Indico DB Technology

An update

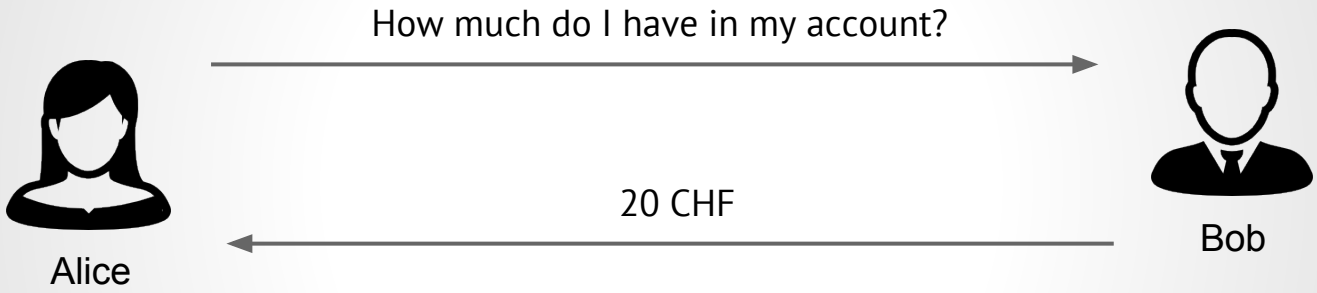
Pedro Ferreira
Ferhat Elmas

- Last presentation
  - Why change?
  - Solutions
  - Database types
- Our solution
- Migration
- Conclusion

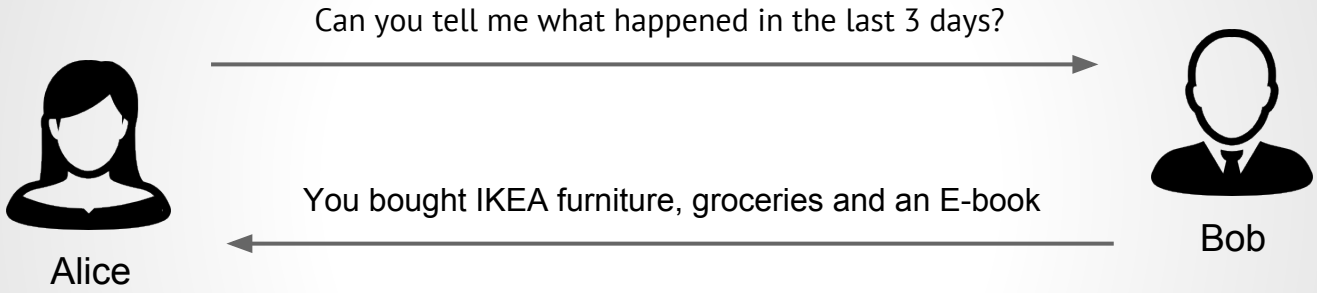
Last time we spoke...

ZODB was not a satisfactory solution.

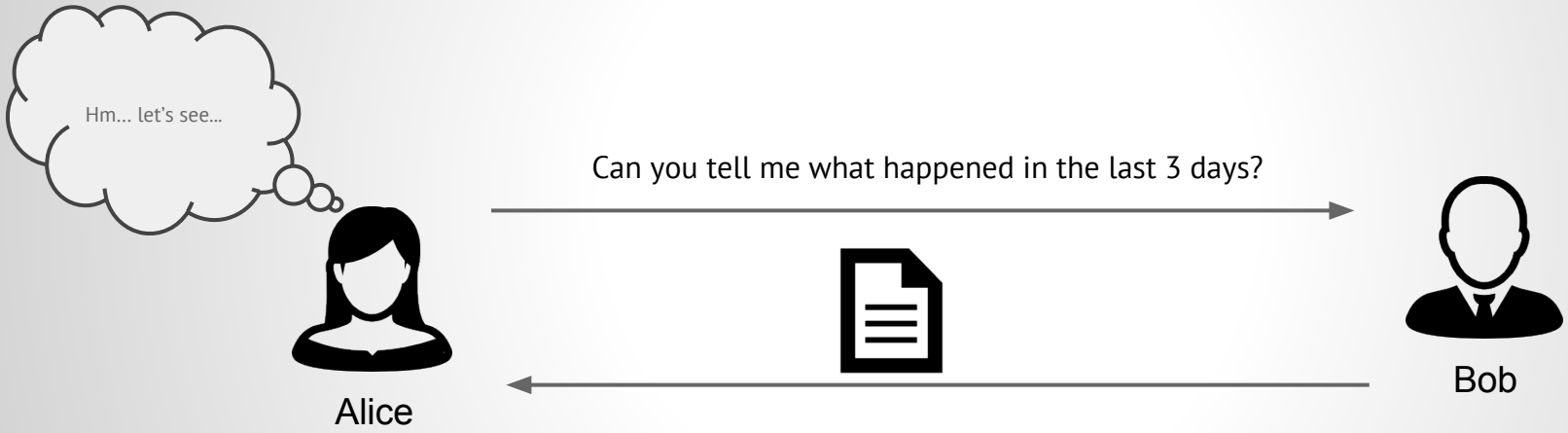
Traditional databases



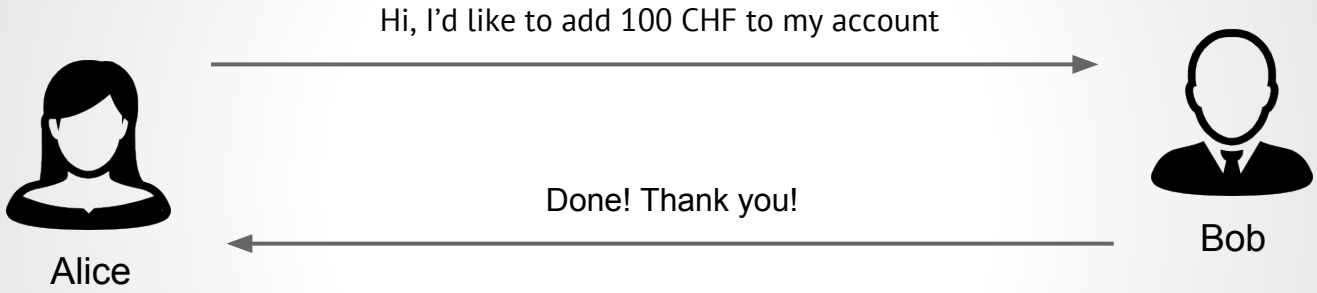
Traditional databases



Object-Oriented Databases



Traditional databases



Object-Oriented Databases

ZODB

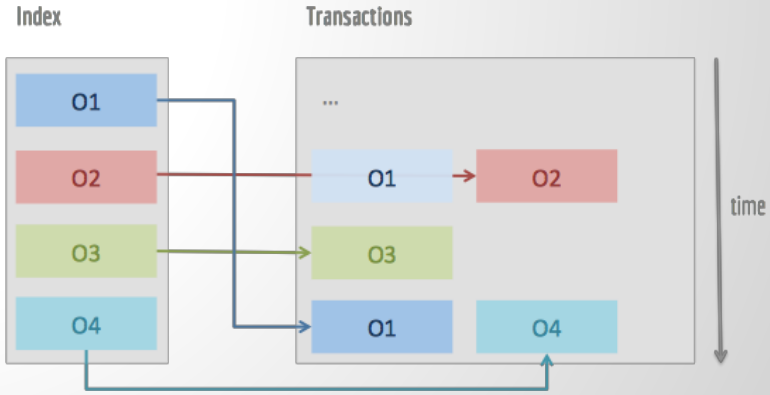
"Glorified pickle store"

Tightly integrated with Python

Object-oriented

Transactional

ACID - no surprises



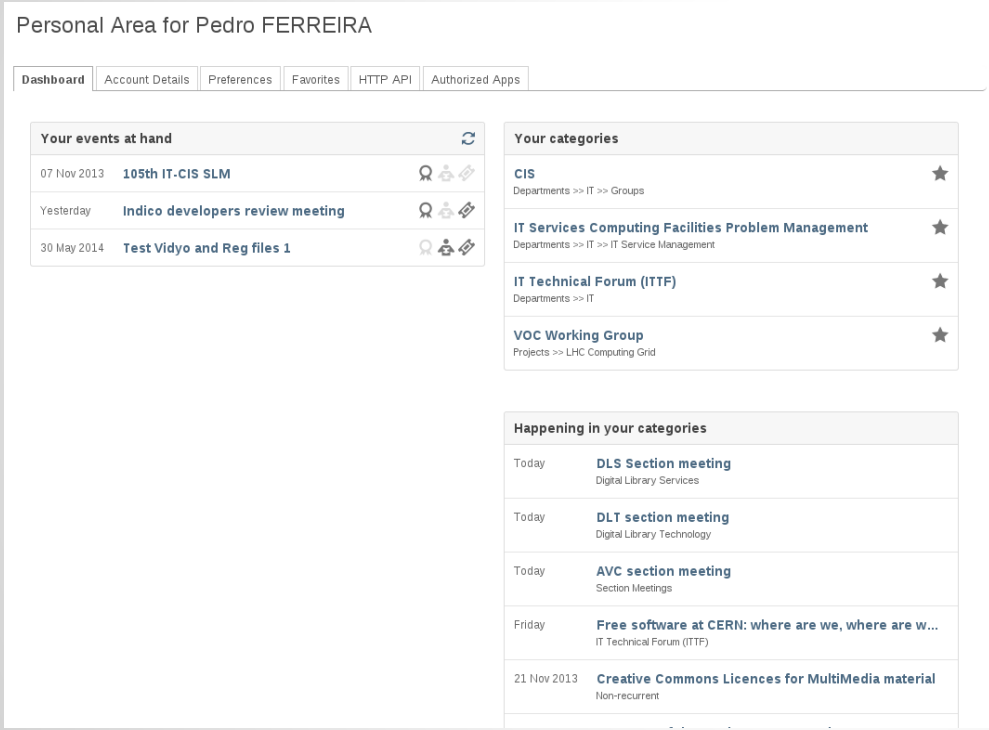
Why change?

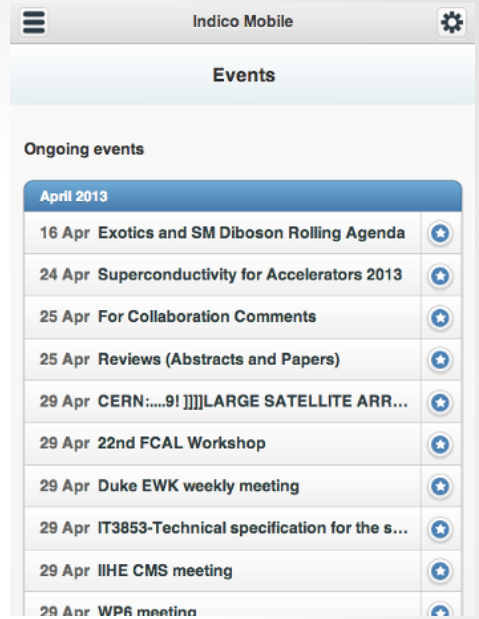
ZODB has no server-side queries

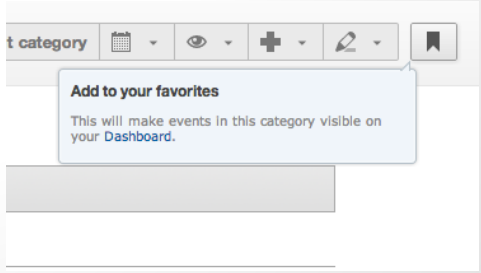
No built-in indexing

Getting data out of it = slow

No way of fetching > 1 object at once


Personal Area for Pedro FERREIRA
Dashboard
Account Details
Preferences
Favorites
HTTP API
Authorized Apps
Your events at hand
07 Nov 2013 105th IT-CIS SLM
Yesterday Indico developers review meeting
30 May 2014 Test Vidyo and Reg files 1
Your categories
CIS
Departments >> IT >> Groups
IT Services Computing Facilities Problem Management
Departments >> IT >> IT Service Management
IT Technical Forum (ITTF)
Departments >> IT
VOC Working Group
Projects >> LHC Computing Grid
Happening in your categories
Today DLS Section meeting
Digital Library Services
Today DLT section meeting
Digital Library Technology
Today AVC section meeting
Section Meetings
Friday Free software at CERN: where are we, where are w...
IT Technical Forum (ITTF)
21 Nov 2013 Creative Commons Licences for MultiMedia material
Non-recurrent


Indico Mobile
Events
Ongoing events
April 2013
16 Apr Exotics and SM Diboson Rolling Agenda
24 Apr Superconductivity for Accelerators 2013
25 Apr For Collaboration Comments
25 Apr Reviews (Abstracts and Papers)
29 Apr CERN:....9!]]]]LARGE SATELLITE ARR...
29 Apr 22nd FCAL Workshop
29 Apr Duke EWK weekly meeting
29 Apr IT3853-Technical specification for the s...
29 Apr IIHE CMS meeting
29 Apr WP6 meeting


t category
Add to your favorites
This will make events in this category visible on your Dashboard.

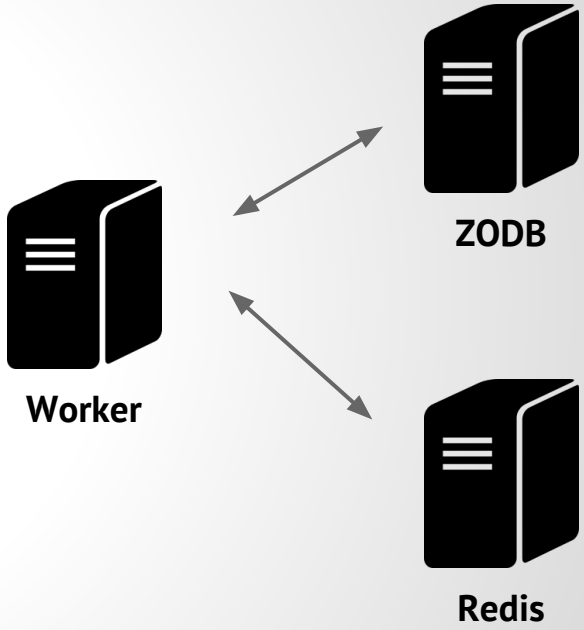
Last time we spoke...

We were relying on workarounds.

Workaround

Example: Dashboard

- Data mirrored in Redis
- Structured for querying
- It works!



Workaround

Example: Dashboard

- 7 LUA scripts for server-side queries
- Synchronizing updates on both storages
- Redundant data, code
- **Hard to maintain!**

Last time we spoke...

We were in a dilemma...

The problem

Working around ZODB’s weaknesses

Development time ⇢ DB

Risking data inconsistency

Shouldn’t things be easier?

Plus...

ZODB has to be packed regularly

No caching on server side (only OS)

Replication not that easy *

Niche project

Last time we spoke...

We were in the middle of a careful analysis.

The Quest for the Holy Grail

Criteria

Availability (OSS)

Scalability / Replication

Easiness of use / development

Transactions / Consistency

Community / Momentum

Costs / Exit Strategy

The Contestants

- The relational suspects
  - MySQL and forks
  - PostgreSQL
- The NoSQL crowd
  - Key-value
  - Document-oriented
  - Column-oriented
  - Graph databases

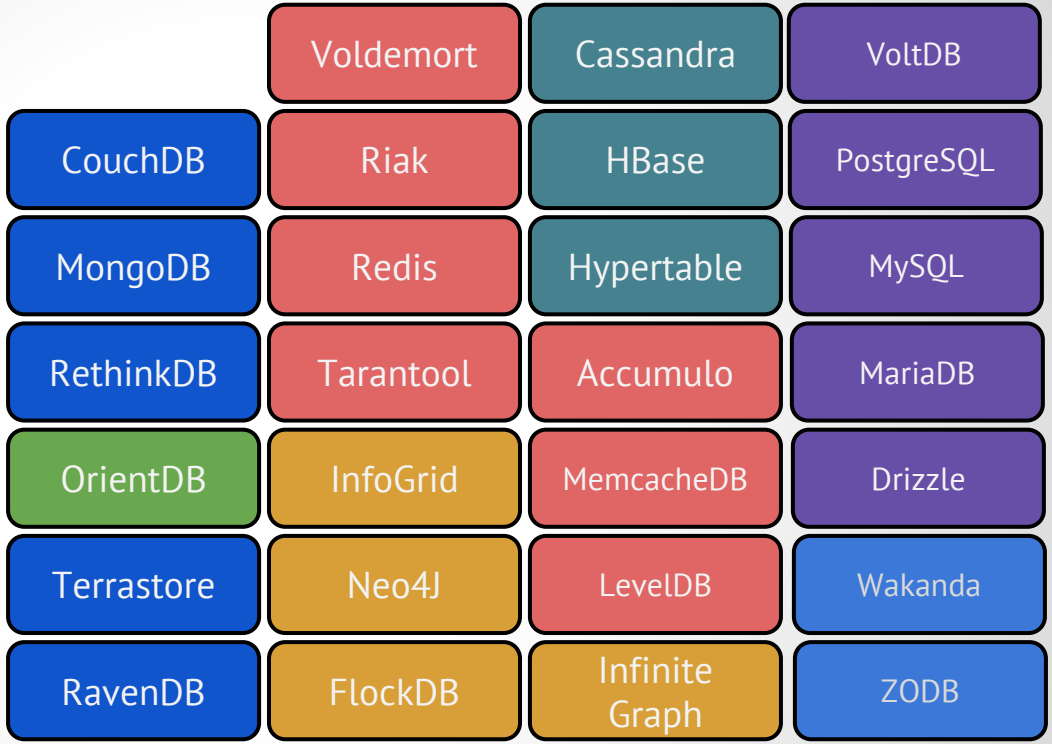
Not so simple...

A single DB?

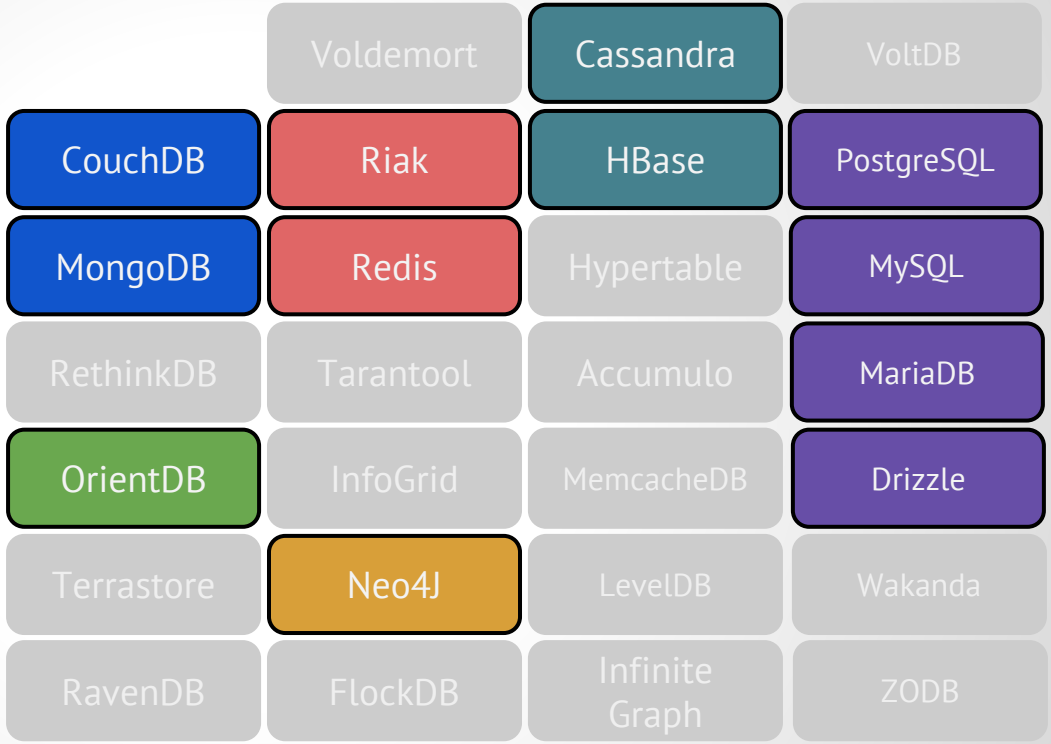
Primary + Secondary DB?

DB + caching?

Narrowing down



Community & Project Activity



Key-value stores

Tunable - in-memory and persistent

Fast, minimalistic

As simple as it can get

Lack of complex data structures *

Namespacing is hard (no tables!)

Values are strings, no data types

Column-Oriented DBs

Closer to the relational model

Faster range queries

Highly distributable, scalable

Eventually consistent

No transactions

Graph-Oriented DBs

ACID compliant

Schema fits nicely to Indico's core concepts

Not everything in Indico is graph-like

Sharding is hard/impossible

Niche-oriented

Document-Oriented DBs

Simple and intuitive

Data is JSON-like

Super-fast querying capabilities

JOINs done client-side

No atomicity (transactions)

Relational DBs

Solid and mature

Arbitrary queries

Fully ACID

Replication out of the box

Schema has to be normalized (less natural)

SQL or ORM needed

Back to focus

©SouthernSass

Key-value stores

Can’t be used by themselves

- Need to work around limitations
- Conceived mostly for caching

Can be very useful as caches

- Current usage in Indico

Column-Oriented DBs

Could be used in particular contexts

- Super-large distributed Indico instance

Killing a mosquito with a nuclear warhead?

- Unnecessary complexity is introduced
- Non-trivial infrastructure

Graph-Oriented DBs

Could be a great way of extracting useful data

- Meetings, Speakers, Interests…?

Not exactly general-purpose

- Useful in a subset of the application (Indico)
- Would introduce complexity

Document-Oriented DBs

Store data in a very natural way

- Indico events, users, etc…
- Closer to what ZODB does
- Good for simple cases such as Indico Mobile's

Tend to lead to redundant data

- Since JOINs are expensive (client-side)

Introduce complexity that is not needed

- e.g. compensating for lack of transactions
- For a performance gain we probably don't need

Relational DBs

They fit Indico’s highly-transactional nature

- ACID, no surprises

In widespread use

- Large user communities (MySQL, PostgreSQL)
- Library availability and know-how

Need for an intermediate layer

- Object-Relational Mapping

Our solution

RDBMS (PostgreSQL)

SQLAlchemy + Flask-sqlalchemy

Redis strictly for caching and web sessions

Scaling according to needs

No over-engineering

Easiness of Use

SQLAlchemy

Community

PostgreSQL

Exit Strategy

Migrating

The Plan

Gradual migration over ~1 year period

Migrating module by module

No external releases in that period

CERN will be the testbed

We'll take care of it, don't worry!


Divide
&
Conquer

Strategy

“Divide and Conquer”

- Room Booking
- Collaboration
- Other plugins
- User data
- ….

A modular Indico would for sure help

Problems

Legacy code will have to co-exist with new code

Need to account for complexity of 2 DBs

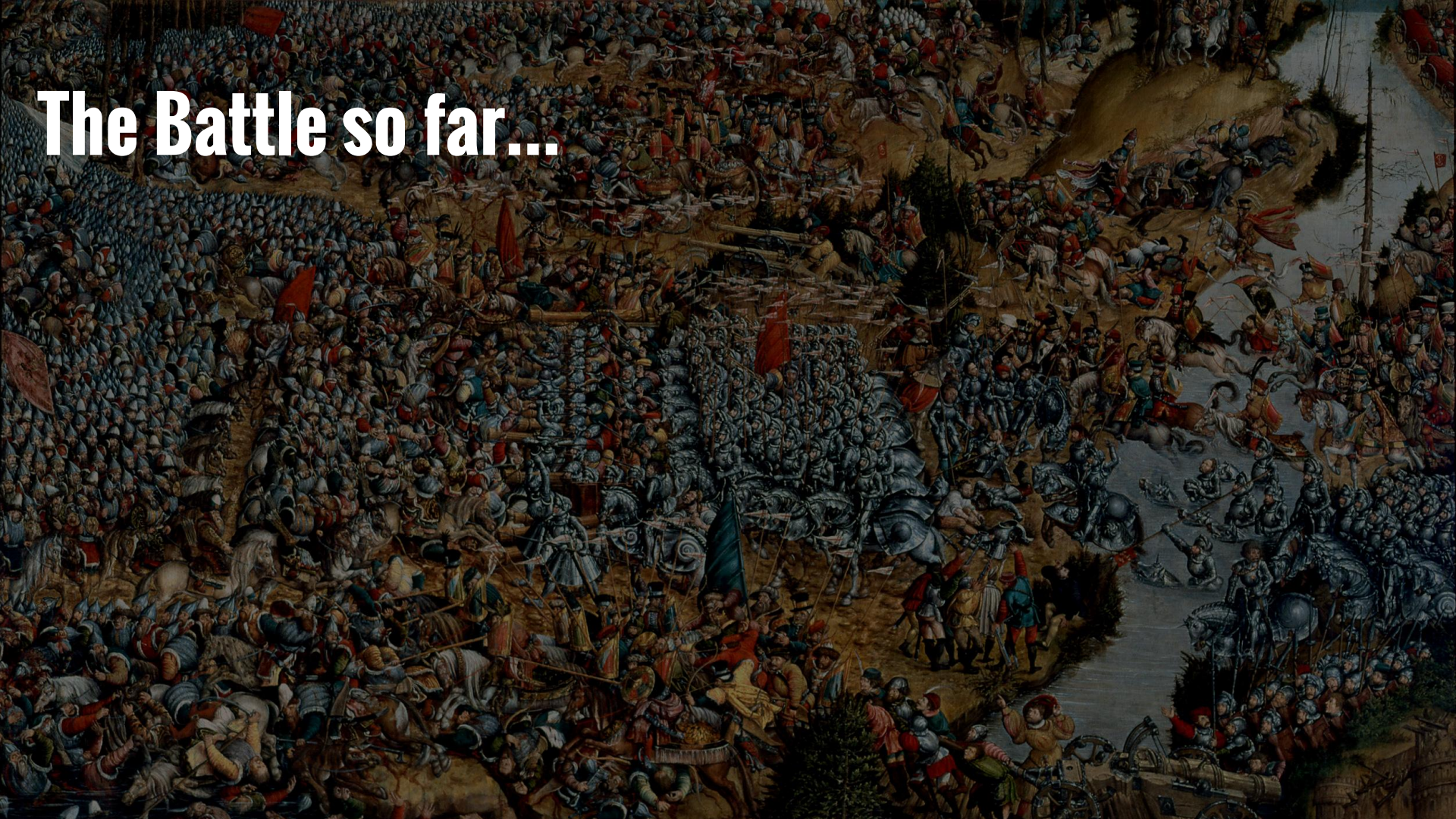
There are risks

Trying it out

Our Guinea Pig: **Room Booking Module**

- DB is independent from rest of Indico
- No cross-DB references (other than IDs)
- Not too big, not too small
- Extrapolating results

The Battle so far...



Our “army”

1 person, full time

Weapons:

```
SQLAlchemy
Flask-SQLAlchemy
zope.sqlalchemy
alembic
```

```
Flask-Testing
Fixture
sqlalchemy_schemadisplay
flask-migrate
```

Regular stand-up meetings (schema, decisions)

The enemy

Lack of Modularity

Deeply nested objects (normalization)

- Complex queries

Conforming to a strict schema

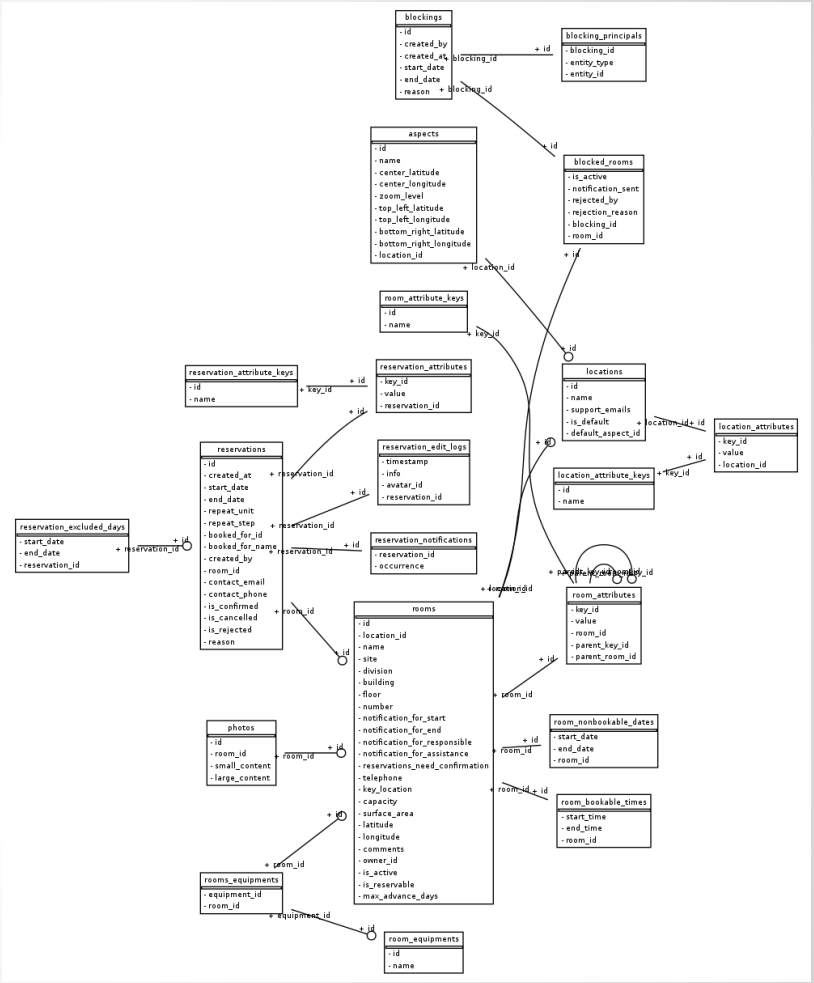
- No custom attributes / no rich data structures

“Temptation to refactor” *

Over-engineering

Room Booking Schema

- Location
  - Map
  - Room
    - Reservation
      - Attributes
      - Notifications
      - Repeats
    - Blocking



Status Quo

~7 weeks of useful work

Schema design & impl. - done

Test infrastructure - ready

Migration script - ready

Front-end

- Location management - done
- Room management - ongoing
- Room booking / booking management - tbd

Estimates

First full-working *alpha* by the end of February

Room Booking: 10-11 weeks of work @ 1 FTE

What about the rest?

Things to consider

The boilerplate is in place

A lot of preparatory work has been done

Know-how has been acquired

Allocated resources will increase from March on

But also...

2 people don’t necessarily do twice the work

Interdependency of tasks

Unpredictability of some rewriting costs

Risks are there (low impact, though)

Conclusions

No ideal scenarios, no silver bullets

Conclusions

Relational seems to provide the best balance

Room Booking experiment - a success so far

If everything goes as expected…

Indico 2.0 world-wide release 1H 2015

Questions?

Thank you!