Novinky v PostgreSQL 9.4

Tomáš Vondra, 2ndQuadrant (tomas@2ndquadrant.com)
http://blog.pgaddict.com (tomas@pgaddict.com)

vývojáři

JSONB

aggregate expressions (FILTER)

```
SELECT
    a,
    SUM(CASE WHEN b <  10 THEN c ELSE NULL END) AS pod_10,
    SUM(CASE WHEN b >= 10 THEN c ELSE NULL END) AS nad_10
FROM tabulka GROUP BY a;
```

```
SELECT
    a,
    SUM(c) FILTER (WHERE b <  10) AS pod_10,
    SUM(c) FILTER (WHERE b >= 10) AS nad_10
FROM tabulka GROUP BY a;
```

~~ordered-set aggregates~~

- pořadí agregovaných hodnot
  - často nepodstatné (MIN, MAX, SUM, …)
  - někdy na něm ale záleží (array_agg, string_agg, …)

```
SELECT
    SUM(b ORDER BY c),        -- zbytečné třídění
    ARRAY_AGG(b ORDER BY c) -- setříděné pole
FROM tabulka;
```

(Toto není ordered-set aggregate!)

ordered-set aggregates

- některé agregační funkce pořadí vyžadují
  - z definice (jinak prostě nedávají smysl)
  - rank, percentil, ...
  - direct / aggregate argumenty (rozdíl)

```
-- medián hodnot „b“
SELECT
    PERCENTILE_DISC(0.5) WITHIN GROUP (ORDER BY b)
FROM tabulka GROUP BY a;
```

ordered-set aggregates

- některé agregační funkce pořadí vyžadují
  - z definice (jinak prostě nedávají smysl)
  - rank, percentil, ...
  - direct / aggregate argumenty (rozdíl)

```
SELECT
    a,
    PERCENTILE_DISC(ARRAY[0.25, 0.5, 0.75])
                    WITHIN GROUP (ORDER BY b) AS median_b
FROM tabulka GROUP BY a;
```

(proprietární rozšíření)

hypotetical aggregates

- Kam by se zařadil hypotetický řádek?
  - pořadí – `rank(..), dense_rank(..)`
  - relativní (0, 1) – `percent_rank(..), cume_dist(..)`

```
SELECT
    a,
    rank('xyz')         WITHIN GROUP (ORDER BY b),
    dense_rank('xyz')   WITHIN GROUP (ORDER BY b),
    percent_rank('xyz') WITHIN GROUP (ORDER BY b)
FROM tabulka GROUP BY a;
```

automaticky updatovatelné pohledy

- pro jednoduché pohledy lze „překládat“ DML příkazy
  - jedna tabulka ve FROM, bez agregací / množinových operací, apod.
  - ~~**pouze jednoduché odkazy na sloupce tabulky**~~
  - ~~**nesmí být označený pomocí „security_barrier“**~~
- od 9.4 lze ve views používat výrazy, konstanty, volání funkcí
  - tyto sloupce samozřejmě nelze měnit

```
CREATE VIEW zamestnanci_view AS
     SELECT emp_id, dept_id,
            (salary*0.6) AS cista_mzda
     FROM zamestnanci_tabulka
    WHERE dept_id IN (10, 20);
```

automaticky updatovatelné pohledy

- řádky odfiltrované přes WHERE nelze updatovat
- řádek ale může „vypadnout“
  - WITH CHECK OPTION tomu zabrání
  - další krok na cestě k „Row Level Security“ (řízení přístupu k řádkům)

```
CREATE VIEW zamestnanci_view AS
     SELECT id_zamestnance, id_oddeleni,
            (plat*0.6) AS cista_mzda
     FROM zamestnanci_tabulka
    WHERE id_oddeleni IN (10, 20)
    WITH CHECK OPTION;

UPDATE zamestnanci_view SET id_oddeleni = 30;
```

admini

MATERIALIZED VIEWS

```
CREATE MATERIALIZED VIEW my_view AS SELECT …;

CREATE UNIQUE INDEX my_index ON my_view (…);

REFRESH MATERIALIED VIEW my_view;

REFRESH MATERIALIED VIEW CONCURRENTLY my_view;
```

SET TABLESPACE ...

```
ALTER TABLE ALL IN TABLESPACE tablespace1
      OWNED BY uzivatel
      SET TABLESPACE tablespace2 [NOWAIT];

ALTER INDEX ...

ALTER VIEW ...

ALTER MATERIALIZED VIEW ...
```

Vylepšení GIN indexů

- pro hodnoty složené z „částí“ (pole, slova, …)
- index se skládá z položek

```
key1 => [rowid1, rowid2, rowid3, …]
key2 => [rowid10, rowid20, rowid30, …]
…
```

- v podstatě bitmapové indexy (zvláštně kódované)
  - jde použít na skalární typy (extenze btree_gin)
- 9.4 přináší dvě významná vylepšení
  - komprese posting listů (seznam odkazů na řádky)
  - fast scan (dotazy typu „častý & vzácný“ výrazně rychlejší)

ALTER SYSTEM

- dosud bylo nutné přímo editovat `postgresql.conf`

```
vim /var/lib/pgsql/9.1/data/postgresql.conf
```

- nově je možné toho dosáhnout přímo z databáze

```
ALTER SYSTEM SET work_mem = '128MB';
```

- vytvoří se nový soubor s konfigurací (include)

```
postgresql.auto.conf
```

- stále nutný explicitní reload / restart

```
SELECT pg_reload_conf();
```

Konfigurační parametry

- `autovacuum_work_mem`
  - odděleno z `maintenance_work_mem`
- `huge_pages`
  - Linuxové systémy s velkým objemem RAM
- `wal_log_hints`
  - standardně se nelogují, ale občas problémy (replikace, pg_rewind)
- `(maintenance_)work_mem / effective_cache_size`
  - navýšení default hodnot (4x)

pg_stat_statements

- texty dotazů se ukládají do souboru
  - odstraňuje limit na délku dotazu
  - šetří sdílenou paměť
  - načítání bez textů (optimalizace motorovacích nástrojů)

```
SELECT * FROM pg_stat_statements(false);
```

- doplnění „query ID“ (interní hash dotazu)
  - identifikace dotazu (monitorovací nástroje)
  - nestabilní mezi major verzemi / platformami

replikanti

Replikace a recovery

- přesun tablespaces v pg_basebackup
- time-delayed standby
- pg_stat_archiver

Infrastruktura

- základ logické replikace (changeset extraction)
- replikační sloty
- background workers

Spousta dalšího ...

9.4 release notes

http://www.postgresql.org/docs/9.4/static/release-9-4.html

článek od Pavla Stěhule (květen 2014)

http://www.root.cz/clanky/postgresql-9-4-transakcni-sql-json-databaze/

What's new in PostgreSQL 9.4 (Magnus Hagander)

http://www.hagander.net/talks/postgresql94_2.pdf

PostgreSQL Conference Europe 2014 (říjen, Madrid)

https://wiki.postgresql.org/wiki/PostgreSQL_Conference_Europe_Talks_2014