Modern Database Systems

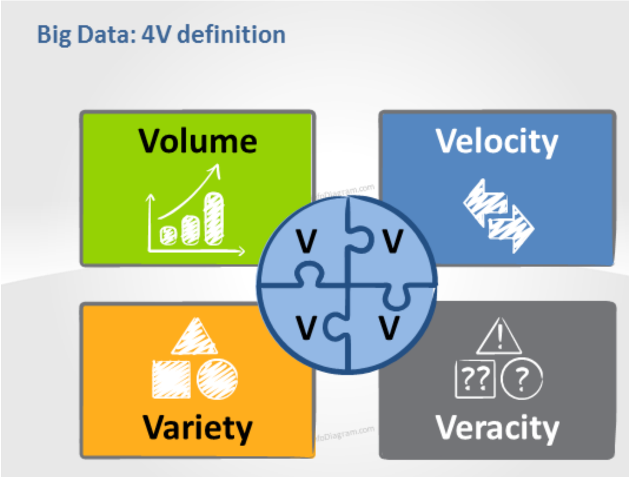
Polystores

Doc. RNDr. Irena Holubova, Ph.D.
Irena.Holubova@matfyz.cuni.cz

Based on the tutorial "Multi-model Databases and Tightly Integrated Polystores: Current Practices, Comparisons, and Open Challenges", Jiaheng Lu, Irena Holubova, Bogdan Cautis, CIKM'18, Turin, Italy.

A Grand Challenge on Variety

- Big data: Volume, Variety, Velocity, Veracity, …
- Variety:
  - Hierarchical data
    - XML, JSON
  - Graph data
    - RDF, property graphs, networks
  - Tabular data
    - CSV
  - …



Motivation

- One application to include multi-model data
  - Relational data: customer databases
  - Graph data: social networks
  - Hierarchical data: catalogue, product
  - Text data: customer review
  - …

Two Solutions

1. Multi-model databases
   - Using one single, integrated backend
2. Polystores
   - Using jointly multiple data storage technologies, chosen based upon the way data is being used by individual applications

Multi-model Database

- One unified database for multi-model data

Polystore

Polyglot persistence

- Use the right tool for (each part of) the job…
  - If you have structured data with some differences
    - Use a document store
  - If you have relations between entities and want to efficiently query them
    - Use a graph database
  - If you manage the data structure yourself and do not need complex queries
    - Use a key-value store

- …and **<u>glue</u>** everything together

An example of multi-model data

Marry (1)

knows knows

William (3) John (2)

Social network graph

```
{ "Order_no":"0c6df508",
  "Orderlines": [
   { "Product_no":"2724f",
     "Product_Name":"Toy",
     "Price":66 },
   { "Product_no":"3424g",
     "Product_Name":"Book",
     "Price":40 } ]
}
```

Order JSON document

"1" -- > "34e5e759"

"2"-- > "0c6df508"

Key/value pairs
(Customer_ID , Order_no)

Customer relation

Customer_ID	Name	Credit_limits
1	Mary	5,000
2	John	3,000
3	William	2,000

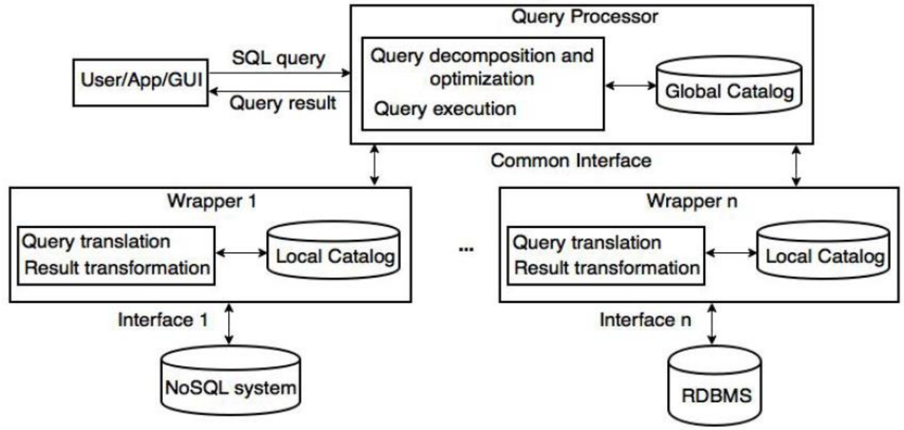
Pros and Cons of Polystores

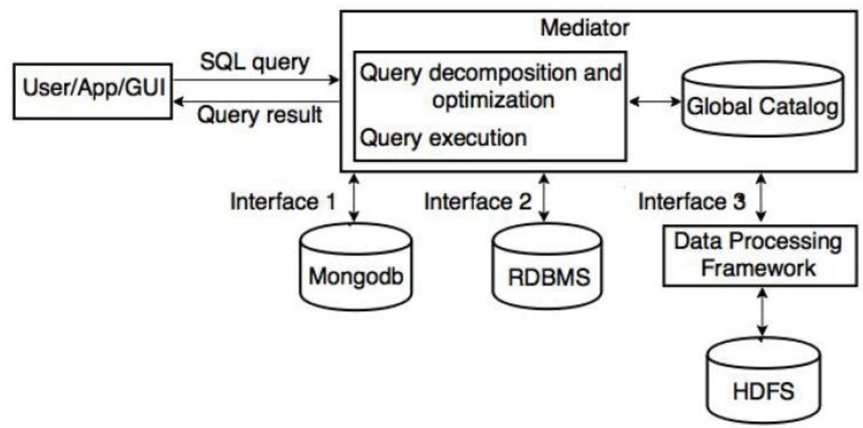
- Handle multi-model data
- Help your applications to scale well
- A rich experience of the single-model stores

- Requires the company to hire people to integrate different databases
- Developers need to learn different databases
- It is a challenge to handle cross-model query and transaction

Three Types of Polystore Systems

- Loosely-coupled systems
  - Similar to mediator-wrapper architecture
  - Common interfaces
  - Autonomy of local stores
- Tightly-coupled systems
  - Exploit directly local interfaces
  - Trade autonomy for performance
    - Materialized views, indexes
- Hybrid





Bondiombouy, Carlyna, and Patrick Valduriez. "Query processing in multistore systems: an overview." International Journal of Cloud Computing 5.4 (2016): 309-346

Polystore	Objective	Data model	Query language	Data stores
Loosely-coupled				
BigIntegrator (Uppsala U.)	Querying relational and cloud data	Relational	SQL-like	BigTable, RDBMS
Forward (UC San Diego)	Unyfing relational and NoSQL	JSON-based	SQL++	RDBMS, NoSQL
QoX (HP labs)	Analytic data flows	Graph	XML based	RDBMS, ETL
Tightly-coupled				
Polybase (Microsoft)	Querying Hadoop from RDBMS	Relational	SQL	HDFS, RDBMS
HadoopDB (Yale U.)	Querying RDBMS from Hadoop	Relational	SQL-live (HiveQL)	HDFS, RDBMS
Estocada (Inria)	Self-tuning	No common model	Native query languages	RDBMS, NoSQL
Hybrid				
SparkSQL (UCB)	SQL atop Spark	Nested	SQL-like	HDFS, RDBMS
BigDAWG (MIT)	Unifying relational and NoSQL	No common model	Island query languages, with CAST and SCOPE operators	RDBMS, NoSQL, Array DBMS, DSMSs

An overview of polystores https://slideplayer.com/slide/13365730/

No „one size fits all"…

- Heterogeneous data analytics: data processing frameworks (Map/Reduce, Spark, Flink), NoSQL, …
- Polystore idea:
  - Package together multiple query engines
    - Union (federation) of different specialized stores, each with distinct (native) data model, internal capabilities, language, and semantics
  - Holy grail: platform agnostic data analytics
- Use the right store for (parts of) each specialized scenario
- Possibly rely on middleware layer to integrate data from different sources

Dimensions of Polystores

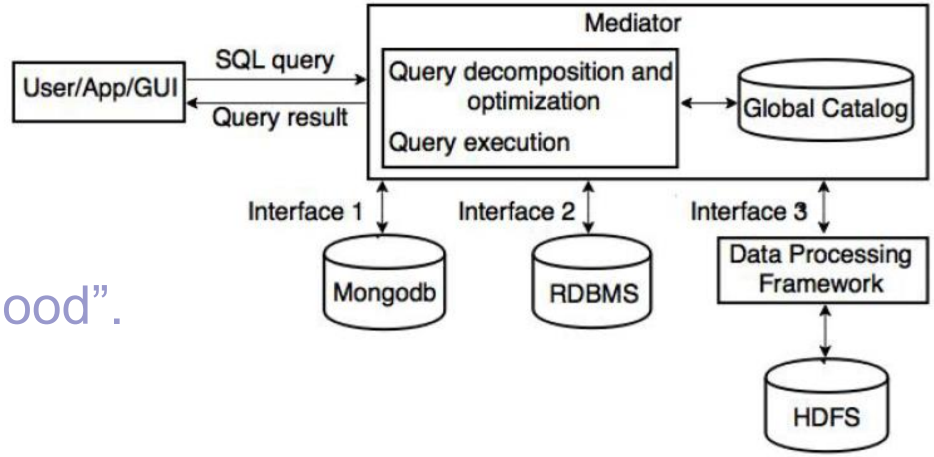
- Heterogeneity
  - Different data models, query models, expressiveness, query engines
- Autonomy
  - Association with the polystore, execution (support of native applications + federation), evolution of own models and schemas
- Transparency
  - Location (data may even span multiple storage engines, user does not know that), transformation / migration of data
- Flexibility
  - User-defined schemata and interfaces (functions), modular architecture
- Optimality
  - Federated plans, data placement

Tan et al. "Enabling query processing across heterogeneous data models: A survey". BigData 2017

Tightly Integrated Polystores (TIPs)

- Examples: Polybase, HadoopDB, Estocada
- Trade autonomy for efficient querying of diverse kinds of data for Big Data analytics
  - Data stores can only be accessed through the multi-store system
  - Less uncertainty with extended control over the various stores
  - Stores accessed directly through their local language
- Efficient / adaptive data movement across data stores
- Number of data stores that can be interfaced is typically limited
- Extensibility
  - Good to have…

Arguably the closest we can get to multi-model DBs, while having several native stores "under the hood".



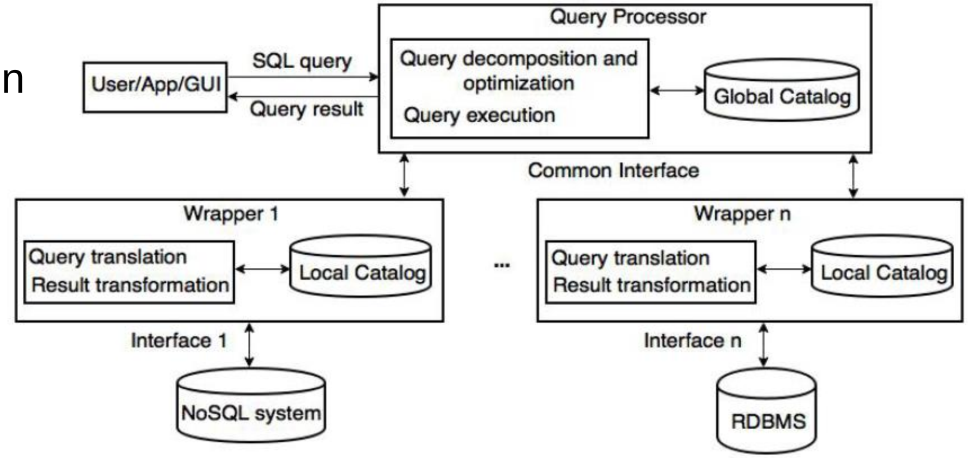
Comparison of MMDs and TIPs

- Common features:
  - Support for multiple data models
  - Global query processing
  - Cloud support

	MMDs	**TIPs**
Engine	single engine, backend	multiple databases (native)
Maturity	lower	higher ???
Usability	read, write and update	read-only
Transactions	global transaction supported	unsupported
Holistic query optimizations	open problem	more challenging
Community	industry-driven	academia-driven
Data migration	difficult	simple

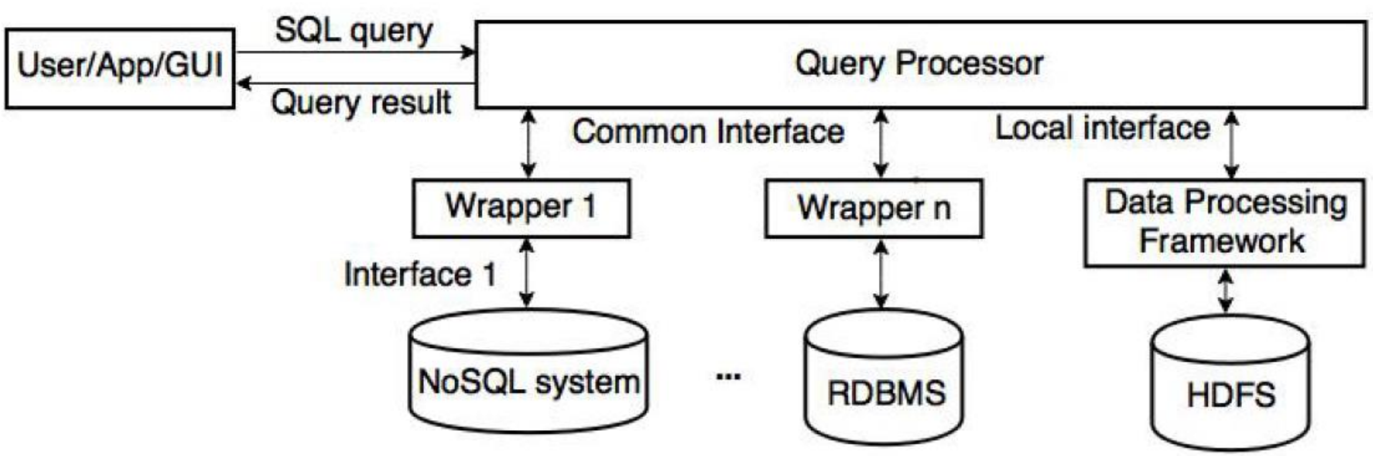
Loosely Integrated Polystores

- Examples: BigIntegrator, Forward/SQL++, QoX
  - Data mediation SQL engines: Apache Drill, Spark SQL, SQL++
    - Allow different sources to be plugged in by wrappers, then queried via SQL
- Reminiscent of multi-database systems
- Follow mediator-wrapper architecture (one wrapper per datastore)
  - One global common language
- General approach
  - Split a query into subqueries
    - Per datastore, still in common language
  - Send to wrapper
  - Translate
  - Get results
  - Translate to common format
  - Integrate

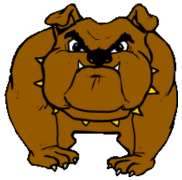


Hybrid Polystores

- Examples: BigDawg, SparkSQL, CloudMdsQL
- Rely on tight coupling for some stores, loose coupling for others
- Following the mediator-wrapper architecture
  - But the query processor can also directly access some data stores



BigDAWG

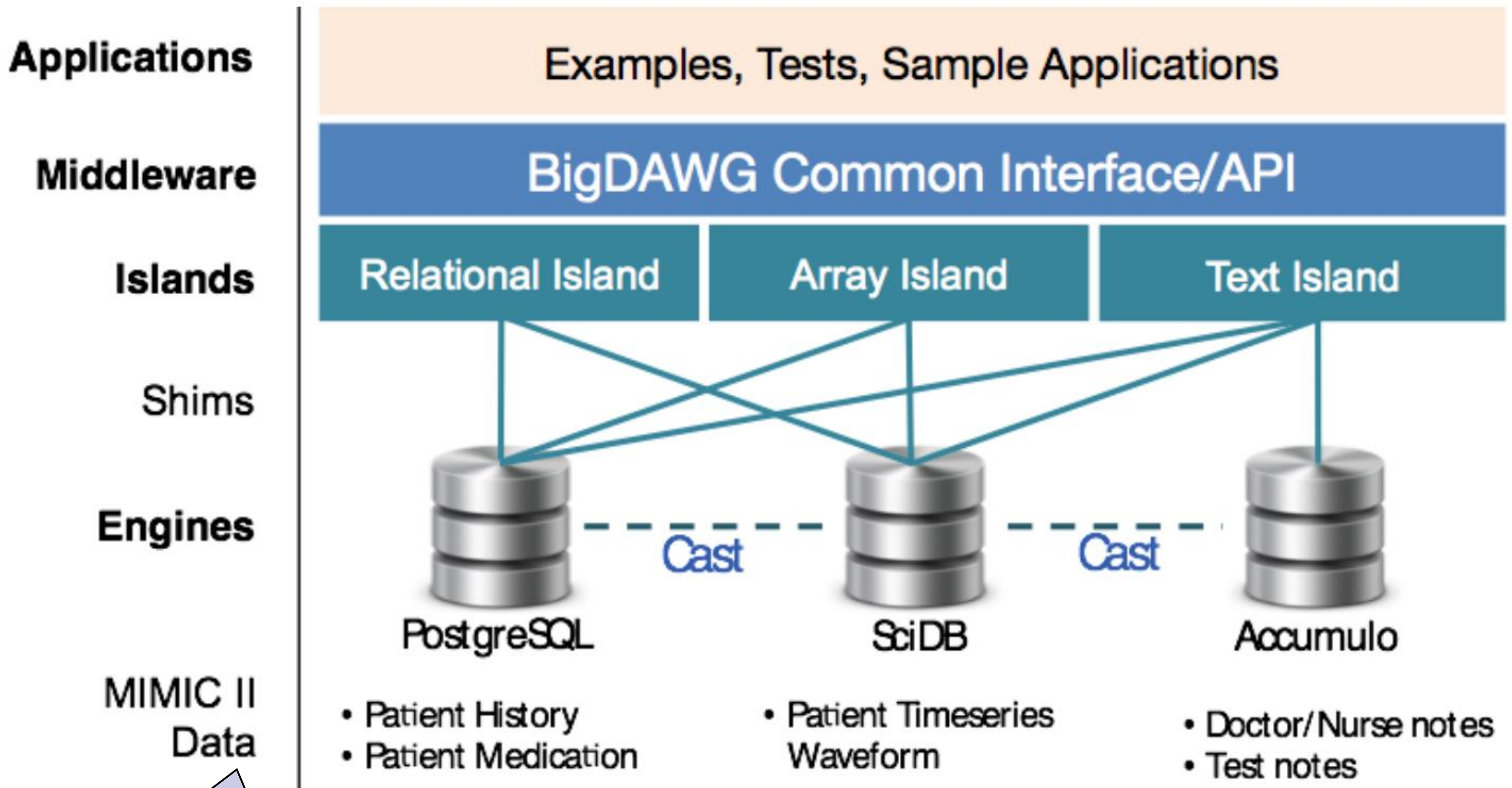


- A collection of data stores accessed with a single query language
- Key abstraction: island of information
  - Data model + operations + storage engine(s)
  - Cross-island queries
- Relies on a variety of data islands
  - Relational, array, NoSQL, streaming, …
  - Currently: PostgreSQL, SciDB, Accumulo
- No common data model, query language / processor
  - Each island has its own
- Shim connects an island to one or more storage engines
  - Maps queries from island language to the native query language of a particular storage engine (or engines)
- Cast = operators for moving datasets between islands
  - Processing in the storage engine best suited to the features of the data

Sorted, distributed key/value store

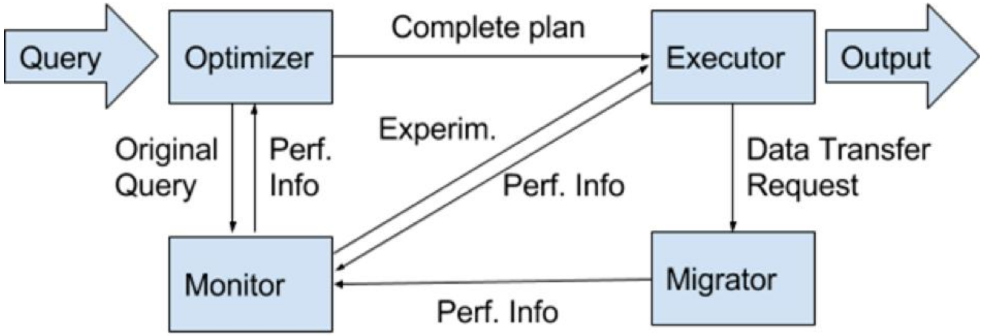
https://bigdawg.mit.edu/

BigDAWG



Openly available health data

BigDAWG



- At its core middleware that supports a common API to a collection of storage engines
- Key elements:
  - Optimizer: parses the input query and creates a set of viable query plan trees with possible engines for each subquery
  - Monitor: uses performance data from prior queries to determine the query plan tree with the best engine for each subquery
  - Executor: figures out how to best join the collections of objects and then executes the query
  - Migrator: moves data from engine to engine when the plan calls for such data motion

Another Classification

- Federated systems:
  - Collection of homogeneous data stores
  - Features a single standard query interface
- Polyglot systems:
  - Collection of homogeneous data stores
  - Exposes multiple query interfaces to the users
- Multistore systems:
  - Data across heterogeneous data stores
  - Supporting a single query interface
- Polystore systems:
  - Query processing across heterogeneous data stores
  - Supports multiple query interfaces

Tan et al. "Enabling query processing across heterogeneous data models: A survey". BigData 2017

Open Problems and Challenges

- Many challenges: query optimization, query execution, extensibility, interfaces, cross-platform transactions, self-tuning, data placement / migration, benchmarking, …
  - High degree of uncertainty
- Transparency: do not require users to specify where to get / store data, where to run queries / subqueries
  - Explain and allow user hints
- More than ever need for automation, adaptiveness, learning on the fly

References

- Kolev, B. et al.: Benchmarking Polystores: the CloudMdsQL Experience
  - https://hal-lirmm.ccsd.cnrs.fr/lirmm-01415582/file/CloudMdsQL-IEEE_v.0.4.1.pdf
- Kharlamov, E. et al.: A Semantic Approach to Polystores
  - http://www.cs.ox.ac.uk/ian.horrocks/Publications/download/2016/KharlamovMBBBJL16.pdf
- Karimov, J. et al.: PolyBench: The First Benchmark for Polystores
  - http://www.redaktion.tu-berlin.de/fileadmin/fg131/dima-feed/polystore_benchmark_TPCTC-1028_crv.pdf
- Meehan, J. et al.: Integrating Real-Time and Batch Processing in a Polystore
  - https://cs.brown.edu/courses/cs227/papers/bigdawg-integration.pdf
- Bondiombouy,C. et al.: Query Processing in Multistore Systems: an overview
  - https://hal.inria.fr/hal-01289759v2/document