Partial Data Materialization Techniques for Hybrid Data Integration

Dmitriy Pochitaev
dmitriy.pochitaev@put.poznan.pl

Supervisors:
Robert Wrembel - Poznan University of Technology;
Oscar Romero, Petar Jovanovic - Universitat Politècnica de Catalunya

Outline

- Motivation
- Data Integration Approaches: Physical vs. Virtual
- Hybrid Data Integration (HDI)
- Limitations of Existing HDI Solutions
- Our Scope and Assumptions
- Challenges
- Query Processing Flow
- Materialized Data Stages and Refreshment
- Future Work - Ideas
- Publication Strategy
- Conclusions

Motivation

Big Data challenges (3 Vs):

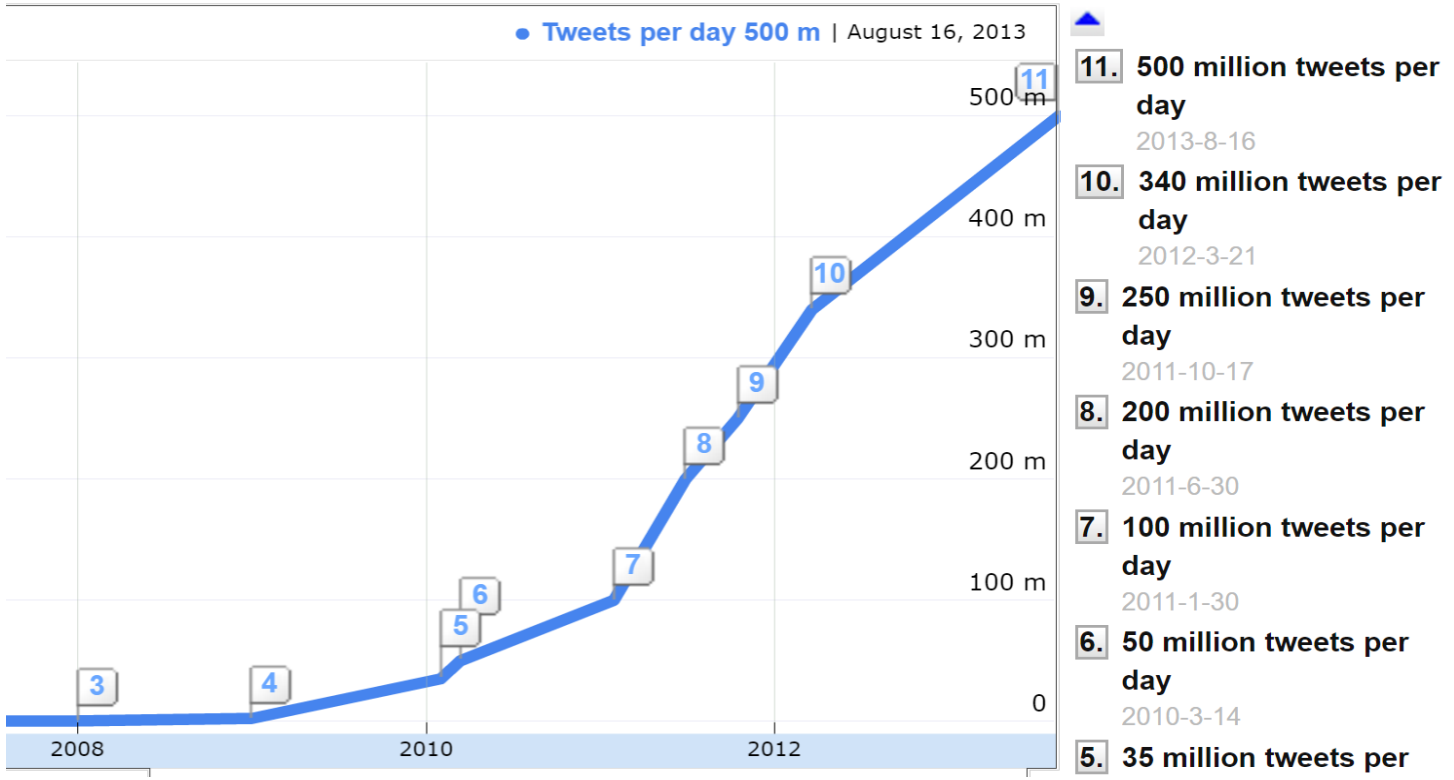
- Large **volume** of accumulated data
- High **velocity** of accumulation of the new data
- **Variety** of data sources

Organizations and data scientists are facing a **challenge of integrating data** from a great number of disparate data sources.

Motivation

Twitter with 2.5 million tweets a day in 2009 rocketing to over **500 million** tweets a day at present - a 20,000 percent increase in less than ten years.

“Gartner, Inc. forecasts that **8.4 billion** connected things will be in use worldwide in 2017, up 31 percent from 2016, and will reach **20.4 billion** by 2020.”
[http://www.gartner.com/newsroom/id/3598917]



[http://www.internetlivestats.com/twitter-statistics]

These examples demonstrate that, the number and variety of data sources, as well as, the speed of accumulation of new data is expected to increase considerably. Thus, the demand for data integration should even increase in the nearest future.

Data Integration

Solution: data integration system that will deal with the complexity of integrating huge amounts of data coming from disparate heterogeneous sources and will provide an integrated view of the data to the users.

Data integration:

1. ***different sources***
2. ***combining data***
3. ***unified view of data***

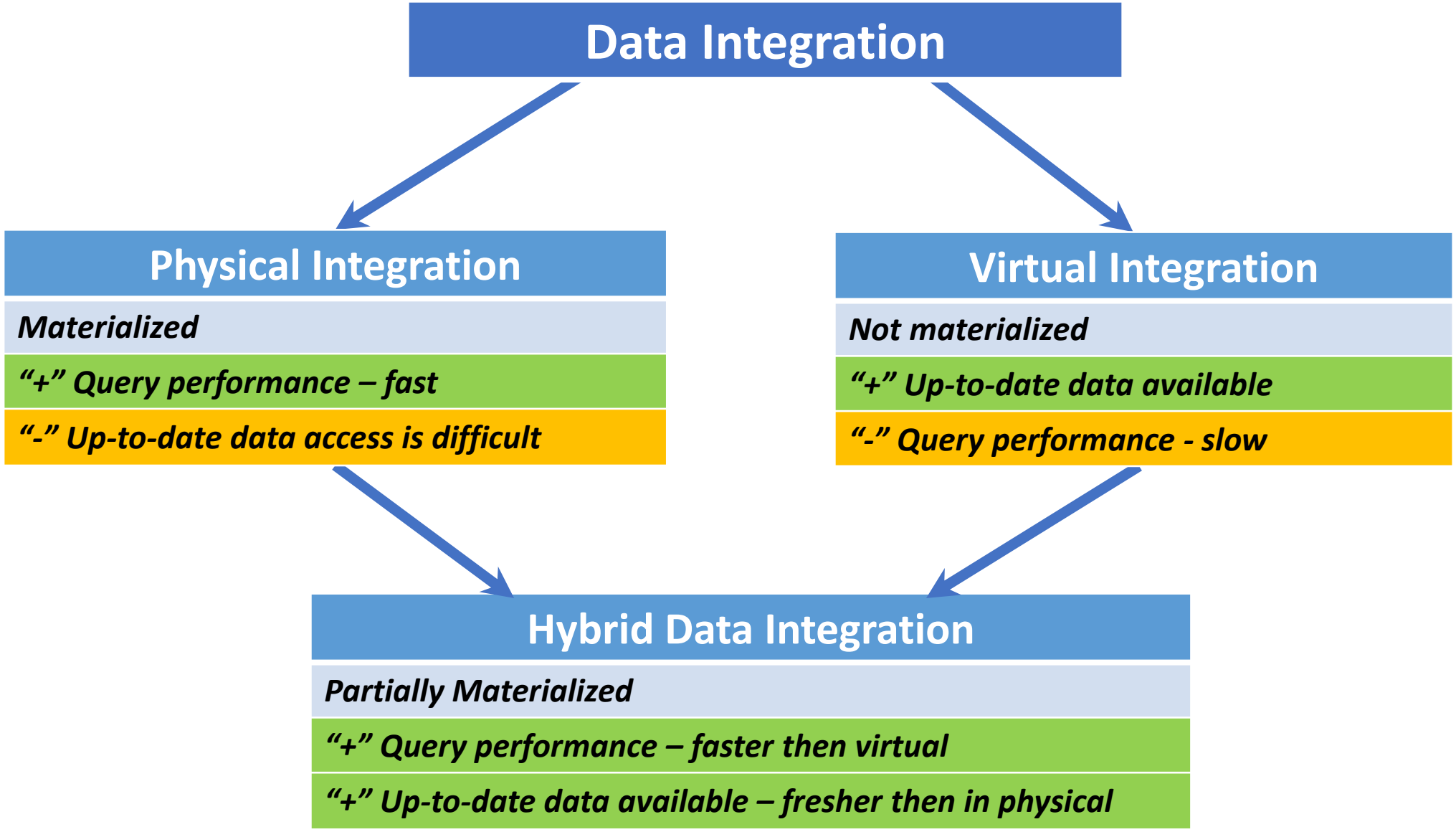
Data Integration Approaches

Two major approaches for data integration:

- **Physical** - data is physically copied (with or without transformation) to a dedicated storage, e.g., Data Warehousing, Data Lakes.

- **Virtual** - data remains in original sources and is accessed during query execution.

Data Integration Approaches



Data Integration Approaches

Hybrid Data Integration (HDI) approach:

virtual data integration combined with ***partial data materialization***.

- Partially materialized data – allows improvement of the ***query performance***.
- Size of materialized data is much smaller then in physical data integration systems – easier to keep ***data fresh***.

Hybrid approach allows ***combining benefits*** of both physical and virtual approaches and minimize their disadvantages.

However, in order to be useful, partially materialized data should be ***carefully selected***.

Partial Data Materialization – Questions to Answer

- Which query results and intermediate query results to materialize?
- Where to materialize data, on disk and/or in RAM?
- How to manage materialized data: which data to refresh incrementally, which data to refresh fully, and when to mark materialized data invalid or outdated?
- Data prefetching - what data to prefetch and when?

Hybrid Data Integration Existing Solutions

- Existing solutions developed for the case when data sources are database systems.
- For example, **H2O system** provides a solution for implementation of a partial data materialization in a virtual data integration system.
  - Considers integration of ***database systems***
  - Implies possibility to ***control data sources***
  - Relies on possibility to ***receive notifications*** about data changes.

[Zhou G, Hull R, King R, Franchitti JC. Data Integration and Warehousing Using H2O. IEEE Data Eng. Bull.. 1995 Jun;18(2):29-40.]

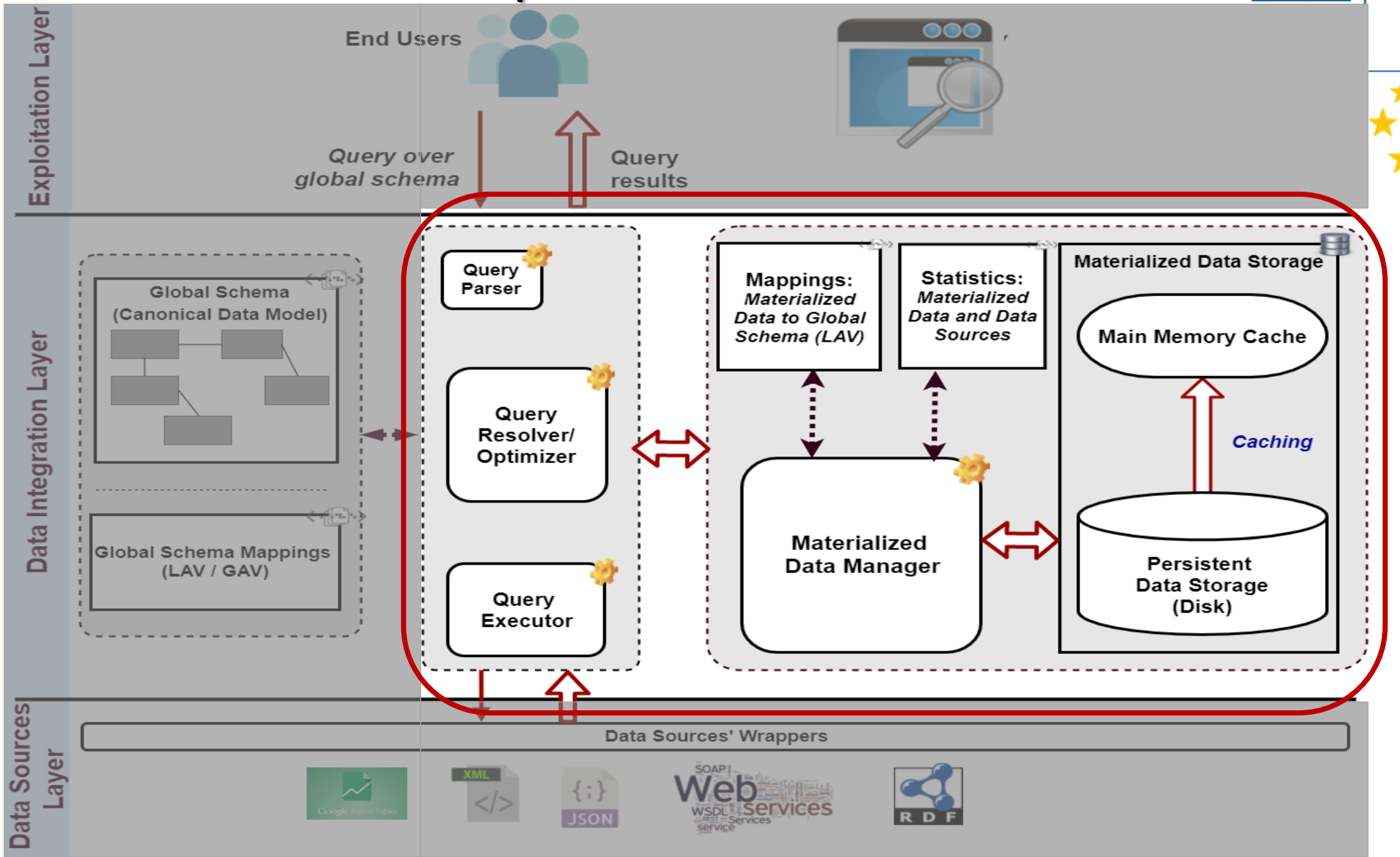
Limitations of Existing Solutions

- Integration of database management systems.
- Assume control over the data sources.
- Heterogeneity of data sources data models is mostly limited to *relational, object-relational* and *object-oriented* models.

Big Data challenges:

- Volume – *existing HDI could handle*
- Velocity – *existing HDI could handle*
- **Variety** – *existing HDI not designed to handle variety*
  - Large number of data sources
  - Heterogeneity of data sources: semantic, syntactic, and system

Architecture and Our Scope



Assumptions

1. Global Schema and Global Schema Mappings assumed to be provided, their maintenance and update are not considered.
   - [Nadal S, Romero O, Abelló A, Vassiliadis P, Vansummeren S. An Integration-Oriented Ontology to Govern Evolution in Big Data Ecosystems. InEDBT/ICDT Workshops 2017.]
   - [Rahm E, Bernstein PA. A survey of approaches to automatic schema matching. The VLDB Journal—The International Journal on Very Large Data Bases. 2001 Dec 1;10(4):334-50.]
   - [Bellahsène Z, Bonifati A, Rahm E, editors. Schema matching and mapping. Springer Science & Business Media; 2011 Feb 14.]
2. The data integration system has no control over the data sources and that all data sources are read-only.
3. Sizes of available persistent storage (disk) and main memory storage are parameters for our problem, i.e., storage size is limited.
4. Types of data sources under consideration: RDF, Web tables, Web pages, and Web Services.

Challenges

At the current stage focusing on the following challenges:

- **What data to materialize?**
- **How to refresh materialized data?**
- **How to process user queries in presence of materialized data?**

What data to materialize?

Selection of data to materialize is known in traditional systems as: ***Materialized views selection***.

[Imene Mami and Zohra Bellahsene. “A survey of view selection methods”.In:SIGMOD Record41.1 (2012), pp. 20–29]

For the case of HDI systems for Big Data, it should be extended to:

- Handle not only relational data
- Handle web data sources:
  - not reliable communications
  - unpredictable performance
  - could be unavailable

What data to materialize?

Proposed solution

Materialize all - maintain selected strategy**.** Materialize results of all queries executed in data sources.

- In classical approach: queries workloads are analyzed and based on the analysis selected views materialized. This requires execution of expensive data fetching from data sources.
- ***Materialize all – maintain selected*** strategy allows to minimize data retrieval from data sources (reuse already available results).

How to refresh materialized data?

In traditional systems known as: ***Materialized views maintenance*** *and relies on "active features" of database systems (e.g., triggers).*

Under our assumptions, data sources are not able to signal about changes of data.

How to refresh materialized data?

Proposed solution

Collection of extensive **statistics** about data sources.

Analysis of collected statistics, application of data mining algorithms.

Detection of **update patterns** in data sources.

> *E.g., some data sources might be updated on certain week days or during certain hours.*

Refresh data according to data sources update patterns – minimization of costly data refresh operations.

How to query data in presence of materialized data?

In traditional systems known as: ***Answering queries using views.*** This is known to be a complex task.

[Alon Y. Halevy. "Answering queries using views: A survey". In:VLDB J.10.4 (2001), pp. 270–294]

Major challenge: detection of part(s) of the user query that could be answered using materialized data.

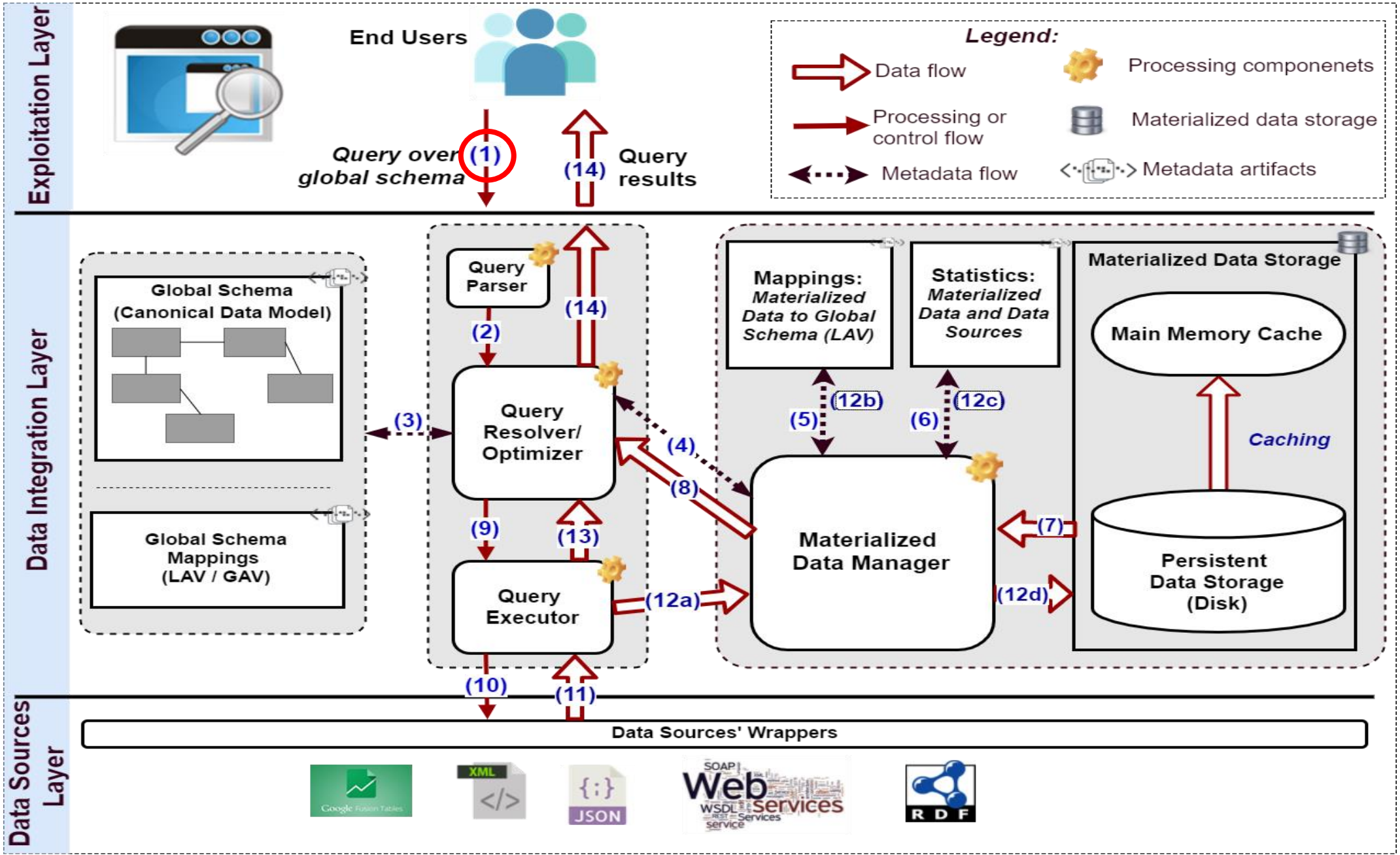
How to query data in presence of materialized data?

Proposed solution

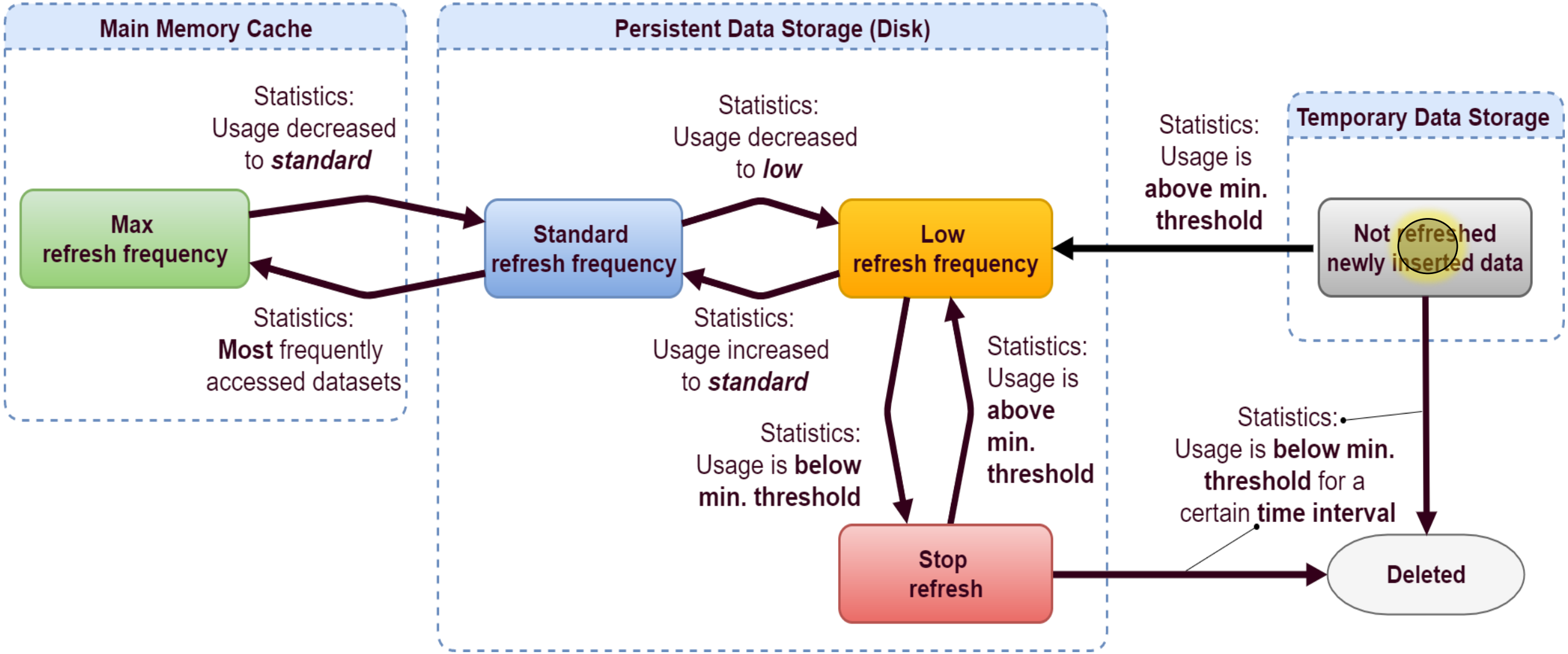
- Local as View (LAV) mappings of materialized datasets to the Global Schema.
- Representing queries and materialized datasets as directed acyclic graphs (DAG).
- Adaptation of the **CoAl** algorithm – detection of common parts in queries of the current workload and already materialized datasets. (e.g., adaptation of the existing CoAl algorithm for multi-flow consolidation).

[Jovanovic P, Romero O, Simitsis A, Abelló A. Incremental consolidation of data-intensive multi-flows. IEEE Transactions on Knowledge and Data Engineering. 2016 May]

Query Processing Flow



Materialized Data Stages



Materialized Data Refresh

Refresh rate depends on: frequency of data source data changes, data source performance, size of materialized dataset, and the overall load of the system. In the simplest case, this dependency could be described by a linear function:

$$Fr = f(Ucoeff, DSperf, DSsize, Sload) = Ucoeff * \left(\frac{DSperf}{DSsize}\right) - Sload$$

- ***Fr*** *- frequency of materialized data refresh.*
- ***Ucoeff*** *- materialized data usage frequency coefficient; the larger the value, the higher is update frequency.*
- ***Dsperf*** *- data source performance coefficient; the higher the value the better is the data source performance.*
- ***DSsize*** *- size of dataset coefficient; the higher the value, the larger the dataset.*
- ***Sload*** *- system overall load coefficient; the higher the value, the higher the system is loaded.*

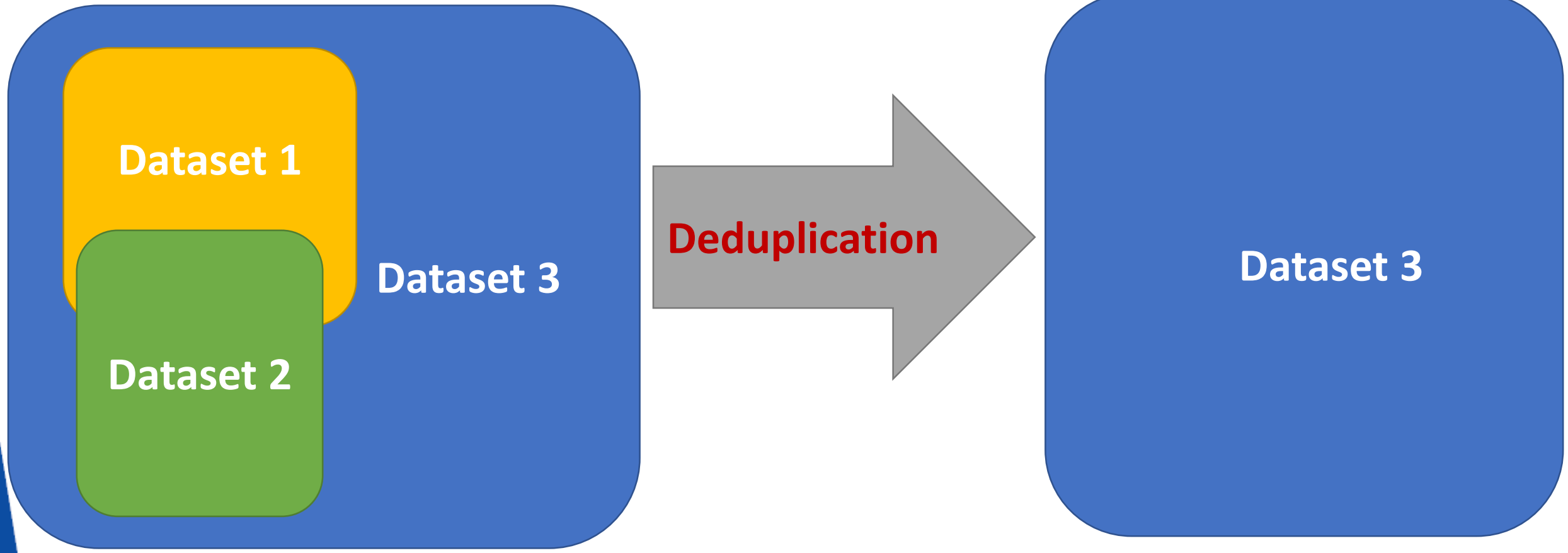
Materialized Data Refresh

However, in practice, the function most likely will be non linear.

For example, with the growth of the system load it may be beneficial (starting from certain system load) to minimize or completely stop materialized data refresh activity.

Materialized Data Management

- Materialized datasets merging and detection of duplicate datasets (using CoAl)



Future Work- Ideas

- Different storage engines for different data
- Dynamic creation of indexes for materialized data based on query workload
- Proactive materialization of ”hot spot” areas
- Data quality issues consideration
- Temporal databases features implementation
- User defined preferences:
  - *Optimization for fast response*
  - *Optimization for max. freshness of the query results*

Publication strategy

- ***"Towards Big Data Integration: Architectures, Challenges, and Solutions"***.

Outlet: The Workshop on Novel Techniques for Integrating Big Data (BigNovelTI) 2017.

State-of-the-art paper, overview of existing approaches, architectures, and identification of challenges.

- ***"Hybrid Data Integration: Enabling Partial Data Materialization of Web Data Sources"***

Outlet: DOLAP 2017

Implementation of partial data materialization support in virtual data integration systems. As a base platform Spark SQL will be used to avoid implementation of typical data integration activities, like connectivity to various data sources, from scratch.

- ***"Enhancing Partial Data Materialization Using Predictive Data Pre-Fetching Techniques"***

Outlet: DAWAK 2018

This paper will be logical continuation of the previous paper, it will enhance the scope by inclusion of predictive data pre-fetching techniques.

- ***"Materialized Data Management for Big Data Integration Architectures"***

Outlet: Information Systems journal 2018.

This journal paper will be a logical continuation of the previous paper, it will enhance the scope by consideration of additional factors such as data source properties (capacity, performance, data format, and data volatility).

- ***"BigCache: Caching Strategies for Big Data Mediated Architecture"***

Outlet: EDBT 2018

This demo paper will present the prototype system, implemented based on the Apache Spark SQL. The prototype system will demonstrate the data materialization techniques developed in the previous papers.

Conclusions

- Big Data Integration poses challenges of:
  - Query performance
  - Data “freshness”
  - Variety and heterogeneity of data sources.
- Virtual Data integration combined with **partial data materialization** are capable of resolving both issues.
- However, materialized data should be **carefully selected**, and data **refreshment policies** should allow efficient updates of materialized data
- New **opportunities** – e.g., incorporation of temporal aspects, improvement of data quality.