Using PostgreSQL in Tantan - From 0 to 350bn rows in 2 years

Victor Blomqvist
vb@viblo.se

Tantan (探探)

December 2, PGConf Asia 2016 in Tokyo

探探

Sweden - Tantan - Tokyo

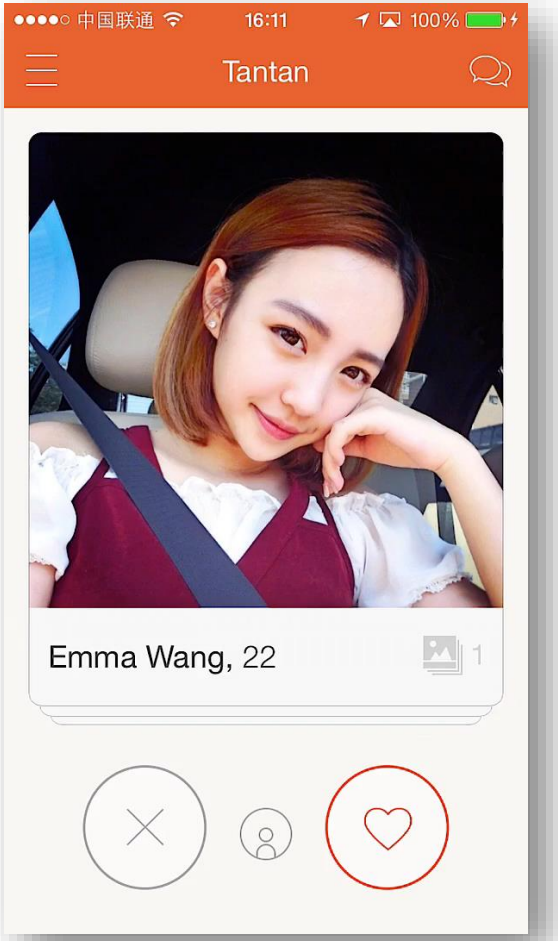
10 Million

11 Million MAU

14 Million

探探

What is Tantan?



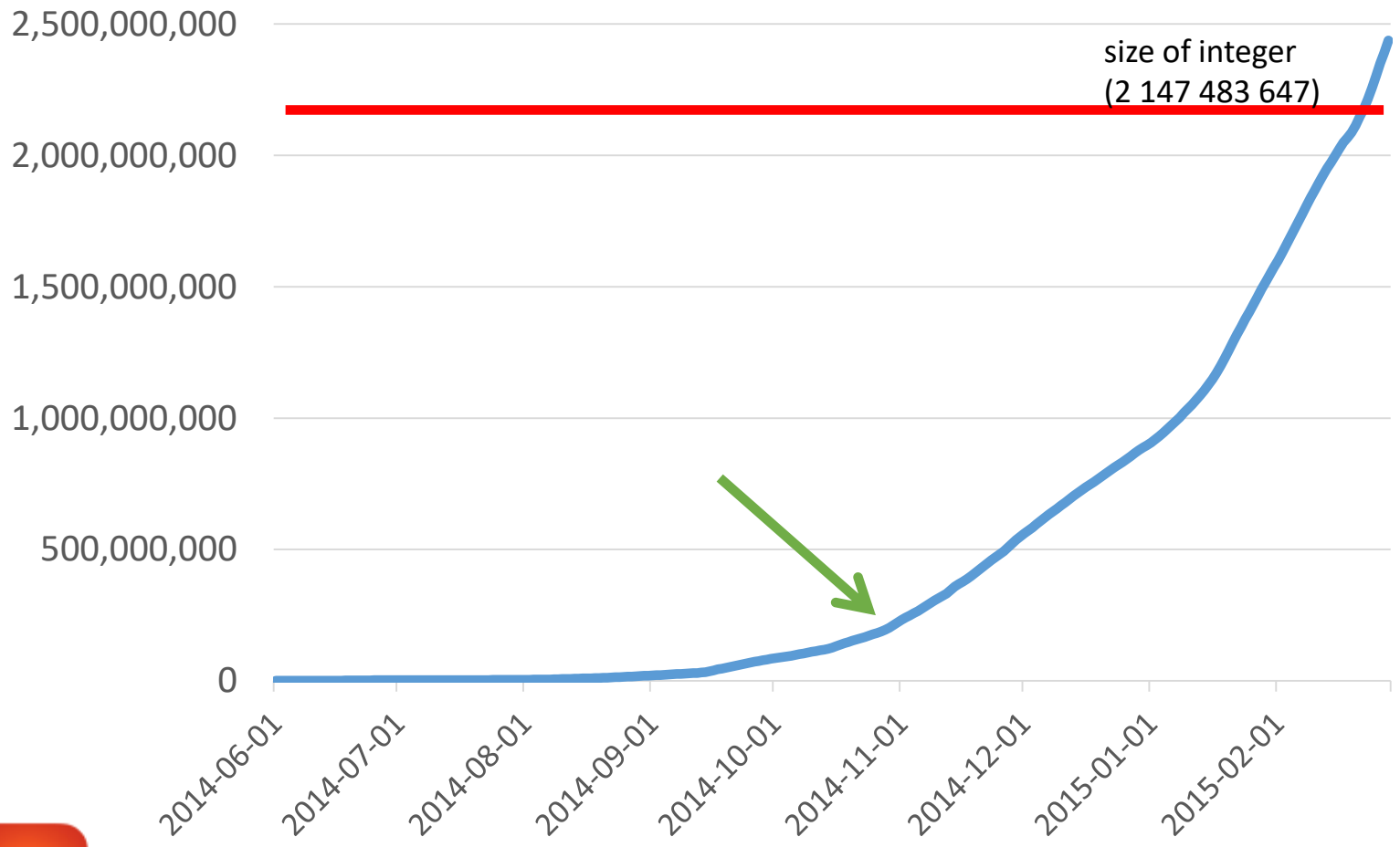
探探

"We have already used 17% of the space in our relationship table, and its growing quickly"

- Email sent on October 31 2014

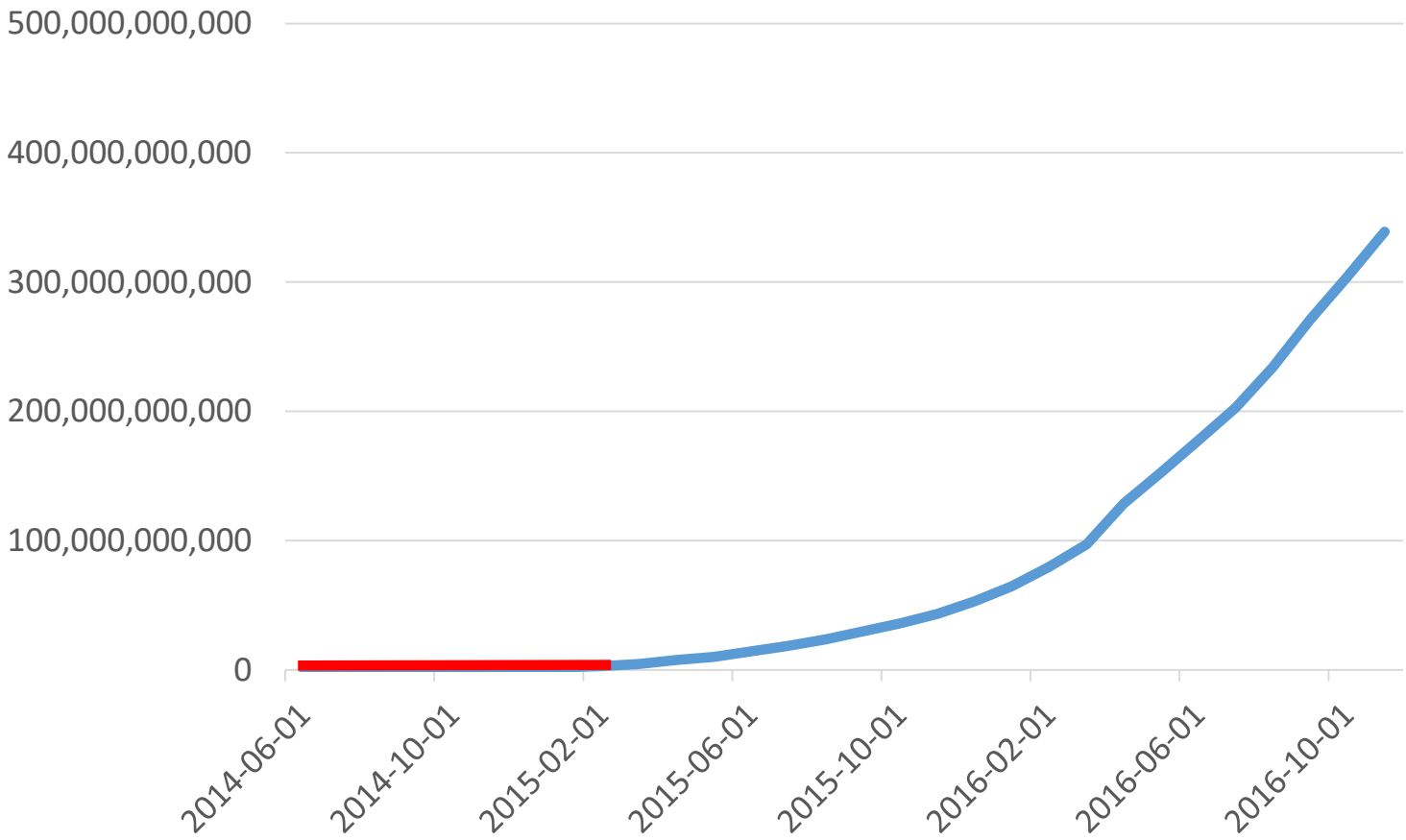
探探

Total Relationships Stored



探探

Total Relationships Stored – Today



探探

1. Intro
2. Architecture
3. Scaling
4. Problems & Challenges
5. End

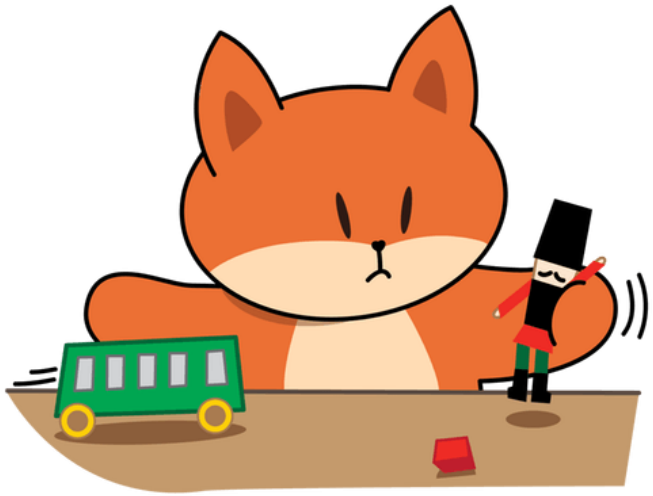
探探

1. Intro
2. **Architecture**
3. Scaling
4. Problems & Challenges
5. End

探探

Architecture – Philosophy

- Keep it simple!
  - Minimize the amount of different subsystems
  - Don’t introduce any new dependencies unless they give huge benefit.



Architecture – First Design

- Q: What can do geo queries, is a SQL database, is well known, and most importantly, is something we have tried?

- A: PostgresSQL + PostGIS ofcourse!

探探

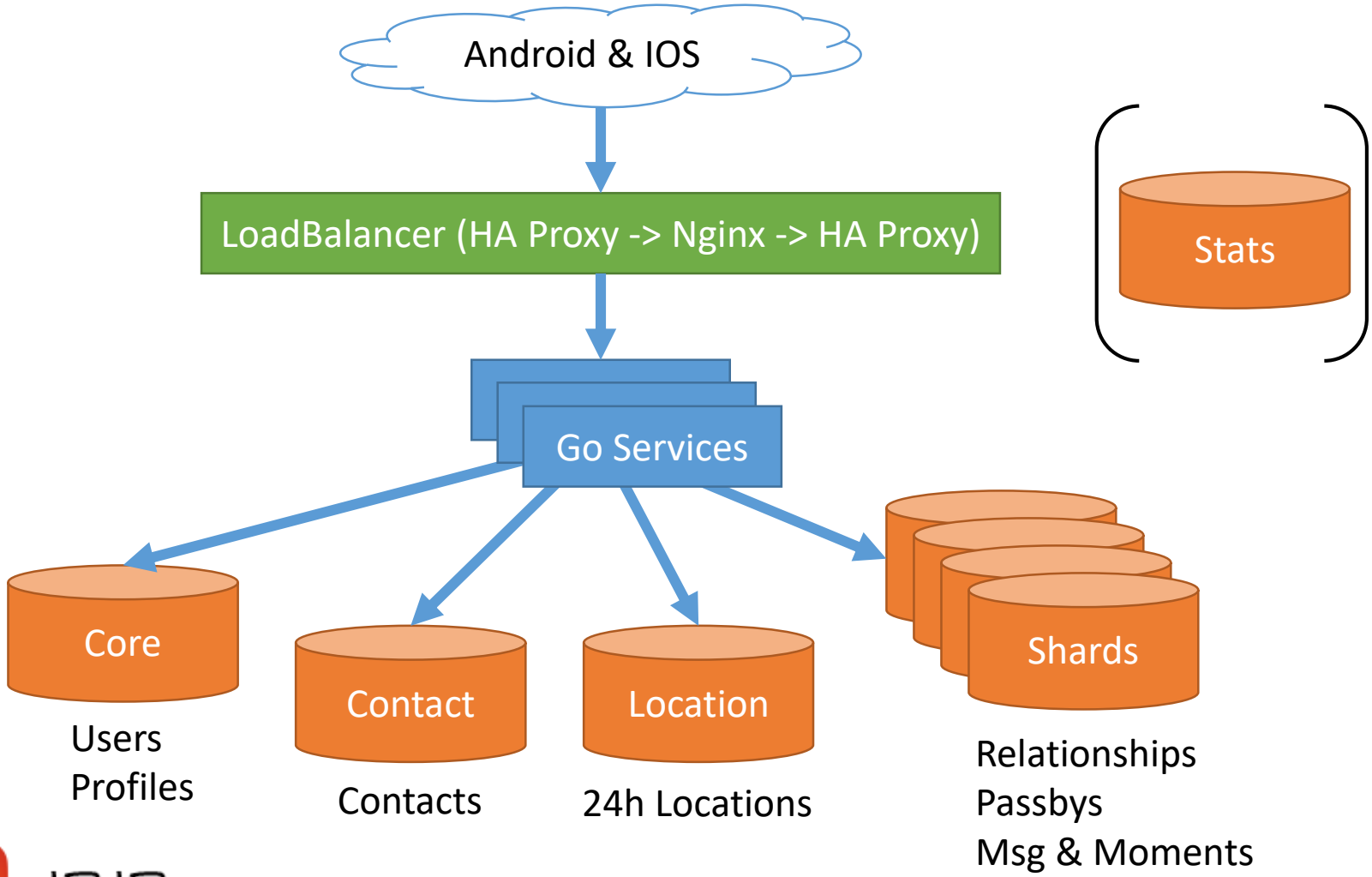
Architecture – In Practice

- All backend business logic is written in Go
- Service based architecture, but only in parts that make sense.
- No caching layer*
- One database: PostgreSQL

* (small in-memory exceptions in the go services for very static data)

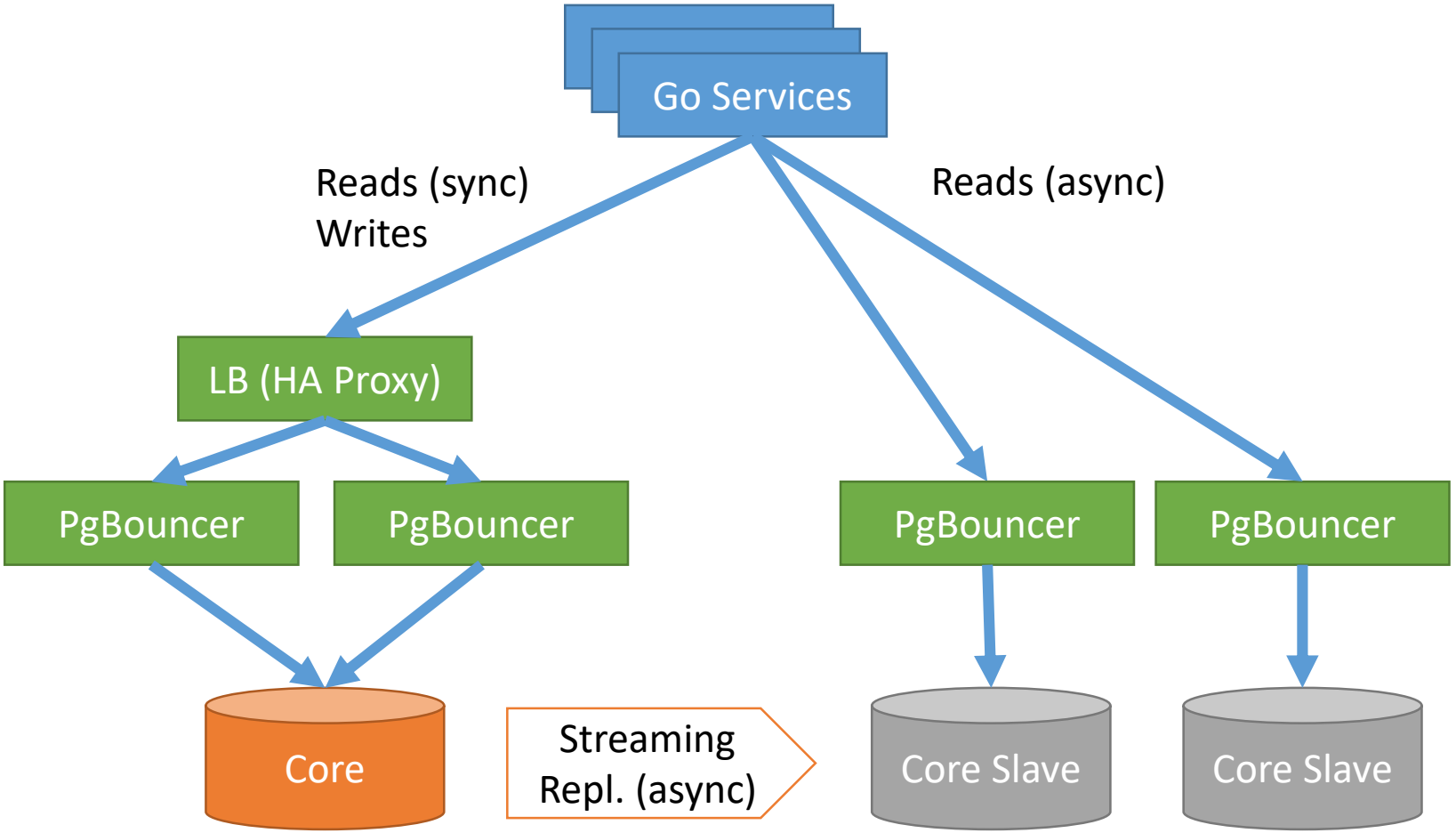
探探

Architecture



探探

Architecture



探探

Functions for all Database Access

- pg_stat_user_functions is much easier than pg_stat_statements
- Easy to identify the query and what parameters it can take
- Easy to “disable” a query by commenting out its body
- Possible to do more advanced / complicated things than what is possible in a single sql statement
- All functions can be kept in one place for version control and easy search/manage/read

探探

Result?

- PostgreSQL is a core component of Tantan, and it has enabled both increase of features and scale

探探

1. Intro
2. Architecture
3. **Scaling**
4. Problems & Challenges
5. End

探探

Current Scale and Load

- Size
  - **1.3TB / 11bn rows** Biggest table (non sharded)
  - **22TB / 350bn rows** Biggest table (sharded)

- Tuple writes per second (peak)
  - **15k tuple writes / sec**. Contact DB
  - **270k tuple writes / sec**. Aggregated over all databases

- Transactions per second (peak)
  - **31k TPS** Core DB Master (read/writes)
  - **49k TPS** Core DB Slave (reads)
  - **1.3M TPS** Aggregated over all databases

探探

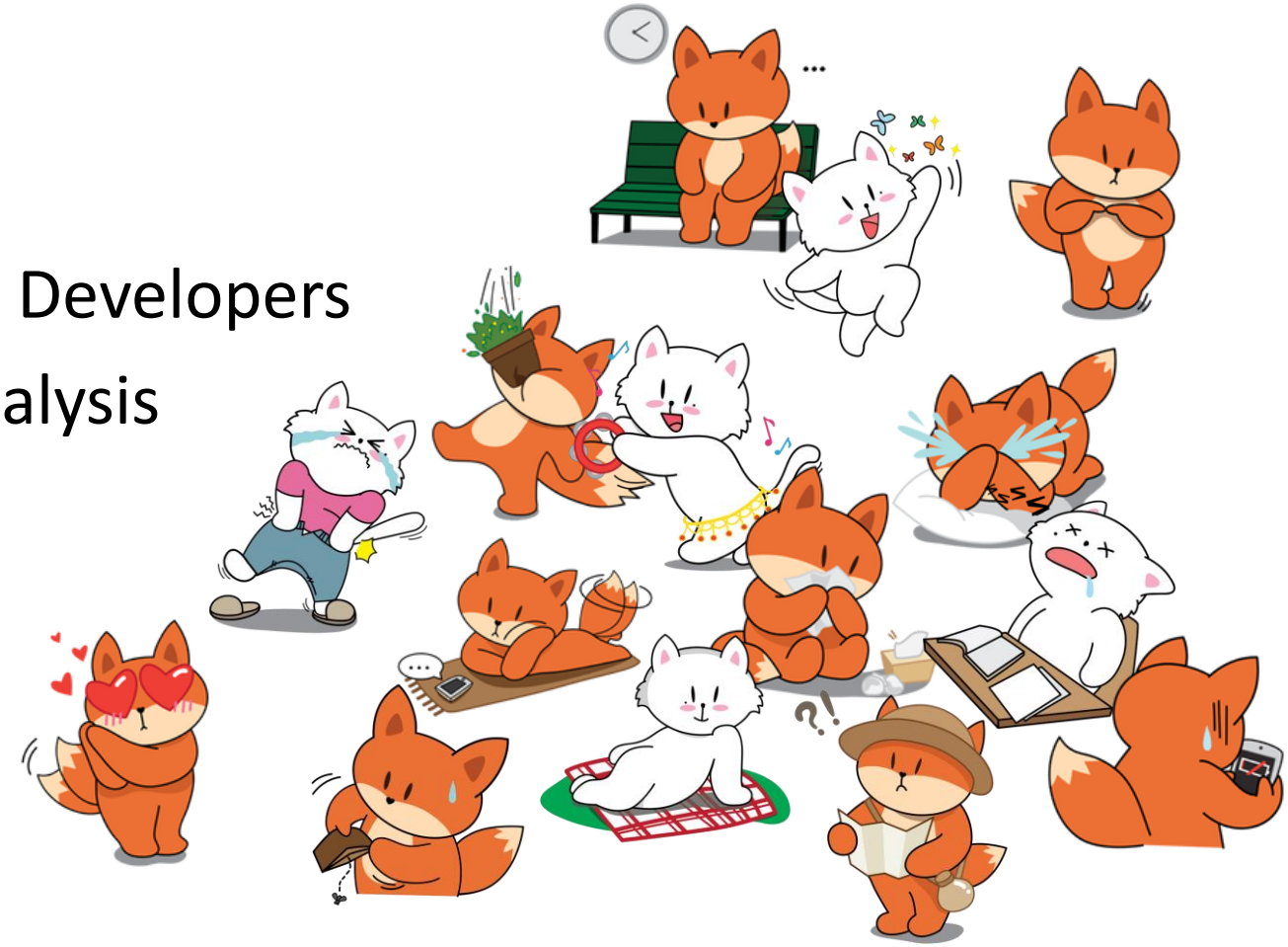
Team day 1

- 4 (5?) (DBA combined Backend)
- 2 half time operations

探探

Team Today

- 3 DBA
- 3 Devops
- 8 Backend Developers
- 2 Stats/Analysis



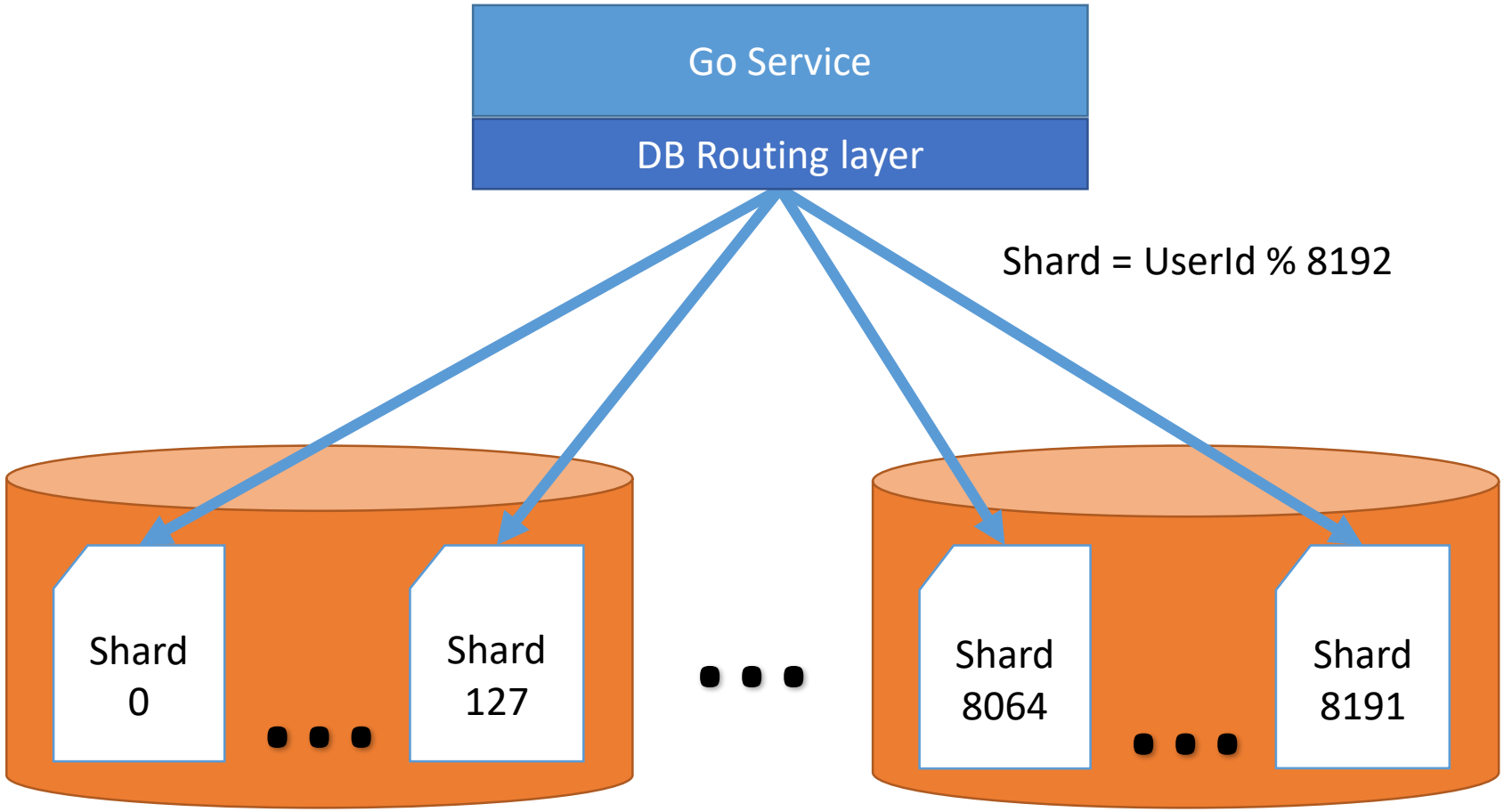
探探

Scale out – Sharding 1

- Homemade version similar to the “Instagram” way
- 8192 logical shards spread out on 64 physical servers
- Go code contains a db routing layer that knows which instance to call

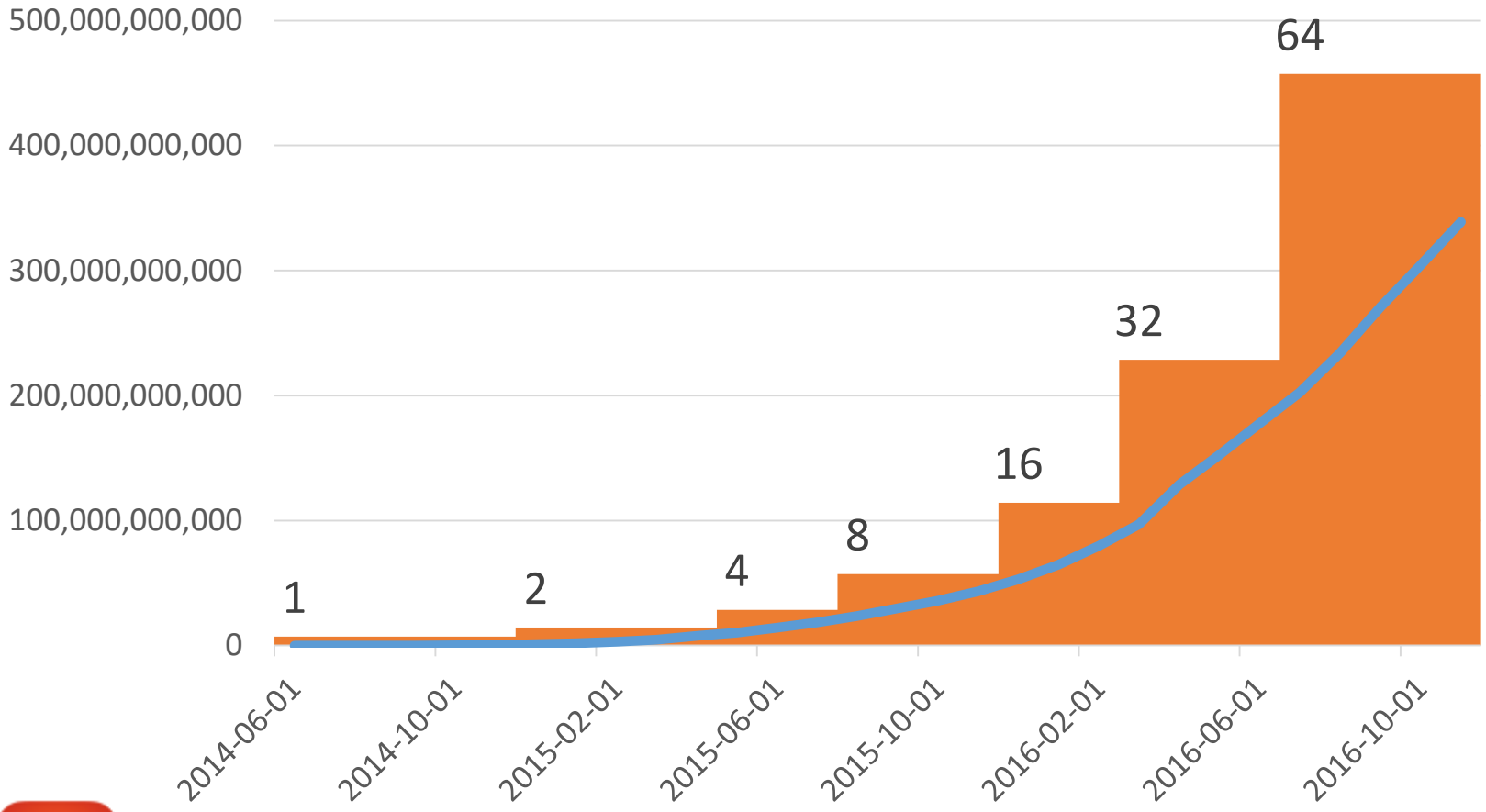
探探

Scale out – Sharding 2



探探

Total Relationships & Number of Shards



探探

Hardware Day 1

- One database for everything:
  Dell R410
  8 cores (2x Intel E5606 2.13GHz)
  128 GB RAM
  SSD
  1 Gbps network

探探

Hardware Today

- Most common type:
  Dell R730
  24 cores/48 threads (2x Intel E5-2680)
  380 GB RAM
  PCI-e SSD 3.2TB
  2x1000Mbps

- ContactDB:
  1 TB RAM
  7 TB SSD

探探

Result?

- Turns out you can scale to 50TB+ data and 1M+ TPS with a classic open source SQL database and some glue code in your app layer

探探

1. Intro
2. Architecture
3. Scaling
4. **Problems & Challenges**
5. End

Problems & Challenges – Team

- Finding PostgreSQL DBAs
- Finding developers with (Postgre)SQL experience

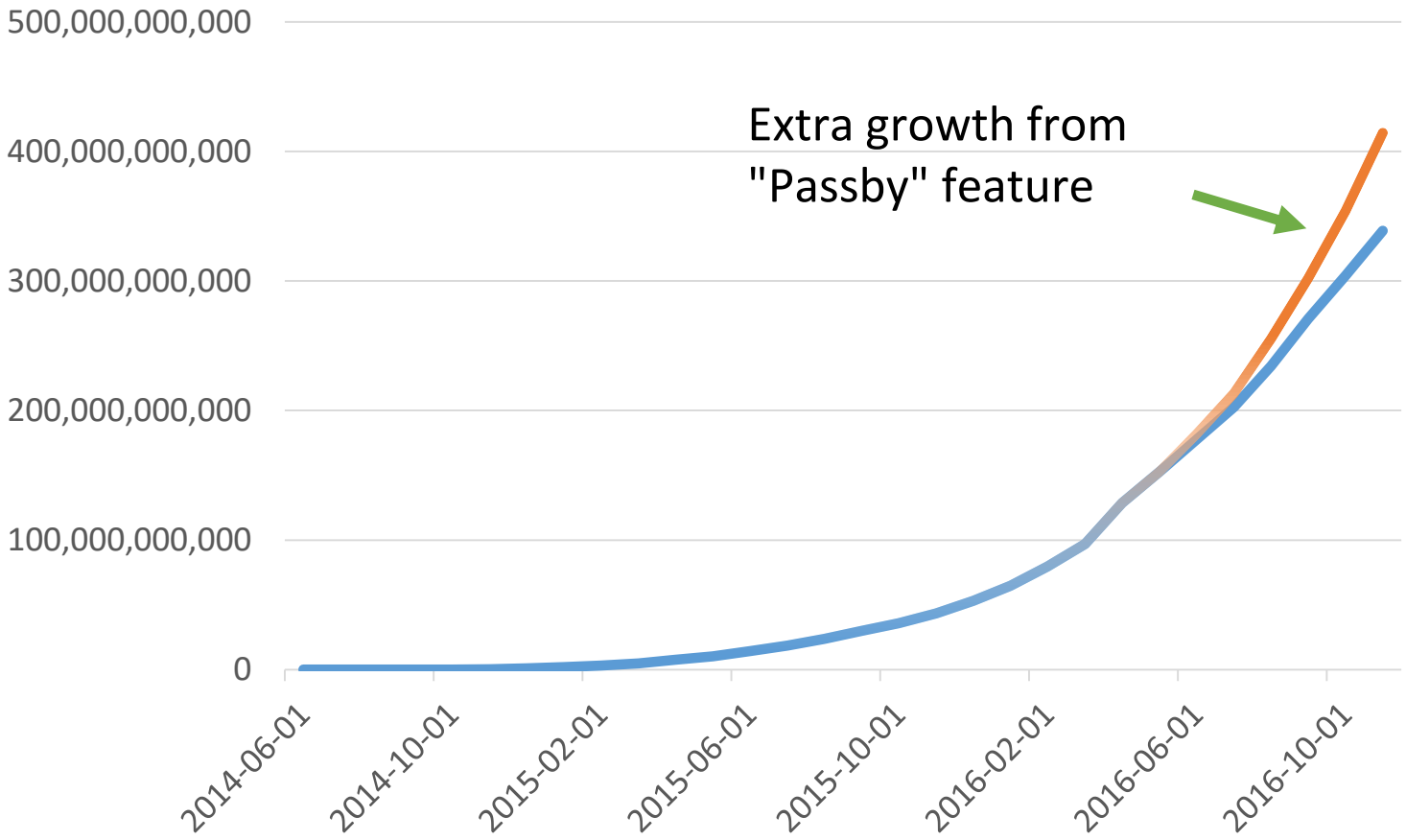
探探

Problems & Challenges – Big DATA

- Fighting the query planner. Usually gets worse as the data gets bigger and poor plans can’t be acceptable anymore
- It can often be difficult to know how a query performs before its tried in production
- Small product changes can lead to huge unforeseen problems
- As the number of servers grow, keep track of your configs!

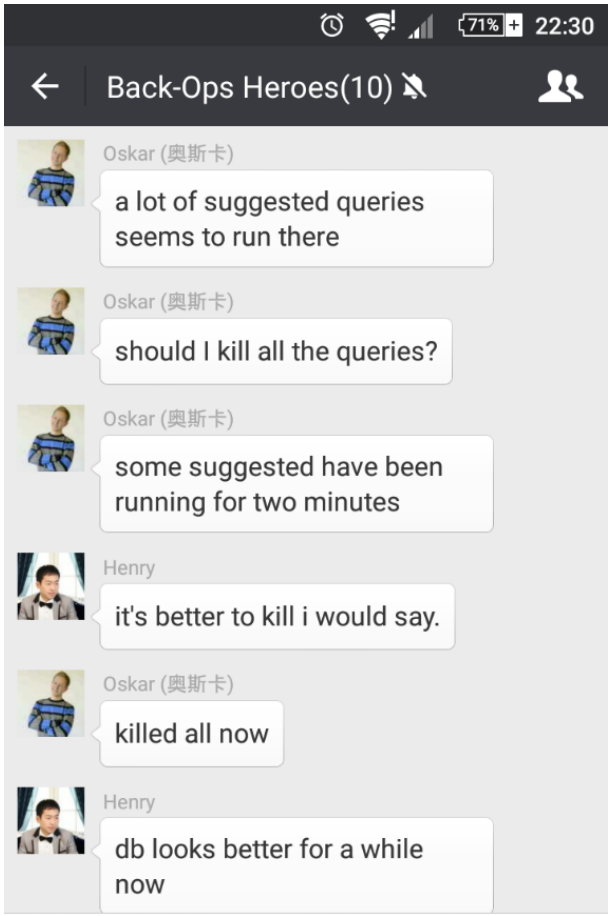
探探

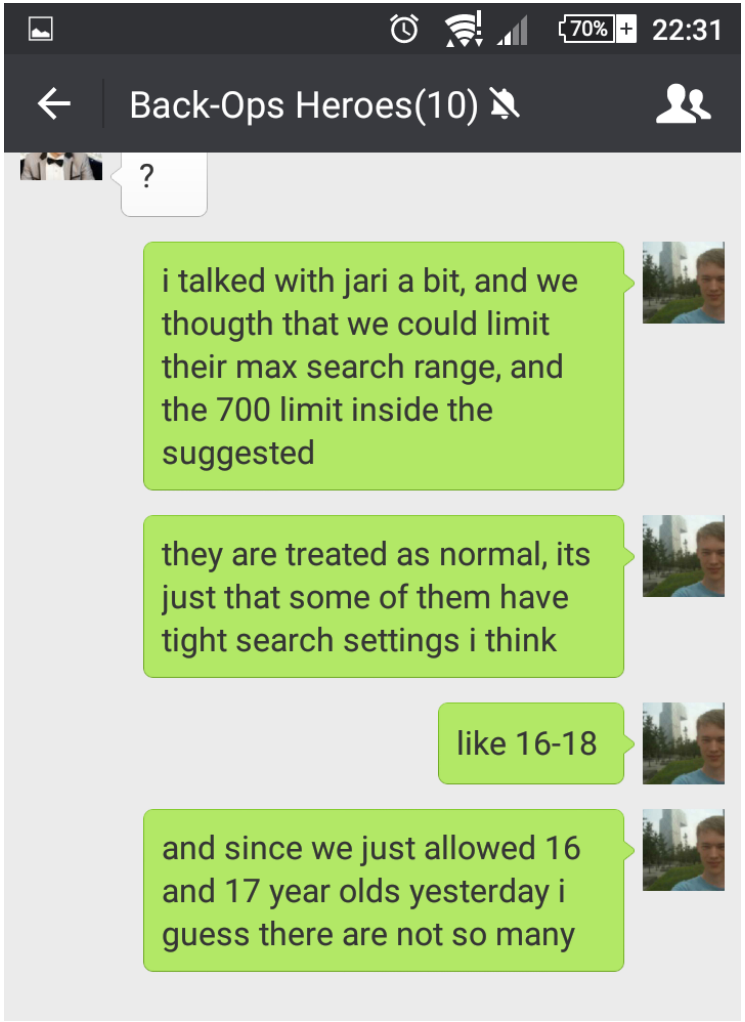
Challenge: Counting Passbys



探探

Challenge: Teenagers





探探

DDL changes

- Often work great, create index concurrently, add null columns
- We mostly follow the post by Braintree, https://www.braintreepayments.com/blog/safe-operations-for-high-volume-postgresql/

- However:
  - Update of return type of function is not instant but breaks queries running while change is happening
  - (make null not null, rewriting a big table is a pain, even just backfilling it with values)

探探

Accidents

- TXID wraparound in DB with TB sized table
- Out of disk space
- Integer primary key out of range
- Wrong query plan! ARGH!
- Hardware failure

...

Wish List

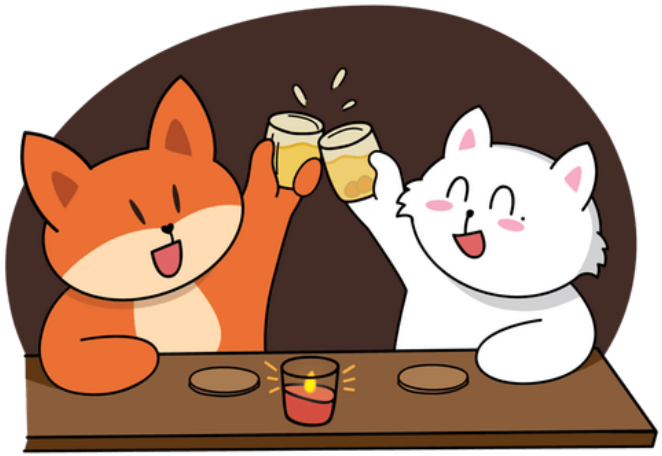
- Make EXPLAIN ANALYZE display the inner part of functions
- Improve locking and propagation of DDL changes under (heavy) load

Conclusion

- PostgreSQL is not perfect and doesn’t solve all the problems. But it is still a very good companion in a fast growing company!

探探

Questions?



Thank You!

vb@viblo.se

探探