PostgreSQL 9.5

Postgres Open 2015
Dallas, TX

Magnus Hagander
magnus@hagander.net

Magnus Hagander

- PostgreSQL
  - Core Team member
  - Committer
  - PostgreSQL Europe
- Redpill Linpro
  - Infrastructure services
  - Principal database consultant

Do you read...

- planet.postgresql.org

Development schedule

- June 10, 2014 - branch 9.4
- June 2014 - CF1
- August 2014 - CF2
- October 2014 - CF3
- December 2014 - CF4
- February 2015 - CF5
- August 2015 - Alpha2!

Current status

- Alpha 2 has been released
- Please help with review and testing!
- Packages now available!

Current status

- Statistics!
  - 2597 files changed
  - 215199 insertions (+)
  - 220459 deletions(-)
- Almost double that of 9.4!
  - But..?

So what's really new

- Developer and SQL features
- DBA and administration
- Performance

New features

- Developer and SQL features
- DBA and administration
- Performance

Multi-column subselect UPDATE

- Update more than one column with subselect
- SQL standard syntax

```
UPDATE tab SET (col1, col2) =
 (SELECT foo, bar FROM tab2)
 WHERE ...
```

Numeric generate_series

- Previously "only" integer
  - And timestamps
- Now decimals and bigger numbers

```
postgres=# SELECT * FROM generate_series(0, 1, 0.1);
 generate_series
-----------------
               0
             0.1
             0.2
             0.3
```

SKIP LOCKED

- Like SELECT NOWAIT
- Except skip rows instead of error

```
postgres=# SELECT * FROM a FOR UPDATE NOWAIT;
ERROR:  could not obtain lock on row in relation "a"
postgres=# SELECT * FROM a FOR UPDATE SKIP LOCKED;
 a  | b  | c
----+----+----
  2 |  2 |  2
  3 |  3 |  3
```

Row level security

- Apply access policies per row
- Limit access to individual rows
  - On top of tables and columns
  - Regular ACLs still apply
- Superusers and owners bypass
  - And BYPASSRLS roles

Row level security

```
postgres=# ALTER TABLE companies ENABLE ROW LEVEL SECURITY;
ALTER TABLE

postgres=# CREATE POLICY companies_manager
postgres-#  ON companies
postgres-#  FOR ALL
postgres-#  TO public
postgres-#  USING (manager = CURRENT_USER);
CREATE POLICY
```

Row level security

```
postgres=# SELECT * FROM companies;
 manager | company
---------+----------
 mha     | Company1
 mha     | Company2
 test    | Company3
postgres=# \c postgres test
You are now connected to database "postgres" as user "test".
postgres=> select * from companies;
 manager | company
---------+----------
 test    | Company3
```

Row level security

- Policies on any "regular" expression
  - No aggregates!
  - But quite complicated
- Multiple policies can be defined per table
  - Results are ORed
- Does not affect cascading RI operations

Row level security

```sql
CREATE POLICY companies_manager_r
ON companies
USING (manager IN (
  WITH RECURSIVE t AS (
    SELECT person,manager FROM managers WHERE manager=CURRENT_USE
   UNION ALL
    SELECT m.person, m.manager FROM managers m
     INNER JOIN t ON t.person=m.manager
  )
  SELECT person FROM t
))
```

INSERT ... ON CONFLICT

- INSERT ... ON CONFLICT DO {UPDATE | IGNORE}
- aka UPSERT
- Similar to MERGE
  - Except better (in some ways)!
  - Based on "speculative insertion"

INSERT ... ON CONFLICT

```sql
INSERT INTO test (id, t)
VALUES (2, 'foobar')
ON CONFLICT
DO NOTHING
```

INSERT ... ON CONFLICT

```
INSERT INTO test (id, t)
VALUES (2, 'foobar')
ON CONFLICT(id) DO
UPDATE SET t=excluded.t
```

INSERT ... ON CONFLICT

```sql
INSERT INTO counters(url, num)
VALUES ('/some/where', 1)
ON CONFLICT(url) DO
UPDATE SET num=counters.num+excluded.num
```

GROUPING SETS

- CUBE and ROLLUP
  - But also fully generic
- "Super-aggregates"
- Partial sums etc

GROUPING SETS

```
postgres=# SELECT dept, count(*) FROM emps
postgres-# GROUP BY ROLLUP(dept);
 dept  | count
-------+-------
 it    |     3
 sales |     2
       |     5
```

GROUPING SETS

```
postgres=# SELECT dept, name, count(*), sum(payout)
postgres-# FROM payouts GROUP BY ROLLUP(dept, name);
 dept  | name  | count | sum
-------+-------+-------+------
 it    | Eva   |     3 |  400
 it    | Johan |     2 |  350
 it    | Olle  |     1 |  200
 it    |       |     6 |  950
 sales | Erik  |     1 |  120
 sales | Lisa  |     2 |  220
 sales |       |     3 |  340
       |       |     9 | 1290
```

New features

- Developer and SQL features
- DBA and administration
- Performance

cluster_name

- New GUC
- Included in process title
- For multi-instance deployments

```
31589 ?        Ss     0:00 postgres: mytestcluster: logger process
31591 ?        Ss     0:00 postgres: mytestcluster: checkpointer p
```

IMPORT FOREIGN SCHEMA

- Import complete schema through FDW
- No need to manually create tables

```
postgres=# CREATE SCHEMA remoteschema;
CREATE SCHEMA
postgres=# IMPORT FOREIGN SCHEMA testschema FROM SERVER otherserve
IMPORT FOREIGN SCHEMA
postgres=# \det remoteschema.*
       List of foreign tables
    Schema    | Table |   Server
--------------+-------+-------------
 remoteschema | test2 | otherserver
 remoteschema | test3 | otherserver
(1 row)
```

Foreign table inheritance

- Foreign tables can be in inheritance trees
- Which is used for partitioning
- Can be used for sharding

SET UNLOGGED

- Unlogged table property can be turned on and off
- Simple ALTER statement

```
postgres=# ALTER TABLE a SET UNLOGGED;
ALTER TABLE
postgres=# ALTER TABLE a SET LOGGED;
ALTER TABLE
```

ALTER SYSTEM RESET

- Reset config variable back to
  - postgresql.conf
  - default value
- Removes from postgresql.auto.conf file

```
postgres=# ALTER SYSTEM RESET work_mem;
ALTER SYSTEM
postgres=# SELECT pg_reload_conf();
```

commit timestamp tracking

- Optional tracking of commit timestamps
  - track_commit_timestamp=on
- See when a row was committed etc?

```
postgres=# SELECT xmin, pg_xact_commit_timestamp(xmin) FROM  a;
 xmin |   pg_xact_commit_timestamp
------+-------------------------------------
  787 | 2015-03-15 15:09:52.253007+00

postgres=# SELECT * FROM pg_last_committed_xact();
 xid |           timestamp
-----+-------------------------------------
 791 | 2015-03-15 15:11:38.709125+00
```

min and max wal size

- checkpoint_segments removed!
- Instead, control min and max size
  - min_wal_size (default 80MB)
  - max_wal_size (default 1GB)
- Checkpoints auto-tuned to happen in between
  - Moving average of previous checkpoints
- Space only consumed when actually needed

recovery_target_action

- What happens when recovery completes
  - pause
  - promote
  - shutdown
- Replaces pause_at_recovery_target

pg_rewind

- Ability to rewind WAL on old master
- Re-use former master without rebuild

SSL code refactoring

- OpenSSL independence
- Though only OpenSSL supported so far...
- Add support for Subject Alternate Name

pg_stat_ssl

- View status of existing SSL connection
- Mostly same info as contrib/sslinfo
- But for all connections

`pg_stat_statements`

- New values for execution times
  - Max
  - Min
  - Mean
  - Stddev

pg_xlogdump

- Now takes --stats argument
- Find out what takes space in the xlog
- (and of course look at details like before)

New features

- Developer and SQL features
- DBA and administration
- Performance

BRIN indexes

- Block Range Index
  - Formerly known as MinMax
  - But supports other opclasses too
- Very small indexes
- Stores only bounds-per-block-range
  - Default is 128 blocks
- Scans all blocks for matches
- Best suited for naturally ordered tables

BRIN indexes

```
postgres=# CREATE INDEX a_brin ON a USING BRIN(a);
CREATE INDEX
postgres=# EXPLAIN SELECT * FROM a WHERE a=3;
                              QUERY PLAN
----------------------------------------------------------------------
 Bitmap Heap Scan on a  (cost=12.01..16.02 rows=1 width=12)
   Recheck Cond: (a = 3)
   ->  Bitmap Index Scan on a_brin  (cost=0.00..12.01 rows=1 widt
         Index Cond: (a = 3)

postgres=# CREATE INDEX a_brin_b ON a
postgres-#   USING BRIN(b) WITH (pages_per_range=1024);
CREATE INDEX
```

GIN pending list

- Max size of GIN pending list configurable
    - Used for GIN fast update
    - Control how often cleanup happens
    - Prefer VACUUM
- Previously controlled by work_mem
- Now gin_pending_list_limit
    - Both GUC and storage parameter

GiST index only scan

- Index only scan for GiST indexes
- Most, but not all, opclasses

WAL compression

- Support for compressing full page images
- Smaller WAL
  - Faster writes, faster replication
  - Costs CPU
- Only compresses FPIs
  - Still useful to gzip archives!
- Also new WAL format and CRC

Sorting enhancements

- Abbreviated keys for sorting
  - text
  - numeric
- Pre-check for equality
  - memcmp is fast!
- more...

Locking enhancements

- Internal atomic operations API
- lwlock scalability increased using this
- Many more lockless operations
  - E.g. triggers and foreign keys
- etc.

There's always more

There's always more

- Lots of smaller fixes
- Performance improvements
- etc, etc
- Can't mention them all!

Tiny favorite?

- psql detects if sent a custom format dump
- We all did this:

```
mha@mha-laptop:~$ 9.4/bin/psql -f /tmp/custom.dump postgres
psql:/tmp/custom.dump:1: ERROR:  syntax error at or near "PGDMP"
LINE 1: PGDMP
```

- Now:

```
mha@mha-laptop:~$ head/bin/psql -f /tmp/custom.dump postgres
The input is a PostgreSQL custom-format dump.
Use the pg_restore command-line client to restore this dump to a
```

What's your biggest feature?

- UPSERT?
- GROUPING SETS?
- RLS?
- Foreign Table Inheritance?
- BRIN?
- Other?

Thank you!

Magnus Hagander
magnus@hagander.net
@magnushagander
http://www.hagander.net/talks/