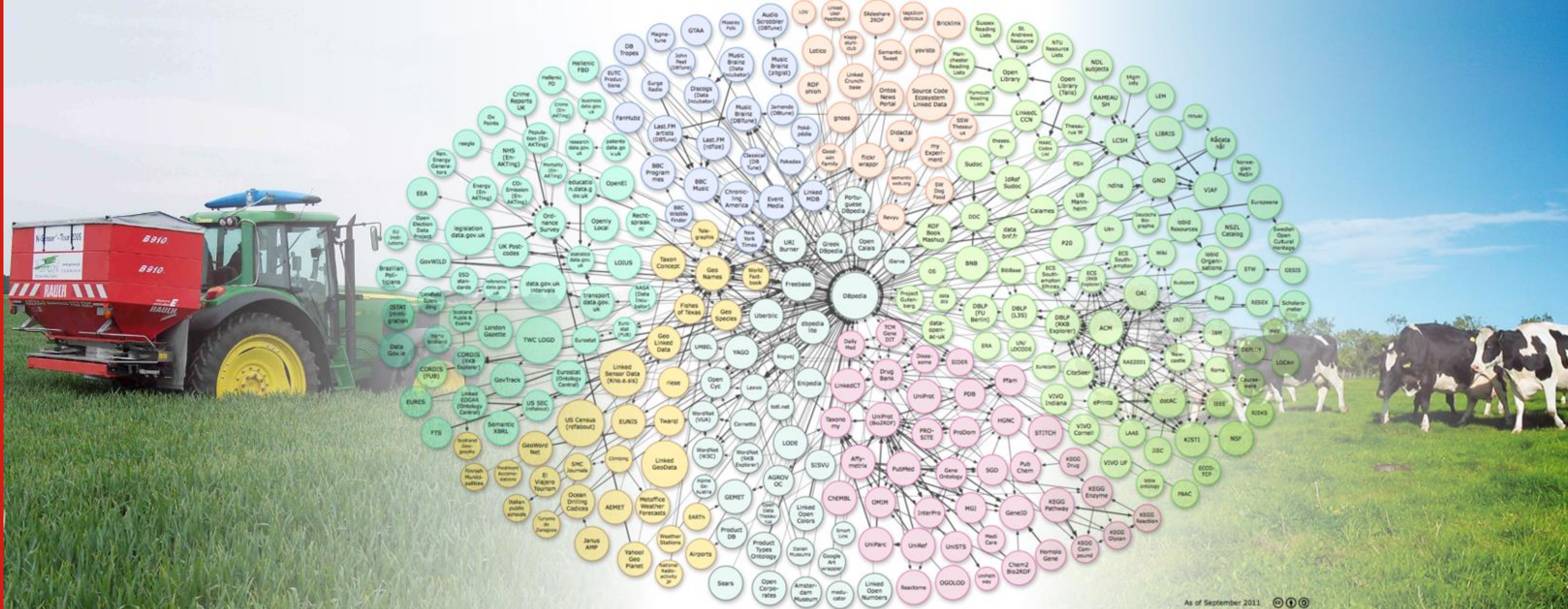


Linked Data architectural components

How-to attach linked data services to legacy infrastructure?

Daniel Martini, Mario Schmitz, Günter Engelhardt

Berlin, 27th of June, 2017

Organization

- Registered Association (non-profit):
  - Funded ~ 2/3 by the german ministry for nutrition and agriculture
  - ~ 400 members: experts from research, industry, extension…
  - ~ 70 employees working in Darmstadt
  - Managing lots of working groups, organizing expert workshops, represented in other committees, maintaining an expert network
- Tasks:
  - Knowledge transfer from research into agricultural practice
  - Supporting policy decision making by expertises
  - Evaluating new technologies: economics, ecological impact…
  - Providing planning data (investment, production processes…) to extension and farmers
- Role of Information Technology:
  - Data acquisition: harvesting open data sources
  - Data processing: calculating planning data from raw data
  - Information provision: delivery to clients via ebooks, web, apps

Goals and requirements

Deliver KTBL planning data in human and machine readable form alike:

- Machine classes: purchase prices, useful life, consumption of supplies…
- Standard field work processes: working time, machines commonly used under different regimes…
- Operating supplies: average prices, contents…
- Facilities and buildings: stables, milking machines and their properties
- …

to reach a broader audience and enable further processing within software applications for the use of farmers, extension…

Problem statement

- There's data that wants to get shared available at an organization
- We want to comply to FAIR principles:
  - Findable
  - Accessible
  - Interoperable
  - Reusable

➔ So we *have to* use standard specifications:
  - RDF
  - HTTP
  - SPARQL
  - ...

- But alas, data exists within a legacy infrastructure
- What's in our toolbox to get it unlocked with the least effort possible?

Graph based data model

- Every data structure can be converted to a directed graph with relative ease
- Extensions can flexibly be implemented

Resource Description Framework (RDF):

Subject	Predicate	Object
FarmerXY	owns	Machine0815
Machine0815	type	tractor
Maschine0815	purchasePrice	83000 Euro

→ Rich representation

→ Advantages when it comes to search, navigation and decision support

"A traditional relational database may tell you the average age of everyone in this pub, but a graph database will tell you who is most likely to buy you a beer." *Andreas Kollegger*

Step 1: Create vocabulary
Most important: reuse

- No name properties. Recommendation:
  "rdfs:label is an instance of rdf:Property that may be used to provide a human-readable version of a resource's name."
  http://www.w3.org/TR/rdf-schema/
- Persons, addresses, phone numbers:
  - vcard: http://www.w3.org/2006/vcard/ns#
  - foaf: http://xmlns.com/foaf/0.1/
- Units and dimensions:
  - QUDT: http://qudt.org
- Geospatial data:
  - Geovocabulary: http://geovocab.org/
  - GeoSPARQL: http://www.opengeospatial.org/standards/geosparql
- Prices, Products, etc.:
  - Good Relations Ontology:
    http://www.heppnetz.de/projects/goodrelations/

Problems and solution approaches

The RDF data model does not support n-ary relations

Representation of physical quantities requires n-ary relations:
value and dimension form an unseparable unit

- It gets worse, if the „what“ needs to be represented as well: „consumption 7.8 l diesel per h“, „fat content 35 g/l of milk“
- Three approaches to solving the problem:
  1. *data types*
     advantage: compact notation
     disadvantage: deprives you of the usage of XML schema data types (e. g. xsd:float) on the numerical value of the quantity
  2. *additonal ressource nodes in the graph*
     advantage: simplifies reasoning
     disadvantage: difficult to represent properly in services
  3. *blank nodes*
     advantage: compact notation in different syntaxes, intuitive, possibility to add further datatypes to values
     disadvantage: not that easy to handle in reasoning

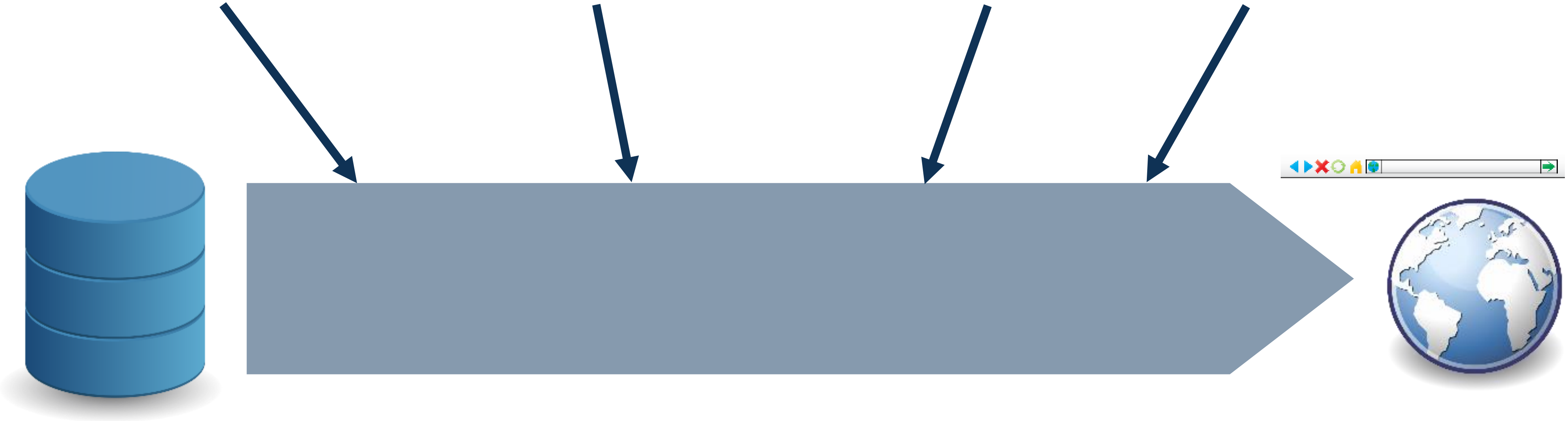
Infrastructure: What‘s needed?

pull data out of a relational data base

efficiently query a graph based data model

serialize in semantic web formats

service HTTP requests

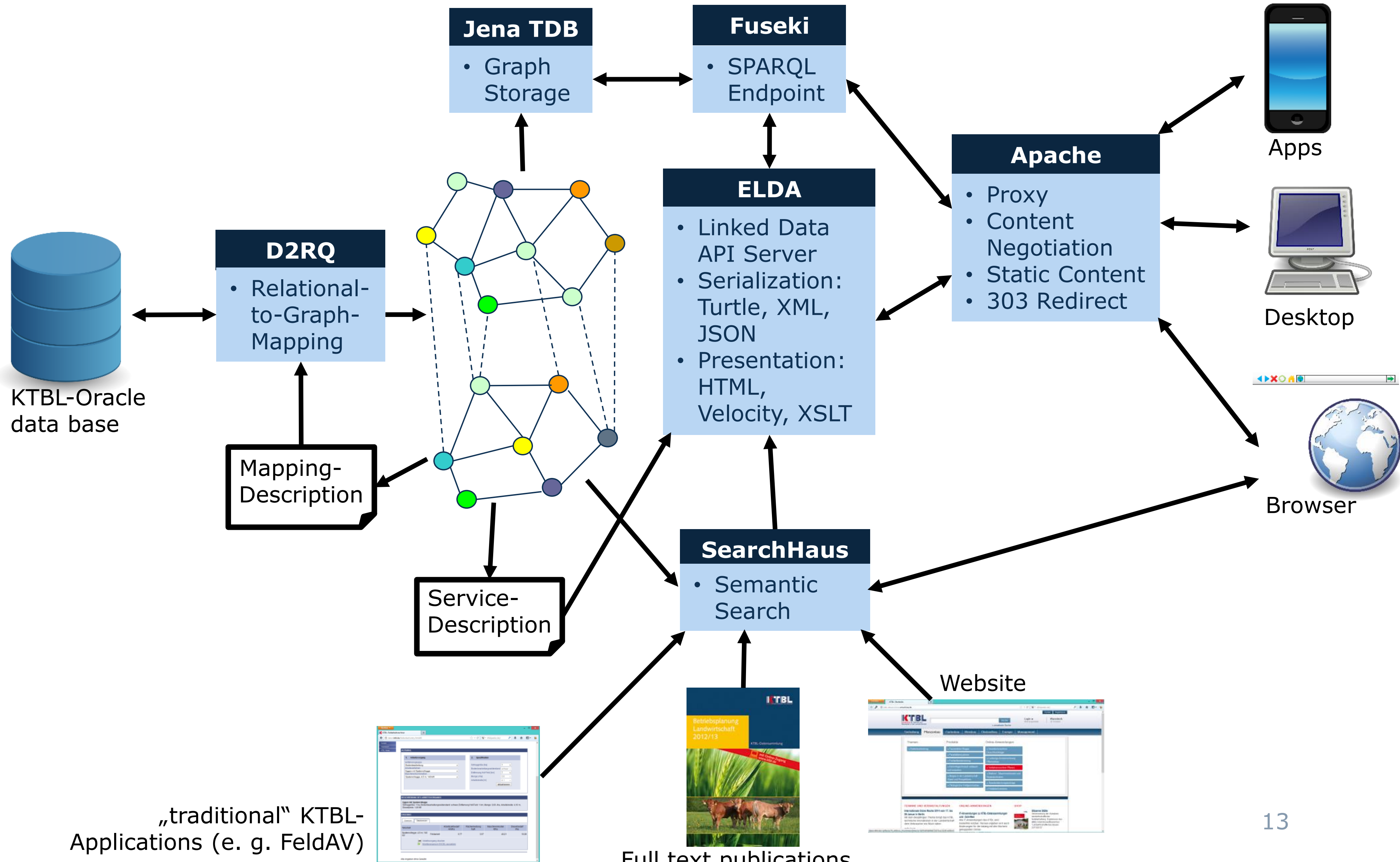


- Relational-to-graph/RDF mapping tool
- Triple/quad store, SPARQL query engine
- Serializer/Linked data server component

Evaluation

Component	Alternatives	Decision
Relational-to-graph/RDF mapping tool	DB2triples Virtuoso R2RML Parser Xsparql Karma …	**D2RQ:** - Supports Oracle data bases via JDBC - Experience was available from a project
Triple/quad store, SPARQL query engine	Sesame Stardog 4Store Owlim …	**Jena Fuseki:** - Easy to use and configure - (relatively) lightweight
Serializer/ linked data server component	D2R server Pubby Callimachus Apache Marmotta Virtuoso …	**ELDA:** - Supports different serialization formats - Allows adjustment of the HTML layout via velocity templates

Architecture



ELDA

Website: http://www.epimorphics.com/web/tools/elda.html

Source code: https://github.com/epimorphics/elda

- used e. g. by data.gov.co.uk
- an implementation of the Linked Data API as specified at: https://github.com/UKGovLD/linked-data-api
- using the Apache velocity template engine http://velocity.apache.org
- one template for the whole server: templates can become rather complex, if you want to do path specific rendering or localization
- no native content negotiation: that requires Apache upfront

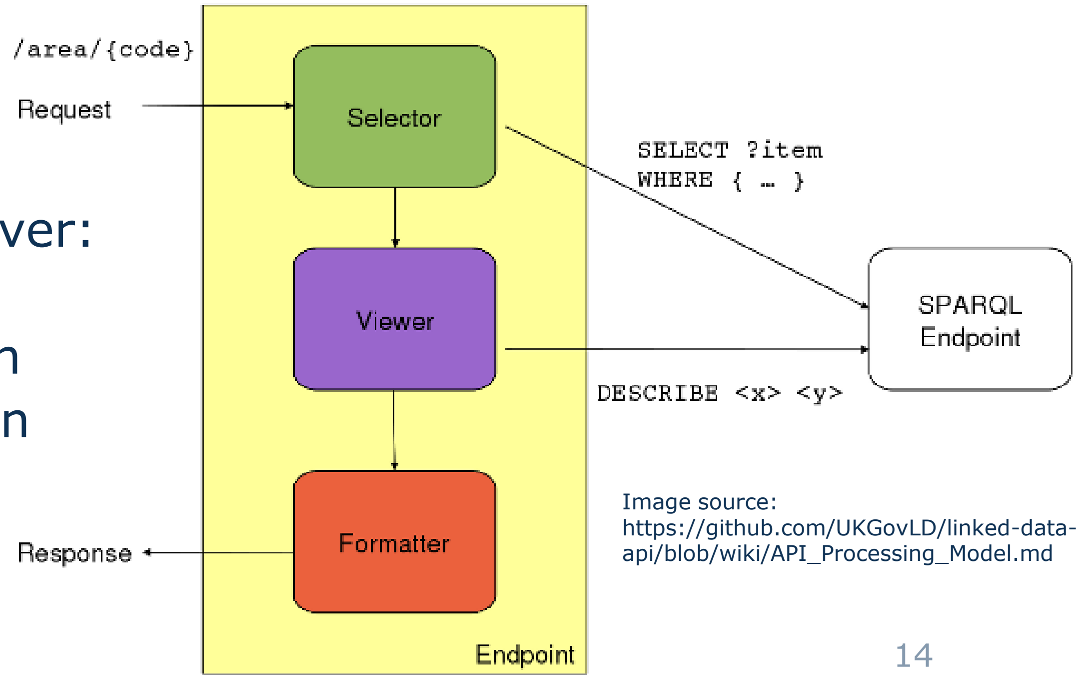


Image source: https://github.com/UKGovLD/linked-data-api/blob/wiki/API_Processing_Model.md

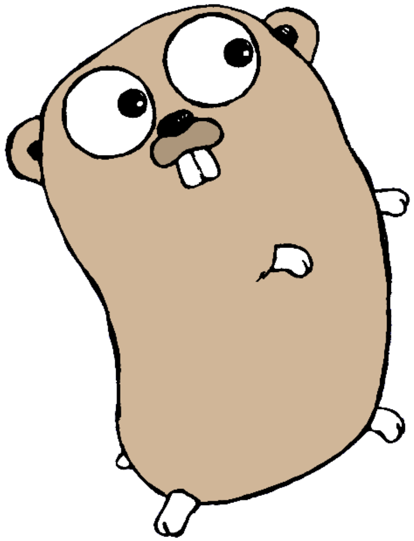
Next **G**eneration **D**ata **A**PI in **G**o

- Allow for differing HTML Renderings and SPARQL backend queries depending upon URL path requested and Accept*-headers:
  - template driven HTML frontend
  - SPARQL query templates with variable expansion
  - Each URL path can have its own HTML as well as SPARQL templates
- Content-Negotiation (HTTP Accept: header + filename suffix)
- Frontend Localization Support (HTTP Accept-Language: header + LDA _lang parameter)
- Support most of the additional query parameters in the LDA spec
- Replace LDA spec JSON compliant serialization by JSON-LD as specified by the recent W3C recommendation

Why Go?

http://golang.org

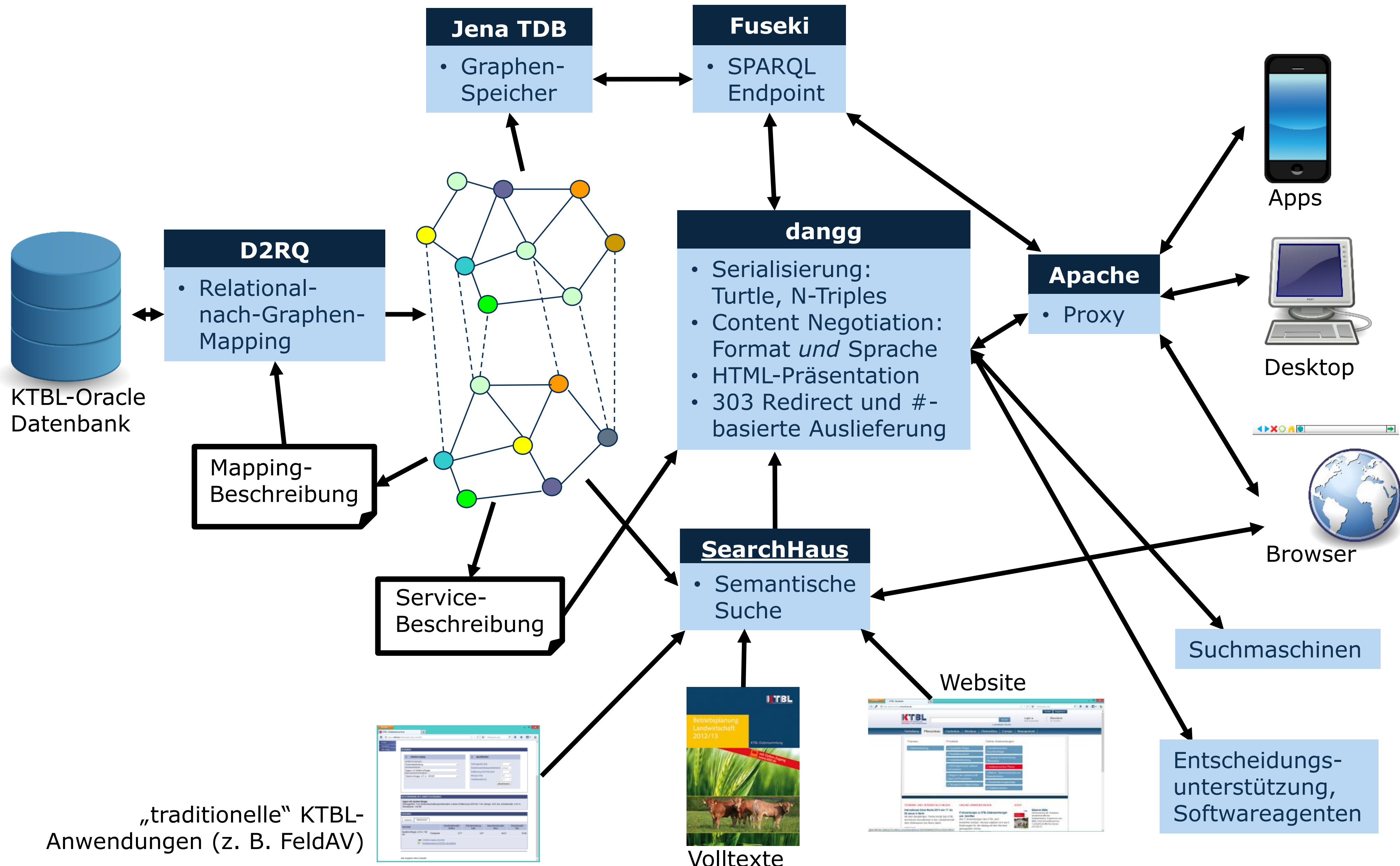
Created by some Google Engineers and former AT&T/Bell Labs Unix System Laboratories employees around 2009: Rob Pike, Robert Griesemer, Ken Thompson
Inspired by their former work at Bell Labs: Plan9



Designed by Renee French
http://reneefrench.blogspot.com/
licensed under the Creative Commons 3.0 Attributions license

- Features:
  - The best of three worlds: Python, C/C++, Java
  - Compiled language with a clean, portable compiler design
  - Consistent syntax
  - Easy to use build and packaging framework included
  - Adjusted to modern hardware architectures: concurrency, networking
  - Performant (~ C++)
  - Non-object-oriented, but has interfaces and methods
  - Static typing, pointers but no pointer arithmetic, function closures…
- Used in some high profile, large-scale projects:
  - Soundcloud's Prometheus monitoring system: http://prometheus.io
  - Google's download server: http://dl.google.com serving Chrome, Android SDK, Earth… downloads

Architektur LOD-Service am KTBL



Dangg: features so far

- Done:
  - Content Negotiation
  - Per-Endpoint-Templates: SPARQL and HTML (standard go template engine: https://golang.org/pkg/html/template/)
- Not yet:
  - JSON-LD
  - IP-based logging
  - Configuration files
  - Complete LDA/LDP support
- In-memory label processing:
  - Speed (avg. 8 ms page load time -> huge improvement vs. ELDA)
  - Might require redesign with datasets with lots of labels
- ~2000 SLOC

Dangg: Core data structure

- „Item“ struct:
  - Either a Subject/Object or a Predicate in a RDF triple
- Item struct fields:
  - P (**P**arent Node)
  - T (Node **T**ype at parse time: subj/obj or pred)
  - L (human readable **L**abel, filled from in-memory map)
  - U (**U**RL)
  - V (**V**alue: only filled for literals)
  - D (**D**imension: only filled for physical quantities, requires units to be represented as blank nodes)
  - N (**N**ext Level of Items)
- All fields referencable from HTML templates
- Can feed any RDF data to it, as long as units are represented using the blank node strategy

Linked Data: Entitätenansicht

☰ MENÜ

? HILFE   i INFORMATIONEN   ✉ KONTAKT

DUNGZANGE FÜR FRONTLADER - 1,9 M³ FÜR FRONTLADER AN 138-KW-TRAKTOR

Eigenschaften:	
Typ	Ladewerkzeug, Maschinenklasse,
Anschaffungspreis	3050 EUR
Nutzungsdauer	10 a
Nutzungspotenzial nach Massendurchsatz	57000 t
Reparaturkosten nach Masse	0.01 EUR/t
Abstellmaß: Breite	2600 mm
Abstellmaß: Höhe	950 mm
Abstellmaß: Länge	1000 mm

RDF-Daten herunterladen im: Turtle Format / N-Triples Format / JSON Format

Diese Seite drucken

Conclusions

- Free tools, not too difficult to setup are available
  - Usually, the problem exists between keyboard and chair
  - There are rough edges
  - Replacing certain components by own code is doable, when you are fluent in graph based data models

Alternatives:

1. Buy an all in-one-solution with a service contract
2. Program each and every data service from scratch

Thanks for listening!

Questions?

Contact: d.martini@ktbl.de