Data Lakes: A Solution or a new Challenge for Big Data Integration

Christoph Quix, DATA 2016
christoph.quix@fit.fraunhofer.de
Frequent Problems of a Big Data Project

- Which data sources are available?
- Where is the data which I require for my application?
- What are the interfaces of the data source?
- Which API can be used to retrieve the data?
- How can integrate the data with other datasets?
- How can I transform the data into my desired target structure & system?
- How can I keep the data up-to-date?
- ...

Data Lakes as a general-purpose data store
Source of the Data Lake Idea

If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples.

James Dixon (Pentaho)
Agenda

- Motivation & Introduction
- State-of-the-Art & Practice
- Architecture for a Data Lake System
- Challenges for Data Lake Systems

- Conclusion and Outlook
Agenda

- Motivation & Introduction
- **State-of-the-Art & -Practice**
- Architecture for a Data Lake System
- Challenges for Data Lake Systems

- Conclusion and Outlook
State-of-the-Art & -Practice

- Frequently mentioned properties of a data lake system
  - Storage of data in its original structure
  - Data of any source can be added to the data lake
  - A data lake has multiple sources
  - Metadata is important
  - Governance is necessary

- But
  - Only few details about the required functions and data models
  - No reference architectures
  - Consulting business
Abstract DL Architecture

Hadoop is not the architecture of a DL system, but it might be an important component.
DL Architecture of a Flight Tracking System

Boci, E. & Thistlethwaite, S.: A novel big data architecture in support of ADS-B data analytic

Fraunhofer FIT
Goods: Google’s Datasets

Agenda

- Motivation & Introduction
- State-of-the-Art & -Practice
  - Architecture for a Data Lake System
  - Challenges for Data Lake Systems
- Conclusion and Outlook
Proposal for a DL Architecture

- User
  - Metadata Manager
    - Schema Manager
    - Enrichment
  - Data Exploration
    - Query Formulation
    - Data Interaction
  - Application-specific Data Marts
- Data Scientist
  - Metadata Model
    - Schemas, Mappings, DQ, Lineage
  - Data Access Interface
    - Query Rewriting
    - Data Transformation
  - Metadata Store
  - Raw Data Stores
- DL Admin
  - DS Config
  - DQ Control
  - Metadata Extraction
  - Data Ingestion
  - Load as-is
- Keynote DATA 2016

Heterogeneous Data Sources
Ingestion Layer

- Low effort for loading data (ELT, not ETL!)
- Support for the extraction of metadata and data
  - Degree of automatic extraction?
  - Schema for semi-structured data (JSON, XML)
  - Schema-on-Read
  - Lazy Loading
Storage Layer

- Type of data storage
  - HDFS? NoSQL? RDBMS?
  - Hybrid!
  - Uniform access interface for all data stores
  - Rewriting of queries and data transformation

- Metadata repository and metadata model
  - Store information about schemata, mappings, data quality, data provenance, ...
  - Tight integration of data and metadata
Interaction Layer

- Search & Exploration in Data Repository
  - Rather keyword-like queries, than direct SQL-like queries
  - Queries for metadata and data

- User interaction should be captured as metadata
  - Definition of precise queries (SQL-like)
  - Recognition of data relationships

- Metadata Management
  - Exploration of the DL System (What data is available?)
  - Semantic Annotations
- Human-centered support for incremental & interactive data integration in the life sciences (Research project 2015-2018)
- Integration applies **pay-as-you-go** idea
- Data is **incrementally** collected and **integrated**
- **Interactive tools** for the exploration and selection of data, definition of semantic relationships, and visualization
- Separation of storage of raw data in data lake and its processing/analysis in data marts
charMant

Data Lakes in a Production Environment

Collect sensor data from machines, integrate with ERP/MES data, to enable quality management applications
Agenda

- Motivation & Introduction
- State-of-the-Art & -Practice
- Architecture for a Data Lake System
- Challenges for Data Lake Systems
- Conclusion and Outlook
Uniform Data Access Layer

- DL system should provide a uniform way to query & retrieve data, ignoring its original format
- Query language should use a generic data model that can be easily mapped to concrete data models
  - JSON $\rightarrow$ JSONiq
  - Graphs $\rightarrow$ Cypher, GraphLog, ...
- Challenge: Rewriting the input query into the query language of the underlying storage system
- Consider also Schema Mappings? $\rightarrow$ GAV/LAV rewriting
Query Rewriting Engine

[Diagram showing the flow of query, options, logic, and results through QueRJS and a Plugin with MRTranslator and ... ]
JSONiq

Based on Xquery

http://jsoniq.org

Few implementations by 28.io, Zorba, IBM, ...

```xml
for $x in collection("cities")
where $x.state eq "AL"
return {
    "id" : $x.id,
    "city" : $x.city
}

for $x in coll("cities")
group by $s := $x.state
return {
    "state" : $s,
    "ids" : $x._id
}
```
Logical Representation

Datalog-like notation

\( \text{in}(x,c): x \text{ is member of } c \text{ (coll./array)} \)
\( \text{a}(x,k,v): x \text{ has a key } k \text{ with value } v \)

Queries are translated into rules

\[
q(f(x_1, \ldots, x_n)) :- /* \text{HEAD} */ \\
\text{CQ for body } \rightarrow \\
\text{CQ to define result structure}
\]
Query Rewriting

1. JSONiq \(\rightarrow\) Logical Representation

```json
for $x$ in collection("cities")
where $x$.state eq "AL"
return {
    "id" : $x$.id,
    "city" : $x$.city
}
```

```json
q(r(Vx)) := /* Head */
in(Vx,:collection("cities")), a(Vx,"state",V0), /*B*/
eq(V0,"AL"), a(Vx,"_id",V2), a(Vx,"city",V3) ->
let(r(Vx),V1), a(V1,"_id",V2), a(V1,"city",V3). /*R*/
```
Query Rewriting
2. Logic \(\rightarrow\) MR-Query

```
q(r(Vx)) := /* Head */
in(Vx,:collection("cities"), a(Vx,"state",V0), /*B*/
eq(V0,"AL"), a(Vx,"_id",V2), a(Vx,"city",V3) ->
let(r(Vx),V1), a(V1,"_id",V2), a(V1,"city",V3). /*R*/
```

```
function map() {
  if (this.state == "AL") {
    var r = {};
    r._id = this._id;
    r.city = this.city;
    emit(this._id, r);
  }
}
```

```
function reduce(key, values) {
  var r = {};
  for (_v in values) {
    var value = values[_v];
    for (_a in value) {
      if (_a == null) r[_a] = [];
      r[_a].push.apply(r[_a],value[_a]);
    }
    return r;
  }
}
```

Note: MapReduce-Query is only one possible executable output format, there can be other languages.
Lazy & Pay-As-You-Go

- Data loading and integration usually requires high manual efforts.
- Complex tasks for data integration (mapping, reconciliation, etc.) should be only addressed if data is required for a specific application.
- Application context might simplify integration tasks as not the "universal" integrated schema needs to be defined, but only one for a specific application.
- Balance between "early investments" and later user interaction.
  - Which data marts should be materialized?
  - What kind of transformations and cleaning activities should be already performed on the datasets?
Schema-on-Read & Schema Evolution

- Schema-on-Read
  - Schemas are only created when data is accessed (Lazy)
  - Schema extraction from semi-structured data is required
    - Structure and constraints

- Schema Evolution
  - Schemas are evolving, especially in NoSQL systems
  - Incremental maintenance of extracted schema information is necessary
  - In addition, schema definitions can be revised while using the data
    - additional constraints, semantic annotations, ...
Data Quality

- Data Quality Management requires a holistic approach in DL systems

- Data quality needs to be controlled already at the ingestion layer to avoid the data swamp
  - Specify minimal requirements for ingested data
  - Implement quality control by (metadata) queries

- Data quality information should be made accessible in the metadata repository
Conclusion

- Data analysis is often hampered by limited data availability.
- Making the data available in a data lake system provides query, search and exploration features to the users.
- Data lake is in early concept and requires more research.
- We have addressed some challenges:
  - Metadata extraction (→ CAiSE Forum 2016)
  - Constance – Data Lake Framework (→ SIGMOD 2016)
  - Data quality management (→ QDB workshop at VLDB 2016)
  - User interaction and data visualization
Solution or Challenge?

- Both
  - Good implementation of metadata management, data quality management, and data governance might simplify data integration tasks
  - Transition from ETL to ELT
  - Typical integration problems (heterogeneity in various aspects) are still there, but the problem is postponed to later stage
  - DL systems are complex systems which need to be assembled from various components, not only HDFS