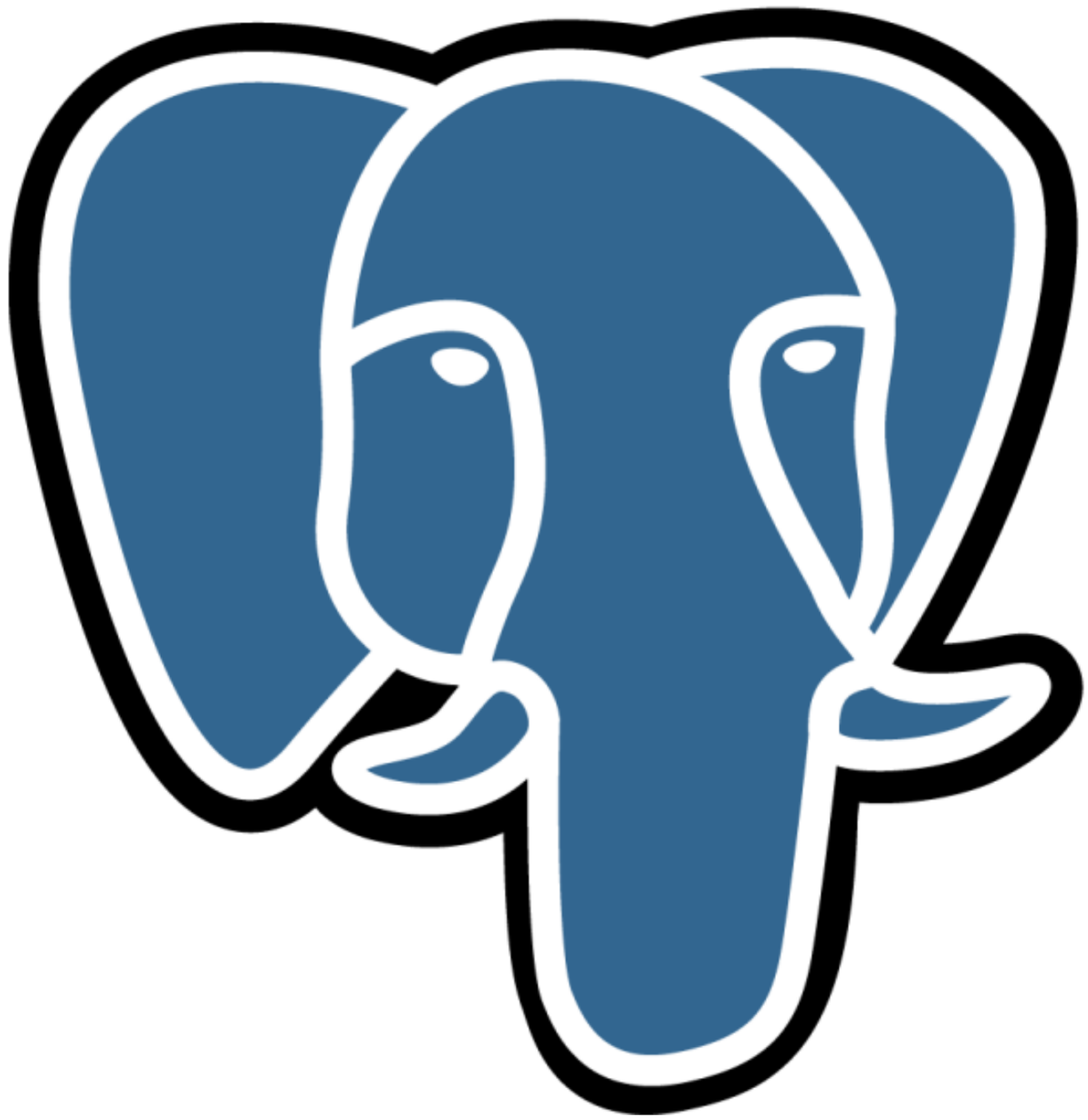
LUGBZ BabsCamp2013

PostgreSQL

Chris Mair <chris@1006.org>

http://www.1006.org
http://www.pgtraining.com

About PostgreSQL



- sistema per la gestione di database relazionali (RDBMS)
- Open Source & Free Software (licenza BSD)
- Postgres dal 1986, Uni Berkeley, Stonebraker
- PostgreSQL dal 1996, progetto indipendente
- Postgres è PostgreSQL sono sinonimi, entrambi i nomi sono corretti

- supporta molte parti di ANSI SQL:2008 - è di **gran lunga** il più avanzato RDBMS libero

- dalla 8.0 (2005) ha avuto un incredibile mole di nuovi feature (versione attuale 9.2)

- attualmente: circa 50 main contributor full time sponsorizzati da diverse aziende guidate da un core team di 6 persone

- **non** c'è una **singola** azienda dietro Postgres, ma più aziende (che offrono supporto per Postgres e/o sono a loro volta utilizzatori di Postgres)

Comunity

Photo (C) Steve Evans - License: Creative Commons Attribution 2.0 Generic

- Sito internazionale

  → http://www.postgresql.org

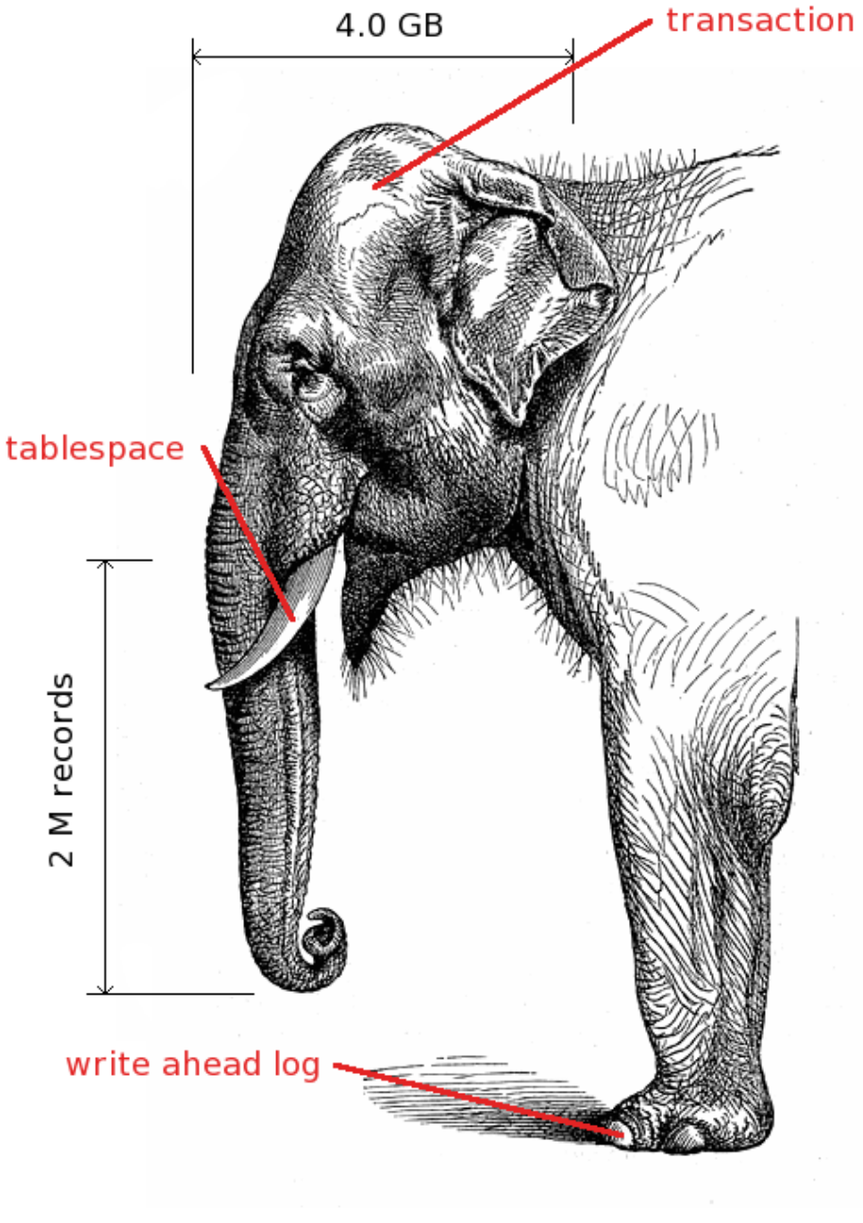
- ITPUG (advocacy, conferenze in Italia)

  → http://www.itpug.org

- psql.it (principale mailing list italiana)

  → http://www.psql.it

Features



- PostgreSQL è un RDBMS di classe “enterprise” - sul sito c’è uno storico elenco (che per pigrizia non copio qui ;)

- da alcuni anni supporta replicazione out-of-the-box, tacendo una storica critica

- da parecchio tempo supporta XML o fulltext search e recentemente sta integrando elementi stile “NoSQL”, come supporto nativo per JSON, la possibilità di avere tabelle *unlogged*, o PL/V8 - serverside JavaScript con V8 (ancora beta)

Let's run it

Photo (C) Richard Lair - used with permission

Esperimento 1

“sì, ma l’SQL è *webscale*?”

lugbzbook

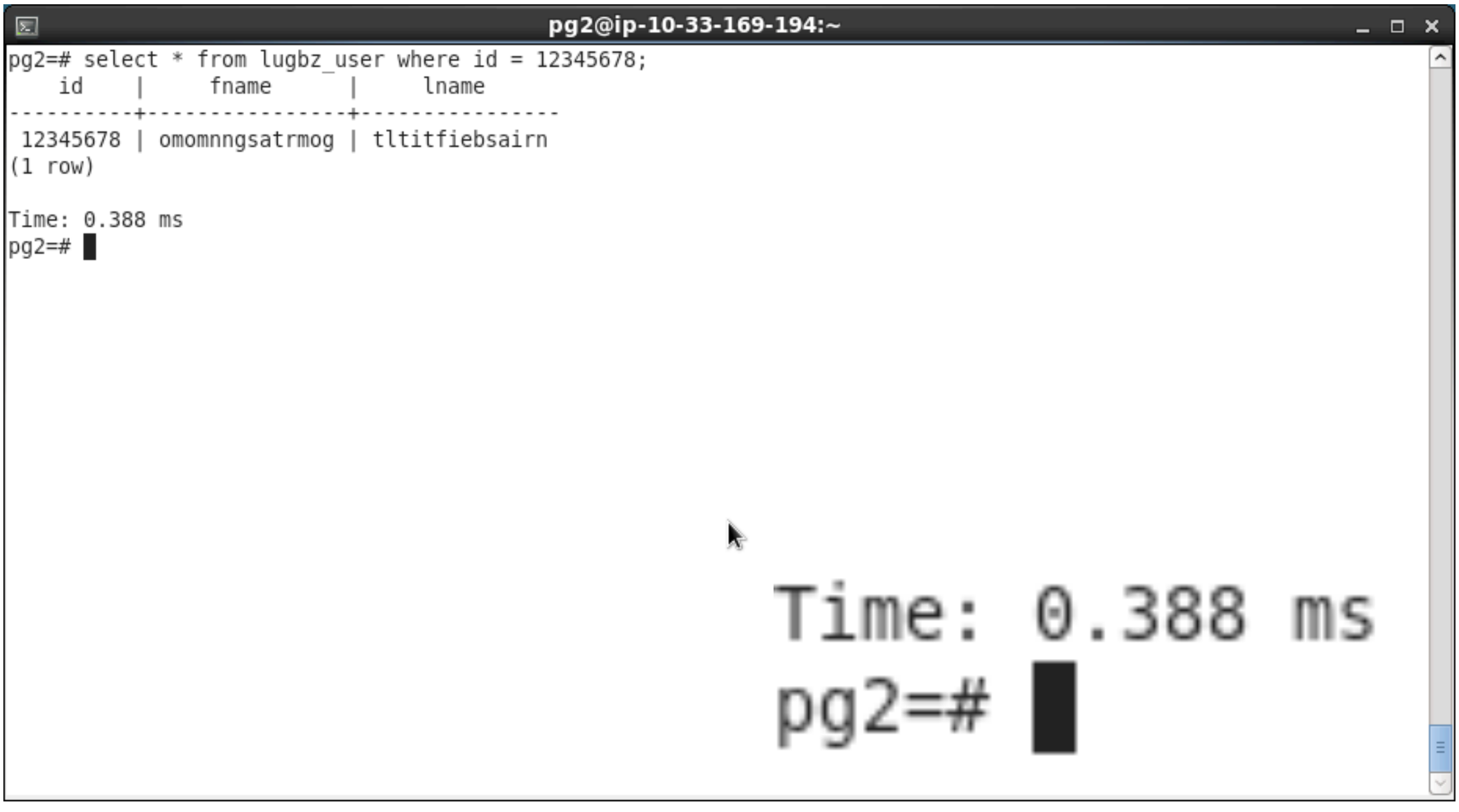
- tabella lugbz_user con 100 000 000 record ("utenti")

- tabella lugbz_link con 1 050 000 360 relazioni tra utenti (ciascuno ha 1 - 20 "amici")

- di ogni utente salviamo, oltre l'id a 64 bit solo nome e cognome (è un esperimento con elevato numero di record, non dati)

DB object	Record	Dimensione	Dimensione Indici
lugbz_user	1.00E+08	6.49 GB	2.09 GB
lugbz_link	~1.05E+09	43.3 GB	21.96 GB 21.96 GB

- ok, troviamo l'utente 12345678



- oooook, ma... troviamo i suoi amici

pg2@ip-10-33-169-194:~

```
pg2=# select * from lugbz_link where user_id = 12345678;
 user_id  | friend_id
----------+-----------
 12345678 |  47811272
 12345678 |  19382652
 12345678 |  79186839
 12345678 |  26233642
 12345678 |  53578934
 12345678 |  47667128
 12345678 |  51926936
 12345678 |  85401995
 12345678 |  27549362
 12345678 |  68538234
 12345678 |  73952680
 12345678 |  36261743
 12345678 |   4122419
(13 rows)

Time: 0.307 ms
pg2=#
```

Time: 0.307 ms
pg2=#

- ma... un join tra 100 000 000 e 1 050 000 360 record?!

pg2@ip-10-33-169-194:~

```
pg2=# select u.* from lugbz_user u inner join lugbz_link l on (u.id = l.friend_id) where l.user_id = 12345678;
    id    |      fname       |      lname
----------+------------------+-----------------
 47811272 | gtmggemepampgs   | fsnitbnttaiifss
 19382652 | segsrgsrpagrtr   | flsnfsrtbraltnnf
 79186839 | setgnottraegtr   | snsflnbtsnaetbl
 26233642 | peeesnoosgarser  | lbtfbfarsfnen
 53578934 | ssmtmgeoomansomg | bnblrblralrbf
 47667128 | ogstmsgnnamog    | iffbirrinbaslri
 51926936 | ggrpoonsnsansgrr | nlbrflellbasif
 85401995 | sgoetpnnapgmsg   | rrrbntilisatsei
 27549362 | snmsonnnogantnpr | fsilnrsrnanrbb
 68538234 | msnteesnetaoert  | esbfnftlarbsb
 73952680 | rpmstrgsppanrrmt | nltninrtsnaseirb
 36261743 | toengggtsammt    | tiseitrfbaeseil
  4122419 | oesrsgonpratpomg | iliinlfbrasei
(13 rows)

Time: 0.557 ms
pg2=#
```

Time: 0.557 ms
pg2=#

- ne SQL ne join sono lenti (lento sono solo i benchmark sui prodotti degli altri :)

- con un indice, passare da 1 000 a 1 000 000 di record vuol dire fare ~ 20 passi anzichè ~ 10 per trovare un record (il logaritmo è il tuo amico)

- questa macchina ha 15GB di RAM, quindi buona parte degli indici usati nella join (2.09 + 21.96GB) possono essere in cache e effettivamente trovare i dati in sub-millisecondi anche se i record fossero più larghi

Esperimento 2

“sì, ma il load deve essere distribuito su più macchine”

- Postgres scala benissimo in verticale (vedere p.e. questa presentazione per benchmark correnti), ciò nonostante, soluzioni orizzontali hanno il loro uso

- guardiamo un'altra istanza di Postgres (sulla stessa macchina dell'esperimento 1) che in questo momento è replicato **sul mio portatile** nella modalità *hot standby* con *streaming replication*

- in pratica è un setup master-slave, dove il slave (o i slave, il numero non è limitato) seguono con pocchissimo lag il master e possono essere normalmente usati per query read-only

pg1@ip-10-33-169-194:~

```
Every 1.0s: psql -c "select * from msg order by ts desc limit 10"      Fri Feb 22 22:48:05 2013

  id  |             ts              | author |                   txt
------+-----------------------------+--------+-----------------------------------------
 1233 | 2013-02-22 22:48:02.50092   | daemon | The time is Fri Feb 22 22:48:02 UTC 2013
 1232 | 2013-02-22 22:47:52.484081 | daemon | The time is Fri Feb 22 22:47:52 UTC 2013
 1231 | 2013-02-22 22:47:42.467345 | daemon | The time is Fri Feb 22 22:47:42 UTC 2013
 1230 | 2013-02-22 22:47:32.450531 | daemon | The time is Fri Feb 22 22:47:32 UTC 2013
 1229 | 2013-02-22 22:47:22.433832 | daemon | The time is Fri Feb 22 22:47:22 UTC 2013
 1228 | 2013-02-22 22:47:12.416906 | daemon | The time is Fri Feb 22 22:47:12 UTC 2013
 1227 | 2013-02-22 22:47:02.399898 | daemon | The time is Fri Feb 22 22:47:02 UTC 2013
 1226 | 2013-02-22 22:46:52.38156  | daemon | The time is Fri Feb 22 22:46:52 UTC 2013
 1225 | 2013-02-22 22:46:42.364963 | daemon | The time is Fri Feb 22 22:46:42 UTC 2013
 1224 | 2013-02-22 22:46:32.347978 | daemon | The time is Fri Feb 22 22:46:32 UTC 2013
(10 rows)
```

chris@mercury:~

```
Every 2.0s: psql -c "select * from msg order by ts desc limit 10"      Fri Feb 22 23:48:04 2013

  id  |             ts              | author |                   txt
------+-----------------------------+--------+-----------------------------------------
 1233 | 2013-02-22 22:48:02.50092   | daemon | The time is Fri Feb 22 22:48:02 UTC 2013
 1232 | 2013-02-22 22:47:52.484081 | daemon | The time is Fri Feb 22 22:47:52 UTC 2013
 1231 | 2013-02-22 22:47:42.467345 | daemon | The time is Fri Feb 22 22:47:42 UTC 2013
 1230 | 2013-02-22 22:47:32.450531 | daemon | The time is Fri Feb 22 22:47:32 UTC 2013
 1229 | 2013-02-22 22:47:22.433832 | daemon | The time is Fri Feb 22 22:47:22 UTC 2013
 1228 | 2013-02-22 22:47:12.416906 | daemon | The time is Fri Feb 22 22:47:12 UTC 2013
 1227 | 2013-02-22 22:47:02.399898 | daemon | The time is Fri Feb 22 22:47:02 UTC 2013
 1226 | 2013-02-22 22:46:52.38156  | daemon | The time is Fri Feb 22 22:46:52 UTC 2013
 1225 | 2013-02-22 22:46:42.364963 | daemon | The time is Fri Feb 22 22:46:42 UTC 2013
 1224 | 2013-02-22 22:46:32.347978 | daemon | The time is Fri Feb 22 22:46:32 UTC 2013
(10 rows)
```

- questo meccanismo è sorprendentemente facile da mettere su (info qui con qualche dettaglio qui)

- ed è anche sorprendentemente robusto (sul server c'è un insert ogni 10 secondi che il mio portatile replica da ieri sera anche se non era sempre online!) - si può settare l'ammontare di tx log che il server tiene salvato nel caso il slave non ce la fa a replicarlo in tempo reale

- ci sono varianti: p.e. si può rendere sincrona la *streaming replication* o limitarsi a un *warm standby*

Domande?

The End

Chris Mair <chris@1006.org>

http://www.1006.org
http://www.pgtraining.com