Ruby and a Graph

Piotr Walkowski,
Ruby dev,
Kraków, Polska
Twitter: @walu911

Snapchat

- 100 million daily active users
- Total Number of Snapchat Users: 200 million+

Tinder

- 10 million daily active users
- Total est. 50 Mo users

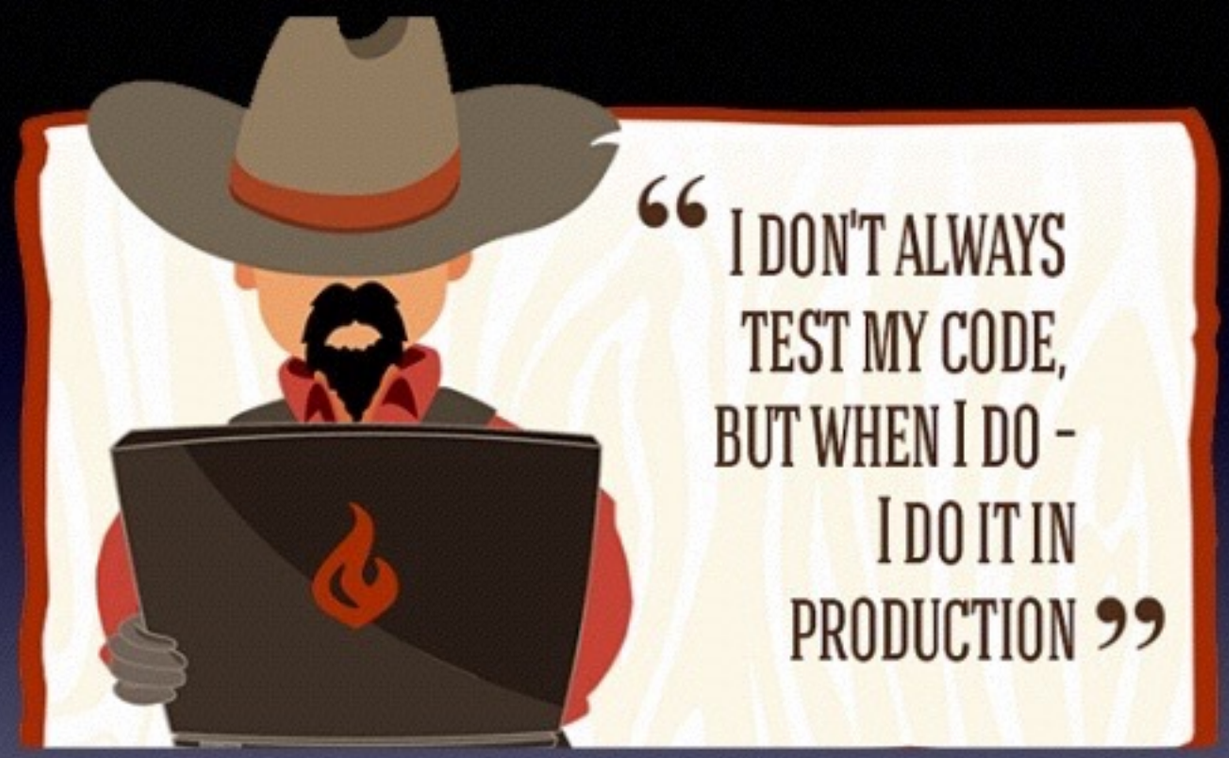
- *sources:*
- *http://expandedramblings.com*
- *http://www.omnicoreagency.com/snapchat-statistics/*

Problems I faced with

- My main task: to rewritte exising „dating-oriented” portal to more modern technology (here RoR)
- Cope with legancy code
- Big awful monolith+ shortage of tests (sound familiar?)
- Written in ColdFusion, anyone heard?

Community of CF programmers in Krakow

Problems I faced with

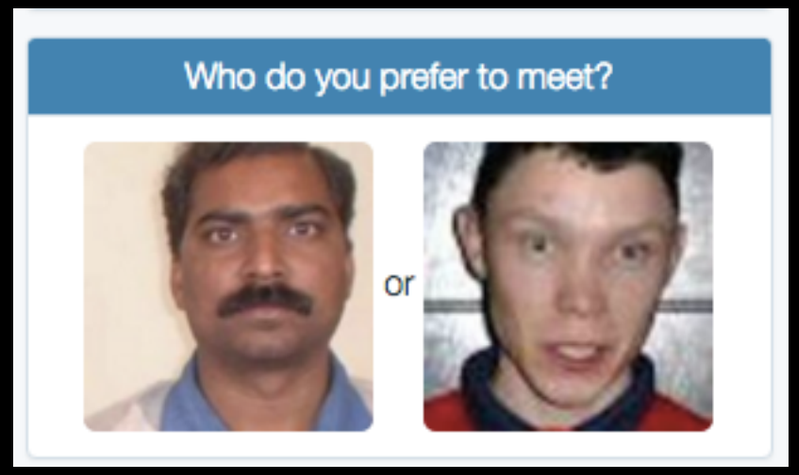


Don’t be a Cowboy Coder

Problems I faced with part 2

- Quite big legacy RDBMS (… on SQL server)
- Roughly ~450 tables, hundrets of triggers, stored procedures etc..
- **Crucial part** of system…

Cruicial part of system



- Feature called: **„Meet me”**

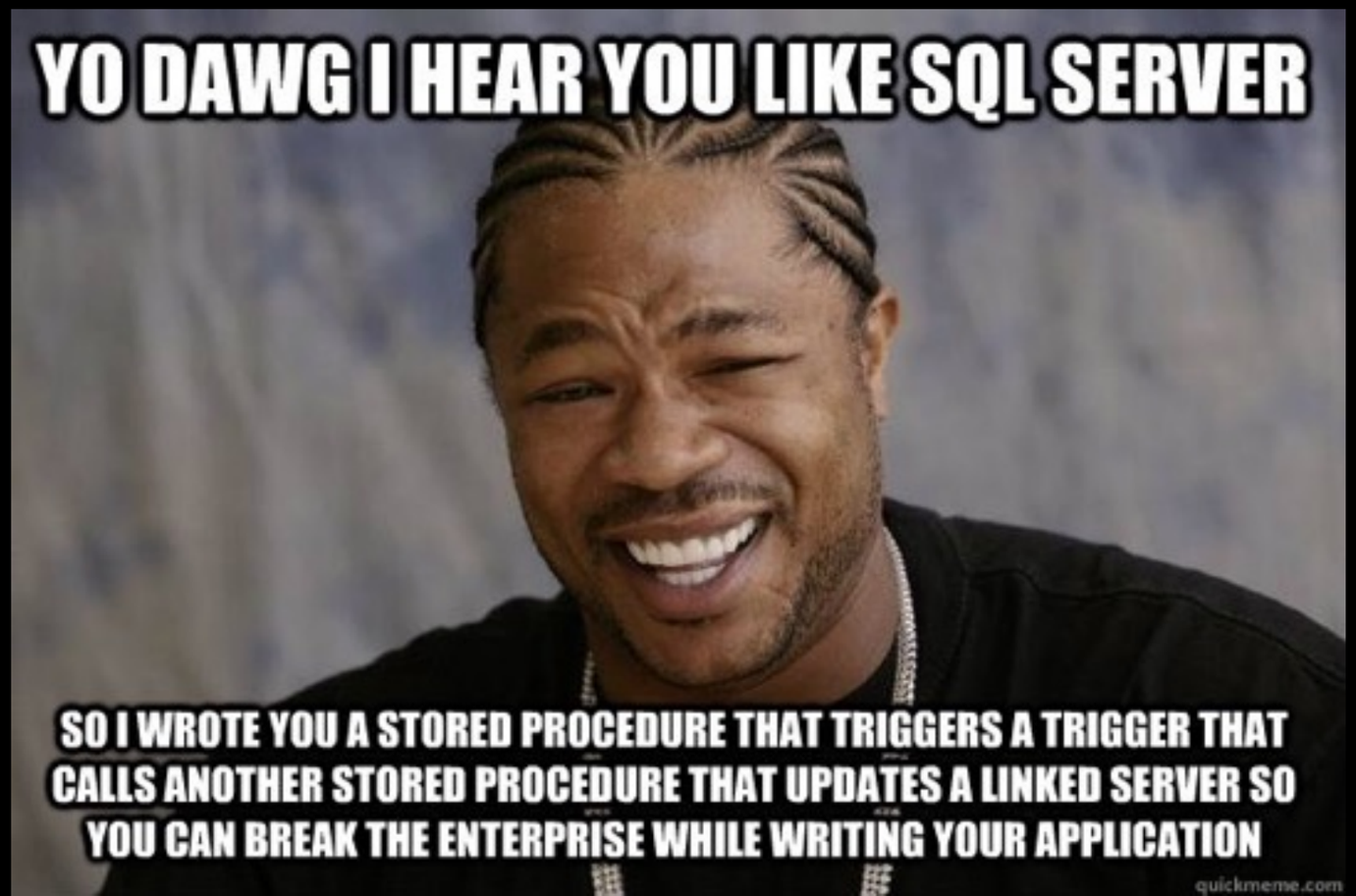
- Recommendation system responsible for making suggesting of new profiles to discover by other users (in order to keep them on page … and encourage to pay for premium features)

- It suggests according to profile fields, and other characteristics (metrics) best profiles to potential match (see „swipe right” ™).

- It has siginificant meaning.

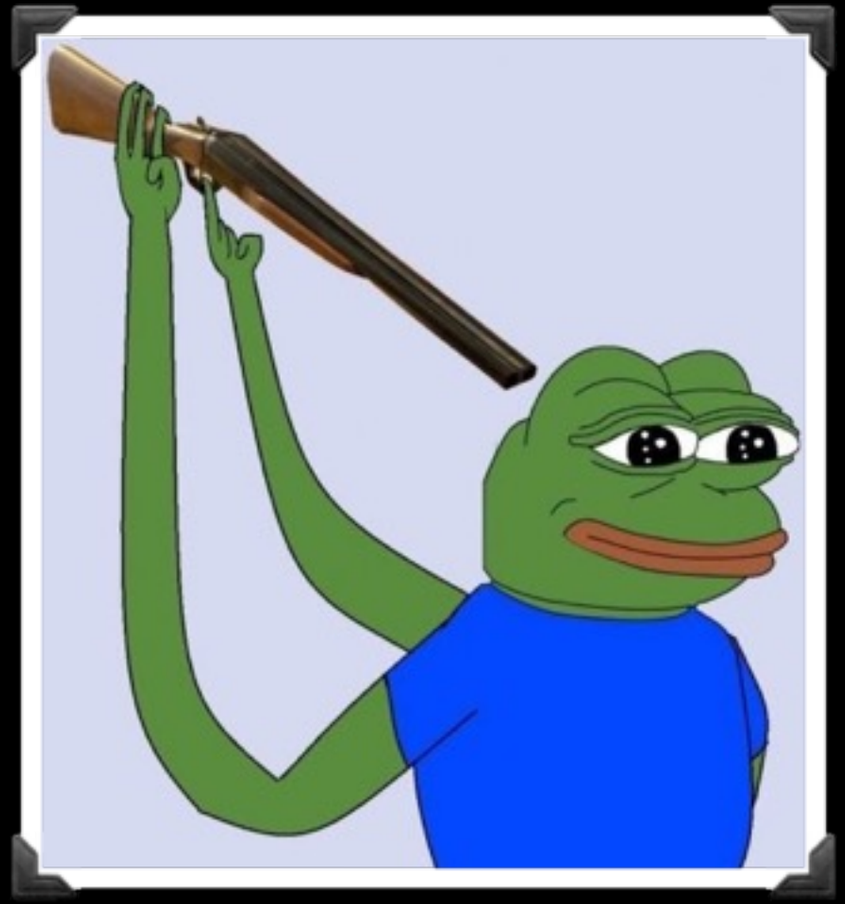
How it worked previously

- Most of logic was done by MSSQL
- Code were scattered to many stored procedures, which were running periodically (usually by night) by many different workers.
- Because of relational character of dabases. It generates inefficient comparasions / calculations on each users (almost ~ cartesian product of rows…)



How it worked previously?

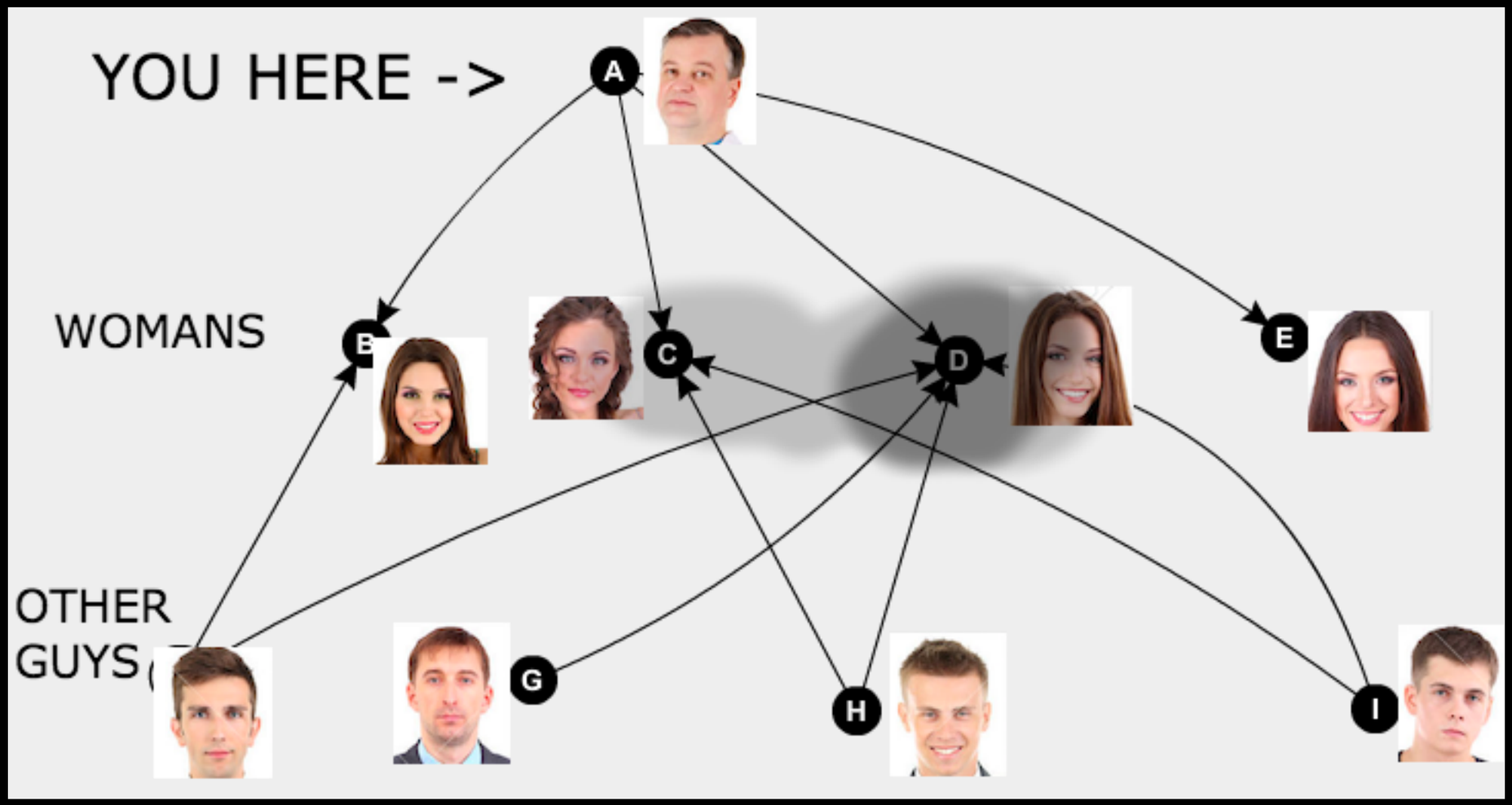
- Worth to mention: The „best” query consisted of 23 joins (sounds ridiculous but true).
- (un?)fortunately dont have snippets ;(



Graph database

- Why? Graphs are almost ideal structures for any recommendation systems, node traversing etc.
- IMPORTANT: **Requires** different way of thinking
- Our type - Neo4j
- There are few libraries for Ruby (http://neo4jrb.io)
- Abovementioned abstraction layer resembles AR… but there are still few bugs (project currently maintained by only 2 busy devs…)
- Much, much better then in 2012 (Imo only interesting detail, not useful tool), but still bit buggy. Progress! (͡° ͜ʖ ͡°)
- GOOD INFORMATION: We can use **cypher** (Some SQL language equivalent for Neo4j) - Why not use it?

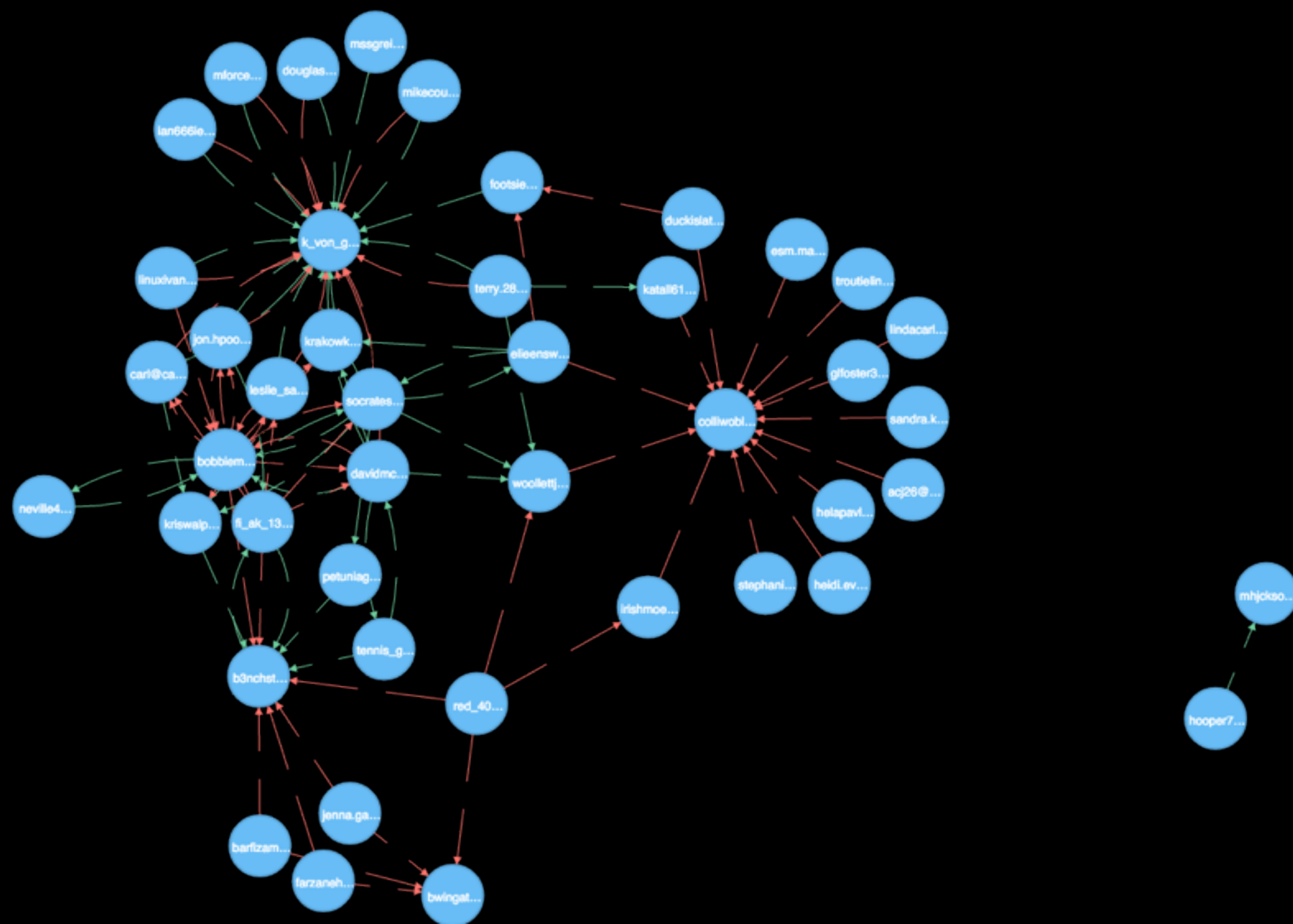
Graph visualitions of our recommendation system



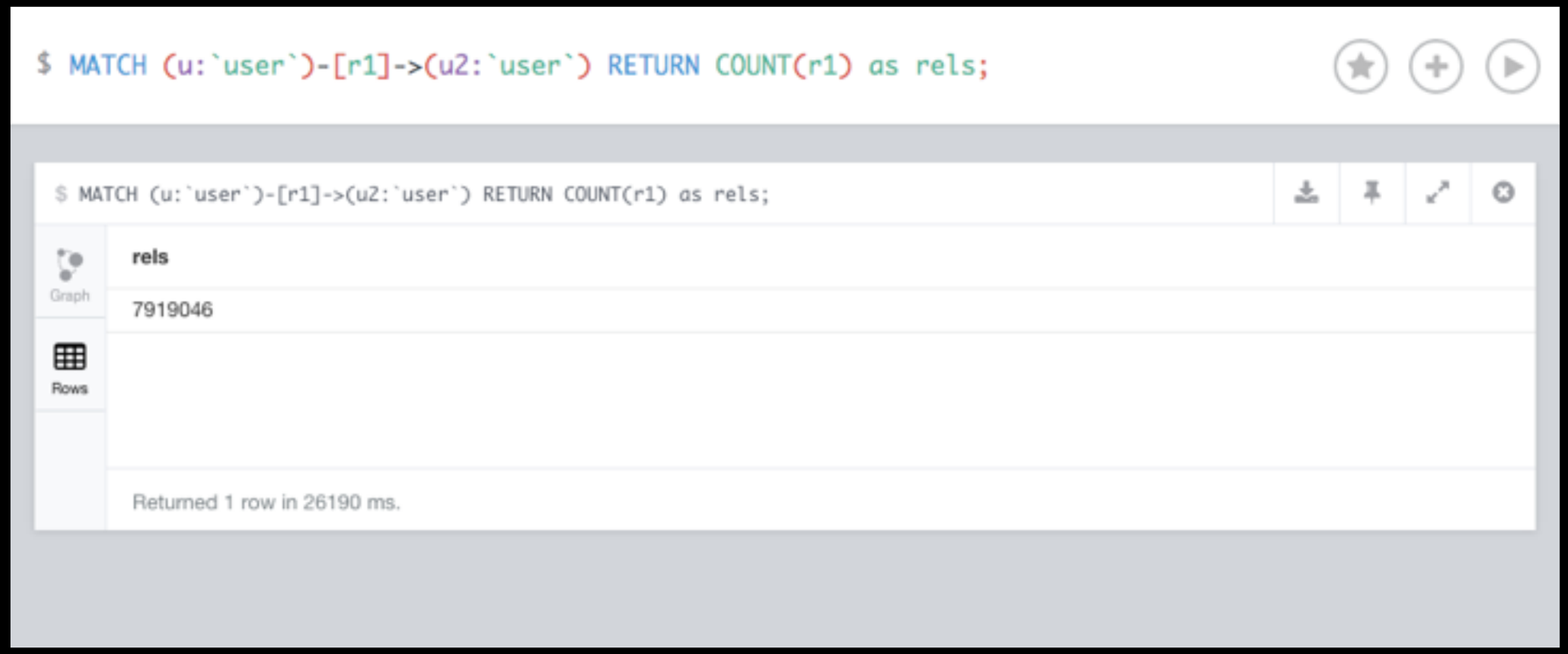
Lets try cypher in practice

```
def get_similar_profiles(limit = 8)
  query = "MATCH (profile:`user` {pid: #{self.pid} })<-[:`view`]-(guys:`user`)-[:`view`]->(other_pretties:`user`)
           WHERE (profile.gender = guys.looking_for)
            AND (profile.looking_for = guys.gender)
            AND (profile.looking_for = other_pretties.looking_for)
            AND (profile.gender = other_pretties.gender)
            AND (guys.looking_for = other_pretties.gender)
            AND (guys.gender = other_pretties.looking_for)
          RETURN DISTINCT other_pretties as uu
          LIMIT #{limit};"

  parse_and_evaluate(query)
end

def meetme_candidate
  query = "MATCH (c:`user`), (me:`user` {pid: #{self.pid}})
           WHERE (NOT (c)<-[:`want`]-(me)) AND (c.pid <> me.pid)
             AND c.active = 1 AND c.scammer = false
             AND c.gender = me.looking_for
            RETURN c LIMIT 1"

    parse_and_evaluate(query)
end

def number_of_viewers
  self.viewers.count
end
```

Our graph (very small part) example visualisation



Does it scale?



- Typical query is evaluates ~0.1s (in real time) - Significant improvement (previously it took few days until all workers were finished their „comparation jobs”).
- We just setup all on AWS machines (one of averages) - no magic here ;-)
- Database consists of 1.75*10^6 nodes, has ~8*10^7 verticles (with properties). Physical size ~ 3GB.
- Most complex query (involving 5 levels nesting) is evaluated ~ 3 sec.

Why it rocks?

- Sorry no time for graph theory review ;-) [probably few seconds left.. maybe next year?]
- http://neo4j.com/why-graph-databases/
- http://radar.oreilly.com/2013/07/why-choose-a-graph-database.html

- To cut a long story short:
  - Graph databases traverse relationships **very** quickly (but perform less well on mass/bulk queries)
  - MATCH n-[r:foo|bar*..7]->m RETURN m; (Many have tried this in SQL, and it requires ninja skill to express this…)
  - On queries like this, neo4j is going to kick RDBMS's ass.
  - Another cool thing are diffrerent graph algorithms (e.g calculating transitive clousure) hard or unable to do in RDBMS

Why it sucks? ;-)

- Any RDBMS would rather blow away neo4j in performance on quey like this:
- MATCH p WHERE p.name='Piotruś' RETURN p;
- Despite no explicite relationships it forces the DB to scan all nodes :(. Might be improved by setting indexes but anyway much worse than typical RDBMS
- Big imports (like CSV ~ few GBs) works not efficient. At current stage, transations are not perfect.

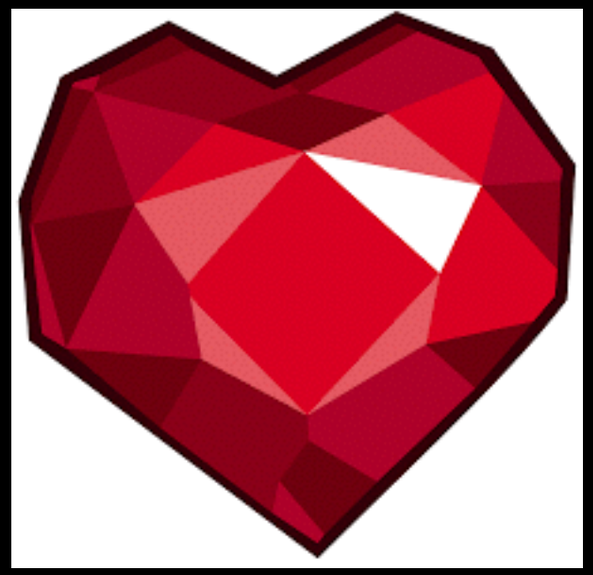
Neo4jrb features (quick review)

- Properties
- **Indexes / Constraints**
- Callbacks
- Validation
- Assocations
- **Relationships are first-class citizens in Neo4j, models can be created for both nodes and relationships**
- A chainable arel-inspired query builder
- Transactions
- Migration framework

Nice quote from SO.

- *„Don't use neo4j like a relational database, or it will perform about as well as if you tried to use a screwdriver to pound nails”*

Thank you for attention



- Want know more? - ping me at twitter **@walu911** (Despite Im not ninja expert - we can discuss some more details)
- Snapchat me too (if you want) .. @walu2
- Thx for my mate @flyerlsk ;-)