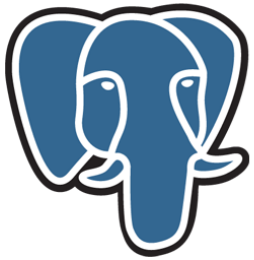

PostgreSQL

PGDay.IT 2011
Monash University Prato Centre
Venerdì 25 Novembre 2011

Scrivere un'estensione per PostgreSQL 9.1

Marco Nenciarini
Italian PostgreSQL Users Group

www.itpug.org
www.postgresql.org

Marco Nenciarini

- DBA, sviluppatore e sysadmin presso 2ndQuadrant
  - Database OLTP business critical
  - Data warehousing
- Membro della comunità di PostgreSQL
  - Co-Fondatore di ITPUG
- Debian Developer

Sommario

- Perché le estensioni
- Anatomia di un'estensione
- Installazione
- Manutenzione e aggiornamento
- PGXN
- Conclusioni

Ecosistema PostgreSQL

- Estensibilità
- Ricca documentazione
- Grande quantità di moduli aggiuntivi
  - 46 Contrib, molteplici estensioni, estensioni private
  - PostGIS, hstore, adminpack, dblink, ltree, pgq, ip4r, intagg, cube, pgfincore, pgcrypto, citext, pg_trgm, wildspeed, temporal, prexpg_stattuple, pg_freespacemap, pg_stat_statements, …

Prima di PostgreSQL 9.1

- Installazione tramite script

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

- Lo script sql può contenere molti oggetti
- Tutti gli oggetti vengono creati nello schema di default
- Difficoltà di gestione in database multi-schema

Perché le estensioni

- Supporto migliore per dump e restore
- Separazione logica fra l'estensione e gli altri oggetti presenti nel database
  - Non fa parte dei dati
  - È una dipendenza dei dati
- Installazione in uno schema specifico
- Operazioni di manutenzione semplificate

```
CREATE EXTENSION IF NOT EXISTS cube WITH SCHEMA public;
```

Anatomia di un'estensione

- File di controllo
  - Metadati
  - Impostazioni
  - Dipendenze
- Script SQL di installazione
  - Nessuna nuova sintassi da imparare
  - Ambiente di esecuzione controllato
- Script SQL di aggiornamento (opzionali)
- Moduli binari (opzionali)

Il file di controllo

- In `${SHAREDIR}/extension`
- Si deve chiamare `<nome_estensione>.control`
- Sintassi chiave/valore (come postgresql.conf)

```
$ cat /usr/share/postgresql/9.1/extension/pair.control
# pair extension
comment = 'A key/value pair data type'
default_version = '1.0'
relocatable = true
```

Lo script di installazione

- *<estensione>*--*<versione>*.sql
- È un normale script SQL
- Viene eseguito all'interno di una transazione
  - No BEGIN/COMMIT/ROLLBACK
- Ambiente controllato
  - Tutto quello che viene creato fa parte dell'estensione
- Due sostituzioni possibili
  - MODULE_PATHNAME
  - @extschema@

Esempio: pair--1.0.sql

```
CREATE TYPE pair AS ( k text, v text );
ALTER EXTENSION pair DROP TYPE _pair; -- bug 9.1

CREATE OR REPLACE FUNCTION pair(text, text)
RETURNS pair
LANGUAGE SQL AS 'SELECT ROW($1, $2)::pair;';

CREATE OPERATOR ~> (
   LEFTARG = text,
   RIGHTARG = text,
    PROCEDURE = pair);
```

Esempio: estensione non rilocabile

- Se l'estensione necessita conoscere schema di destinazione durante l'installazione essere marcata come non rilocabile
  - Usare @extschema@ come segnaposto per lo schema
  - relocatable = false
- Se l'estensione necessita di essere installata in uno schema specifico
  - relocatable = false
  - schema = *schema_di_destinazione*

Installazione

- CREATE EXTENSION
  - IF NOT EXISTS
  - SCHEMA <*schema_esistente*>
  - VERSION <*versione*>
  - FROM <*versione_modulo*>

```
CREATE EXTENSION hstore;
```

```
CREATE EXTENSION cube FROM unpackaged;
```

Ispezione da psql

```
postgres=# \dx
             List of installed extensions
 Name | Version | Schema |         Description
------+---------+--------+-----------------------------
 pair | 1.0     | pair   | A key/value pair data type
(1 row)

postgres=# \dx+
  Objects in extension "pair"
      Object Description
------------------------------
 function pair.pair(text,text)
 operator pair.~>(text,text)
 type pair.pair
(3 rows)
```

Supporto per il dump/restore

- L'estensione è una dipendenza
  - Una sola riga nel dump (commenti esclusi)
- L'estensione può contenere tabelle
  - Le tabelle non compaiono nel dump
  - Ma le tabelle possono essere modificate!

```
$ pg_dump | grep pair
-- Name: pair; Type: EXTENSION; Schema: -; Owner:
CREATE EXTENSION IF NOT EXISTS pair WITH SCHEMA public;
-- Name: EXTENSION pair; Type: COMMENT; Schema: -; Owner:
COMMENT ON EXTENSION pair IS 'A key/value pair data type';
```

Tabelle

```
CREATE TABLE extconfig (
  key text,
  value text,
  standard_entry boolean
);

SELECT pg_catalog.pg_extension_config_dump(
  'extconfig', 'WHERE NOT standard_entry'
);
```

Supporto per tabelle di configurazione

- Dump del solo contenuto
- Possibilità di discriminare fra dati di default e dati aggiuntivi

Manutenzione

- Aggiornamento atomico

```
ALTER EXTENSION pair UPDATE;
```

- Rilocazione in uno schema diverso

```
ALTER EXTENSION pair SET SCHEMA pair;
```

Gli script di aggiornamento

- *<estensione>*--*<da_versione>*--*<a_versione>*.sql
- Stessa sintassi dello script di installazione
- Strada più corta per la versione richiesta
- Nessuna assunzione su ordinamento
  - Downgrade
  - Nomi simbolici
- Funzione pg_extension_update_paths('estensione')
- Versione “unpackaged”

Esempio: pair--unpackaged--1.0.sql

- Si basa su un'estensione vuota
- Aggiunge manualmente il contenuto del vecchio modulo

```
ALTER EXTENSION pair ADD
  TYPE pair;
ALTER EXTENSION pair ADD
  FUNCTION pair(text, text);
ALTER EXTENSION pair ADD
  OPERATOR ~> (text,text);
```

Esempio: pair--1.0--1.1.sql

```
CREATE OR REPLACE FUNCTION pair(anyelement, anyelement)
RETURNS pair LANGUAGE SQL AS 'SELECT ROW($1, $2)::pair';
CREATE OR REPLACE FUNCTION pair(text, anyelement)
RETURNS pair LANGUAGE SQL AS 'SELECT ROW($1, $2)::pair';
CREATE OR REPLACE FUNCTION pair(anyelement, anyelement)
RETURNS pair LANGUAGE SQL AS 'SELECT ROW($1, $2)::pair';
CREATE OPERATOR ~>
  (LEFTARG = text, RIGHTARG = anyelement, PROCEDURE = pair);
CREATE OPERATOR ~>
  (LEFTARG = anyelement, RIGHTARG = text, PROCEDURE = pair);
CREATE OPERATOR ~>
  (LEFTARG = anyelement, RIGHTARG = anyelement,
    PROCEDURE = pair);
```

Esempio: pg_extension_update_paths

```
postgres=# SELECT * FROM pg_extension_update_paths('pair');
   source    |   target    |          path
-------------+-------------+-----------------------
 1.0         | 1.1         | 1.0--1.1
 1.0         | unpackaged  |
 1.1         | 1.0         |
 1.1         | unpackaged  |
 unpackaged  | 1.0         | unpackaged--1.0
 unpackaged  | 1.1         | unpackaged--1.0--1.1
(6 rows)
```

PostgreSQL eXtension Network - PGXN

- http://pgxn.org/
- Obiettivo: raccogliere tutte le estensioni open source
- PGXN Client
  - simile a apt-get o yum
    - **`pgxn install`** *`estensione`*
    - **`pgxn search`** *`parola`*
- PGXN Utils
  - Ausilio per lo sviluppatore di estensioni

META.json

- Specifiche http://pgxn.org/spec/

```
{
   "name": "pair",
   "abstract": "A key/value pair data type",
   "version": "0.1.0",
   "maintainer": "Marco Nenciarini <info@2ndQuadrant.it>",
   "license": "postgresql",
   "meta-spec": {
      "version": "1.0.0",
      "url": "http://pgxn.org/meta/spec.txt"
   },
}
```

Conclusioni

- Usare moduli aggiuntivi non è mai stato così facile
- Vantaggi anche per chi sviluppa estensioni
  - Robustezza
  - Manutenibilità
- Applicazioni come estensioni
  - Aggiornamento atomico
  - Packaging
  - Dump selettivo
- Lenta migrazione dei moduli esistenti

Domande?

- E-Mail: marco.nenciarini@2ndquadrant.it
- URL: www.2ndquadrant.it
- Blog: blog.2ndquadrant.it

Licenza Creative Commons

Attribuzione

Non commerciale

Condividi allo stesso modo

2.5 Italia

http://creativecommons.org/licenses/by-nc-sa/2.5/it/