2016-02-27

PostgreSQL Open Day, Rimini

Primi passi con BDR

Chris Mair

http://www.pgtraining.com

replicazione - l'inizio

Core team statement on replication in PostgreSQL

From: Tom Lane <tgl(at)sss(dot)pgh(dot)pa(dot)us>
To: pgsql-hackers(at)postgreSQL(dot)org
Subject: Core team statement on replication in PostgreSQL
Date: 2008-05-29 14:12:55

The Postgres core team met at PGCon to discuss a few issues, the largest
of which is the need for simple, built-in replication for PostgreSQL.
Historically the project policy has been to avoid putting replication
into core PostgreSQL, so as to leave room for development of competing
solutions, recognizing that there is no "one size fits all" replication
solution. However, it is becoming clear that this policy is hindering
acceptance of PostgreSQL to too great an extent, compared to the benefit
it offers to the add-on replication projects. Users who might consider
PostgreSQL are choosing other database systems because our existing
replication options are too complex to install and use for simple cases.
In practice, simple asynchronous single-master-multiple-slave
replication covers a respectable fraction of use cases, so we have
concluded that we should allow such a feature to be included in the core
project. We emphasize that this is not meant to prevent continued
development of add-on replication projects that cover more complex use
cases.

We believe that the most appropriate base technology for this is
probably real-time WAL log shipping, as was demoed by NTT OSS at PGCon.
We hope that such a feature can be completed for 8.4. Ideally this
would be coupled with the ability to execute read-only queries on the

replicazione - l'evoluzione

- 8.x - log shipping e PITR possono essere usati per creare dei warm standby
- 9.0 - streaming replication & hot standby
- 9.1 - synchronous replication
- 9.2 - cascading replication
- 9.4 - replication slots & **logical decoding**

replicazione - le alternative

- Slony (master-slave basato su trigger)
- Londiste (idem)
- Bucardo (master-slave e master-master, basato su triggers)
- Pgpool-II (sistema ibrido: pool / load balancer / replicatore a livello di statement / parallel query system)

replicazione - il futuro

- **sistemi che si agganciano al logical decoding** introdotto con la 9.4
- risolve il problema del "o replico tutto o niente" dei metodi basati su WAL (shipping o streaming)
- risolve il problema della performance e scalability dei metodi basati su trigger
- un prodotto (recente), portato avanti da 2ndQuadrant: **BDR** (Bi-Directional Replication)

BDR

- asynchronous multi-master logical replication
- molto veloce: modifiche vengono commitate sul nodo locale prima di essere replicate
- se più nodi toccano lo stesso dato contempo-raneamente conflitti sono inevitabili (Simon Riggs: "The Physics of Multi-Master")
- use case ideale: sistema distribuito tra siti con elevata latenza che vuole mettere a disposizione un master veloce presso tutti i siti

BDR

- BDR è un estensione PostgreSQL che richiede dei feature che 2ndQuadrant et.al. stanno commitando pezzo per pezzo a PostgreSQL:
  - Background workers (9.3)
  - Event Triggers (9.3 and 9.4)
  - Replication slots (9.4)
  - Logical decoding (9.4)
  - Enhanced DROP event triggers (9.4)
  - REPLICA IDENTITY (9.4)
  - Commit Timestamps (9.5)
  - Replication Origins (9.5)
  - DDL Deparse hooks (9.5)
    Fonte: http://blog.2ndquadrant.com/when-are-we-going-to-contribute-bdr-to-postgresql/
- purtroppo, tuttora, anche con 9.5, c'è bisogno di una versione patchata di PostgreSQL per BDR

BDR

- l'ultima versione disponibile ad oggi è BDR 0.9.3 per PostgreSQL (patchato) 9.4.

- docs & download:

  http://bdr-project.org/docs/0.9/index.html

- abbiamo creato un setup con tre nodi seguendo le istruzioni della versione 0.9.3

i demo server

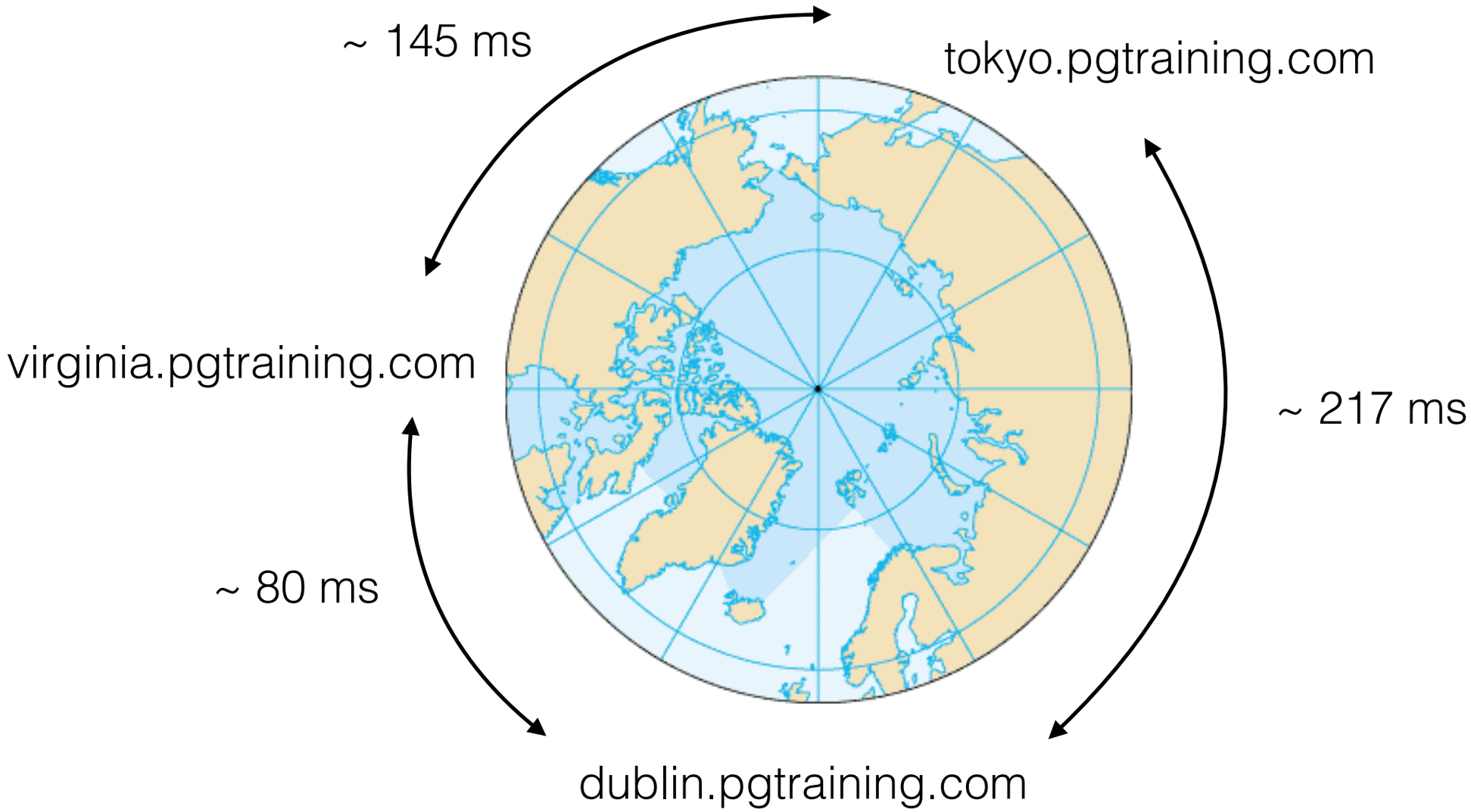
- ~~tre nodi che girano sul mio portatile~~

i demo server

- ~~tre nodi che girano su macchine virtuali su un server qui dietro~~

i demo server!

(su Amazon EC2 ;)



i demo server

- i tre nodi hanno mutualmente la porta 5432 aperta (firewall) e entry in pg_hba.conf per connettersi; usano SSL (notare i tempi):

setup dei tre nodi

```
# virginia

    CREATE EXTENSION btree_gist;
    CREATE EXTENSION bdr;

    SELECT bdr.bdr_group_create(
    local_node_name := 'node_virginia',
    node_external_dsn := 'host=virginia.pgtraining.com dbname=pgtraining');
    SELECT bdr.bdr_node_join_wait_for_ready();

# dublin (join via virginia)

    CREATE EXTENSION btree_gist;
    CREATE EXTENSION bdr;

    SELECT bdr.bdr_group_join(
    local_node_name := 'node_dublin',
    node_external_dsn := 'host=dublin.pgtraining.com dbname=pgtraining',
    join_using_dsn := 'host=virginia.pgtraining.com dbname=pgtraining');
    SELECT bdr.bdr_node_join_wait_for_ready();

# tokyo (join via virginia)

    CREATE EXTENSION btree_gist;
    CREATE EXTENSION bdr;

    SELECT bdr.bdr_group_join(
    local_node_name := 'node_tokyo',
    node_external_dsn := 'host=tokyo.pgtraining.com dbname=pgtraining',
    join_using_dsn := 'host=virginia.pgtraining.com dbname=pgtraining');
    SELECT bdr.bdr_node_join_wait_for_ready();
```

status

```
pgtraining=# \x
Expanded display is on.

pgtraining=# select node_status, node_name, node_local_dsn, node_init_from_dsn
             from bdr.bdr_nodes;

-[ RECORD 1 ]------+----------------------------------------
node_status        | r
node_name          | node_virginia
node_local_dsn     | host=virginia.pgtraining.com dbname=pgtraining
node_init_from_dsn |
-[ RECORD 2 ]------+----------------------------------------
node_status        | r
node_name          | node_dublin
node_local_dsn     | host=dublin.pgtraining.com dbname=pgtraining
node_init_from_dsn | host=virginia.pgtraining.com dbname=pgtraining
-[ RECORD 3 ]------+----------------------------------------
node_status        | r
node_name          | node_tokyo
node_local_dsn     | host=tokyo.pgtraining.com dbname=pgtraining
node_init_from_dsn | host=virginia.pgtraining.com dbname=pgtraining
```

prime prove (demo)

chris — ec2-user@ip-172-30-0-230:~ — ssh — 62×10

```
[ec2-user@virginia ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# select * from test;
 id | name
----+------
(0 rows)

pgtraining=#
```

chris — ec2-user@ip-10-0-1-228:~ — ssh — 62×7

```
[ec2-user@dublin ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# insert into test values (1, 'Chris');
INSERT 0 1
pgtraining=#
```

chris — ec2-user@ip-172-31-3-5:~ — ssh — 62×11

```
[ec2-user@tokyo ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# select * from test;
 id | name
----+-------
  1 | Chris
(1 row)

pgtraining=#
```

alcune cose non sono possibili

- "The Physics of Multi-Master" in azione...

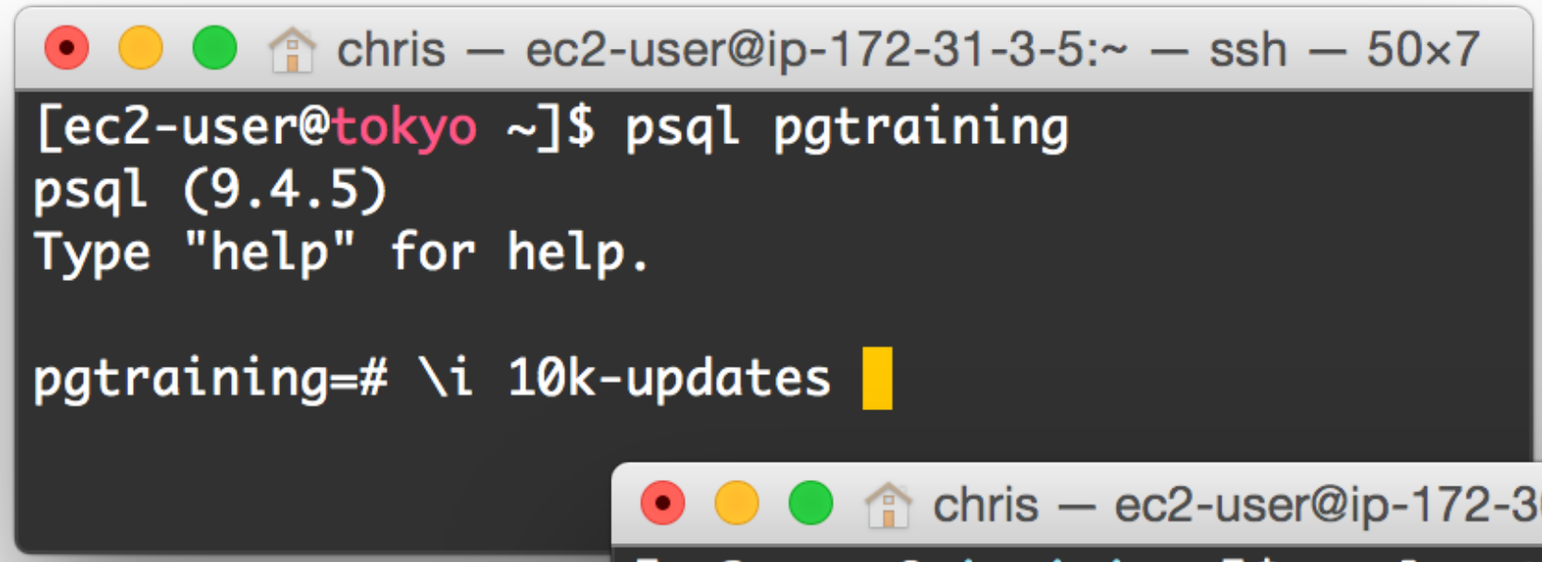
chris — ec2-user@ip-10-0-1-228:~ — ssh — 95×17

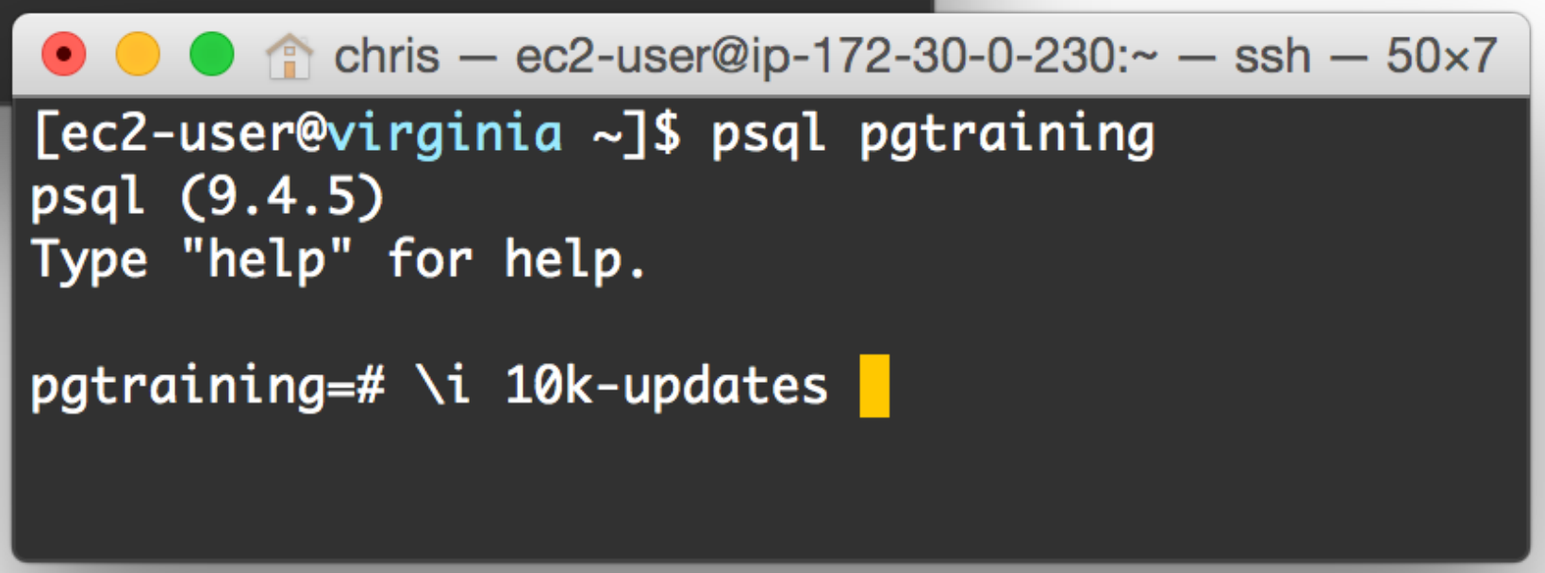
```
https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
[ec2-user@dublin ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# create table test2 (id int, name varchar);
CREATE TABLE
pgtraining=# insert into test2 values (1, 'Chris');
INSERT 0 1
pgtraining=# update test2 set name = 'Postgres' where id = 1;
ERROR:  Cannot run UPDATE or DELETE on table test2 because it does not have a PRIMARY KEY.
HINT:  Add a PRIMARY KEY to the table
pgtraining=# update test2 set name = 'Postgres' where name = 'Chris';;
ERROR:  Cannot run UPDATE or DELETE on table test2 because it does not have a PRIMARY KEY.
HINT:  Add a PRIMARY KEY to the table
pgtraining=#
```

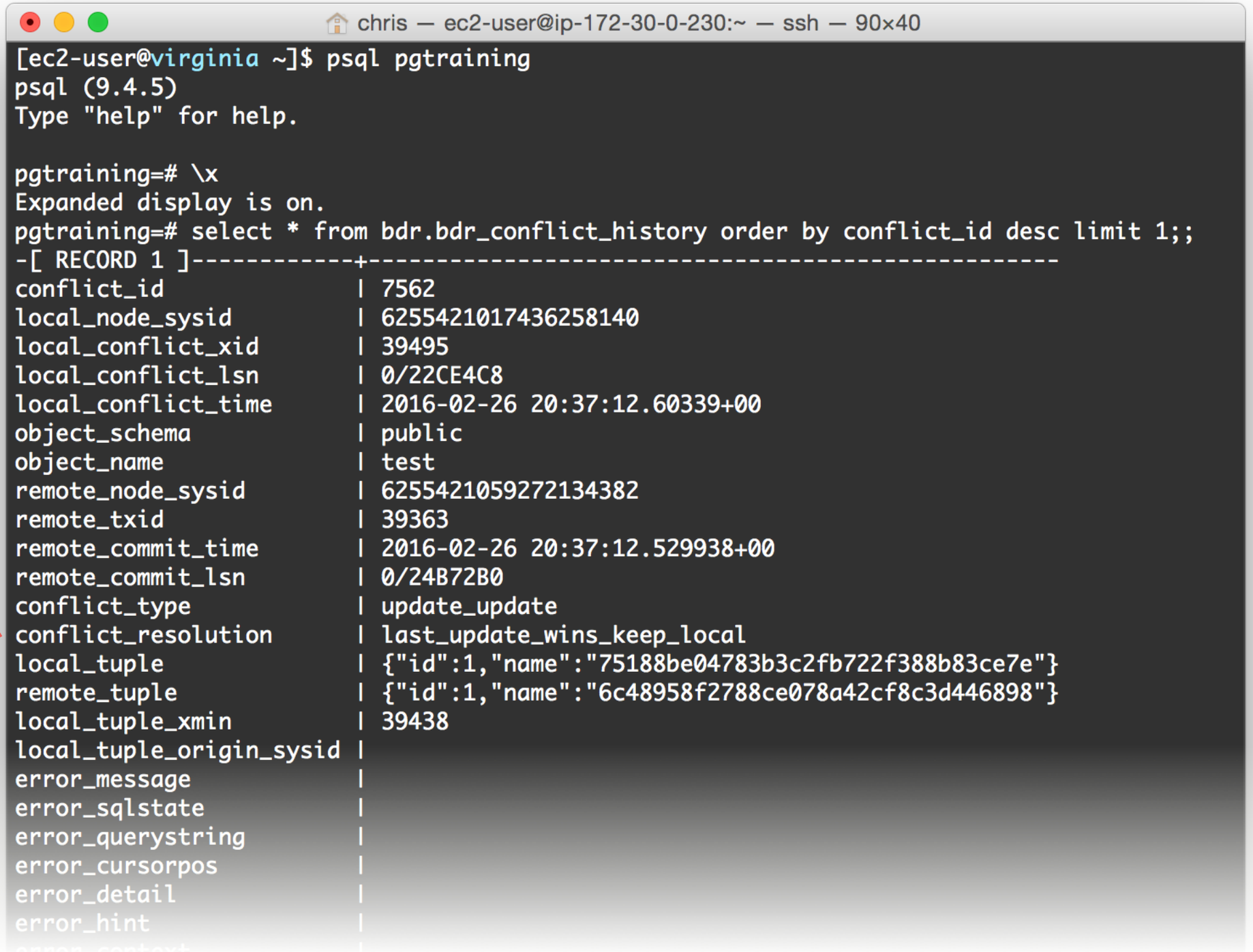
conflitti!

- 10k update sullo stesso record (= worst case)





risoluzione dei conflitti



ancora conflitti

- http://bdr-project.org/docs/next/conflicts-types.html
- UPDATE/UPDATE:
  - non aggiornare la stessa tupla nello stesso istante su nodi diversi ;)
  - custom conflict handlers
  - distributed locking a livello applicativo
  - se questo succede spesso, forse non volevi il multi-master...
- e INSERT/INSERT? ... sequenze globali!

sequenze globali

- `CREATE SEQUENCE test_seq USING bdr;`
- oppure setting: `default_sequenceam = true`
- promettono di dare `nextval()` unici su tutti i nodi riservando intervalli distinti per nodo
- questi intervalli vengono assegnati con un algoritmo distribuito di voting (maggioranza N/2+1)
- non sono obbligatorie: si può anche scegliere di usare qualche tipo di UUID

sequenze globali

chris — ec2-user@ip-10-0-1-228:~ — ssh — 53×17

```
[ec2-user@dublin ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# create table seqtest(id serial);
CREATE TABLE
pgtraining=# insert into seqtest values (default);
INSERT 0 1
pgtraining=# select * from seqtest;
  id
------
 5001
    2
(2 rows)

pgtraining=#
```

chris — ec2-user@ip-172-30-0-230:~ — ssh — 53×7

```
[ec2-user@virginia ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# insert into seqtest values (default);
INSERT 0 1
pgtraining=#
```

limitazioni delle s. globali

- non si può utilizzare **`setval()`**!
- le sequenze globali sono sempre incrementate di 1
- a differenza delle sequenze non-BDR, nextval() può fallire se un nodo "consuma" l'intervallo che ha a disposizione prima che la rete di nodi può votare per assegnargli un nuovo intervallo:

```
ERROR:  could not find free sequence value for global
sequence public.seqtest_id_seq

HINT:  The sequence is refilling from remote nodes. Try again
soon. Check that all nodes are up if the condition persists.
```

pg_dump?

- non avendo setval() non si riesce a semplicemente importare un dump non-BDR...

- un trucco:

```
set default_sequenceam = '';
\i bla.dump
alter sequence bla_seq using bdr;
call nextval('bla_seq') in un ciclo per portare la
sequenze oltre il massimo usato in bla
```

- per pg_dump + BDR: bit.ly/1n3mDmm

DDL

- fino a qui, ogni operazione (che non fallisce ;) può essere eseguita direttamente dal nodo che la riceve

- questo **NON** vale per il DDL (CREATE, DROP, ALTER)
  - c'è bisogno di una votazione tra i nodi per cui **TUTTI** devono essere d'accordo (quasi come per i d.d.l. ;)

- se anche un solo nodo non risponde, l'operazione non procede

- questo, perché è difficile/impossibile accettare (e risolvere) conflitti sui DDL

DDL

tokyo non risponde... e dublin aspetta

chris — ec2-user@ip-172-31-3-5:~ — ssh — 55×7

```
[ec2-user@tokyo ~]$ pg_ctl -l log stop
waiting for server to shut down.... done
server stopped
[ec2-user@tokyo ~]$
```

chris — ec2-user@ip-10-0-1-228:~ — ssh — 55×7

```
[ec2-user@dublin ~]$ psql pgtraining
psql (9.4.5)
Type "help" for help.

pgtraining=# create table test3 (id serial);
```

forse era godzilla?

- oltre il caso DDL, è importare monitorare il sistema: se un nodo va giù, gli altri nodi accumulano i WAL, potenzialmente fino a riempire il disco...
- si possono togliere nodi con:

```
SELECT bdr.bdr_part_by_node_names(ARRAY['node_tokyo']);
```

- occhio che a fare il join dello stesso nodo dopo ho incontrato problemi (vedi anche mailing list) - forse un problema di documentazione / beta-ness...

'k thx bye ;)