Using a graph database for resource authorization
Roar Audun Myrset and Sebastian Verheughe

2nd Workshop on Graph-based Technologies and Applications «Graph-TA»
Barcelona, Spain
February 2014
Customer model

Customer

Account

Subscription (cell phone)
The problem: Which resources does a user have access to?
The problem domain

The resource authorization logic was written in SQL and had been running for several years. But we started to get into trouble:

• Calculation of resources for a large user could take up to twenty minutes

• To get acceptable response times we needed to introduce caching which again caused:
  – Inability to scale for more users with access to a large set of resources
  – Data not being up to date

• Complicated to understand the SQL code
Solution part 1

• Question: Re-implement the existing SQL/relational database or make something new?
  – Decision was made to use a graph database based on gut feeling and a Proof of Concept
  – The graph database chosen was Neo4J
Solution part 2

• ~30 million nodes/vertices in the graph
  – ~3 million user nodes
  – ~27 million resource nodes
• ~40 million relationships/edges
• At runtime the database is loaded into memory (in-heap) and is using about 20GB.
Results

- We don’t need to cache anymore hence we are able to scale with regards to the number of users.
- The graph query logic is easier to read than the old SQL logic.
- Users are getting acceptable response times:
Reasons to meet us at the poster

• Experience with using Neo4J
• Performance optimizations done
• All other stuff we didn’t get to tell you