RDF Compression

HDT

Antonio Fariña, Javier D. Fernández and Miguel A. Martinez-Prieto

3rd KEYSTONE Training School
Keyword search in Big Linked Data

23rd August 2017
General agenda

- **Session I** (09:00 - 10:30) "Basics of Compression for Big Linked Data Management"
  - Big (Linked) Semantic Data Compression: motivation & challenges
  - Compact Data Structures

- **Session II** (13:30 - 15:00) "RDF Compression"
  - RDF Compression. HDT
  - RDF Dictionaries
  - RDF Triples

- **Session III** (15:30-17:00) "Compressed RDF: Practical Uses & Hands-on"
  - Practical Uses (LOD-a-lot, RDF Archiving, etc.)
  - Hands on

images: [zurb.com](https://zurb.com)
Agenda of this session

- Motivation
  - Linked Data limitations

- HDT
  - Foundations
  - Components
  - Achievements & Challenges
The greatness of Linked Data

> 150B triples
1K-6K datasets
>557 SPARQL Endpoints

http://lod-cloud.net/
https://datahub.io
http://stats.lod2.eu/
http://sparqles.ai.wu.ac.at
But what about Web-scale queries

- Use case: Query answering
  - E.g. ask “something” about Axel Polleres?
    - First, retrieve all entities in LOD with the label “Axel Polleres”

```
select distinct ?x {
    ?x rdfs:label "Axel Polleres" .
}
```

- Solutions?
Let’s fish in our Linked Data Eco System
A) Federated Queries!!

1. Get a list of potential SPARQL Endpoints
   - datahub.io, LOV, other catalogs?
2. Query each SPARQL Endpoint

- Problems?
  - Many SPARQL Endpoints have low availability

http://sparqles.ai.wu.ac.at/
The Web of Data Eco System

A) Federated Queries!!
1. Get a list of potential SPARQL Endpoints
   - datahub.io, LOV, other catalogs?
2. Query each SPARQL Endpoint

- Problems?
  - Many SPARQL Endpoints have low availability
  - SPARQL Endpoints are usually restricted (timeout, #results)
  - Moreover, it can be tricky with complex queries (joins) due to intermediary results, delays, etc

- Linked Data Fragments is a good alternative! But it is not extensively deployed and it would require a large client.
B) Follow-your-nose

1. Follow self-descriptive IRIs and links
2. Filter the results you are interested in

- Problems?
  - You need some initial seed
    - DBpedia could be a good start
  - It’s slow (fetching many documents)
  - Where should I start for unbounded queries?
    - ?x rdfs:label "Axel Polleres"
C) Use the RDF dumps by yourself

1. Crawl de Web of Data
   - Probably start with datahub.io, LOV, other catalogs?

2. Download datasets
   - You better have some free space in your machine

3. Index the datasets locally
   - You better are patience

4. Query all datasets
   - You better are alive by then

- Problems?
  - Hugh resources!
  - + Messiness of the data
D) Access lodlaundromat.org a central repository of LD

- Problems?
  - Still you need to query 650K datasets
  - Of course the solution is not complete, but “a good approximation”
Why is this not working?

1. Distributed approach
   - Linked Data is inherently distributed
   - But we need
     - Clear guidelines, practices and tools to solve Web scale queries
     - Linked Data Fragments is a good start
Why is this not working?

2. The publication model

"the published RDF dumps are actually bulks with no structure, no design, no final user in mind. They resemble unwanted creatures whose owners are keen to be rid of them"
The problem is in the roots (Big tree = big roots)

- Publication, Exchange and Consumption of **large RDF datasets**
  - Most RDF formats (N3, XML, Turtle) are text serializations, designed for human readability (not for machines)
    - Verbose = High costs to write/exchange/parse
    - A basic offline search = (decompress)+ index the file + search

[DBpedia](http://www.dbpedia.org) 431 M.triples~ 63 GB
Header Dictionary Triples (HDT)

HDT compresses big RDF datasets while maintaining search operations

“HDT is like a duck, it may not walk, swim and fly as fast as other animals ... ... but he does everything!”
Motivation

- Lightweight Binary RDF (HDT)
  - Highly compact serialization of RDF
  - Allows fast RDF retrieval in compressed space (without prior decompression)
    - Includes internal indexes to solve basic queries with small (3%) memory footprint.
    - Very fast on basic queries (triple patterns), x 1.5 faster than Virtuoso, RDF3x.
    - Complex queries (joins) on the same scale of current solutions (Virtuoso, RDF3x).

DBpedia

431 M.triples ~ 63 GB

NT + gzip 5 GB
HDT 6.6 GB
HDT + gzip 2.7 GB

rdfhdt.org
Compress and share ready-to-consume RDF datasets

Transfer large data between servers
- mirror, partial replication

Embedded Systems & Phones
- E.g. offline query answering?

Fast –low cost- SPARQL Query Engine
- Via LDF
- HDT-Jena
- HDT-Cliopatra
HDT (Header-Dictionary-Triples) Overview

**Header**
- Metadata describing the RDF dataset

**Dictionary**
- Mapping between IDs ↔ elements in the dataset

**Triples**
- Structure of the data after the ID replacement

RDF

- Structure of the data after the ID replacement
Header

- Publication Metadata:
  - Publisher, Public Endpoint, ...

- Statistical Metadata:
  - Number of triples, subjects, entities, histograms...

- Format Metadata:
  - Description of the data structure, e.g. Triples Order.

- Additional Metadata:
  - Domain-specific.

- ... In RDF
<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://purl.org/HTD/hdt#DataType">http://purl.org/HTD/hdt#DataType</a></td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#type">http://www.w3.org/1999/02/22-rdf-syntax-ns#type</a></td>
<td><a href="http://rdfs.org/ns/void#Dataset">http://rdfs.org/ns/void#Dataset</a></td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://rdfs.org/ns/void#triples">http://rdfs.org/ns/void#triples</a></td>
<td>&quot;4789981&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://rdfs.org/ns/void#properties">http://rdfs.org/ns/void#properties</a></td>
<td>&quot;212&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://rdfs.org/ns/void#distinctSubjects">http://rdfs.org/ns/void#distinctSubjects</a></td>
<td>&quot;505751&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://rdfs.org/ns/void#distinctObjects">http://rdfs.org/ns/void#distinctObjects</a></td>
<td>&quot;1717095&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#statisticalInformation">http://purl.org/HTD/hdt#statisticalInformation</a></td>
<td>_statistics</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#publicationInformation">http://purl.org/HTD/hdt#publicationInformation</a></td>
<td>_publicationInformation</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#formatInformation">http://purl.org/HTD/hdt#formatInformation</a></td>
<td>_format</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#dictionary">http://purl.org/HTD/hdt#dictionary</a></td>
<td>_dictionary</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#triples">http://purl.org/HTD/hdt#triples</a></td>
<td>_triples</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#dictionarynumSharedSubjectObject">http://purl.org/HTD/hdt#dictionarynumSharedSubjectObject</a></td>
<td>&quot;383152&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#dictionarymapping">http://purl.org/HTD/hdt#dictionarymapping</a></td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#dictionarysizeStrings">http://purl.org/HTD/hdt#dictionarysizeStrings</a></td>
<td>&quot;22120359&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#dictionaryBlockSize">http://purl.org/HTD/hdt#dictionaryBlockSize</a></td>
<td>&quot;16&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#triplesnumTriples">http://purl.org/HTD/hdt#triplesnumTriples</a></td>
<td>&quot;4789981&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#triplesOrder">http://purl.org/HTD/hdt#triplesOrder</a></td>
<td>&quot;SPO&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#originalSize">http://purl.org/HTD/hdt#originalSize</a></td>
<td>&quot;688768256&quot;</td>
</tr>
<tr>
<td><a href="http://example.org">http://example.org</a></td>
<td><a href="http://purl.org/HTD/hdt#setSize">http://purl.org/HTD/hdt#setSize</a></td>
<td>&quot;41646252&quot;</td>
</tr>
</tbody>
</table>
Dictionary+Triples partition
Dictionary+Triples partition

1. <http://example.org/Vienna>
2. <http://example.org/Javier>
3. <http://example.org/Paul>
5. <http://example.org/Stefan>
6. ex:birthPlace
7. ex:workPlace
8. foaf:mbox
9. foaf:name
10. rdf:type
11. “jfergar@example.org”
12. “jfergar@wu.ac.at”
13. “Vienna”@en
Dictionary

- Mapping of strings to correlative IDs. \{1..n\}
- Lexicographically sorted, no duplicates.
- Prefix-Based compression in each section.
  - Efficient ID↔String operations

We’ll see soon how to optimize the representation and querying of RDF dictionaries.
Bitmap Triples Encoding

- **Bitmap Triples:**
  
  ![Diagram of Bitmap Triples]

- **We index the bitsequences to provide a SPO index**

  This is just the HDT way of representing RDF triples! We’ll see soon more representations!
Remember.... Succinct Data Structures

- Represent and index large volumes of data
- ~ Theoretical minimum space while serving efficient operations:
  - Mostly based on 3 operations:
    - Access
    - Rank
    - Select
Bit Sequence Coding

- **Bitmap Sequence.**
  - Operations in constant time
    - access(position) = Value.
    - rank(position) = “Number of ones, up to position”.
    - select(i) = “Position where the one has i occurrences”.
  - Implementation:
    - \( n + o(n) \) bits
    - Adjustable space overhead: In practice, 37.5% overhead
Bitmap Triples Encoding

- Bitmap Triples:

  - E.g. retrieve (2,5,?)
    - Find the position of the second ‘1’-bit in Bp (select)
    - Binary search on the list of predicates looking for 5
    - Note that such predicate 5 is in position 4 of Sp
    - Find the position of the four ‘1’-bit in Bo (select)
Sequence of Integers Coding

- Wavelet Tree - Compact Sequence of Integers \{0,\sigma\}.

  - Operations in $O(\log \sigma)$ time
  - access(position) = Value at position.
  - rank(entry, position) = Number of appearances of "entry" up to "position".
  - select(entry, i) = Position where "entry" has i occurrences.

- Implementation:
  - $o(n)\log(\sigma)$ bits of overhead
Bitmap Triples Encoding

- **Bitmap Triples:**

- E.g. retrieve (?,5,?)
  - Get the number of occs of 5 \( \rightarrow \text{rank}(5, n)=1 \)
  - For each occ:
    - Find the position of the occurrence \( \rightarrow \text{select}(5, \text{occ}) \)
      - e.g. pos= select (5,1) =4
    - Find the associated subject \( \rightarrow \text{rank} \text{(pos)} \) and access the objects
      - e.g. rank(4) = 1, and access the object as (1,5,?)

```
subjects
\[ S \]
\[ 1 \quad 2 \quad 3 \quad 4 \]
\[ \begin{array}{l}
1 \\
1 \\
1 \\
1 \\
\end{array} \]

Predicates:
- \( B_p \)
- \( S_p \)
- \( B_o \)
- \( S_o \)

Objects:
```

```
\[ \begin{array}{c}
0 \\
1 \\
0 \\
1 \\
1 \\
0 \\
0 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
0 \\
1 \\
0 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1 \< /raw>
Bitmap Triples Encoding

- Bitmap Triples:

```
<table>
<thead>
<tr>
<th>S</th>
<th>P</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

**Predicates:**

\[ B_p \]

\[ S_p \]

\[ B_o \]

\[ S_o \]

```
\[ 1 \]
1 0 0 1 1 1
\[ 4 2 3 5 1 1 \]
1 1 0 1 1 1
5 1 3 4 2 1 1
```

**Objects:**

- \( O_1 \)
- \( O_2 \)
- \( O_3 \)
- \( O_4 \)
- \( O_5 \)

**Bitmap Triples Encoding subjects:**

- \( [2,5,6] \)
- \( [4] \)
- \( [3] \)
- \( [3] \)
- \( [1] \)

**Bitmap Triples Encoding Predicates:**

- \( ? ? O \)
- \( ? ? P \)
- \( O \)
On-the-fly indexes: HDT-FoQ

- From the exchanged HDT to the functional HDT-FoQ:
  - Publish and Exchange HDT
  - At the consumer:

<table>
<thead>
<tr>
<th>Process</th>
<th>Type of Index</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subject</td>
<td>SPO, SP?, S??, S?O, ???</td>
</tr>
<tr>
<td>2</td>
<td>Predicate</td>
<td>?P?</td>
</tr>
<tr>
<td>3</td>
<td>Object</td>
<td>?PO, ??O</td>
</tr>
</tbody>
</table>

- Publish and Exchange HDT
- At the consumer:

![Diagram of the process]

1. **index the bitsequences**
   - Subject: SPO, SP?, S??, S?O, ???
2. We index the position of each predicate (with a wavelet*)
   - Predicate: PSO
   - Pattern: ?P?
3. We index the position of each object (just a position list)
   - Object: OPS
   - Pattern: ?PO, ??O
(HDT creation, as requested by audience)

- Classifying RDF terms.
  - Triple-by-triple parsing.
  - Three hash tables: S, P, O.
  - Triples are encoded by temporary ID
  - SO hash table is built after parsing.
- Building HDT Dictionary.
  - Each section is sorted lexicographically.
  - Relation between definitive IDs and temporary IDs are stored.
- Building HDT Triples.
  - Triples have their temporary IDs replaced by definitive IDs.
  - ID-triples are sorted by subject, predicate and object ID
  - Bitmap Triples are obtained.
HDT

Achievements and Challenges
HDT Acknowledged as W3C member submission:

- [http://www.w3.org/Submission/2011/03/](http://www.w3.org/Submission/2011/03/)
Some numbers on size

http://dataweb.infor.uva.es/projects/hdt-mr/


| Dataset        | Triples | |S0| |S| |O| |P| | NT | NT+lzo | HDT | HDT+gz |
|----------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|-------|-------|-------|
| LinkedGeoData  | 0.27BN  | 41.5M   | 10.4M   | 80.3M   | 18.3K   | 38.5    | 4.4     | 6.4     | 1.9     |
| DBpedia        | 0.43BN  | 22.0M   | 2.8M    | 86.9M   | 58.3K   | 61.6    | 8.6     | 6.4     | 2.7     |
| Ike            | 0.51BN  | 114.5M  | 0       | 145.1K  | 10      | 100.3   | 4.9     | 4.8     | 0.6     |
| Mashup         | 1.22BN  | 178.0M  | 13.2M   | 167.2M  | 76.6K   | 200.3   | 18.0    | 17.1    | 4.6     |
| LUBM-1000      | 0.13BN  | 5.0M    | 16.7M   | 11.2M   | 18      | 18.0    | 1.3     | 0.7     | 0.2     |
| LUBM-2000      | 0.27BN  | 10.0M   | 33.5M   | 22.3M   | 18      | 36.2    | 2.7     | 1.5     | 0.5     |
| LUBM-3000      | 0.40BN  | 14.9M   | 50.2M   | 33.5M   | 18      | 54.4    | 4.0     | 2.3     | 0.8     |
| LUBM-4000      | 0.53BN  | 19.9M   | 67.0M   | 44.7M   | 18      | 72.7    | 5.3     | 3.1     | 1.0     |
| LUBM-5000      | 0.67BN  | 24.9M   | 83.7M   | 55.8M   | 18      | 90.9    | 6.6     | 3.9     | 1.3     |
| LUBM-6000      | 0.80BN  | 29.9M   | 100.5M  | 67.0M   | 18      | 109.1   | 8.0     | 4.7     | 1.6     |
| LUBM-7000      | 0.93BN  | 34.9M   | 117.2M  | 78.2M   | 18      | 127.3   | 9.3     | 5.5     | 1.9     |
| LUBM-8000      | 1.07BN  | 39.8M   | 134.0M  | 89.3M   | 18      | 145.5   | 10.6    | 6.3     | 2.2     |
| LUBM-12000     | 1.60BN  | 59.8M   | 209.9M  | 133.9M  | 18      | 218.8   | 15.9    | 9.6     | 2.9     |
| LUBM-16000     | 2.14BN  | 79.7M   | 267.8M  | 178.6M  | 18      | 292.4   | 21.2    | 12.8    | 3.8     |
| LUBM-20000     | 2.67BN  | 99.6M   | 334.8M  | 223.2M  | 18      | 366.0   | 26.6    | 16.3    | 5.5     |
| LUBM-24000     | 3.74BN  | 119.5M  | 401.7M  | 267.8M  | 18      | 439.6   | 31.9    | 19.6    | 6.6     |
| LUBM-28000     | 3.74BN  | 139.5M  | 468.7M  | 312.4M  | 18      | 513.2   | 37.2    | 22.9    | 7.7     |
| LUBM-32000     | 4.27BN  | 159.4M  | 535.7M  | 357.1M  | 18      | 586.8   | 42.5    | 26.1    | 8.8     |
| LUBM-36000     | 4.81BN  | 179.3M  | 602.7M  | 401.8M  | 18      | 660.5   | 47.8    | 30.0    | 9.4     |
| LUBM-40000     | 5.32BN  | 198.4M  | 666.7M  | 444.5M  | 18      | 730.9   | 52.9    | 33.2    | 10.4    |
Evaluation. Querying (TP)
Results

- Data is ready to be consumed 10-15x faster.
  - HDT $\ll$ any other RDF format $||$ RDF engine

- Competitive query performance.
  - Very fast on triple patterns, $\times$ 1.5 faster (Virtuoso, RDF3x).

- Integration with Jena
  - Joins on the same scale of existing solutions (Virtuoso, RDF3x).
Get more than 650K HDT datasets from LOD Laundromat...

LOD Wardrobe
This is where the cleaned data is stored. You can download both clean and dirty (i.e., the original) data. Each data document contains a meta-data description that includes all the stains that were detected.
And query them online
Still room for optimization

- **Dynamically:**
  - Most compact data structures are “static”, but data may evolve
  - Tradeoff between compression and fast generation

- **Advanced capabilities:**
  - Reasoning (entailment)

- **Query optimization:**
  - Leverage precomputed statistics (stored in the Header)
  - Leverage on-demand statistics (Triple Pattern cardinality)
Let’s the lecture continue…

RDF Dictionaries