Práce s fulltextem v databázi PostgreSQL 8.0

Lukáš Zapletal

PostgreSQL 8.0 je kvalitní, svobodná, relační databázová platforma fungující pod unixovými operačními systémy a nově také pod OS Windows. Mezi hlavní přednosti patří dobrá podpora SQL standardů, bohatá škála podporovaných datových typů, velmi propracovaná ochrana integrity, dobrá nabídka procedurálních jazyků na straně serveru a kvalitní dokumentace.

Vznikla v 90 letech na univerzitě v Berekeley (USA) – tehdy jako projekt POSTGRES. Projekt se postupně vyvíjel až do zcela nového projektu Postgres95, který již byl SQL kompatibilní. Záhy byl projekt přejmenován na PostgreSQL.

PostgreSQL podporuje, podobně jako databáze Firebird, technologii MVCC. Aby databáze zajistila konzistentní stav během jedné transakce, musí hlídat, aby data nějaký jiný uživatel v danou chvíli nezměnil. Toho může docílit různými způsoby (například různé druhy zámků), technologie MVCC používá takzvanou historii záznamů, kdy se podobně jako revizní systém CVS může vrátit ke starším verzím.

Databáze podporuje pohledy, cizí klíče, transakce a jednoduchou tabulkovou dědičnost. Příklad:

```
CREATE TABLE cities (
  name       text,
  population real,
  altitude   int     -- (in ft)
);


CREATE TABLE capitals (
  state      char(2)
) INHERITS (cities);
```

PostgreSQL je SQL-2003 kompatibilní a poskytuje bohatou škálu datových typů, včetně nestandardních (geometrické a geografické útvary včetně operací s nimi).

Mezi hlavní přednosti PostgreSQL je její multigenerační architektura (MVCC). Konzistence není udržována pomocí zámků (ačkoliv zámky tato databáze plně podporuje, a to jak na úrovni tabulek, tak i záznamů), ale pomocí zpětného udržování historie jednotlivých záznamů.

PostgreSQL nabízí programování na straně serveru, a to v jak vestavěném jazyku PL/pgSQL, tak i jazycích Tcl, Perl a Python – včetně spouští (triggerů). Databázový server lze velmi snadno rozšiřovat, a to nejen o nové datové typy, ale i o nové jazyky. Existuje velké množství

zásuvných modulů, rozšiřující funkčnost (cubes, fuzzy strings, stromy, soundex, XML...)

Ve verzi 8.0 najdeme kromě jiného novinky, jako jsou body návratu transakcí (savepoints), tabulkové prostory (tablespaces), průběžné (přírůstkové) zálohování a mnoho výkonnostních optimalizací.

Modul Tsearch2

Pokud máme nainstalovaný modul, je aktivace full-textu hračkou:

```
createdb test

psql test

test=# \i /usr/share/pgsql/contrib/tsearch2.sql
```

Skript zaregistruje a vytvoří potřebné struktury a tabulky a aktivuje nový datový typ tsvector. Přesvědčíme se, jak jsme na tom:

```
test=# select dict_name, dict_comment, dict_initoption from pg_ts_dict;
    dict_name    |                  dict_comment                  |
dict_initoption
-----------------+------------------------------------------------
+----------------------
 simple          | Simple example of dictionary.                  |
 en_stem         | English Stemmer. Snowball.                     |
contrib/english.stop
 ru_stem         | Russian Stemmer. Snowball.                     |
contrib/russian.stop
 ispell_template | ISpell interface. Must have .dict and .aff files |
 synonym         | Example of synonym dictionary                  |
```

Vidíme, že skript zaregistroval několik slovníků, nás ovšem pochopitelně zajímá podpora češtiny, kterou nyní zavedeme. Nejdřív vyzkoušíme, že funguje stemování (vracení kořenů) slov, které je pro fulltextové rejstříkování nejdůležitější:

```
test=# SELECT lexize('en_stem','looking');
 lexize
--------
 {look}
```

Modul umí načíst slovník pro ISPELL, což je program z projektu GNU sloužící k automatické kontrole textu. Tyto slovníky nejen že obsahují obrovské množství českých slov, ale také všechny jejich tvary a základní lexémy. Nejprve zaregistrujeme slovník a soubor se stop slovy:

```
INSERT INTO pg_ts_dict (
 SELECT 'cz_ispell', dict_init,
 'DictFile="/var/tmp/czech.dict",
 AffFile="/var/tmp/czech.aff",
 StopFile="/var/tmp/czech.stop"',
 dict_lexize FROM pg_ts_dict WHERE
 dict_name='ispell_template');
```

Nyní vše vyzkoušíme:

```
test=# SELECT lexize('cz_ispell','databázové');
    lexize
-------------
 {databázový}
```

Kromě stemmingu a stop slov lze také naimportovat synonyma, která mohou výsledky hledání vylepšit. Soubory se synonymy mají podobný tvar jako se stop slovy – obyčejné textové soubory, co řádek to pojem, mezerou jsou odděleny jednotlivé další výrazy.

O rozložení textu na jednotlivá slova se stará parser. Modul Tsearch2 obsahuje jednoduchý parser, který si poradí s obyčejným textem a HTML zdrojovým kódem. Otevřená architektura umožňuje napsat si vlastní parser. Otestujme jej:

```
test=# SELECT * FROM parse('V databázích jsou uloženy všechny informace o nás.');
 tokid |   token
-------+------------
     1 | V
    12 |
     3 | databázích
    12 |
     1 | jsou
    12 |
     3 | uloženy
    12 |
     3 | všechny
    12 |
     1 | informace
    12 |
     1 | o
    12 |
     3 | nás
    12 | .
(16 řádek)
```

Ačkoli je to parser jednoduchý, zvládá speciality typu URL adresy nebo e-mailu. A nyní již můžeme přistoupit k vlastní tvorbě tsvektorů a práce s nimi. Nejprve je nutno nastavit parser (kódování a sloník):

```
INSERT INTO pg_ts_cfg VALUES ('default_czech','default','cs_CZ');

INSERT INTO pg_ts_cfgmap SELECT 'default_czech',tok_alias,dict_name FROM pg_ts_cfgmap WHERE ts_name='default_russian';

UPDATE pg_ts_cfgmap SET dict_name='{cz_ispell,simple}' WHERE ('ru_stem'=ANY(dict_name) OR 'en_stem' = ANY(dict_name)) AND ts_name='default_czech';
```

Parser nyní funguje správně, lematizuje slova:

```
test=# select to_tsvector('V databázích jsou uloženy všechny informace o nás.');
to_tsvector
-------------------------------------------------------
 'nás':8 'uložit':4 'všechny':5 'databáze':2 'informace':6
```

Nyní již můžeme snadno indexovat dokumenty nebo jiné texty:

```
CREATE TABLE dokumenty (
 id SERIAL PRIMARY KEY,
 t text,
 v tsvector);
CREATE INDEX idxFTI_idx ON dokumenty USING gist(v);
CREATE TRIGGER tsvectorupdate BEFORE UPDATE OR INSERT ON dokumenty FOR EACH ROW EXECUTE PROCEDURE tsearch2(v, t);
INSERT INTO dokumenty (t) VALUES ('Text dokumentu...');
```

Díky GIST indexu je nyní možné fulltextově vyhledávat pomocí binárního operátoru @@. Ve vyhledávacím dotazu je možné použít &, | nebo negaci !

```
test=# SELECT t FROM dokumenty WHERE v @@ to_tsquery('Oracle');
            t
-------------------------
 Oracle je super server!
(1 řádka)


test=# SELECT t FROM dokumenty WHERE v @@ to_tsquery('!něco');
              t
-----------------------------
 Oracle je super server!
 Ale to je PostgreSQL taky!!
(2 řádek)


test=# SELECT t FROM dokumenty WHERE v @@ to_tsquery('(něco & ještě) | server');
                       t
-----------------------------------------------
 Oracle je super server!
 A ještě něco, ať je tam nějaká diakritika...
```

Modul Tsearch2 poskytuje jednoduchou implementaci fulltextu. Je snadno rozšiřitelný a díky tomu, že se jedná o open-source projekt, má i slibnou budoucnost.

Odkazy:

http://www.postgres.org
Domovská stránka projektu

http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2/
Modul Tsearch2