< 0.8)
```

Step 2: INSTALL EXTENSION

Example (install extension)

```
begin;
install extension foo;
commit;
```

Example (drop extension)

```
begin;
drop extension foo [cascade];
commit;
```

pg_execute_from_file()

This is the easiest part to implement, so that's done... available in git already. Does it make sense on its own?

Example (execute from file)

```
+Datum
+pg_execute_from_file(PG_FUNCTION_ARGS)
+{
+ text    *filename_t = PG_GETARG_TEXT_P(0);
+
+ foreach(parsetree_item, parsetree_list)
+
```

tracking objects

We'll need to add a *backend local variable* and some action on each and every `CREATE` command.

Conclusion

We need to talk.