PostgreSQL & the Postgres community

JPUG pgCon 2013, Tokyo

Peter Geoghegan
(Rhymes with "Ronald Reagan")
pg@heroku.com
Twitter: @sternocera

About me

- An enthusiast of PostgreSQL since around 2006, though started contributing fairly recently - early 2011.
- Work for Heroku; Cloud platform that lets developers focus on their application. Heroku can take some credit for popularizing PostgreSQL among "new wave" of application developers.
- Used to work as a consultant, so have seen many types of PostgreSQL databases and use-cases.

About me (Cont.): Projects

- pg_stat_statements normalization - built on the work of Takahiro Itagaki of NTT.
- Performance features.
- Currently working on “UPSERT”.

What is “UPSERT”?

- Very simple idea - developer wants to either insert a new row, or update an existing one instead if that is appropriate (because to insert would create a undesired duplicate).

- Very notable omission, though can be worked around.

- Complex concurrency issues frustrate implementation. Simple at first, but the longer you look, the more complex it becomes.

- I have been looking for quite a while now!

```
WITH rej AS
(
        INSERT INTO test(a, b)
        VALUES(123, 'Vancouver'),
              (456, 'Dublin'),
              (789, 'Tokyo')
        ON DUPLICATE KEY LOCK FOR UPDATE
        RETURNING REJECTS *
)
UPDATE test SET test.b = rej.b FROM rej
WHERE test.a = rej.a;
```

- For each tuple, INSERTs, or "projects" rejects for UPDATE.
- I expect use with *writeable* common table expressions as shown here will become idiomatic.
- Relies on unique btree indexes (Primary keys/unique constraints).
- Some degree of *generality* - can be composed, with ability to, for example, DELETE rather than UPDATE.
- Some MERGE-like capabilities (e.g. insert-rejected tuples can be "pipelined", or UPDATE'd or DELETE'd based on some criteria), but not as powerful.

Why work on PostgreSQL?

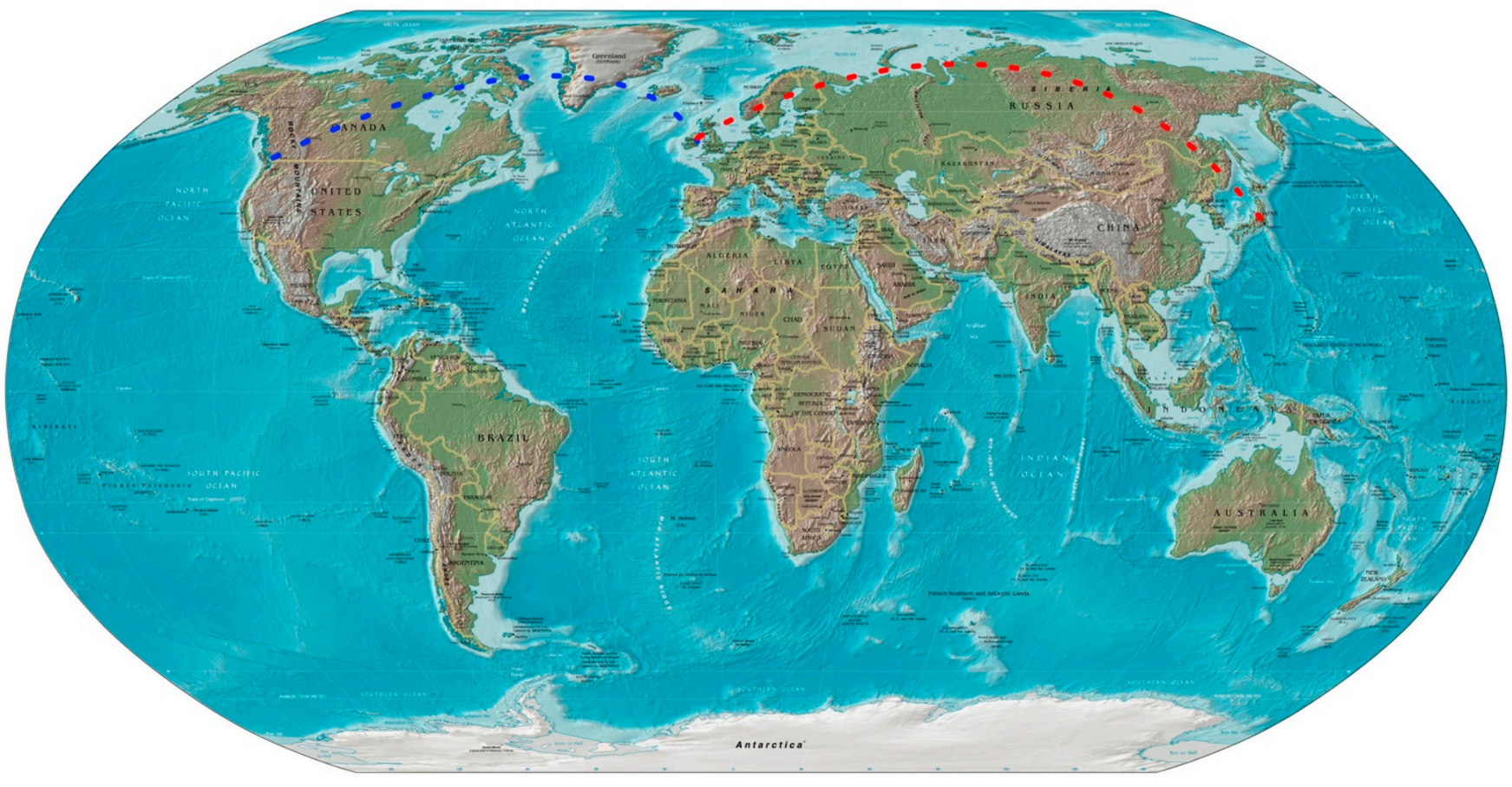
- Found underlying philosophy - emphasis on extensibility, consistency, rich datatype support and support for semi-structured data, and extensible indexing - elegant and deeply compelling.
- Code quality very high - the mark of a project built to last many years. When making such a big personal investment, we want it to pay off!
- Great community. Opportunity to learn from others with many years of experience.

Why work on PostgreSQL? (Cont.)

- There are many interesting problems to solve. Database systems touch upon many areas of computer science.
- Apart from core concepts in transaction processing, an understanding of other computer science concepts is helpful too. This includes for example compiler theory, operating system design, and a general understanding of both common algorithms (e.g. Quicksort) and more novel algorithms (e.g. HyperLogLog).
- Desire to do something socially useful - to "make a difference" and help people in my own small way.

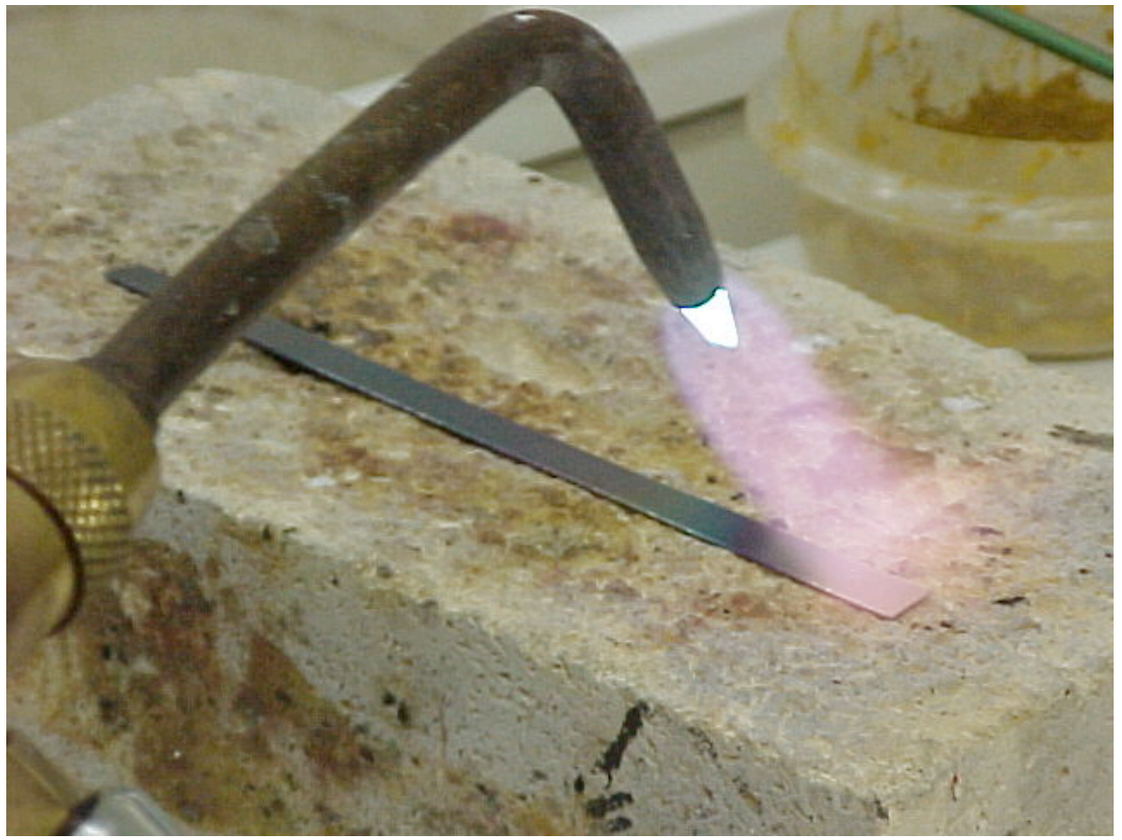
Why work on PostgreSQL? (Cont.)

Bonus: I get to travel around the world and learn about the interesting ways in which people are working with PostgreSQL!



Many contributors, one great database system

- Each person's contribution "anneals" into a single, coherent whole.



- Every contributor and company has their own motives, presumably many similar to my own.

- The "scratch my own itch" motivation is perhaps overall the single strongest.

PostgreSQL

Not so much a database as a *data platform*

What's a data platform?

- Fundamentally, it's the job of a database to provide a *generalized* solution to data management problems. They offer a flexible, reusable solution.
- Basically, all database internals developers solve complex problems around things like concurrency control, crash safety and general data integrity *so you don't have to*.
- In the 1980s, professor Stonebraker created the POSTGRES project to make rich datatypes a first-order concern.

What's a data platform? (Cont.)

- At that time, it proved necessary to solve problems like alternative date representations (e.g. in financial applications) in the database.
- Ad-hoc methods work, but managing that complexity does not scale.
- Asking complicated questions about the data requires custom indexing.
- PostgreSQL has many great strengths, but I believe that its greatest, enduring strength is its flexibility.

What's a data platform? (Cont.)

- If the core function of a database is to provide flexibility, and a generalized solution, a data platform could be described as bringing that to its logical extreme: virtually any use-case can be supported, including support for very diverse data types. It's easier to describe it with reference to real examples than in the abstract, though.
- Synergy with other projects is important here. The prime example is PostGIS, but they are only one example.

Example: Logical replication

- Essentially, core infrastructure for plugins that are interested in doing *something* with data as it is written.
- Plugins "decode" WAL to logical representation.
- Logical as opposed to physical representation can be constructed by extra write-ahead log (WAL) information.

The difference in WAL representation for replication

Physical	Logical
Tiny overhead	Modest overhead
Inflexible; cannot take out information about just one thing of interest.	Totally flexible; as much or as little as you want.
Refers to unstable identifiers like relfilenode. Ties binary representation to data (e.g. machine endianness, build time settings).	Logical records describe changes in a way that makes sense to anyone or anything.
Great for just simple replication and disaster recovery	Will support online in-place upgrades, prevent VACUUM from causing recovery conflicts, cross-database replication, writes on standbys, multi-master replication and more.

Example: *writeable* foreign-data wrappers

- PostgreSQL 9.3 feature. Used by postgres_fdw.
- Clever
  - Only sends *needed* data.
  - Foreign cost estimates influence local plan - statistics locally held.
  - Predicate (WHERE clause) push-down.

Example: Enhanced JSON type

- In the past, JSON seemed to me like a rough "specification" for an interchange format.
- I understand now, though: JSON is an interchange format that's very easy to work with from scripting languages, often manipulated as built-in data structures.
- Vagueness of spec around things like numeric precision turns out to be okay: that's the price of a truly flexible interchange format.
- If Postgres did better than the "lowest common denominator", what would be the point? You'd have to worry about the lowest common denominator breaking things, and that isn't very flexible!
- Not for every application, though can of course be used selectively within an application. That makes it compelling where flexibility is most important.

JSON for semi-structured data

```
{
   "name": "Brooks DuBuque",
   "age": 36,
   "siblings": 2,
   "numbers": [
      {
         "type":   "work",
         "number": "684.573.3783 x368"
      },
      {
         "type":   "home",
         "number": "625.112.6081"
      }
   ]
}
```

JSON manipulation

```
CREATE OR REPLACE FUNCTION
get_numeric(key text, data json)
RETURNS numeric $$
  return JSON.parse(data)[key];
$$ LANGUAGE plv8 IMMUTABLE STRICT;

select avg(get_numeric('age', data))
from people;
-[ RECORD 1 ]-------------
avg | 24.491387000000000
Time: 6641.060 ms
```

JSON futures

- Work underway for 9.4 - binary representation of JSON.
- GIN and GiST index support is being worked on too. Should be possible to query data in an almost arbitrary fashion efficiently.
- Also proposals around JSON CRUD functions.

Thanks!

ご清聴ありがとうございました

Questions?