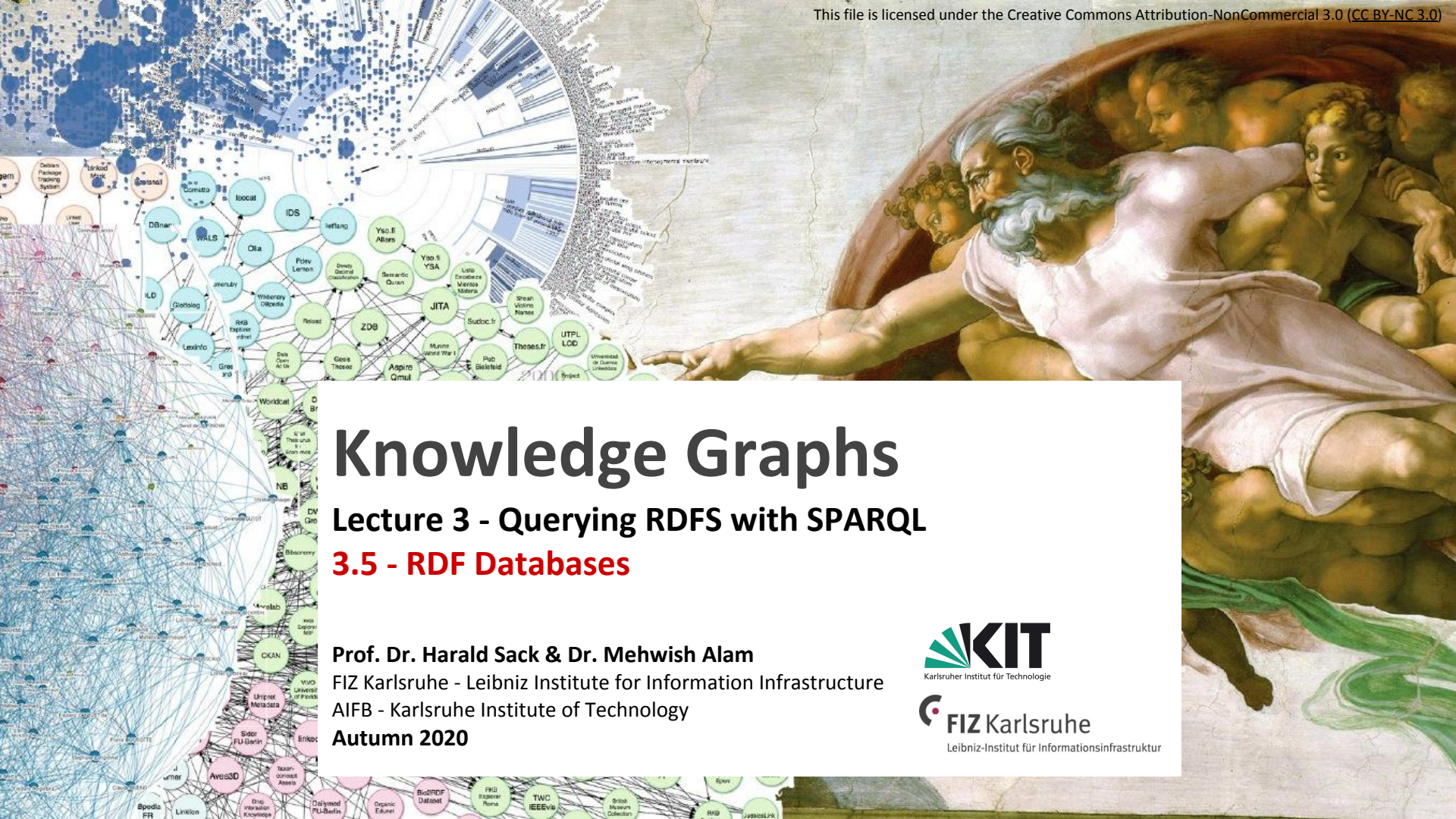


Knowledge Graphs

Lecture 3 - Querying RDFS with SPARQL

3.5 - RDF Databases

Prof. Dr. Harald Sack & Dr. Mehwish Alam
FIZ Karlsruhe - Leibniz Institute for Information Infrastructure
AIFB - Karlsruhe Institute of Technology
Autumn 2020

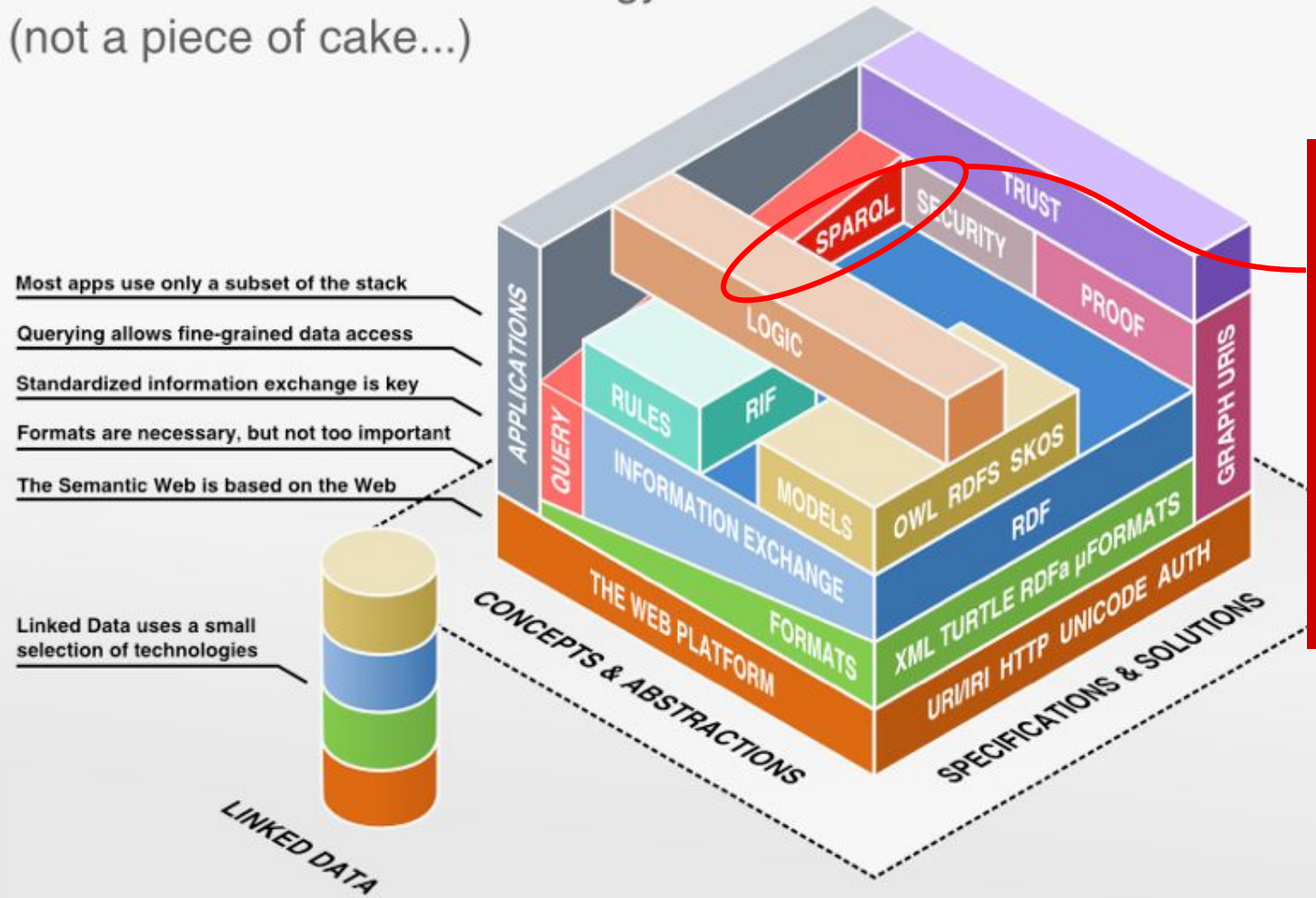
Karlsruher Institut für Technologie

FIZ Karlsruhe
Leibniz-Institut für Informationsinfrastruktur

Knowledge Graphs

Lecture 3: Querying RDF(S) with SPARQL

The Semantic Web Technology Stack
(not a piece of cake...)



USE **SPARQL** and QUERY ON

[1,2]

[3]

RDF in RDBMS

- RDF Graph can be represented by a set of Triples (s,p,o).
- Triples can simply be stored in a **relational database management system** (RDBMS).
- **Motivation**: Use a specific relational schema for RDF data and benefit from 40 years of research in the DB community.
- **3 steps for SPARQL query processing**:

  (1) Convert SPARQL query to SQL query (w.r.t. the schema).

  (2) Use RDBMS to answer SQL query.

  (3) Generate SPARQL query result from SQL query result.

RDF in RDBMS - The Problem

- How to store RDF triples to carry out efficient SPARQL queries?

- 4 alternatives:
  (1) Monolithic Triple Storage
  (2) Property Tables
  (3) Vertically Partitioned Tables (Binary Tables)
  (4) Hexastore

Monolithic Triple Storage

Subj	Property	Obj
ID1	type	FullProfessor
ID1	teacherOf	'AI'
ID1	bachelorFrom	'MIT'
ID1	mastersFrom	'Cambridge'
ID1	phdFrom	'Yale'
ID2	type	AssocProfessor
ID2	worksFor	'MIT'
ID2	teacherOf	'DataBases'
ID2	bachelorsFrom	'Yale'
ID2	phdFrom	'Stanford'
ID3	type	GradStudent
ID3	advisor	ID2
ID3	teachingAssist	'AI'
ID3	bachelorsFrom	'Stanford'
ID3	mastersFrom	'Princeton'
ID4	type	GradStudent
ID4	advisor	ID1
ID4	takesCourse	'DataBases'
ID4	bachelorsFrom	'Columbia'

- Basic Idea
  - Store all RDF Triples in a single table
  - Performance depends on efficient indexing
- Pros:
  - Easy to implement
  - Works for a huge number of properties, if indexes are chosen with care
- Cons:
  - Many self joins necessary

Monolithic Triple Storage

Subj	Property	Obj
ID1	type	FullProfessor
ID1	teacherOf	'AI'
ID1	bachelorFrom	'MIT'
ID1	mastersFrom	'Cambridge'
ID1	phdFrom	'Yale'
ID2	type	AssocProfessor
ID2	worksFor	'MIT'
ID2	teacherOf	'DataBases'
ID2	bachelorsFrom	'Yale'
ID2	phdFrom	'Stanford'
ID3	type	GradStudent
ID3	advisor	ID2
ID3	teachingAssist	'AI'
ID3	bachelorsFrom	'Stanford'
ID3	mastersFrom	'Princeton'
ID4	type	GradStudent
ID4	advisor	ID1
ID4	takesCourse	'DataBases'
ID4	bachelorsFrom	'Columbia'

- Basic Idea
  - Store all RDF Triples in a single table
  - Performance depends on efficient indexing

```
SELECT ?university WHERE {
   ?v rdf:type :GradStudent;
      :bachelorsFrom ?university . }
```

```
SELECT t2.o AS university
FROM triples AS t1, triples AS t2
WHERE t1.p='type' AND
      t1.o='GradStudent' AND
      t2.p='bachelorsFrom' AND
      t1.s=t2.s
```

ID Based Triple Storage

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
“AI”	5
:bachelorsFrom	6
“MIT”	7
:mastersFrom	8
“Cambridge”	9

S	P	O
1	2	3
1	4	5
1	6	7
1	8	9
1	10	11
12	13	14
12	15	7
12	4	16

- **Resource table** for indexing URIs and Literals with numerical identifier
- **RDF Triple table** uses numerical identifier for each RDF term in the dataset
- Saves space and enhances efficiency

Quad Tables

RDF Term	ID
:ID1	1
rdf:type	2
:FullProfessor	3
:teacherOf	4
“AI”	5
:bachelorsFrom	6
“MIT”	7
:mastersFrom	8
“Cambridge”	9

g	S	P	O
100	1	2	3
100	1	4	5
101	1	6	7
100	1	8	9
102	1	10	11
100	12	13	14
102	12	15	7
101	12	4	16

- Storing multiple RDF graphs
- Used for provenance, versioning, contexts, etc.

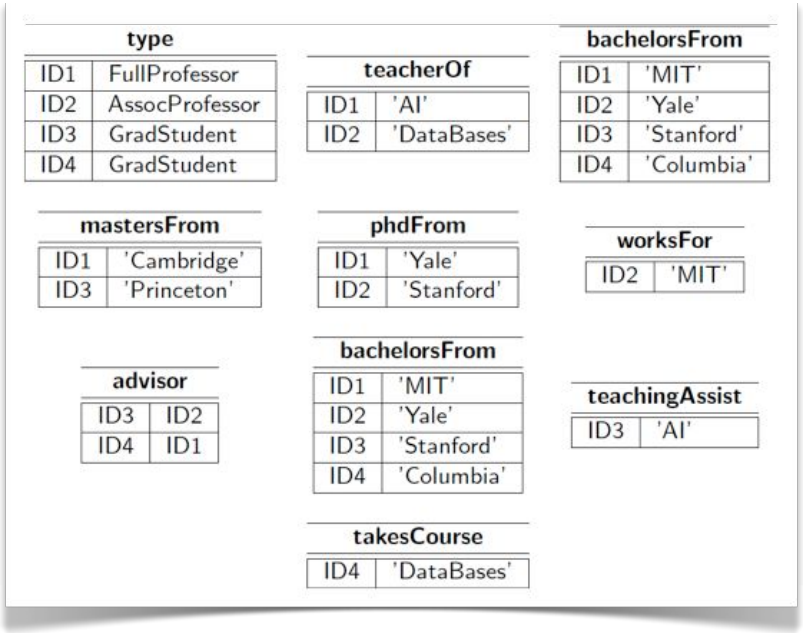
Property Tables

ID	type	teacherOf	bachelorsFrom	mastersFrom	phdFrom	worksFor
ID1	FullProfessor	“AI”	“MIT”	“Cambridge”	“Yale”	NULL
ID2	AssocProfessor	“DataBases”	“Yale”	NULL	“Stanford”	“MIT”

ID	type	advisor	bachelorsFrom	mastersFrom	teachingAssist	takesCourse
ID3	GradStudent	ID2	“Stanford”	“Princeton”	“AI”	NULL
ID4	GradStudent	ID1	“Columbia”	NULL	NULL	“DataBases”

- Combine all (or some) properties of similar subjects in n-ary tables
- Use ID based encoding for efficiency
  - **Pros:**
    - Fewer joins
    - If the data is structured, it’s a relational DB
  - **Cons:**
    - Potentially many NULLs
    - Clustering is not trivial
    - Multi-valued properties are complicated

Vertically Partitioned Tables



type

ID1	FullProfessor
ID2	AssocProfessor
ID3	GradStudent
ID4	GradStudent

teacherOf

ID1	'AI'
ID2	'DataBases'

bachelorsFrom

ID1	'MIT'
ID2	'Yale'
ID3	'Stanford'
ID4	'Columbia'

mastersFrom

ID1	'Cambridge'
ID3	'Princeton'

phdFrom

ID1	'Yale'
ID2	'Stanford'

worksFor

ID2	'MIT'

advisor

ID3	ID2
ID4	ID1

bachelorsFrom

ID1	'MIT'
ID2	'Yale'
ID3	'Stanford'
ID4	'Columbia'

teachingAssist

ID3	'AI'

takesCourse

ID4	'DataBases'

- For each unique property create a two column table
- Use ID based encoding for efficiency

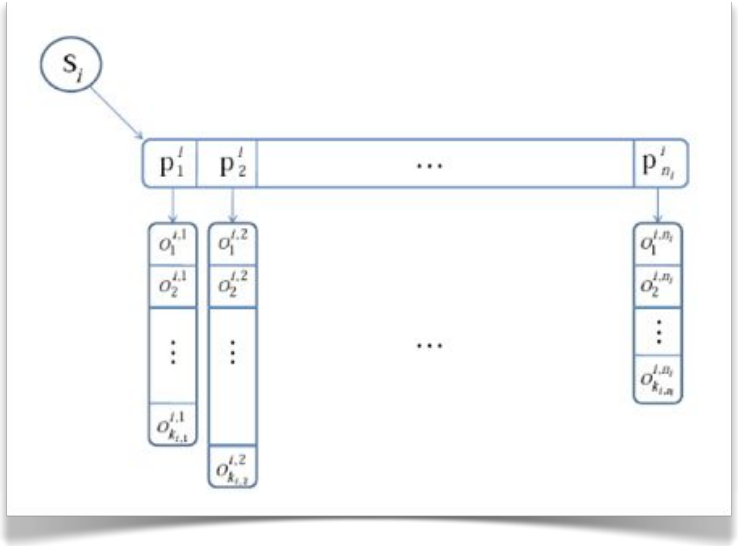
Pros:

- Supports multi-value properties
- No NULLs
- Read only needed attributes (i.e. less I/O)
- No clustering
- Excellent performance (if the number of properties is small, queries with bound properties)

Cons:

- Expensive inserts
- Bad performance (large number of properties, queries with unbound properties)

Hexastores



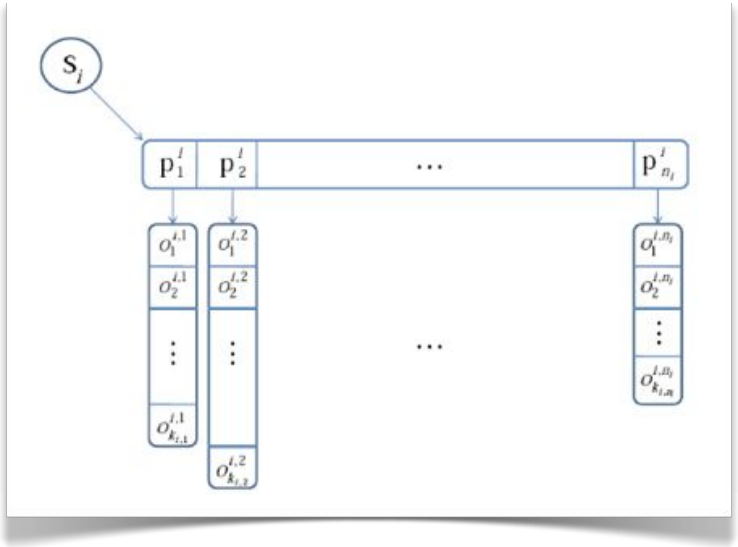
- Create an index for every possible combination to enable efficient processing
  - spo, pos, osp, sop, pso, ops

The **spo** index:

Subject key S_i points to sorted vector of n_i property keys $\{p^i_1, p^i_2, \ldots, p^i_{nj}\}$
Each property key p^i_j is linked to a sorted list of object keys.

The object key lists are shared with the **pso** index.

Hexastores



- Create an index for every possible combination to enable efficient processing
  - spo, pos, osp, sop, pso, ops

Pros:

- fast joins (in the beginning)

Cons:

- 5 times more storage
- weak performance when disk access is necessary

Some Triple Stores

- Native RDF Stores
  - Apache Jena TDB - http://jena.apache.org
  - AllegroGraph - http://www.franz.com/agraph/allegrograph/
  - GraphDB - http://ontotext.com/products/ontotext-graphdb-owlim/
  - Blazegraph - https://www.blazegraph.com/
- DBMS backed RDF Stores
  - Apache Jena SDB - http://jena.apache.org
  - Oracle Spatial and Graph: RDF Semantic Graph http://www.oracle.com/technetwork/database/options/spatialandgraph/overview/rdfsemantic-graph-1902016.html
- Hybrid RDF Stores
  - Open Link Virtuoso - http://virtuoso.openlinksw.com
  - Sesame - http://rdf4j.org/
- W3C maintains a list of triple stores: https://www.w3.org/2001/sw/wiki/Category:Triple_Store

Next Lecture...

Karlsruher Institut für Technologie
FIZ Karlsruhe
Leibniz Institute for Information Infrastructure

Picture References:

- [1] Benjamin Nowack, *The Semantic Web - Not a Piece of cake…*, at bnode.org, 2009-07-08 , [CC BY 3.0]
  http://bnode.org/blog/2009/07/08/the-semantic-web-not-a-piece-of-cake
- [2] British Crown vector illustration, publicdomainvectors.org, [Public Domain]
  https://publicdomainvectors.org/en/free-clipart/British-Crown-vector-illustration/12150.html
- [3] The New Fred Meyer on Interstate on Lombard, Lyza @ flickr, [CC BY-SA 2.0]
  https://www.flickr.com/photos/lyza/49545547
- [4] Ernst Haeckel, Kunstformen der Natur (1904), plate 61: Phaeodaria, [Public Domain]
  https://commons.wikimedia.org/wiki/File:Haeckel_Phaeodaria_61.jpg