Introduction to PostgreSQL

Don't Panic!

Federico Campoli

Brighton PostgreSQL Users Group

13 June 2016

Table of contents

Table of contents

1. An elephant never forgets
2. A quick look to a powerful tool
3. NOSQL on Acid
4. Wrap up

An elephant never forgets

PostgreSQL, an history of excellence

- Created at Berkeley in 1982 by database's legend Prof. Stonebraker
- In the 1994 Andrew Yu and Jolly Chen added the SQL interpreter
- In the 1996 becomes an Open Source project.
- The project's name changes in PostgreSQL

PostgreSQL, an history of excellence

- Created at Berkeley in 1982 by database's legend Prof. Stonebraker
- In the 1994 Andrew Yu and Jolly Chen added the SQL interpreter
- In the 1996 becomes an Open Source project.
- The project's name changes in PostgreSQL
- Fully ACID compliant
- High performance in read/write with the MVCC
- Tablespaces

PostgreSQL, an history of excellence

- Created at Berkeley in 1982 by database's legend Prof. Stonebraker
- In the 1994 Andrew Yu and Jolly Chen added the SQL interpreter
- In the 1996 becomes an Open Source project.
- The project's name changes in PostgreSQL
- Fully ACID compliant
- High performance in read/write with the MVCC
- Tablespaces
- Runs on almost any unix flavour
- From the version 8.0 is native on *cough* MS Windows *cough*
- HA with hot standby and streaming replication
- Heterogeneous federation

PostgreSQL, an history of excellence

- Created at Berkeley in 1982 by database's legend Prof. Stonebraker
- In the 1994 Andrew Yu and Jolly Chen added the SQL interpreter
- In the 1996 becomes an Open Source project.
- The project's name changes in PostgreSQL
- Fully ACID compliant
- High performance in read/write with the MVCC
- Tablespaces
- Runs on almost any unix flavour
- From the version 8.0 is native on *cough* MS Windows *cough*
- HA with hot standby and streaming replication
- Heterogeneous federation
- Procedural languages (pl/pgsql, pl/python, pl/perl...)
- Support for NOSQL features like HSTORE and JSON

Development

- Old ugly C language
- New development cycle starts usually in June
- New version released usually by the end of the year
- At least 4 LTS versions
- Can be extended using shared libraries
- Extensions (9.1+)
- BSD like license

Limits

- Database size. No limits.
- Table size 32 TB
- Row size 1.6 TB
- Field size 1 GB
- Rows in table. No limits.
- Fields in table 250 - 1600 depending on data type.
- Tables in a database. No limits.

Table of contents

1. An elephant never forgets
2. A quick look to a powerful tool
3. NOSQL on Acid
4. Wrap up

A quick look to a powerful tool

Image by Hein Waschefort -
http://commons.wikimedia.org/wiki/User:Hein_waschefort

Data types

PostgreSQL comes with an incredibly rich data type set.

Data types

Numerical

- smallint, integer, bigint
- decimal, numeric, user-specified precision, exact
- real,double precision, variable precision, inexact

Data types

Character

- character
- character varying with max size
- text, character varying

Data types

Binary

- bytea

Data types

Alongside the general purpose data types PostgreSQL have some exotic types.

- Range (integers, date)
- Geometric (points, lines etc.)
- Network addresses
- XML
- JSON
- JSONB
- HSTORE (extension)

UPSERT (9.5+)

```
INSERT INTO t_table (i_id, v_value)
    VALUES (1, ’fake value’), (2, ’another fake value’)
    ON CONFLICT (i_id) DO UPDATE SET v_value = EXCLUDED.v_value;
```

Row Level Security (9.5+)

Row Level Security, allows security "policies" filtering rows per database user.

Big Data! (9.5+)

BIG DATA!

- BRIN - Block Range Indices
- IMPORT FOREIGN SCHEMA, Federation with steroids
- TABLESAMPLE

GROUPING SETS

GROUPING SETS (shameless copied from the on line manual)
The data selected by the FROM and WHERE clauses is grouped separately by each specified grouping set, aggregates computed for each group just as for simple GROUP BY clauses, and then the results returned. For example:

```
=> SELECT * FROM items_sold;
 brand | size | sales
-------+------+-------
 Foo   | L    |  10
 Foo   | M    |  20
 Bar   | M    |  15
 Bar   | L    |  5
(4 rows)

=> SELECT brand, size, sum(sales) FROM items_sold GROUP BY GROUPING SETS ((
    brand), (size), ());
 brand | size | sum
-------+------+-----
 Foo   |      |  30
 Bar   |      |  20
       | L    |  15
       | M    |  35
       |      |  50
(5 rows)
```

The future

PostgreSQL 9.6

Currently in beta

- Parallel sequential scans, joins and aggregates
- Push down for the postgresql foreigh data wrapper
- Full text search for phrases, YAY!
- Multiple synchronous standby servers
- Snapshot too old!

Table of contents

1 An elephant never forgets

2 A quick look to a powerful tool

3 NOSQL on Acid

4 Wrap up

NOSQL on Acid

JSON

JSON - JavaScript Object Notation

- The version 9.2 adds JSON as native data type
- The version 9.3 adds the support functions for JSON
- JSON is stored as text
- JSON is parsed and validated on the fly
- The 9.4 adds JSONB (binary) data type
- The 9.5 improves JSONB

JSON

JSON - Examples
From record to JSON

```
postgres=# SELECT row_to_json(ROW(1,’foo’));
     row_to_json
---------------------
 {"f1":1,"f2":"foo"}
(1 row)
```

Expanding JSON into key to value elements

```
postgres=# SELECT * from json_each(’{"a":"foo", "b":"bar"}’);

 key | value
-----+-------
 a   | "foo"
 b   | "bar"
(2 rows)
```

JSONB

Because JSON is parsed and validated on the fly it could be a bottleneck.

The new JSONB introduced with PostgreSQL 9.4 is parsed, validated and transformed at insert/update's time. The access is then faster than the plain JSON but the storage cost can be higher.
The functions available for JSON are also available in the JSONB flavour.

Some numbers

Let's create three tables with text,json and jsonb type fields.
Each record contains the same json element generated on
http://beta.json-generator.com/4kwCt-fwg

```
[ {
"_id": "56891aba27402de7f551bc91",
"index": 0,
"guid": "b9345045-1222-4f71-9540-6ed7c8d2ccae",
"isActive": false,
............
3,
{
"id": 1,
"name": "Bridgett Shaw"
}
],
"greeting": "Hello, Johnston! You have 8 unread messages.",
"favoriteFruit": "apple"
}
```

Some numbers

```
DROP TABLE IF EXISTS t_json ;
DROP TABLE IF EXISTS t_jsonb ;
DROP TABLE IF EXISTS t_text ;

CREATE TABLE t_json as
SELECT
'<JSON ELEMENT>'::json as js_value
FROM
generate_series(1,100000);
Query returned successfully: 100000 rows affected, 14504 ms execution time.


CREATE TABLE t_text as
SELECT
'<JSON ELEMENT>'::text as t_value
FROM
generate_series(1,100000);
Query returned successfully: 100000 rows affected, 14330 ms execution time.


CREATE TABLE t_jsonb as
SELECT
'<JSON ELEMENT>'::jsonb as jsb_value
FROM
generate_series(1,100000);
Query returned successfully: 100000 rows affected, 14060 ms execution time.
```

Table size

```
SELECT
       pg_size_pretty(pg_total_relation_size(oid)),
       relname
FROM
       pg_class
WHERE
       relname LIKE ’t\_%’
;

pg_size_pretty | relname
----------------+---------
 270 MB         | t_json
 322 MB         | t_jsonb
 270 MB         | t_text
(3 rows)
```

Sequential scans

TEXT

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT * FROM t_text;
```

```
Seq Scan on t_text  (cost=0.00..1637.00 rows=100000 width=18) (actual time
    =0.016..17.624 rows=100000 loops=1)
   Buffers: shared hit=637
 Planning time: 0.040 ms
 Execution time: 28.967 ms
(4 rows)
```

Sequential scans

JSON

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT * FROM t_json;

Seq Scan on t_json  (cost=0.00..1637.09 rows=100009 width=32) (actual time
    =0.018..15.443 rows=100000 loops=1)
   Buffers: shared hit=637
 Planning time: 0.045 ms
 Execution time: 25.268 ms
(4 rows)
```

Sequential scans

JSONB

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT * FROM t_jsonb;

Seq Scan on t_jsonb  (cost=0.00..1637.00 rows=100000 width=18) (actual time
    =0.015..18.943 rows=100000 loops=1)
   Buffers: shared hit=637
 Planning time: 0.043 ms
 Execution time: 31.072 ms
(4 rows)
```

Sequential scan with json access

TEXT

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT t_value::json->'index' FROM t_text;

Seq Scan on t_text  (cost=0.00..2387.00 rows=100000 width=18) (actual time
    =0.159..7748.381 rows=100000 loops=1)
   Buffers: shared hit=401729
 Planning time: 0.028 ms
 Execution time: 7760.263 ms
(4 rows)
```

Sequential scan with json access

JSON

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT js_value->'index' FROM t_json;

Seq Scan on t_json  (cost=0.00..1887.11 rows=100009 width=32) (actual time
     =0.254..5787.267 rows=100000 loops=1)
   Buffers: shared hit=401730
 Planning time: 0.044 ms
 Execution time: 5798.153 ms
(4 rows)
```

Sequential scan with json access

JSONB

```
EXPLAIN  (BUFFERS, ANALYZE) SELECT jsb_value->'index' FROM t_jsonb;

Seq Scan on t_jsonb  (cost=0.00..1887.00 rows=100000 width=18) (actual time
     =0.138..1678.222 rows=100000 loops=1)
   Buffers: shared hit=421729
 Planning time: 0.048 ms
 Execution time: 1688.752 ms
(4 rows)
```

Table of contents

1. An elephant never forgets

2. A quick look to a powerful tool

3. NOSQL on Acid

4. Wrap up

Wrap up

PostgreSQL is a powerful RDBMS with dozens of features and capabilities.

Choosing the correctly what to use can be tricky.

The lack of horizontal scalability in PostgreSQL can be a problem for large data sets. However, an interesting project for a distributed cluster is PostgreSQL XL - http://www.postgres-xl.org/

Another cool project is CitusDB, a powerful extension to add to PostgreSQL horizontal scale capabilities.
CitusDB recently un-forked from PostgreSQL becoming an open source extension. Yay!

Wrap up

- Schema less data are useful. They are flexible and powerful.
- Never forget PostgreSQL is a RDBMS
- Get a DBA on board

Questions

Questions?

Contacts

- Twitter: 4thdoctor_scarf
- Personal blog: http://www.pgdba.co.uk
- PostgreSQL Book: http://www.slideshare.net/FedericoCampoli/postgresql-dba-01
- Brighton PostgreSQL Meetup: http://www.meetup.com/Brighton-PostgreSQL-Meetup/

License and copyright

This presentation is licensed under the terms of the Creative Commons Attribution NonCommercial ShareAlike 4.0

http://creativecommons.org/licenses/by-nc-sa/4.0/

- The cheetah photo is copyright by Hein Waschefort - http://commons.wikimedia.org/wiki/User:Hein_waschefort
- The elephant logos are copyright of the PostgreSQL Global Development Group - http://www.postgresql.org/

Introduction to PostgreSQL

Don't Panic!

Federico Campoli

Brighton PostgreSQL Users Group

13 June 2016