HL7® FHIR® DevDays 2020

HL7 FHIR

FHIR or Relational Model for Storing Data

Grahame Grieve, HL7

HL7 FHIR DevDays 2020, Virtual Edition US, June 15–18, 2020 | @HL7 @FirelyTeam | #fhirdevdays | www.devdays.com/us

Should I store FHIR resources as FHIR resources?

- This is a quite common implementer question
- Doesn’t have a single/simple right answer

Understanding HL7's Mission

- HL7 functions like a vendor association
- Vendors are free to develop their applications as they see fit
- HL7 standardises the interfaces between applications
- HL7 standards say nothing about what happens inside the application
- HL7 standards say everything about what happens inside the application

HL7® International

FHIR is for interfaces

- FHIR is a API
- Applications can do whatever they want to provide the API
- The resources are defined for robust exchange
  - E.g. denormalised to reduce dependency to other exchanges (that might not have happened)
  - Database normalisation principles

HL7® International

Example

- Robust Exchange:

```
<MedicationRequest
xmlns="http://hl7.org/fhir">
  <code>
   <coding>
     <system value="http://www.nlm.."/>
     <value value=""/>
   </coding>
  </code>
</MedicationRequest>
```

- Normalised:

```
<MedicationRequest
xmlns="http://hl7.org/fhir">
  <code key="123431232"/>
</MedicationRequest>
```

Behind the Curtain

- Applications can choose:
- Store data in a relational database
  - Implement the API as a series of adaptors to the relational database
  - *Denormalise in advance*
- Store data as FHIR resources directly
  - As blobs/documents (JSON/BSON) + generic REST implementation
  - *Denormalise on the fly*
- Applications can mix/match (do both, do either)
  - Non-functional requirements (from FHIR pov) – engineering concern

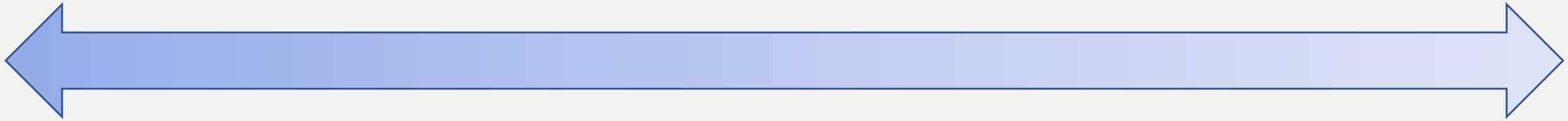
Storing resources natively

Some of what I've seen:

- Storing resources as blobs in SQL database
- Using classic SQL servers with JSON support
- NoSQL servers like Mongodb / Couch etc (or Hadoop)
- Google's Big Query
- Some RDF based store using the RDF in the turtle format

- Just because you can doesn't mean that you should…

Understanding Data Requirements



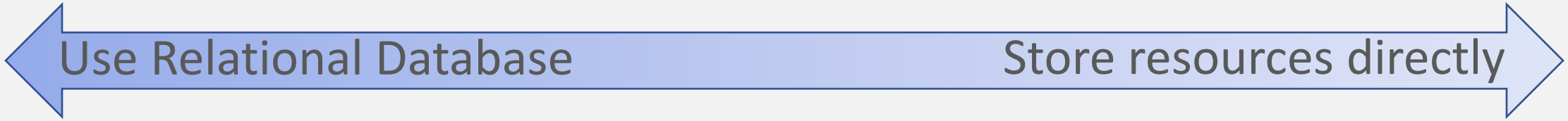
- Highly characterised requirements
- Changes by process/committee
- Requirements -> Product Development
- Changes take years to implement

- E.g. Classic Enterprise IS

- Data design delegated to run time
- Interface specifications change
- No product development (delegated to configuration)
- Must change – days / weeks

- E.g. Classic Data Repository

Choosing whether to store FHIR Resources



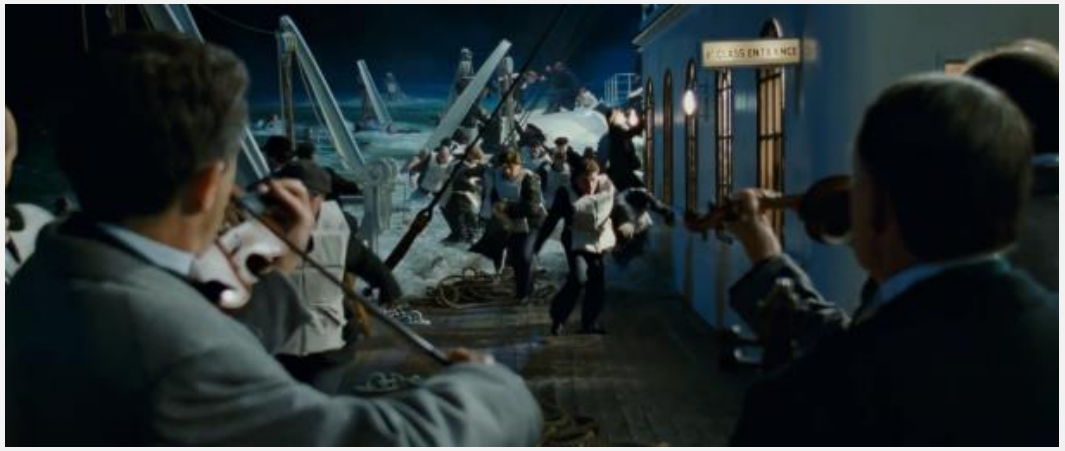
- Most Applications are somewhere in the middle
  - May vary by module/function
- Many applications would benefit from a hybrid approach
  - Low value / static information delegated to resources
  - High value managed/indexed information + analytical denormalizations in relational database

Managing Joins

- If you do store resources natively, you want manage the joins
  - Ensure no broken links (if possible)
  - Resolve more efficiently
- Both references and codes
- The RDF format does this (http://hl7.org/fhir/rdf.html):
  - fhir:reference – resolved local link for Reference
  - fhir:concept – resolved local link for Coding
  - Adopt similar approaches in your own native store

Should I store FHIR resources as FHIR resources?

- That depends! On:
  - Your requirements
  - Your business context/plans
  - Your technology stack + constraints
- But consider storing resources natively and storing information in an atomic/indexed form
- Your pain will always be around managing joins and referential integrity (deck chairs on the titanic)



ORGANIZED BY

PARTNER

HOST SPONSOR