Location based dating in China - 0 to 100000000 daily swipes in a year

Victor Blomqvist

vb@viblo.se

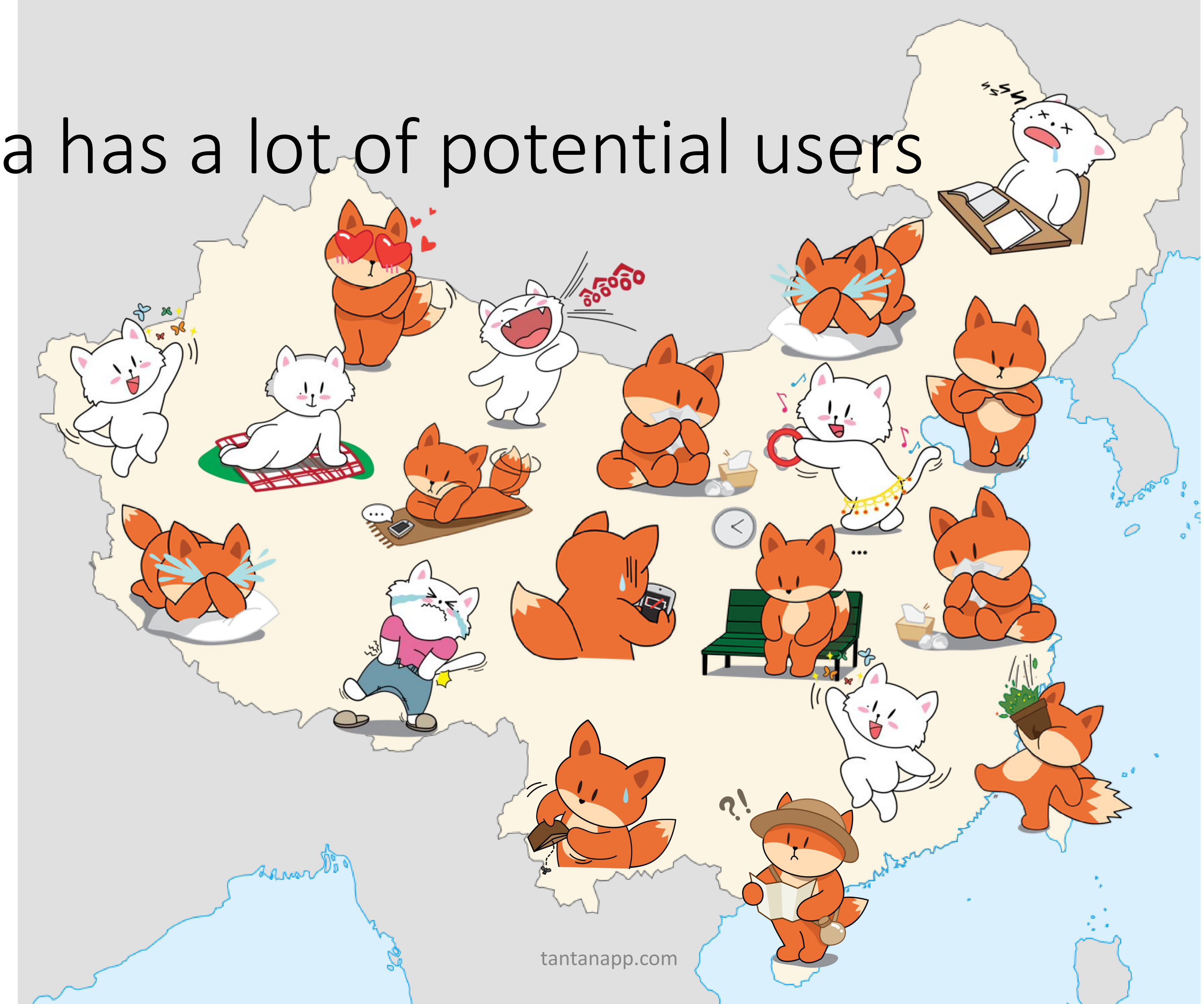
blomqvist@tantanapp.com

Tantan (探探)

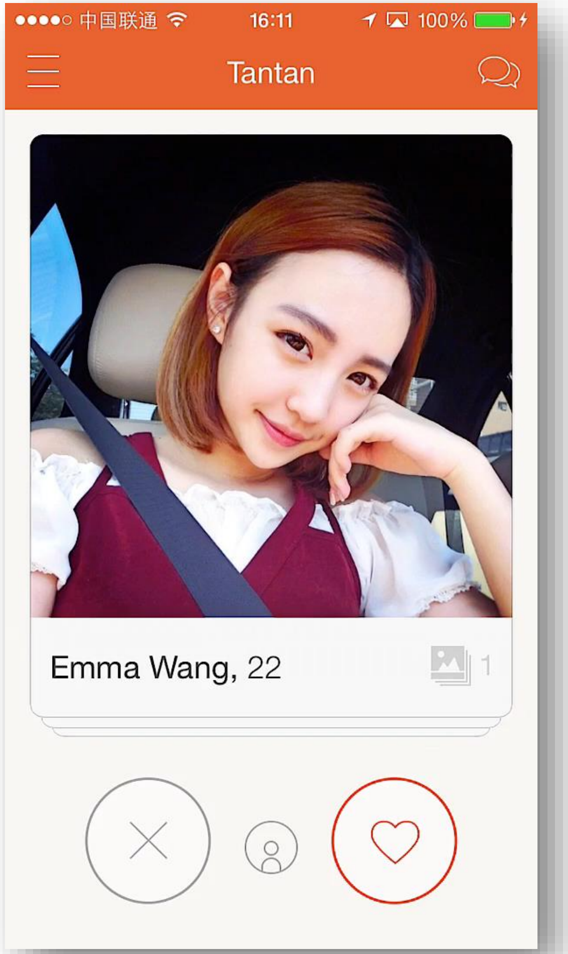
October 28, PGConf.EU 2015 in Vienna

探探

China has a lot of potential users



How Tantan works



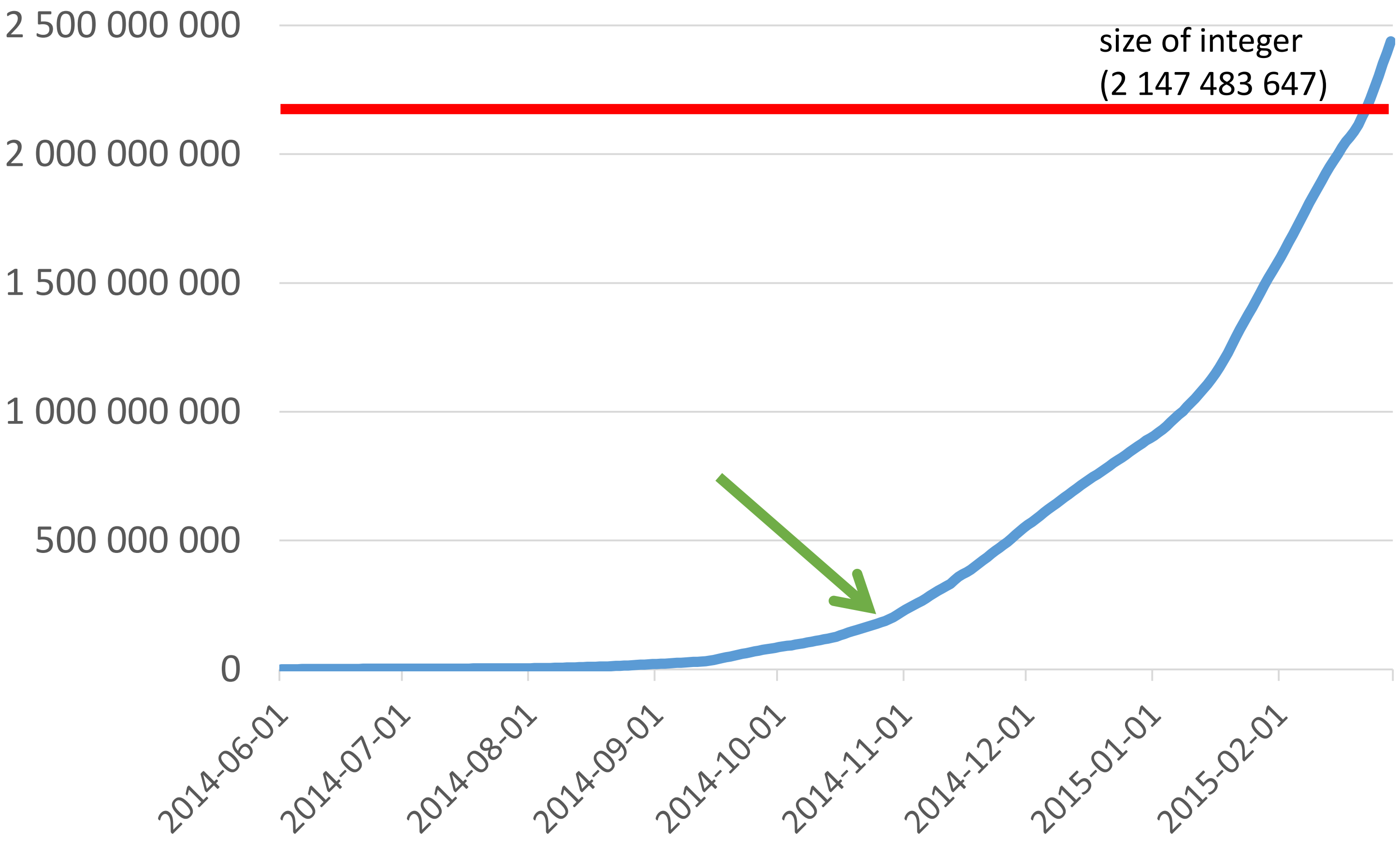
探探

tantanapp.com

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

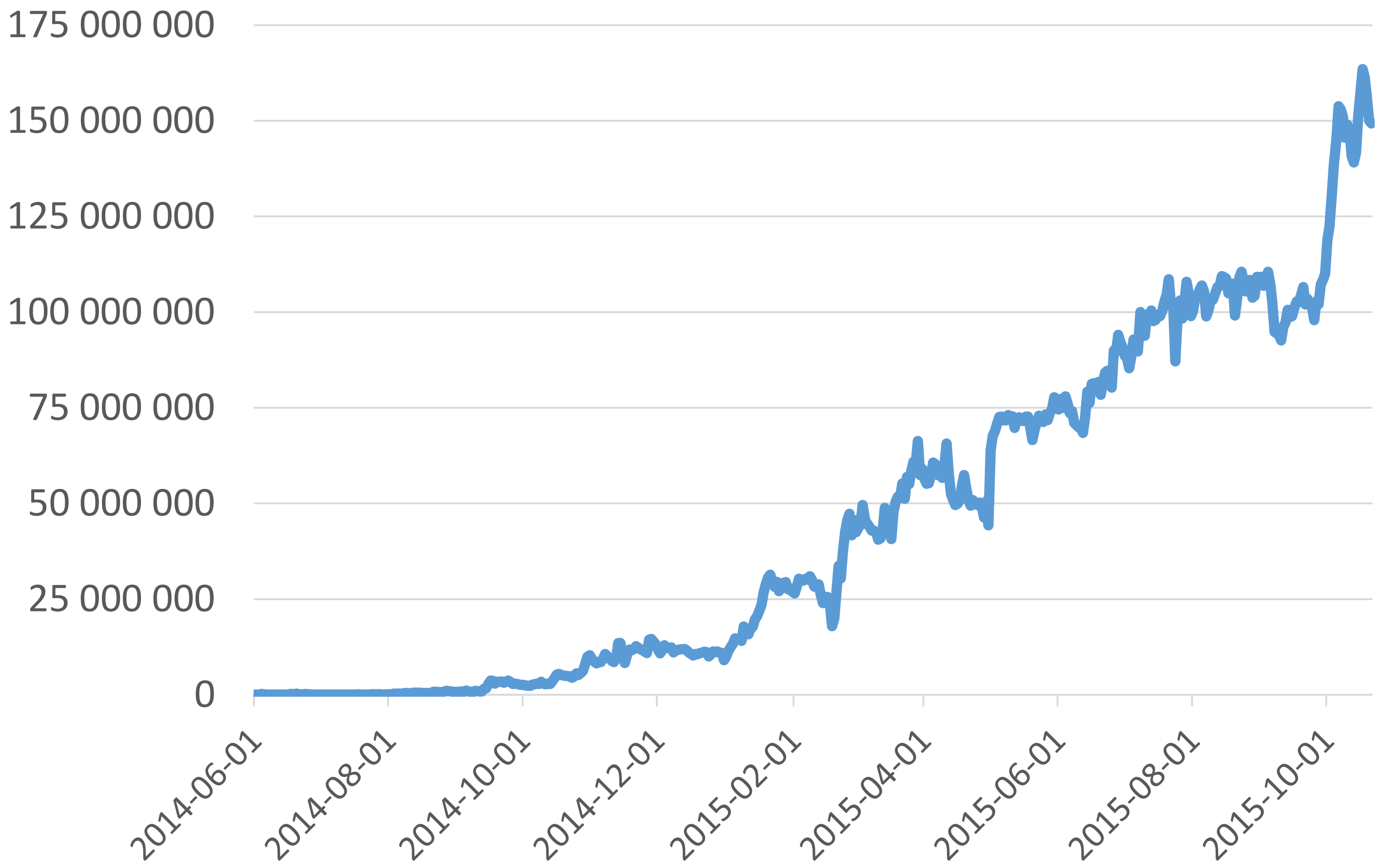
Email sent on October 31 2014

Total relationships stored



探探

Today I will talk about how we got Tantan to scale from 0 to ~~100~~ 160 million daily swipes



探探

Like / dislike other users

探探

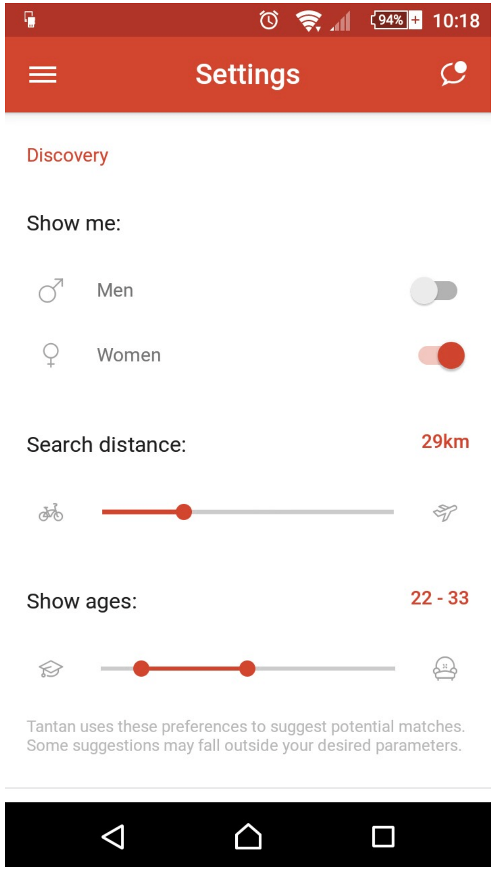
Like / dislike other users - Table

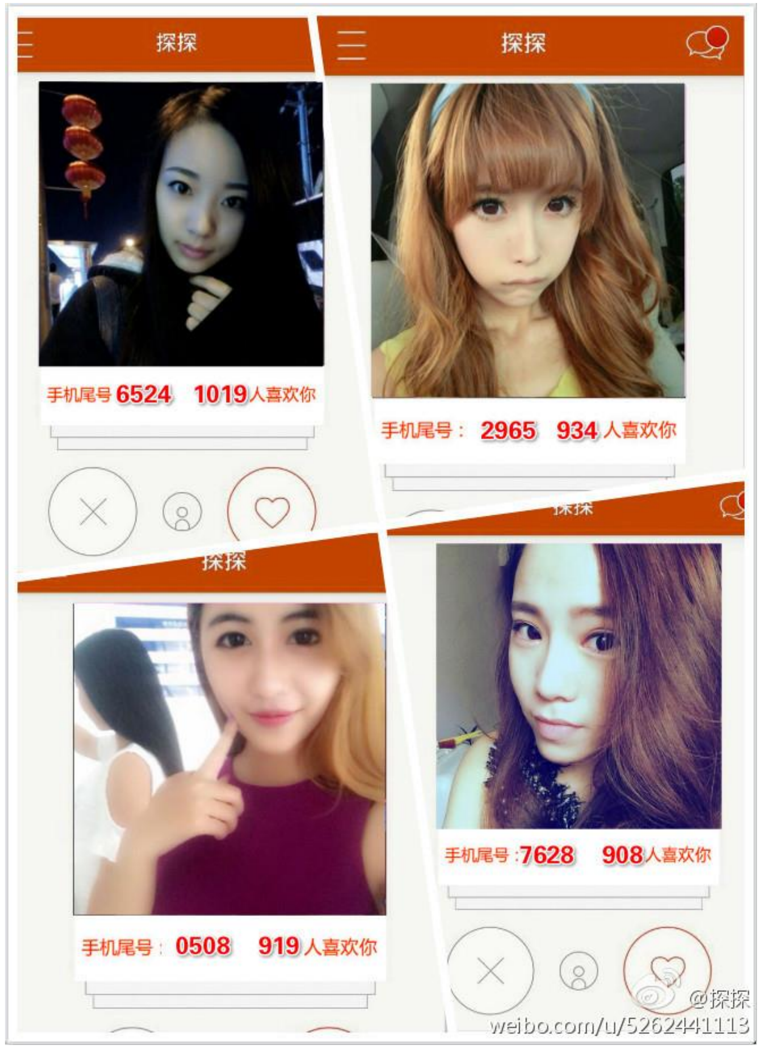
```
CREATE TABLE relationships (
    id serial PRIMARY KEY,
    user1 integer,
    user2 integer,
    state varchar CHECK (state IN ('liked', 'disliked'))
);
```

Like / dislike other users - Query

```
INSERT INTO relationships (user1, user2, state)
    VALUES (1, 2, 'disliked');
```

Suggest users for you





探探

Suggest users for you - Table

```
CREATE TABLE users (
    id serial PRIMARY KEY,
    name varchar,
    gender varchar CHECK (gender IN ('male', 'female')),
    age integer,
    location geometry,
    last_active timestamp
);
```

Suggest users for you - Query

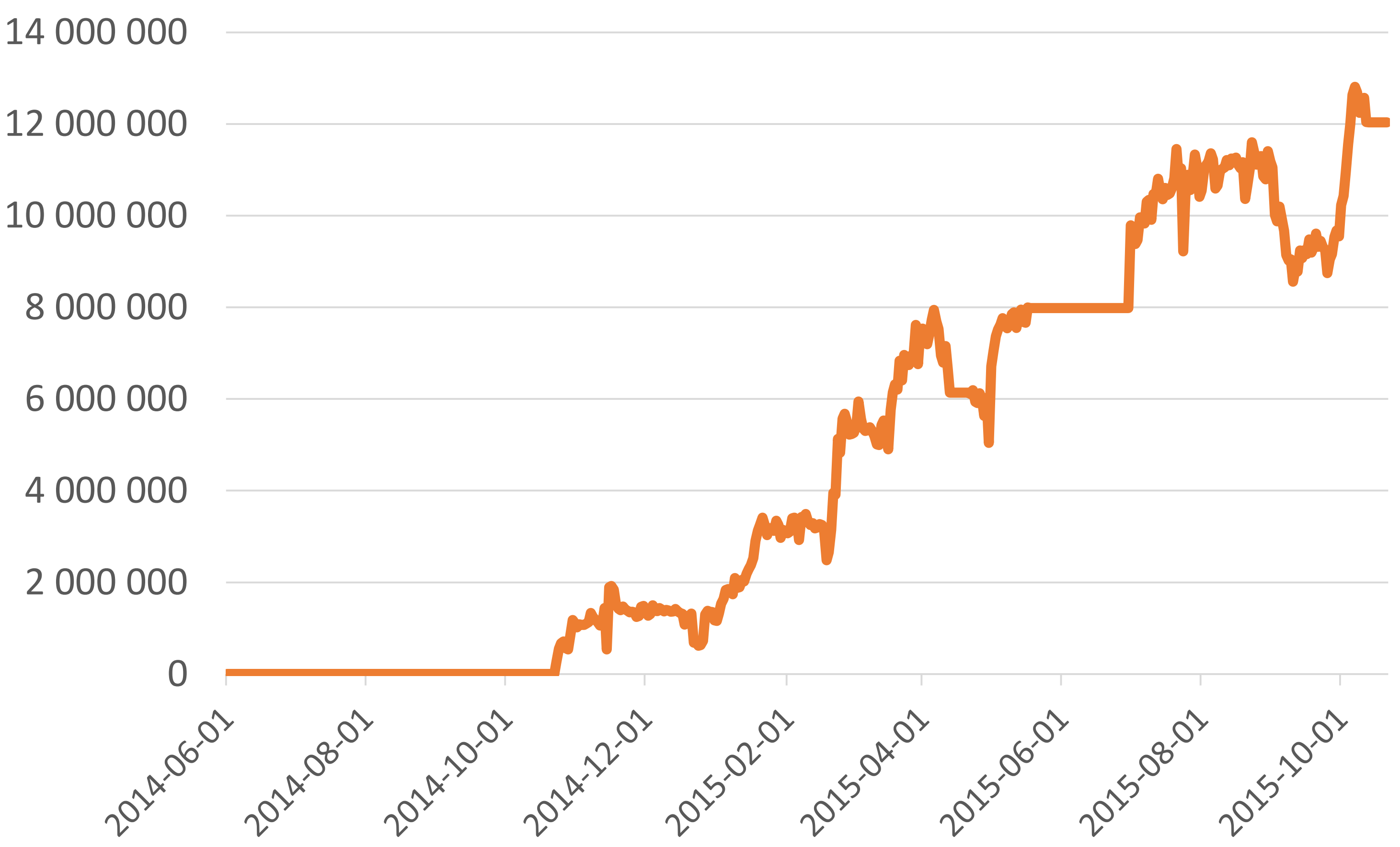
```
\set vienna ST_Point(16.363449, 48.210033)

SELECT * FROM users
    WHERE gender = 'female'
    AND age BETWEEN 22 AND 34
    ORDER BY ST_Distance(location, :vienna)
        * (now() - last_active);
```

Suggest users for you - Query (2)

```
SELECT * FROM users
    WHERE gender = 'female'
    AND age BETWEEN 22 AND 34
    AND NOT EXISTS (SELECT * FROM relationships
        WHERE user1 = 1 AND user2 = users.id)
    ORDER BY ST_Distance(location, :vienna)
        * (now() - last_active);
```

We run the “suggest” query 12 million times per day



What is the first thing to do when the database is slow?

What is the first thing to do when the database is slow?

```
CREATE INDEX users_location_idx ON users
    USING GIST (location);


CREATE INDEX relationships_user1_idx
    ON relationships (user1);
```

```
WITH indexed_query AS (
    SELECT * FROM users
    WHERE gender = 'female'
        AND age BETWEEN 22 AND 34
        AND NOT EXISTS (
            SELECT * FROM relationships
            WHERE user1 = 1 AND user2 = users.id)
    ORDER BY location <-> :Vienna
    LIMIT 100 )
SELECT * FROM indexed_query
    ORDER BY ST_Distance(location, :vienna)
        * (now() - last_active)
```

探探

We can do better

We can do better

```
\set search_area ST_Envelope(ST_Union(ARRAY[
    ST_Project(:vienna, 1000, 0)::geometry,
    ST_Project(:vienna, 1000, 3.14/2)::geometry,
    ST_Project(:vienna, 1000, 3.14)::geometry,
    ST_Project(:vienna, 1000, 3.14*1.5)::geometry
    ]))
```

We can do better

```
WITH indexed_query AS (
    SELECT * FROM users
    WHERE gender = 'female'
        AND age BETWEEN 22 AND 34
        AND NOT EXISTS (
            SELECT * FROM relationships
            WHERE user1 = 1 AND user2 = users.id )
        AND location @ :search_area
    ORDER BY location <-> :vienna
    LIMIT 100 )
SELECT * FROM indexed_query
    ORDER BY ST_Distance(location, :vienna)
        * (now() - last_active)
```

And even better

- Separate index for males and females
- Remove inactive users from index

In production it runs in about 250ms in average, more than 10 million times every day

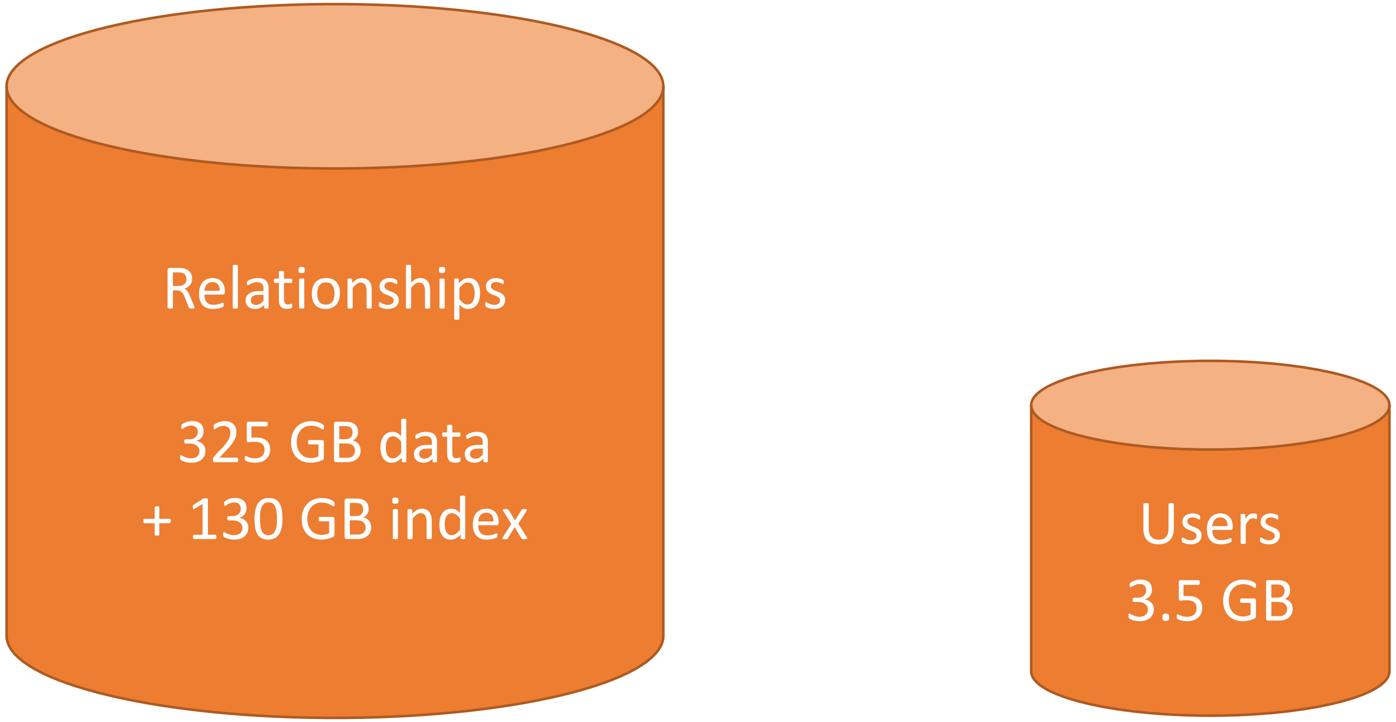
What is the second thing to do when the database is slow?

Buy powerful server(s)!

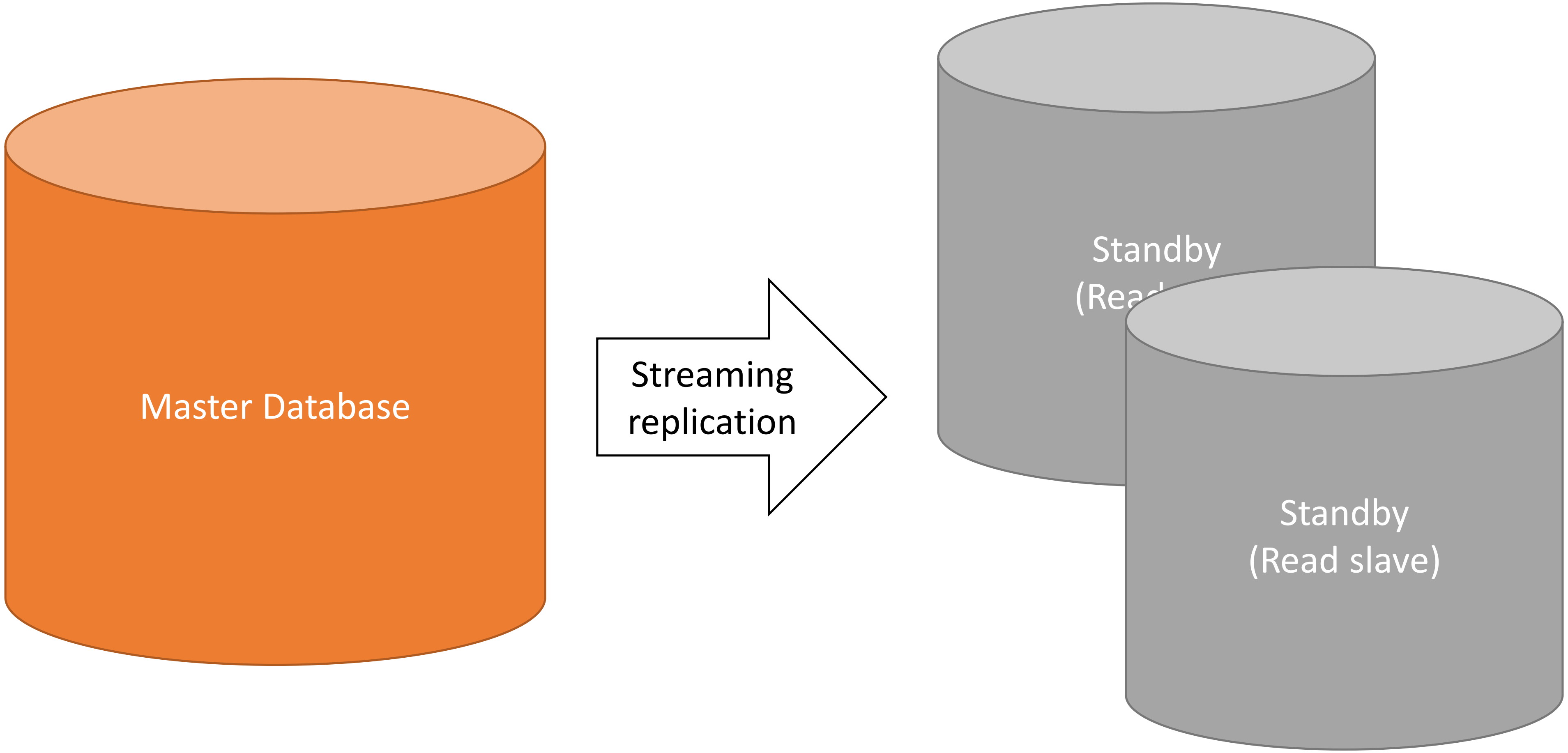
- 2x Intel Xeon CPU E5-2680 v2 @ 2.80GHz (20 cores / 40 threads)
- 256GB RAM
- 4x 600GB Intel SSD DC S3500 (in RAID 10 for PostgreSQL data)
- 2x 240GB Intel SSD 730 (in RAID1 for PostgreSQL transaction log)
- 2x spinning disks (for OS and other file storage)

Data size?

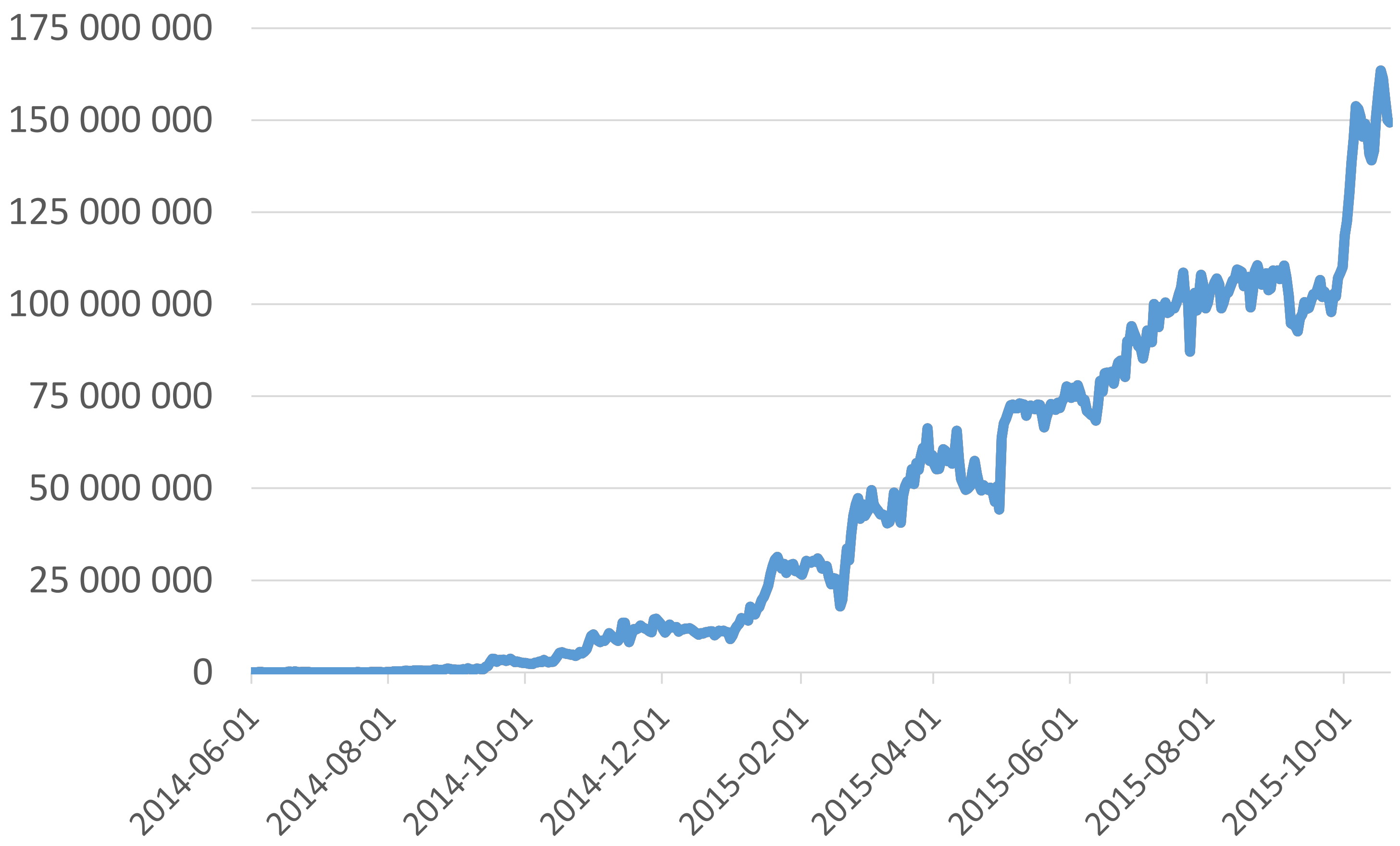
8x



Add standby / read slaves



Optimized queries and good hardware are not enough!



探探

Requirements

- Easy to implement
  (I had <1 year of PostgreSQL experience)
- Easy to implement
  (we are 14 engineers including ops, half working with Android/iOS, other half working on new backend features)
- Easy to implement
  (we need it asap)

Sharding to the rescue!

Let's use “Instagram sharding”*

* http://instagram-engineering.tumblr.com/post/10853187575/sharding-ids-at-instagram

"Instagram Sharding"

Database 1	Database 2

• shard_1
  • pictures-table
• shard_2
  • Pictures-table
	• shard_3
	  • pictures-table
	• shard_4
	  • pictures-table

Recap: We have two things we care about. Users and relationships

- Users can be handled by one server
- Relationships cannot be handled by one server

Insight: We only need to shard the relationships!

Users and relationships - Sharding

Database 1	Database 2

• shard_1
  • relationships
• shard_2
  • relationships
• common
  • users_clone
	• shard_3
	  • relationships
	• shard_4
	  • relationships
	• common
	  • users_clone

How to find the shard?

User ID % Num shards = Shard ID

42 % 4 = 2

Users and relationships - Sharding

Database 1	Database 1*	Database 2	Database 2*

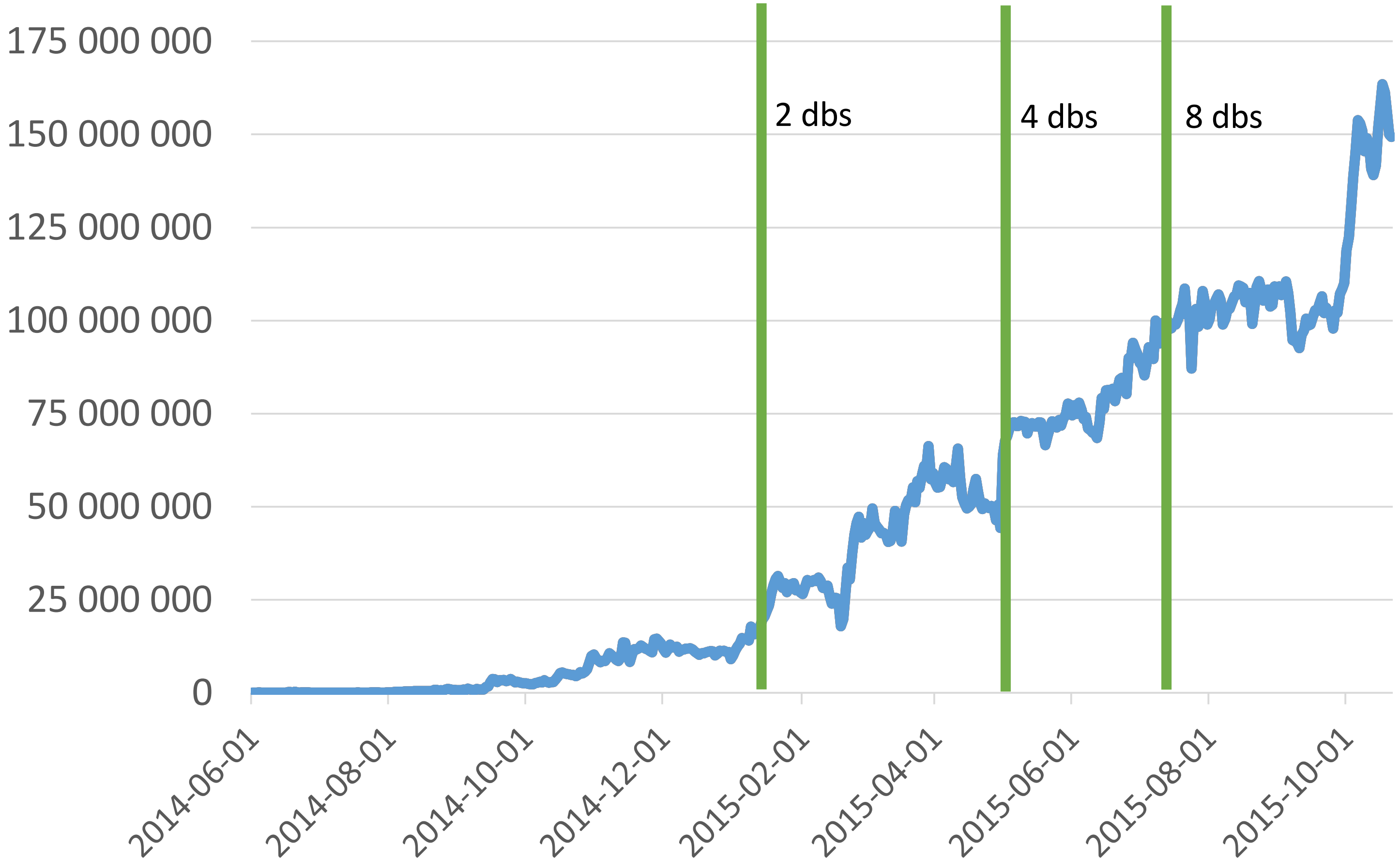
• shard_1
  • relationships
• shard_2
  • relationships
• common
  • users_clone
	• shard_1
	  • relationships
	• shard_2
	  • relationships
	• common
	  • users_clone
		• shard_3
		  • relationships
		• shard_4
		  • relationships
		• common
		  • users_clone
			• shard_3
			  • relationships
			• shard_4
			  • relationships
			• common
			  • users_clone

Users and relationships - Sharding

Database 1	Database 3	Database 2	Database 4

• shard_1
  • relationships
~~• shard_2~~
  ~~• relationships~~
• common
  • users_clone
	~~• shard_1~~
	  ~~• relationships~~
	• shard_2
	  • relationships
	• common
	  • users_clone
		• shard_3
		  • relationships
		~~• shard_4~~
		  ~~• relationships~~
		• common
		  • users_clone
			~~• shard_3~~
			  ~~• relationships~~
			• shard_4
			  • relationships
			• common
			  • users_clone

Swipes per day and number of databases



Summary: When the database is slow

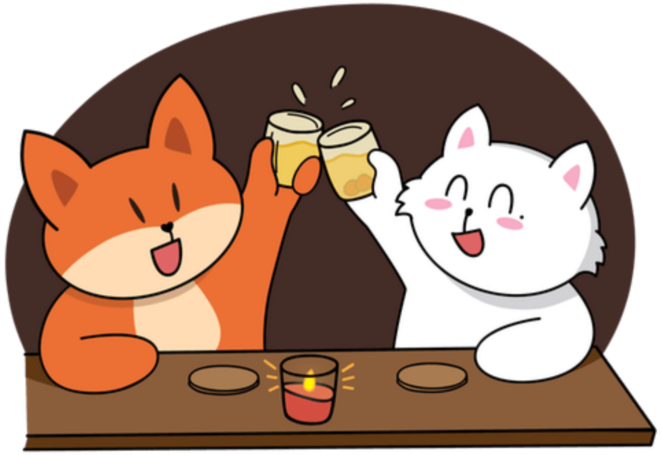
- Optimize with index
- Optimize with query rewrite
- Buy better hardware & standbys
- Sharding

But this is not all we do..

- We monitor with Ganglia
- We use functions for all queries
- We store all database changes in Git
- We use simple Bash scripts for deployments

And we are hiring!

Questions?



Thank You!

blomqvist@tantanapp.com

vb@viblo.se