SQL to NoSQL

NOKIA

Lessons learnt migrating a large and highly-relational database into a "classic" NoSQL

Enda Farrell @endafarrell

“It’s not you, it’s me …”

This doesn’t apply to you

… possibly … probably …

Here's what's coming

What

Why

Complexity

People

Tools

Data

$\left.\begin{array}{l}\text{People}\\\text{Tools}\\\text{Data}\end{array}\right\}$ Lessons

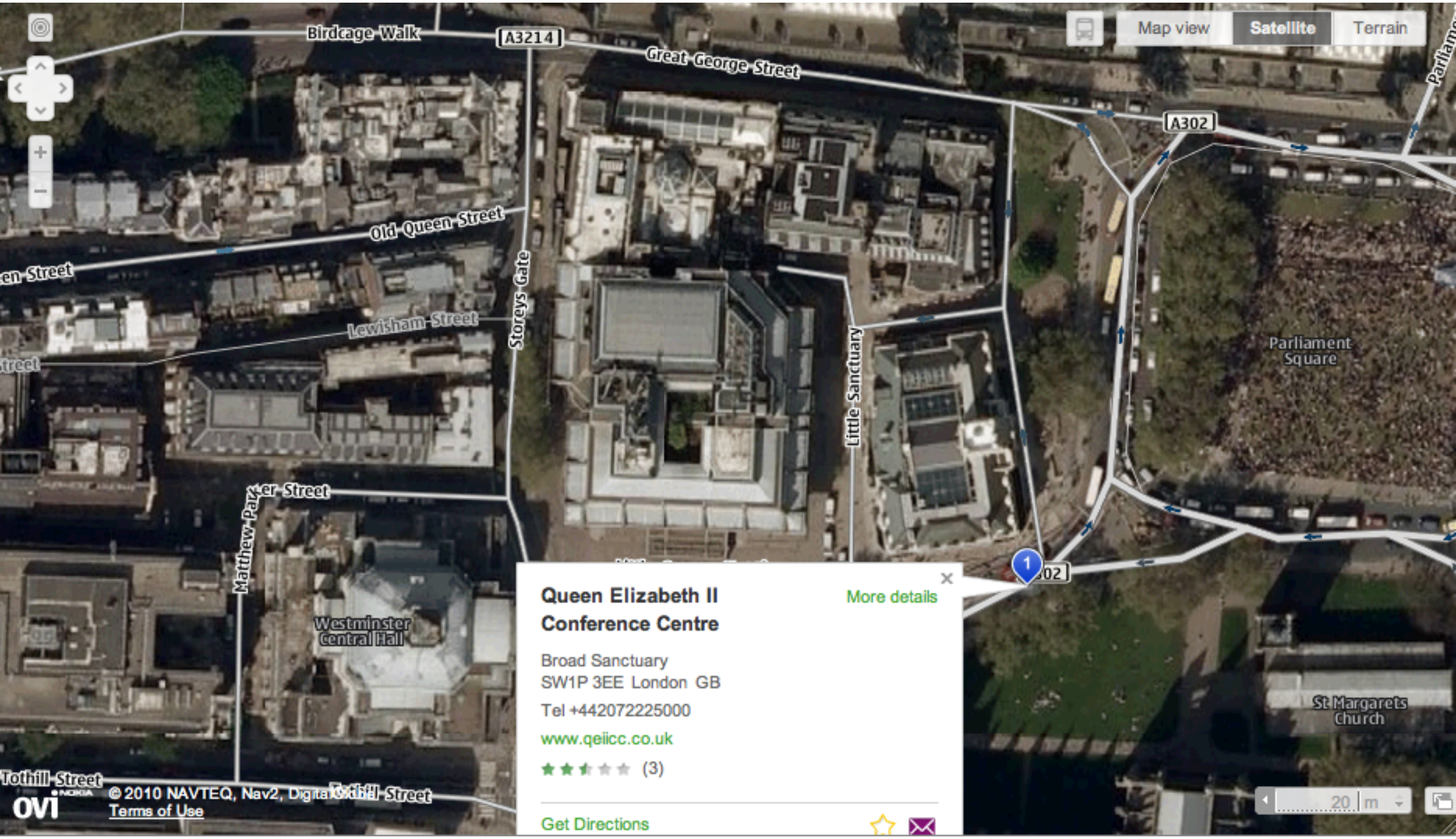
NOKIA

What is this service?

Nokia’s “Ovi Places Registry” aims to be the largest validated point of interest repository in the world

What kind of data?

- Names
- Categories
- Tags
- Location information
  - longitude and latitude
  - postal address
- Contact data



Queen Elizabeth II Conference Centre

More details

Broad Sanctuary
SW1P 3EE London GB

Tel +442072225000

www.qeiicc.co.uk

★★⯪☆☆ (3)

Get Directions

NOKIA

What about it is large and highly relational?

10s of millions points of interest

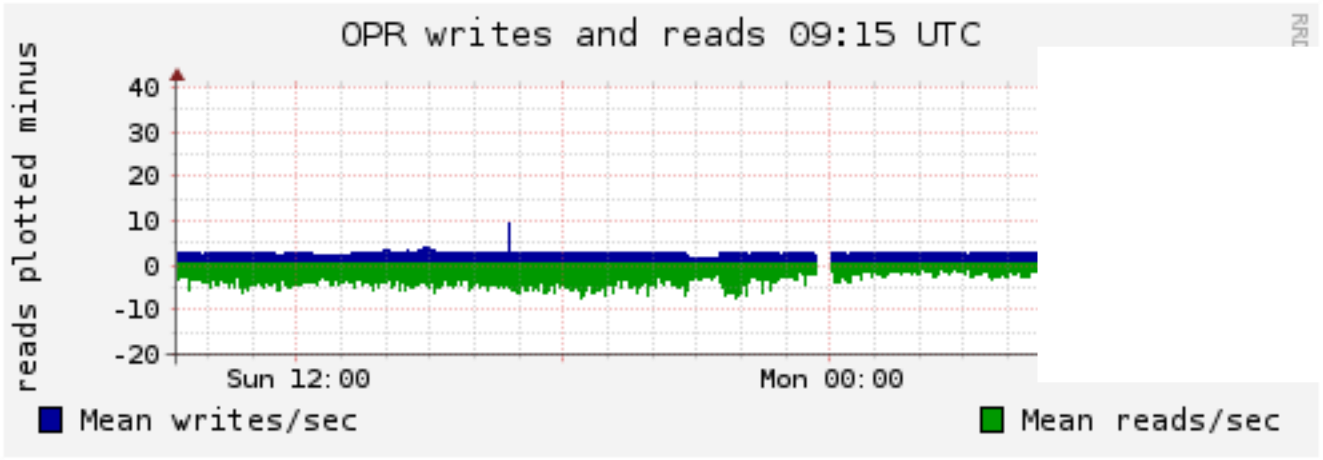
Many many 100s of millions of contributing records

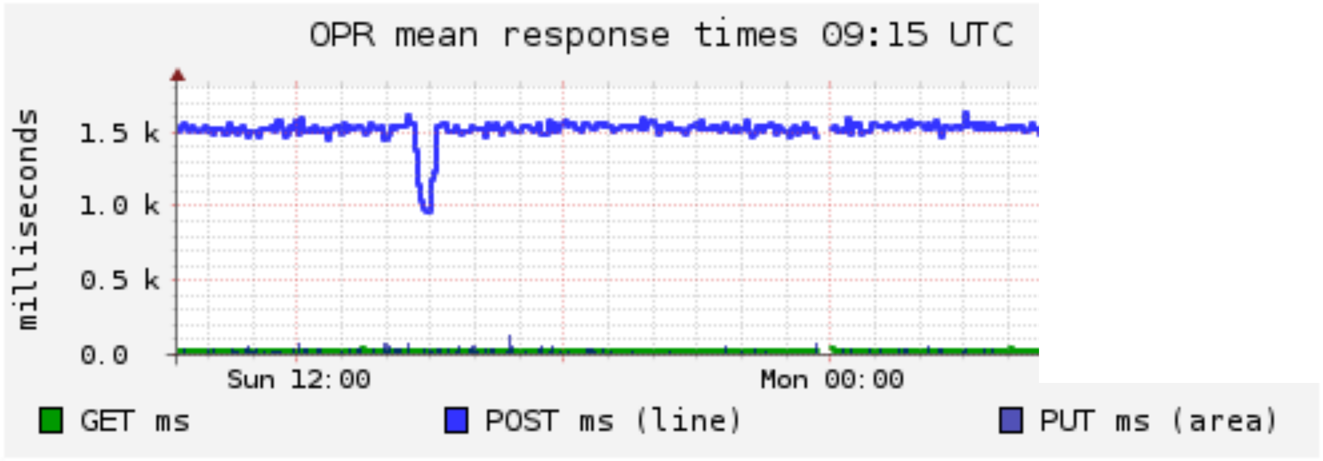
MySQL DB is 600 GB on disk

32 tables, 202 columns

46 non-PRIMARY constraints

What usage patterns do you have?





“Classic” NoSQL? What did you use?

It isn’t CouchDB but a variation of a Nokia internal one

It’s a Key Value store holding JSON

- Without the key you cannot access the value
- It isn’t a “document store” as the store does nothing with the structure

Why did you port this to NoSQL?

bigger and bigger

Nokia maps – web and phone

Yahoo! and soon Bing, => Facebook

Postponed sharding by bigger HDD

We learnt a lot over the last 3 years

Why did you port this to NoSQL? (continued)

SQL databases can be rigid

- The world is a messy place
- “State field”?
- Integrating other organisations’ data

Complicated?

The SQL and NoSQL databases will need to run in parallel for some time

Ops & Disk space

Truth or System of Record

Reconciliation

Syncronisation

Querying

Complicated Ops & disk

Releases, QA, staging and live deployments are more complex when there are two concurrent data storage systems

What assumptions are other people making?

2 x HDD

Complicated truth

When the two systems disagree, which one is “right”?

Complicated reconciliation

How do you know that your two data stores disagree?

Do you check each on on each read/write, or do you have some “batch” code to check equivalence?

Top tip: build a batch reconciler to check keys and revision/etags

Complicated synchronisation

Have you ever tried to keep two different calendars synchronised?

Ever get two email clients telling you you have different numbers of unread mail for the same email account?

Complicated querying

KV stores generally don’t do querying

Some NoSQL stores allow some, but usually more restricted than SQL

We used Solr for performance even though it isn’t as powerful as SQL

The synchronisation complexity is here to stay for us

Complexity

Complexity is often mistaken for “cleverness”

Lessons learnt: people

Why.

It’s a question you will be asked and you will have to answer

Lesson learnt: people

The “DB~A” role is still needed

Here the “~A” is more to do with data/ information architecture than with administration

Top tip: design your JSON. Print it out.

Lesson learnt: people

The effect on your team

- You may have a team of enterprise Java-types who are used to writing Eclipse-enabled code
- In our case we wanted to keep the flexibility that JSON gives us, but it meant we no longer had the same sort of model objects

Lessons learnt: tools

Build “SQL to NoSQL” and “NoSQL to SQL” seeders

You will need to seed your NoSQL from your SQL. You probably have existing DAOs which can form the basis – but this assumes your entities are essentially the same (top tip: keep them so!)

Lessons learnt: tools

Build “SQL to NoSQL” and “NoSQL to SQL” seeders

“NoSQL to SQL” was a seeder we learnt the hard way.

Lessons learnt: tools

What's your unit/integration test coverage?

5 releases post initial launch, is your test data still exercising all code paths?

Lesson learnt: tools

You may find that not everything fits into a Key Value engine

- Even with a queryable index, some data sets really are relational ;-)
- The down-side is that you may therefore have to keep long term the SQL database

NOKIA

Lesson learnt: tools

Visualise your system

- Monitoring: calls, load, response times
- Volumetrics: num docs, HDD, milestones
- Context: draw a systems context diagram
  - (which reminds me …)

Lesson learnt: data

Key generation – it's not sequence numbers nor "auto-increments" anymore

Many are UUIDs and they are long and ugly

- But "guess and check" is ugly too
- Consistent hash?

Lesson learnt: data

Make the revision/etag of the JSON data visible in the JSON

- Not just in a header (assuming HTTP here)
- The data will be taken off-platform, if you want to change it you will need to know that the revision is (or is not) still the same

NOKIA

Lesson learnt: data

Version of the JSON “schema” in the KV

- You have many docs
- You might have to “upgrade” the structure of the KV doc
- Keep the schema version in the JSON

Lesson learnt: data

Many little not few big?

- Easier to replicate
- “Big” docs can be tough on networks
- Trade-off with more client calls (esp error handling)

Probably good ideas

If you're _thinking_ about doing this, do use one of the open source ones

Get one that replicates easily

Build POCs

Thank you! Other questions?

@endafarrell

http://endafarrell.net