HOBBIT

Overview of Big Data Benchmarking

Erik Mannens, Gayane Sedrakyan

Horizon 2020
GA No 688227
01/12/2015 – 30/11/2018

HOBBIT
Holistic Benchmarking
of Big Linked Data

HOBBIT Project goals

- HOBBIT - H2020 European project that aims
  - creating a holistic big linked data benchmarking platform for European companies that will allow them to:

    ➔ assess the fitness of existing solutions for their purposes,

    ➔and for scientific community supporting research and development of big data benchmarking methods, tools and algorithms.

Big Data Benchmarking

- Big data has become a major force of innovation
- New engines with increasing features for managing big datasets

➔ There is lack of means of comparability among such engines

➔ Measuring performance of traditional databases is well-understood, there are no clear performance definitions or metrics for comparing big data systems

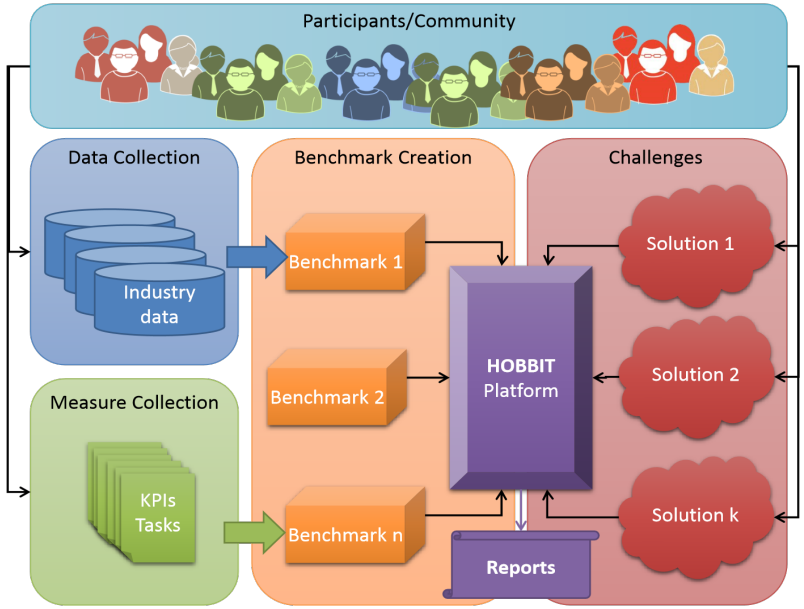
HOBBIT Project goals

- The main objectives of HOBBIT are:
  - Building a generic platform with a family of industry-relevant benchmarks,
  - Implementing a generic evaluation for the Big Linked Data (BLD) value chain,
  - Providing periodic benchmarking results including diagnostics for further improvement of BLD processing tools,
  - (Co-)Organizing challenges and events to gather benchmarking results as well as industry-relevant KPIs and datasets,
  - Supporting companies and academics during the creation of new challenges or the evaluation of tools.

➔ Contributing to best practices and standardization in the field

Community building

- **Aim** ➔ establish as the provider of a benchmarking platform for industry and academia with a focus of Big Linked Data technologies
  - **Key step** ➔ to build up a community of interested parties around the project



  - Gather relevant datasets for a broad context
  - Gather KPIs for the evaluation of the frameworks
  - Gather solutions to benchmark
  - Collect potential members of the Hobbit association (establish as a TF SG within BDVA)

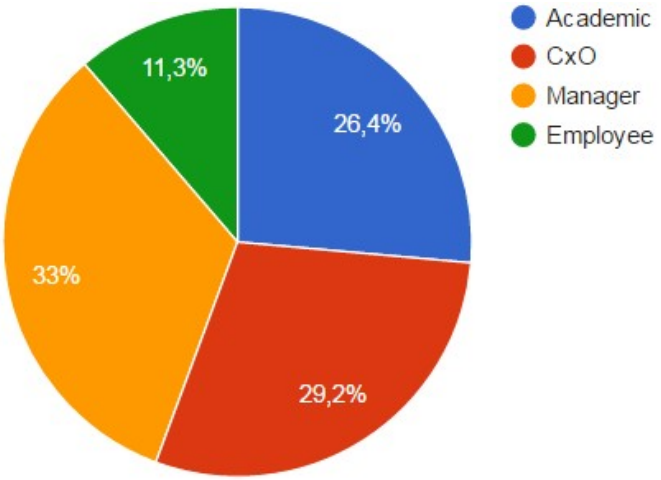
IMEC role

- As a consortium partner IMEC
  - contributes to community building around the project focusing on academic, solution providers and technology users, as well as developing a strategy for establishing a HOBBIT association.
- In the context of benchmarks
  - IMEC-IDLab has also designed and developed an easily pluggable data generator and mimicking algorithms that generate public transport networks using both real and synthetic geographic data.

Contact List

Contact detail

- Email
- Full name, first name and last name
- Role and role type
- Company
- Country
- LinkedIn
- Comment
- Source
- Project Contact



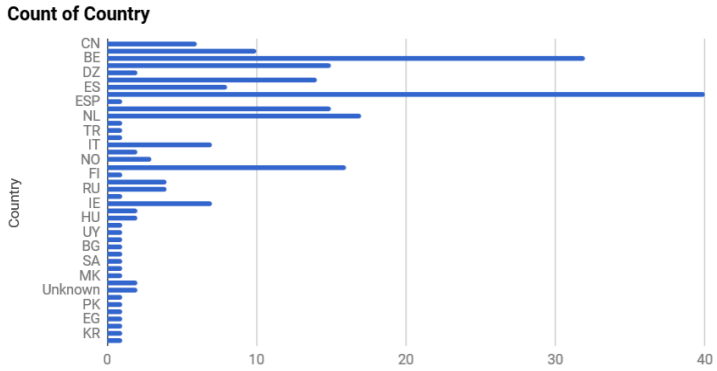
Distribution of roles of Hobbit contacts

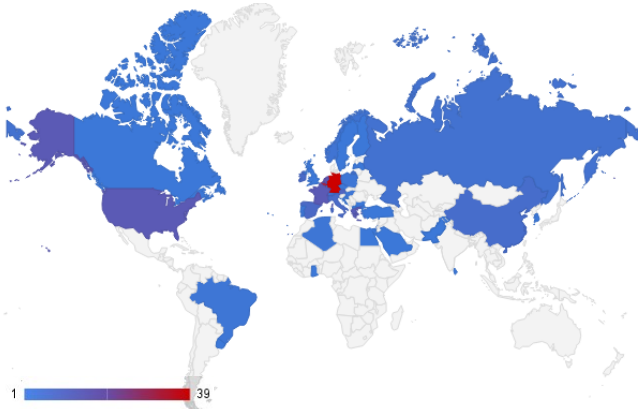
Summary: 250 contacts (135 interacted)

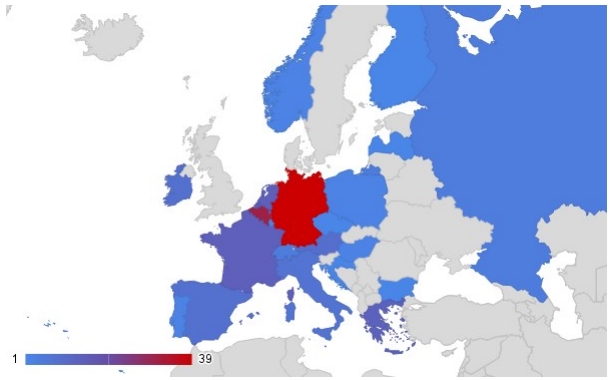
Geographic distribution

Interaction with contacts

- Led to relevant Use Cases
- Led to relevant Data Sets





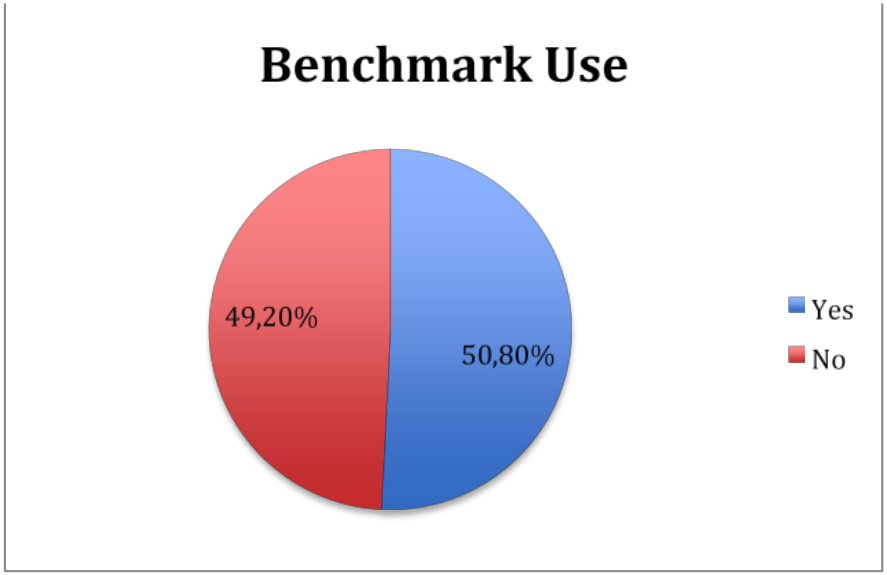


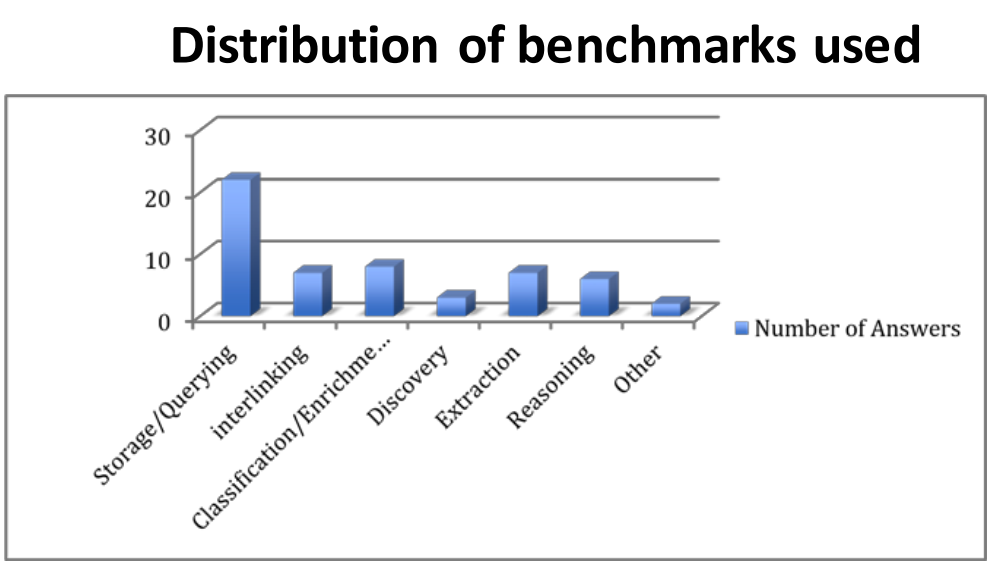
Distribution of Hobbit contacts in the world (left) and in Europe (right)

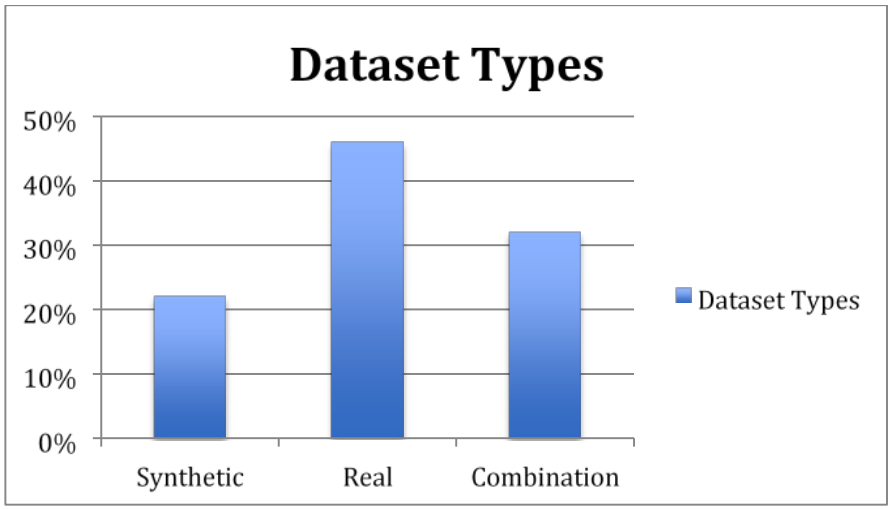
Requirements elicitation & datasets gathering

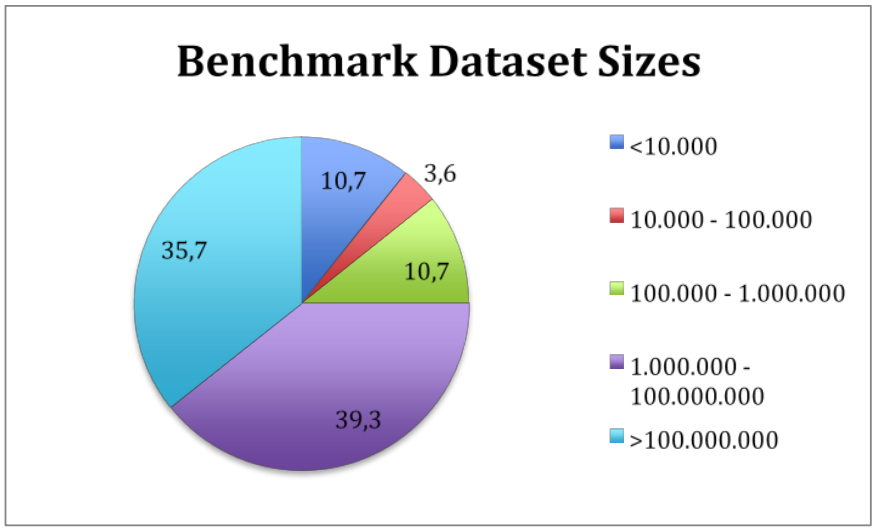
- Strategy for requirements elicitation
  - A survey method has been used to ensure the alignment of the outcomes with real needs of the clientele we target
  - Collected requirements still being used as guidelines for
    - Improving the architecture of the HOBBIT platform
    - Developing and improving the benchmarks
    - Implementing and reporting KPIs

Extracts from HOBBIT survey results



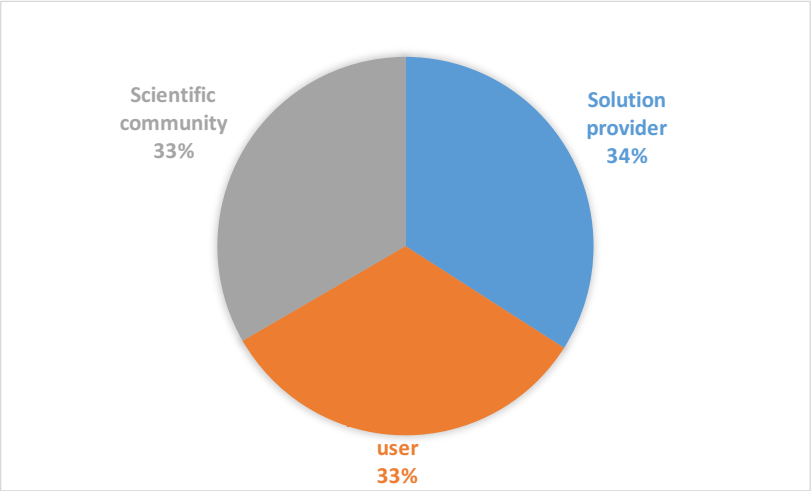




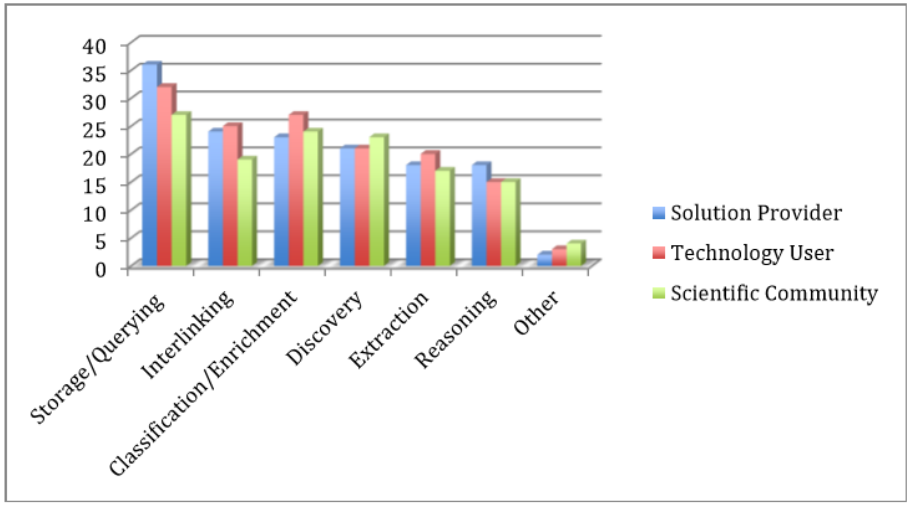


Extracts from HOBBIT survey results

- **Participant profiles**



- **LD solution interests per profile**

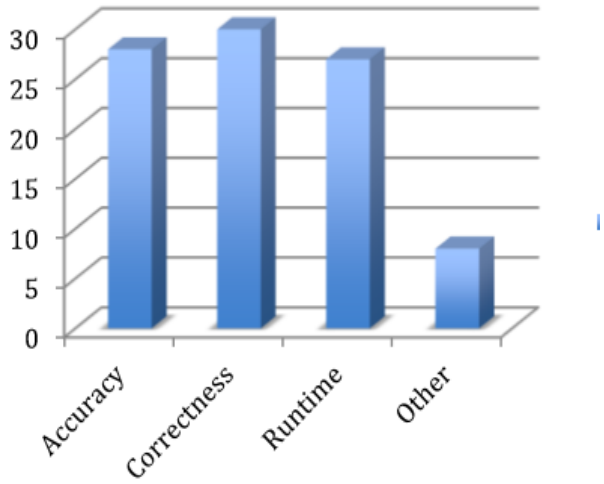


- Other benchmarking solutions such as Reporting/Visualization and Inconsistency Detection ("Other" choice).

Extracts from HOBBIT survey results

- **Key Performance Indicators**

  - KP1. Correctness
  - KP2. Accuracy (precision, recall, F-measure, mean reciprocal rank)
  - KP3. Runtime / Speed
  - KP4. Total triple pattern-wise sources selected
  - KP5. Number of intermediate results
  - KP6. Scalability
  - KP7. Memory usage
  - KP8. CPU usage
  - KP9. Functionality



Requirements analysis results

- Needs for benchmarking solutions

I. Generation & Acquisition

- Include Context in the proposed HOBBIT Benchmark(s)
- Define and Implement Baselines for each Benchmark
- KPIs should be defined independently of the Use Cases
- HOBBIT Benchmarks should be repeatable
- HOBBIT Benchmarks should aim for standard languages
- Semantics need to be accounted for in the Benchmark(s)
- Stream Reasoning should be considered
- Correctness vs Throughput should be considered as KPIs for the Benchmark(s)

II. Analysis & Processing

- Validation Benchmark would be useful
- HOBBIT Benchmark (s) should consider testing Node Traversal in Top-k manner
- In addition to Gold Standard, Human-driven Explanation should be included as well
- Learning patterns should be considered
- The Linking tool should also consider Federated Datasets
- Query Relaxation approaches should be considered
- Answer Evolution regarding relaxed queries should be considered
- KPIs that should be considered are Execution Time, Memory Consumption, CPU Usage
- Benchmarks should be easy to use and interface(s) provided should be simple

III. Storage & curation

- HOBBIT Benchmarks should test different ways of indexing
- HOBBIT Benchmarks should be tested against a range of different types of stores
- HOBBIT Benchmarks should consider security & access rights although it is hard to have a full control of this due to the involvement of users
- The implementation of HOBBIT Benchmarks should be such that it is easy for systems to use them
- HOBBIT Benchmarks should provide an implementation of query workloads in different language(s) or provide appropriate conversion tools (e.g., from SPARQL to SQL)
- HOBBIT Benchmarks should provide a tool to allow for the validation of the results of different engines.

IV. Visualisation & services

- HOBBIT benchmarks should take into account the techniques behind the services/visualisations (no eye-tracking and so on)
- HOBBIT benchmarks should consider the usefulness and the relevance of the returned results (ease of finding the answer on the client)
- HOBBIT developers should consider a comparison regarding the interoperability between recommender systems
- Correctness of the results should be considered as a KPI
- Ranking of the results should be tested in HOBBIT benchmarks
- There is a need for different levels of correctness for different use cases (especially for question answering)
- Response time should be considered as a KPI
- Knowledge-base exploration should be exploited in HOBBIT Benchmarks
- Workloads in HOBBIT benchmarks should be based on actual usage patterns

Datasets

- 23 datasets were gathered by the consortium and in a CKAN repository, accessible through the URL http://hobbit.iminds.be

  - **LinkedSpending** (government spending from all over the worls as Linked Data) ➔ 2 million financial transactions
  - **Dbpedia** (extracts structured information from the Wikipedia, allows answering complex questions using SPARQL) ➔ 3 billion facts, 125 languages (growing 10-20% per year)
  - **Github data** (how people build software, is home to largest community of open source developers) ➔ 12 million people contribution to 31 million projects
  - **LinkedGeoData** (effort to add a spatial information to the web of data-semantic web) ➔ 30 billion facts (5-10% growth per year)
  - **TLC Trip Record Data** (all trips completed in yellow and green taxis in NYC) ➔ 1 billion trips (10-20% growth per year)
  - **LIVED** (Long Device Level Energy Data contains measurments collected from smart plugs multi-sensors) ➔ 2,5 billion measurments
  - **…**

Use cases

- The data collection process returned use cases hint at applications in the following 6 domains:

  - **Industry 4.0**: The use of semantics is of central importance for the creation of machines that can justify their behavior and interact with their users
    - Source: experts in the SAKE and STEP projects
    - Interest: benchmarking link discovery, storage, machine learning and visualization
    - Datasets: CER Smart Metering, LIVED and Weidmüller

Use cases

- **Geospatial data analysis:** Geospatial datasets belong to the largest and most used datasets on the planet.
  - Source ➔ Experts from related projects (GeoKnow, GEISER, SmartRegio, STEP, SLIPO, SAGE)
  - Interest ➔ Hobbit datasets related to geospatial entities and points of interest (LinkedGeoData, Energy Map Germany, LinkedConnections, TLC Record Trip)
  - The benchmarks of interest ➔ Knowledge extraction from structured and unstructured data, storage, versioning and machine learning and visualization

Use cases

- **Weather Data Analysis:** The increasing amount of streaming data from weather sensors demands novel techniques for the semantic analysis of streaming data
  - Key areas ➔ Continuous queries
    - Benchmarking methodologies and unified semantics still need to be dealt with
  - Significance: Smart metering data (LIVED, Weidmüller, CER), storage and acquisition benchmarks are key
- **Human Resource Management:** A rather surprising use case for the HOBBIT datasets, generators and benchmarks for the sake of finding good candidates for job offers
  - Novel applications ➔ Efficient entity recognition, entity linking and relation extraction, which are the area targeted by the knowledge extraction benchmark of HOBBIT
  - Relevant datasets here include the TWIG and the BENGAL

Use cases

- **Enterprise Search**: Searching through streams of ever changing data is of central importance for data-driven companies
  - Use cases: Federated search across several datasets (see projects DIESEL8 and WDAqua9) to search on mobile devices (e.g., project QAMEL10)
  - Datasets: The QALD 6, DBpedia, BioASQ, MESH and BENGAL
  - Benchmarks: Knowledge acquisition
- **European societal challenges:** Through our collaboration with BigDataEurope, we were able to gather use cases for HOBBIT for seven of the societal challenges formulated by the European Union (i.e., health, food and agriculture, energy, transport, climate, social sciences and security)
  - Benchmarks: For example, the CER Smart Metering data and the data storage and knowledge benchmarks are of central importance for the energy domain while LinkedConnections and all other transport datasets are relevant for the transport societal challenge

Generalized knowledge

Benchmarking categorization, purposes for use, design principles, development methodologies, technologies and standards

Business case: tutorials

The Question(s)

- Which are the problems that I wish to solve?
- Which are the relevant key performance indicators?
- Which is the behavior of the existing engines w.r.t. the key performance indicators?

Which are the tool(s) that I should use for my data and for my use case?

The Answer: Benchmark your engines!

- **Querying Benchmark** comprises of
  - **datasets (synthetic or real)**
  - **set of software tools**
    - synthetic data generators
    - query generators
  - **performance metrics**, and
  - set of **clear execution rules**
- **Standardized application scenario(s)** that serve as a basis for **testing** systems
- Must include **a clear set of factors to be measured** and the **conditions** under which the systems should be measured

Role of Benchmarking

- **Benchmarks exist**
  - To allow **adequate measurements** of systems
  - To provide **evaluation of engines** for real (or close to real) use cases
- **Provide help**
  - **Designers** and **Developers** to assess the **performance** of their tools
  - Users to **compare** the **different available tools** and **evaluate suitability for their needs**
  - **Researchers** to compare their work to others
- Leads to **improvements**:
  - **Vendors** can improve their **technology**
  - **Researchers** can address new challenges
  - Current **benchmark design** can be improved to **cover new necessities and application domains**

Benchmark Categories

- Micro-benchmarks
- Standard benchmarks
- Real-life applications

Benchmark Development Methodology

- **Management** and **methodological activities** performed by a group of people
  - **Management:** Organizational protocols to control the process
  - **Methodological**: **principles**, **methods** and **steps** for benchmark creation
- **Benchmark Development**
  - Roles and bodies: people/groups involved in the development
  - Design principles**:** fundamental rules that direct the development of a benchmark
  - Development process: series of steps to develop a benchmark based on **Choke Points**

Choke Points: the set of technical difficulties that force systems to improve their performance

Benchmark Components

- **Datasets**
  - The raw material of the benchmark against which the workload will be evaluated
  - Synthetic & Real Datasets
    - **Synthetic:** Produced with a **data generator** (that hopefully produced data with interesting characteristics)
    - **Real:** Widely used datasets from a domain of interest
- **Query Workload**
  - Sets of queries and/or updates to evaluate the system with
- **Metrics**
  - The performance metric(s) that determine the systems behavior

Following standards

- Linked data
  - using the Web to connect related data that wasn't previously linked, or
  - using the Web to lower the barriers to linking data currently linked using other methods.

➔ recommended best practice for exposing, sharing, and connecting pieces of ***data***, ***information***, and ***knowledge*** on the Semantic Web using ***URIs*** and ***RDF***

- Resource Description Framework (RDF)
  - W3C standard to represent Web data and metadata
  - **generic** and **simple graph based model**
  - information from **heterogeneous** sources merges **naturally**:
    - – resources with the same URI denote the same non-information resource (leading to the Linked Data Cloud)
  - **structure** is added using **schema languages** and is represented as RDF triples
  - Web browsers use URIs to retrieve information

- Adding Semantics to RDF
  - RDF is a **generic**, **abstract** data model for describing resources in the form of **triples**
  - RDF does not provide ways of defining classes, properties, constraints
  - **W3C Standard Schema Languages**
    - RDF Vocabulary Description Language (RDF Schema - RDFS) to define schema vocabularies
    - Ontology Web Language (OWL) to define ontologies

"An RDF Vocabulary is a schema comprising of classes, properties and relationships which can be used for describing data and metadata"

- RDF Vocabularies are **sets of terms** used to describe **notions** in a **domain of interest**
- An RDF term is either a **Class** or a **Property**
  - **Object properties** denote relationships between objects
  - **Data type properties** denote attributes of resources
- **RDFS designed to introduce useful semantics to RDF triples**
- **RDFS Schemas are represented as RDF triples**

- RDF Vocabulary Description Language (RDFS)
  - **Typing**: defining classes, properties, instances
  - **Relationships** between classes and properties: **subsumption**
  - **Constraints**: *domain* and *range* of properties
  - **Inference** rules to entail **new, inferred** knowledge

	Subject	***Predicate***	***Object***
t1	dbo:MusicalWork	**rdfs:subClassOf**	dbo:Album
t2	dbo:MusicalWork	**rdfs:domain**	dbo:artist
t3	dbo:MusicalWork	**rdfs:range**	dbo:march
t4	dbr:Seven_Seas_Of_Rye	**rdf:type**	dbo:MusicalWork
t5	dbo:Album	**rdf:type**	**rdf:Class**

- **SPARQL - querying RDF data: W3C Standard Language for Querying Linked Data**
- **SPARQL 1.0** (2008) only allows accessing the data (**query**)
- **SPARQL 1.1** (2013) introduces:
  - **Query Extensions:** aggregates, sub-queries, negation, expressions in the SELECT clause, property paths, assignment, short form for CONSTRUCT, expanded set of functions and operators
  - **Updates:**
    - **Data management**: Insert, Delete, Delete/Insert
    - **Graph management**: Create, Load, Clear, Drop, Copy, Move, Add
  - **Federation extension**: Service, values, service variables (informative)

- SPARQL semantics based on **Pattern Matching**
  - Queries describe subgraphs of the queried graph
  - SPARQL graph patterns describe the subgraphs to match

Intuitively a triple pattern denotes the triples in an RDF graph that are of a specific form

```
TP1 = (?album, dbpedia-owl:artist, dbpedia:The_Beatles)
```

matches all albums of the Beatles

```
TP2 = (dbpedia_The_Beatles, ?property, ?object )
```

matches all information about The Beatles

Storing and Querying RDF data

- **Schema agnostic**
  - triples are stored in a large triple table where the attributes are (subject, predicate and object) - "Monolithic" triple-stores

	Subject	Predicate	Object
t1	dbr:Seven_Seas_Of_Rye	rdf:type	dbo:MusicalWork
t2	dbr:Starman_(song)	rdf:type	dbo:MusicalWork
t3	dbr:Seven_Seas_Of_Rye	dbo:artist	dbo:Queen

  - But it can get a bit more efficient

id	URI/Literal
1	dbr:Seven_Seas_Of_Rye
2	dbr:Starman_(song)
3	dbo:MusicalWork
4	dbo:Queen
5	dbo:artist
6	rdf:type

Subject	Predicate	Object
1	6	3
2	6	3
1	5	4

RDF-3X maintains 6 indexes, namely, SPO, SOP, OSP, OPS, PSO, POS. To avoid storage overhead, indexes are compressed! [NW09]

Storing and Querying RDF data

- **schema aware:**
  - one table is created per property with subject and object attributes (Property Tables [Wilkinson06])

Booktype

Subject	Title	Author	copyright
ID1	"XYZ"	"Fox, Joe"	"2001"
ID3	"MNO"	NULL	NULL
ID6	NULL	NULL	"2004"

CDType

Subject	Title	artist	copyright
ID2	"ABC"	"Orr, Tim"	"1985"
ID5	"GHI"	NULL	"1985"

Subject	Predicate	Object
ID2	language	"French"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"

Property-class Table

Subject	Predicate	Object
ID1	type	BookType
ID1	title	"XYZ"
ID1	author	"Fox, Joe"
ID1	copyright	"2001"
ID2	type	CDType
ID2	title	"ABC"
ID2	artist	"Orr, Tim"
ID2	copyright	"1985"
ID2	language	"French"
ID3	type	BookType
ID3	title	"MNO"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"
ID5	type	CDType
ID5	title	"GHI"
ID5	copyright	"1995"
ID6	type	BookType
ID6	copyright	"2004"

Subject	Type	Title	copyright
ID1	BookType	"XYZ"	"2001"
ID2	CDType	"ABC"	"1985"
ID3	BookType	"MNO"	NULL
ID4	DVDType	"DEF"	NULL
ID5	CDType	"GHI"	"1995"
D6	BookType	NULL	"2004"

ubject	Predicate	Object
ID1	author	"Fox, Joe"
ID2	artist	"Orr, Tim"
ID2	language	"French"
ID3	language	"English"

Multi-Value P

Subject	Object
…	…
…	…

Clustered Property Table

Storing and Querying RDF data

- Vertically partitioned RDF [AMM+07]

Subject	Predicate	Object
ID1	type	BookType
ID1	title	"XYZ"
ID1	author	"Fox, Joe"
ID1	copyright	"2001"
ID2	type	CDType
ID2	title	"ABC"
ID2	artist	"Orr, Tim"
ID2	copyright	"1985"
ID2	language	"French"
ID3	type	BookType
ID3	title	"MNO"
ID3	language	"English"
ID4	type	DVDType
ID4	title	"DEF"
ID5	type	CDType
ID5	title	"GHI"
ID5	copyright	"1995"
ID6	type	BookType
ID6	copyright	"2004"

type

Subject	Object
ID1	BookType
ID2	CDType
ID3	BookType
ID4	DVDType
ID5	CDType
ID6	BookType

copyright

Subject	Object
ID1	"2001"
ID2	"1985"
ID5	"1995"
ID6	"2004"

author

Subject	Object
ID1	"Fox, Joe"

title

Subject	Object
ID1	"XYZ"
ID2	"ABC"
ID3	"MNO"
ID4	"DEF"
ID5	"GHI"

artist

Subject	Object
ID2	"Orr, Tim"

language

Subject	Object
ID2	"French"
ID3	"English"

To get the most out of this particular decomposition, a column-oriented DBMS is recommended.

Comparison of Storage Techniques [BDK+13]

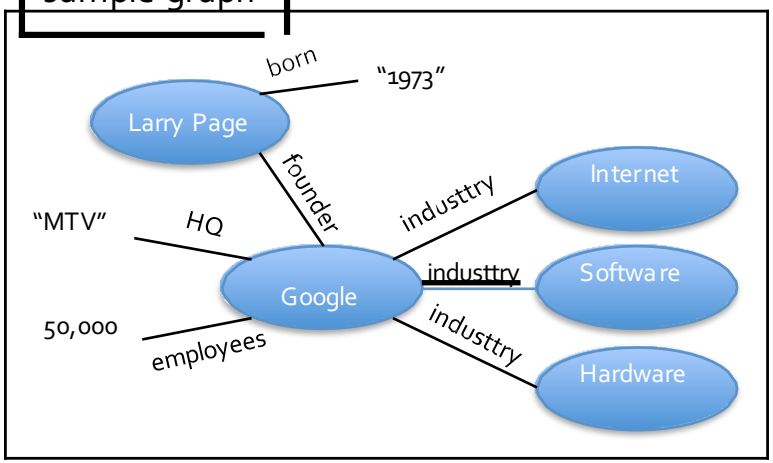
Columns are overloaded

Triple store

subject	predicate	object
Larry Page	born	"1973"
Larry Page	founder	Google
Google	HQ	"MTV"
Google	employees	50,000
Google	industry	Internet
Google	industry	Software
Google	industry	Hardware
Google	developer	Android

Schema does not change on updates

sample graph



Predicate-oriented store

born

subject	object
Larry Page	"1973"

HQ

subject	object
Google	"MTV"

founder

employees

developer

Traditional relational column treatment

Static mix of overloaded and *normal* columns

Type-oriented

person	born	founder
Larry Page	"1973	Google

company	HQ	employees
Google	"MTV"	50,000

subject	predicate	object
Google	industry	Internet
Google	industry	Software
Google	industry	Hardware

movie	released
Google	Android

Schema might change on updates

*Storing Linked Data: **Query Processing***

- **Schema Agnostic**
  - algebraic plan obtained for a query involves a large number of self joins
  - queries are favorable when the predicate is a variable
- **Hybrid Approach** and **Schema-aware**
  - algebraic plan contains operations over the appropriate property/class tables (more in the spirit of existing relational schemas)
  - saves many self-joins over triple tables
  - if the predicate is a variable, then one query per property/class must be expressed

Further questions of interest

1. How can one come up with **the right benchmark** that **accurately** captures use cases of interest?
2. How can a benchmark capture the fact that RDF data originate from a **multitude of formats**
   - Structured: relational and/or XML data to RDF
   - Unstructured
3. How can a benchmark capture **the different data and query patterns** and provide a **consistent picture** for system behavior across different **application settings**?
4. How can one select the **right benchmark** for her system, **data** and **workload**?

- Much more to be found in the series of HOBBIT tutorials

- If interested to include your company in HOBBIT contact list please contact GAYANE.SEDRAKYAN@UGENT.BE