Bigdatastack.eu

@bigdatastackeu

BigDataStack

Holistic stack for big data applications and operations

Seamless Analytical Framework

Pavlos Kranas

LeanXcale

pavlos@leanxcale.com

Co-funded by the European Commission
Horizon 2020 - Grant # 779747

European Commission

Seamless Analytical Framework

- Modern enterprises use
  - Operational databases for OLTP load
  - Key-Value for IoT data
  - Data warehouses for data analytics
  - Datalakes
  - etc ...
- Need polyglot capabilities

Seamless Analytical Framework

- Nowadays: Data Federation using Spark

Seamless Analytical Framework

- Nowadays: Data Federation using Spark

- BUT:
  - Can be very resource consuming
  - Cannot exploit the specific capabilities of each different datastore

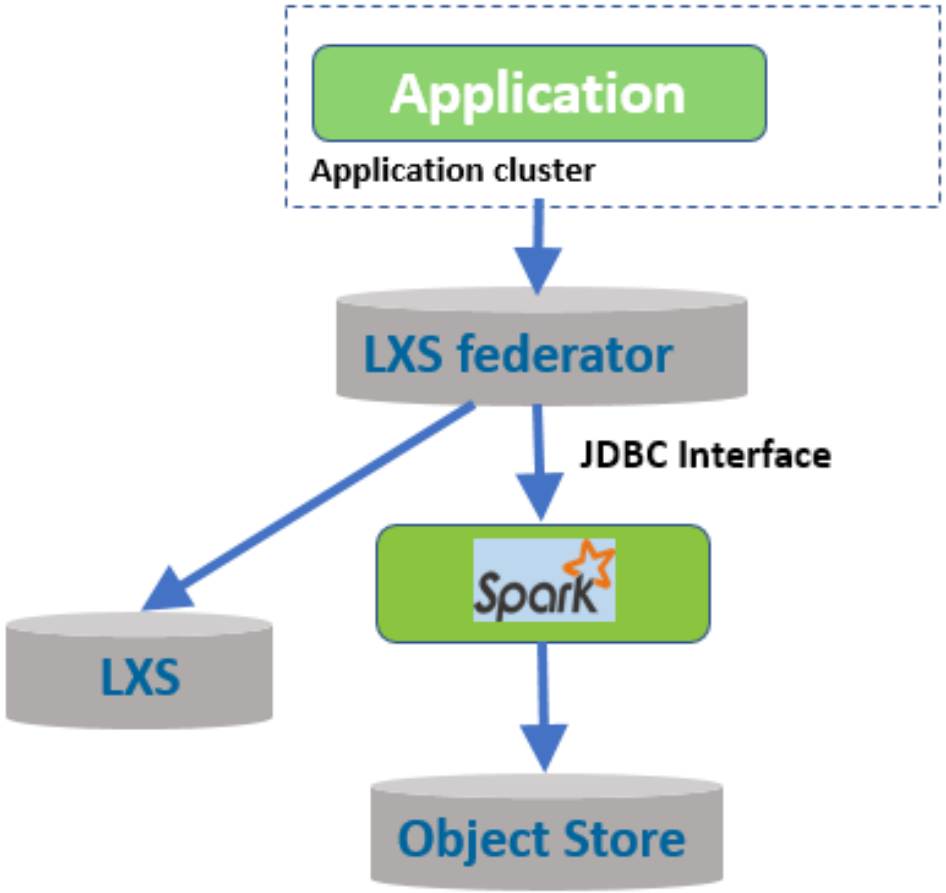
Seamless Analytical Framework – User Story

- Data ingestion in operational datastore (LeanXcale)
- Old data becomes historical, with no modifications
- Data Warehouse to perform analytics on big data volumes
- Distribution of datasets is problematic
  - Data to be retrieved from both stores
  - To be merged in the application level
  - Data consistency considerations when moving datasets

Seamless Analytical Framework - Solution

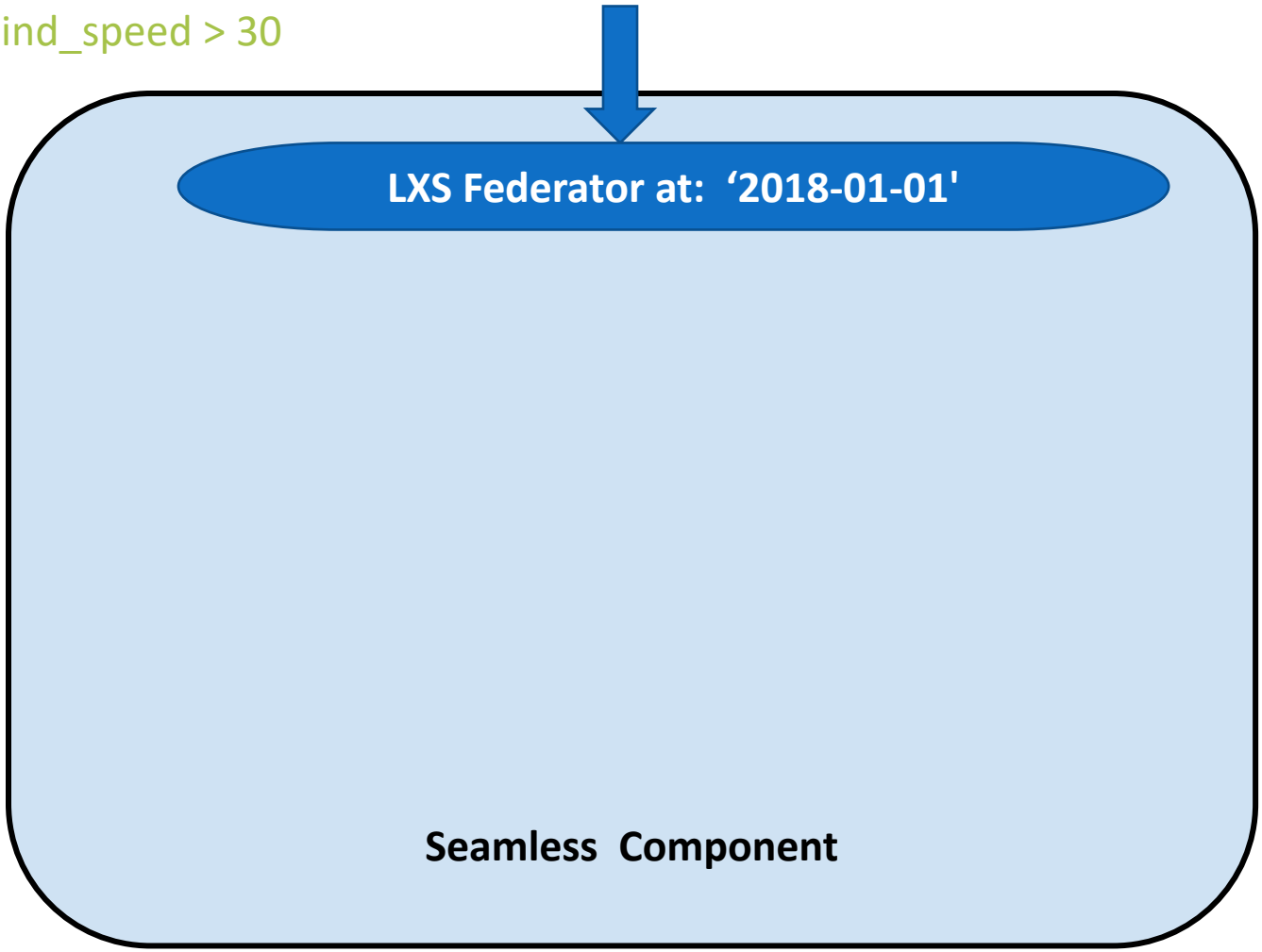
- Seamless Analytical Framework
  - Federate data coming from two different datastores:
    - HTAP Relational LXS Datastore
    - IBM Object store
  - sharing the **SAME** dataset
  - Single (black box) component that
    - consists of two datastores
    - exploits unique characteristics of each one
    - transparently from the user
    - does not compromise some requirements for the benefits of others

Query Federation



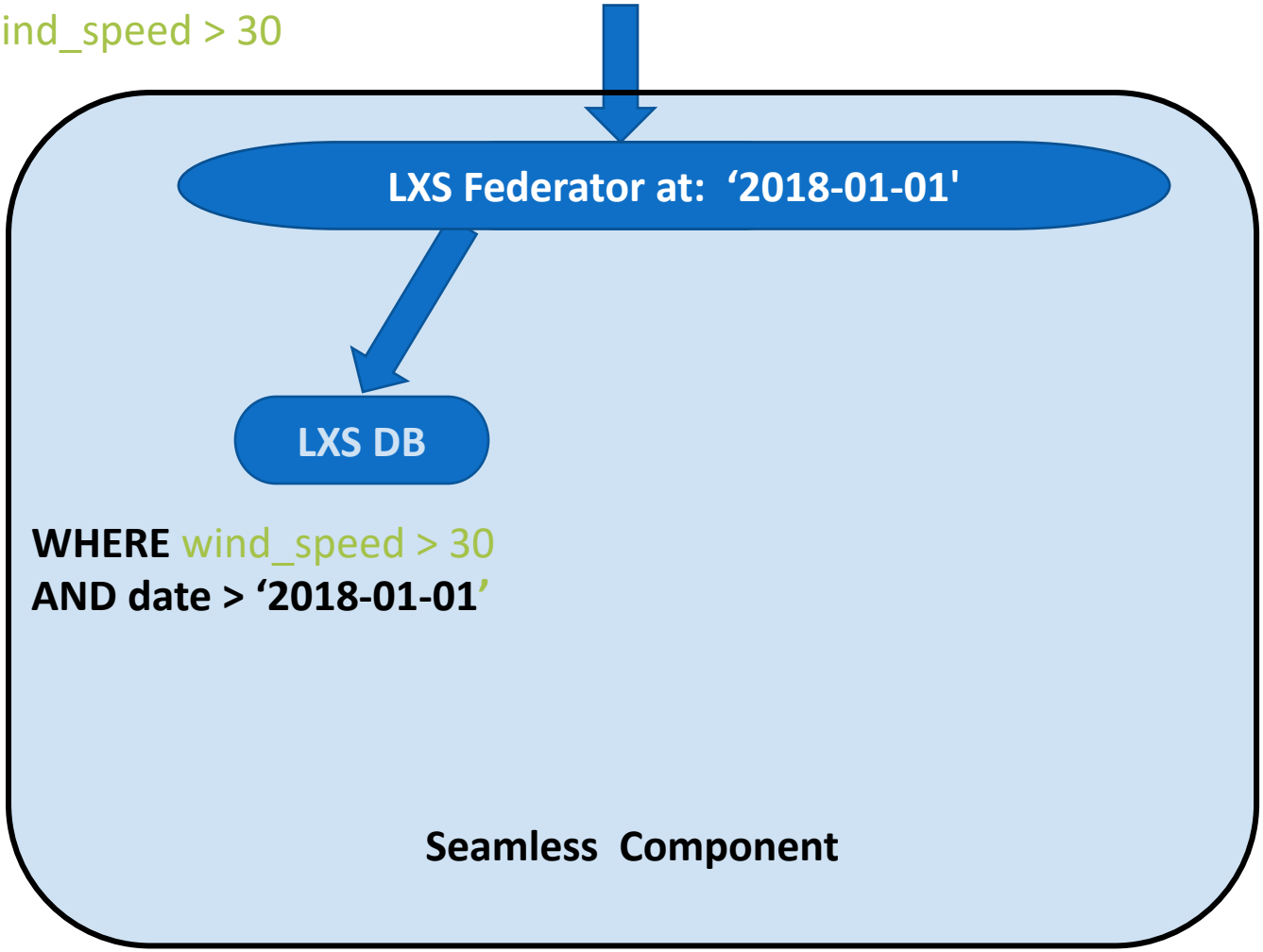
Query path

SELECT vessel_code, datetime, longitude, latitude, wind_speed
FROM danaos
WHERE wind_speed > 30



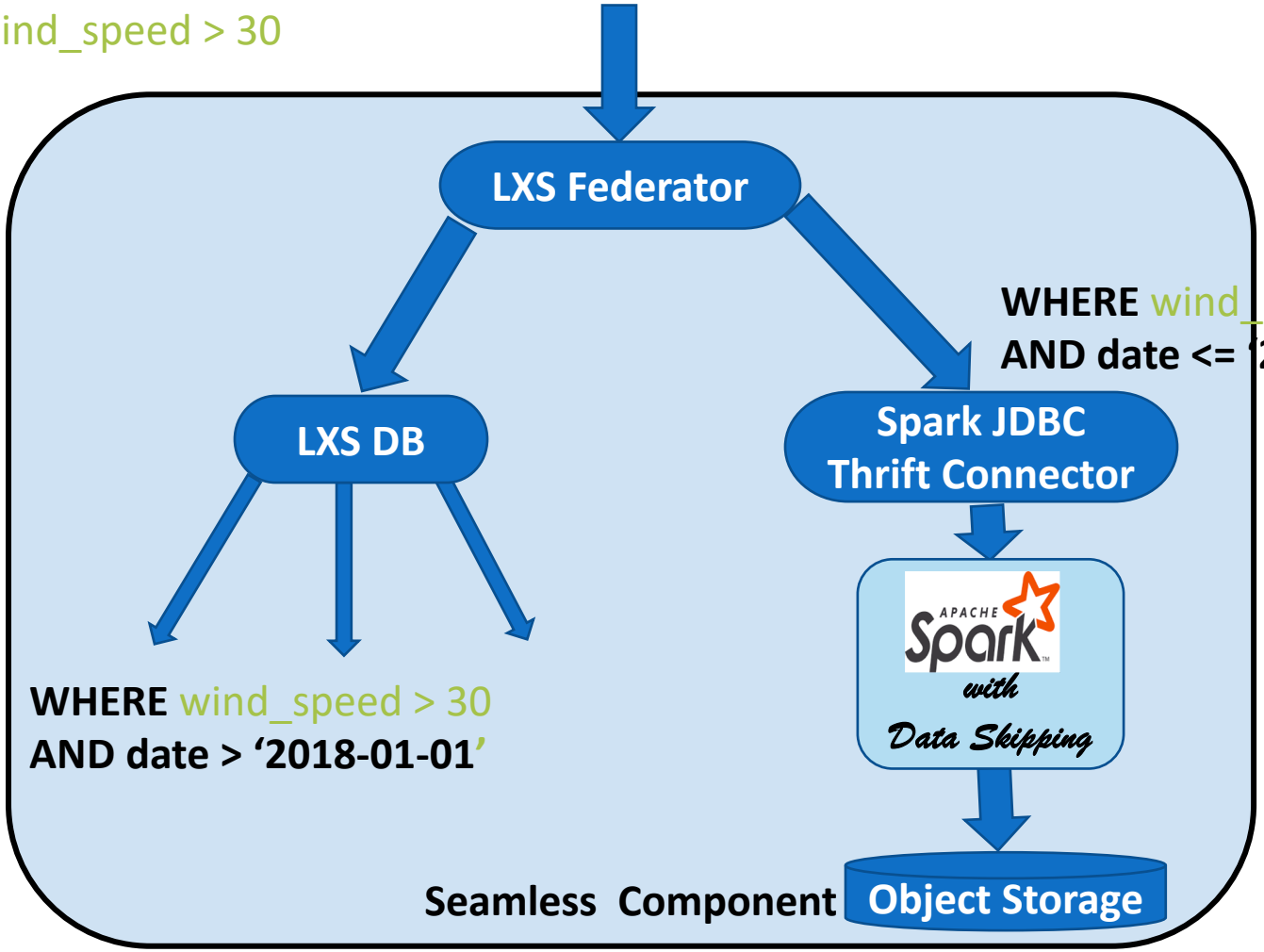
Query path

SELECT vessel_code, datetime, longitude, latitude, wind_speed
FROM danaos
WHERE wind_speed > 30



Query path

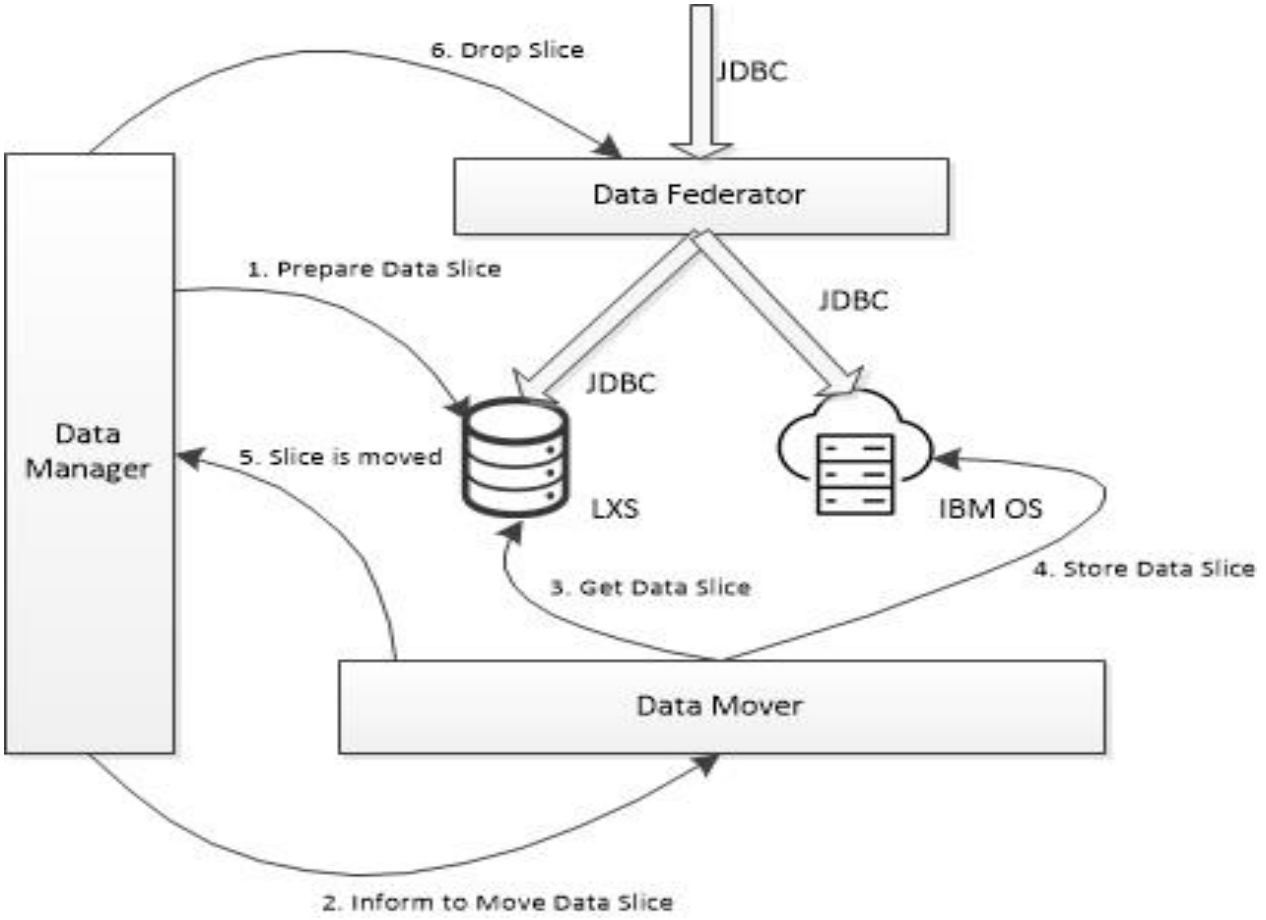
SELECT vessel_code, datetime, longitude, latitude, wind_speed
FROM danaos
WHERE wind_speed > 30



Supported Operations

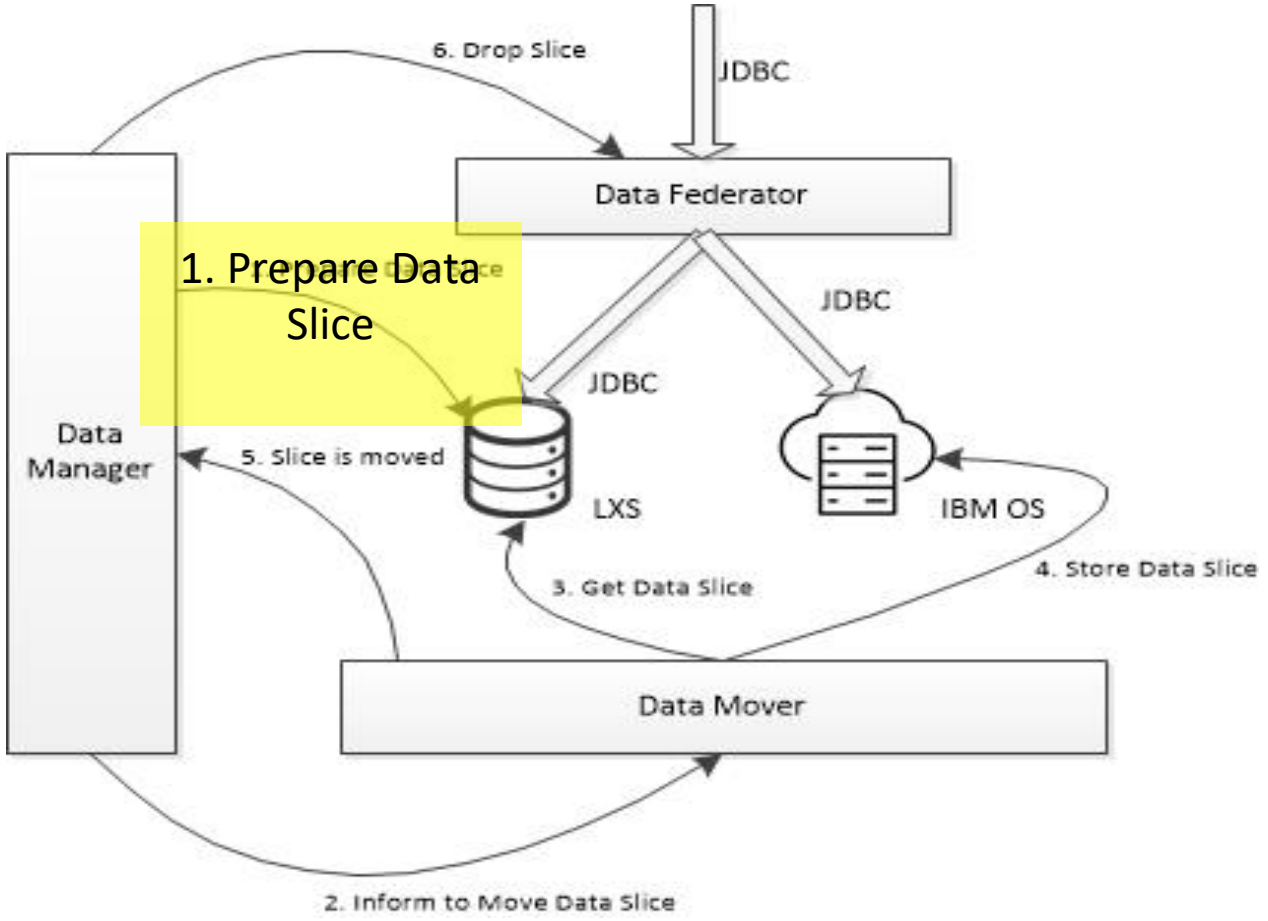
- Currently supports
  - Full Scan
  - Ordered Scan
  - LIMIT
  - Aggregations
  - Group By Aggregations
  - Ordered Group By Aggregations
- Does not yet support
  - JOIN on fragmented data tables

Data Movement High Level



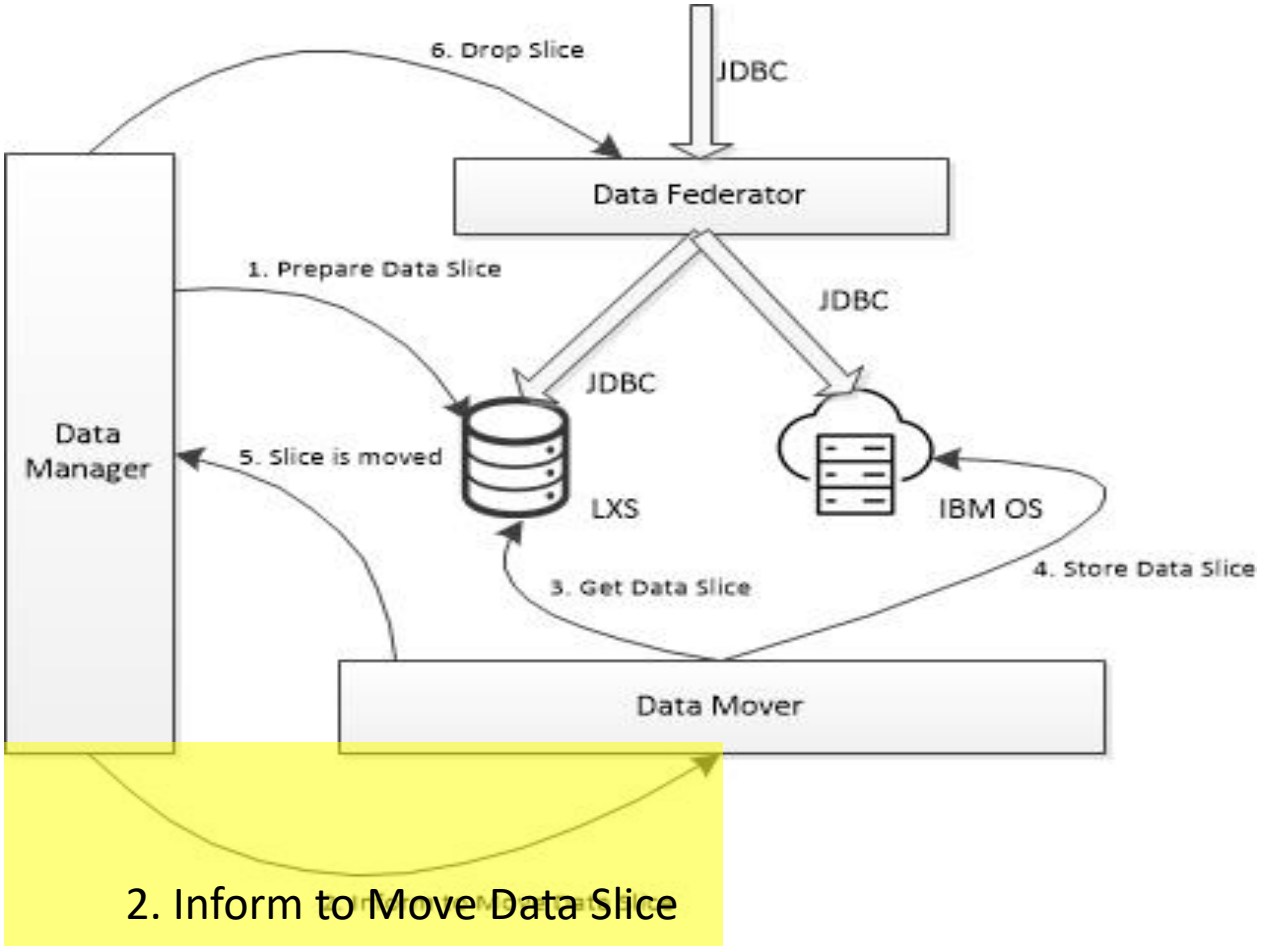
Data Movement High Level

- **Prepare Data Slice** – Data Manager informs LXS to move data slice to a new data region, so that it can easily drop it later



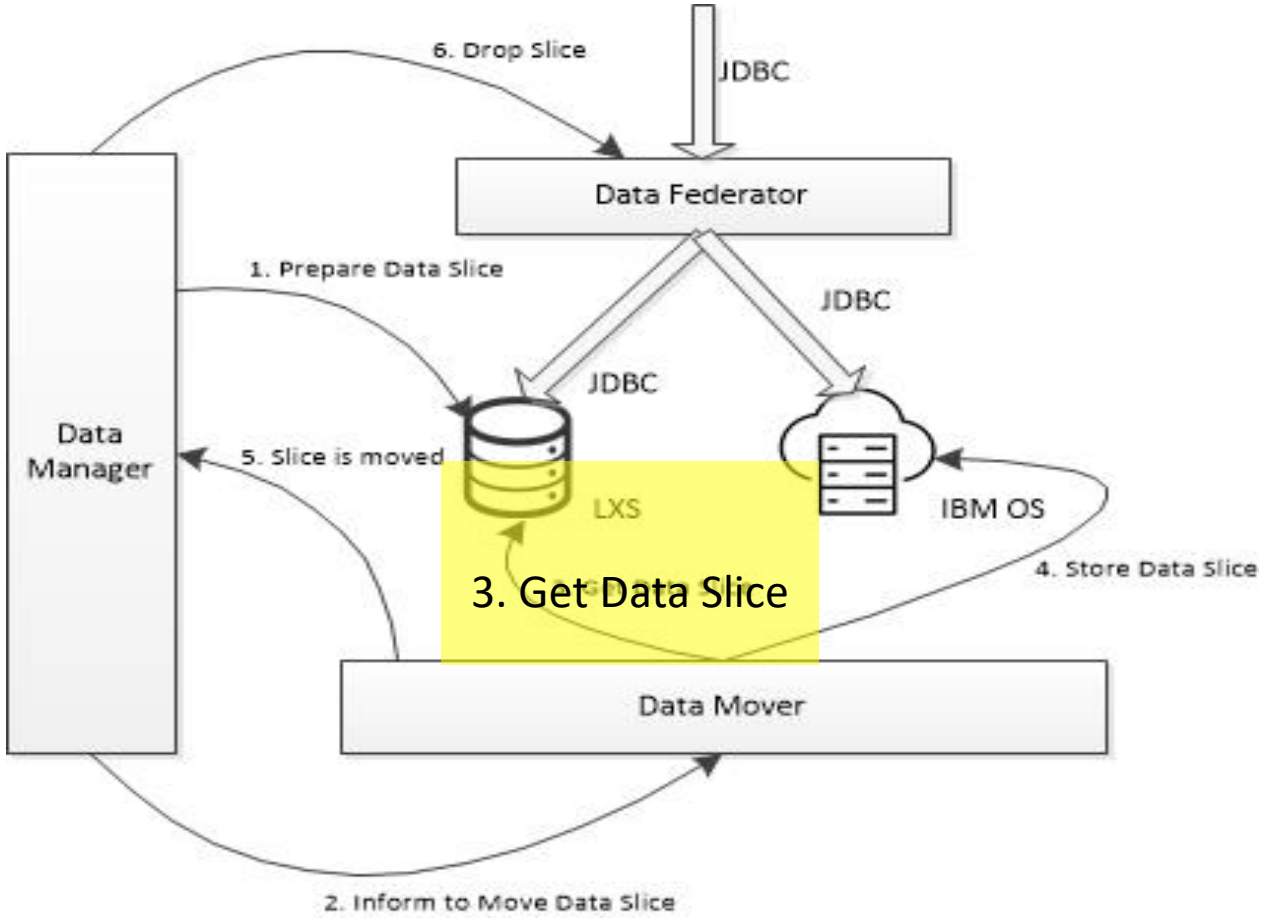
Data Movement High Level

- **Inform to Move Data Slice** – Data Manager informs the Data Mover that it can start the movement process, sending the SQL statement to grab the data slice from LXS



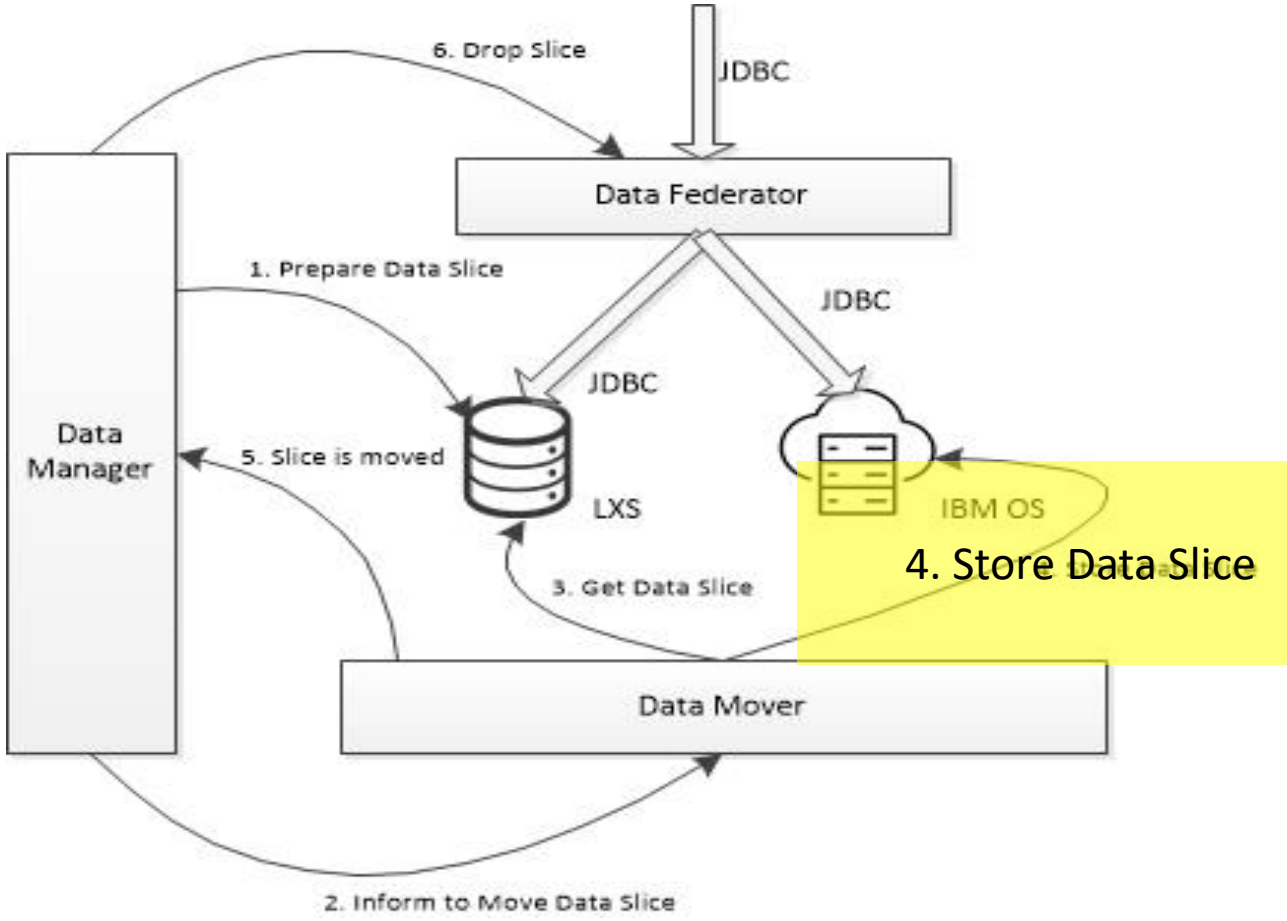
Data Movement High Level

- **Get Data Slice –** Upon receiving a message from the RabbitMQ the data mover fetches the data slice from LXS via JDBC



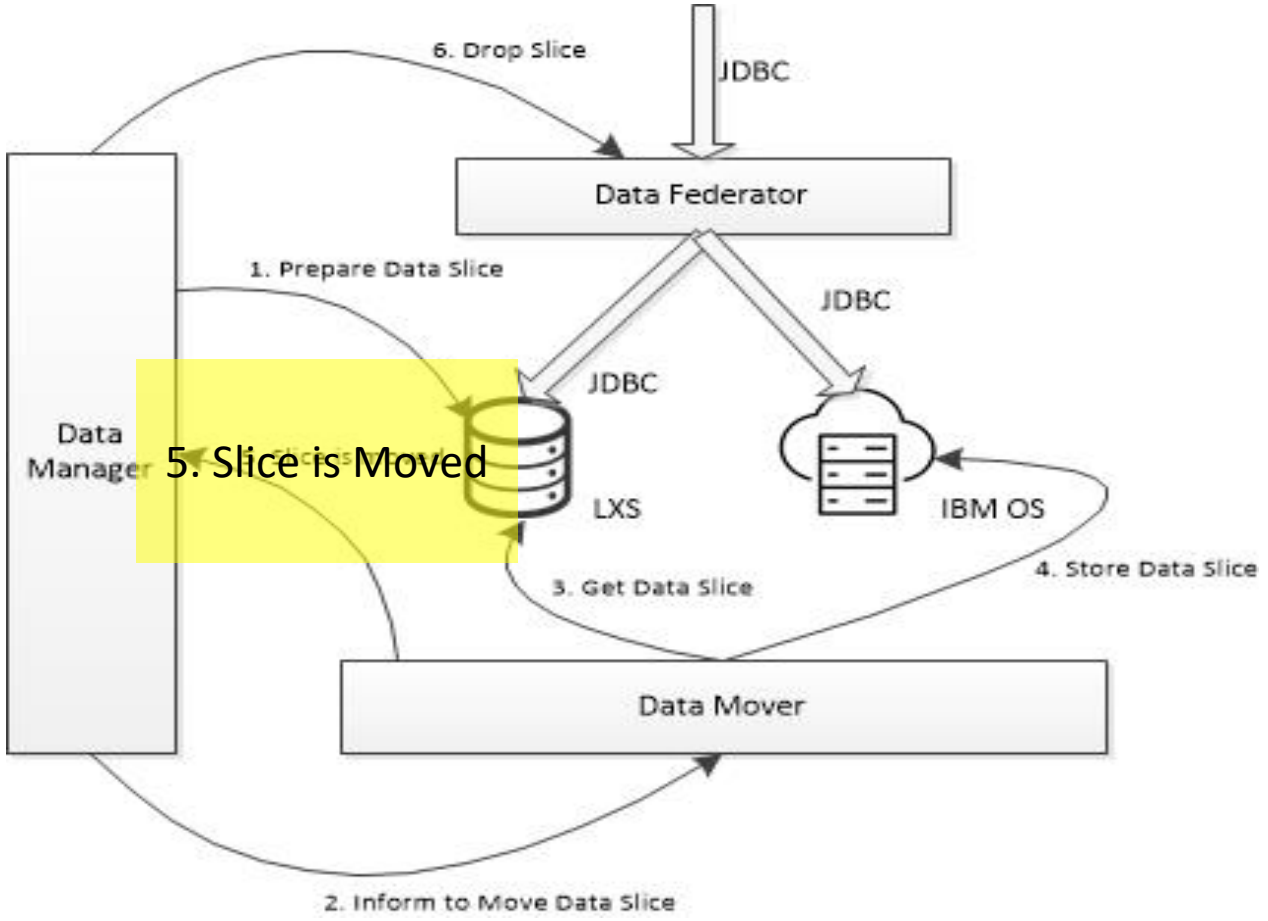
Data Movement High Level

- **Store Data Slice** – Data Mover layout the data, build data skipping indexes and stores the data to Object Storage



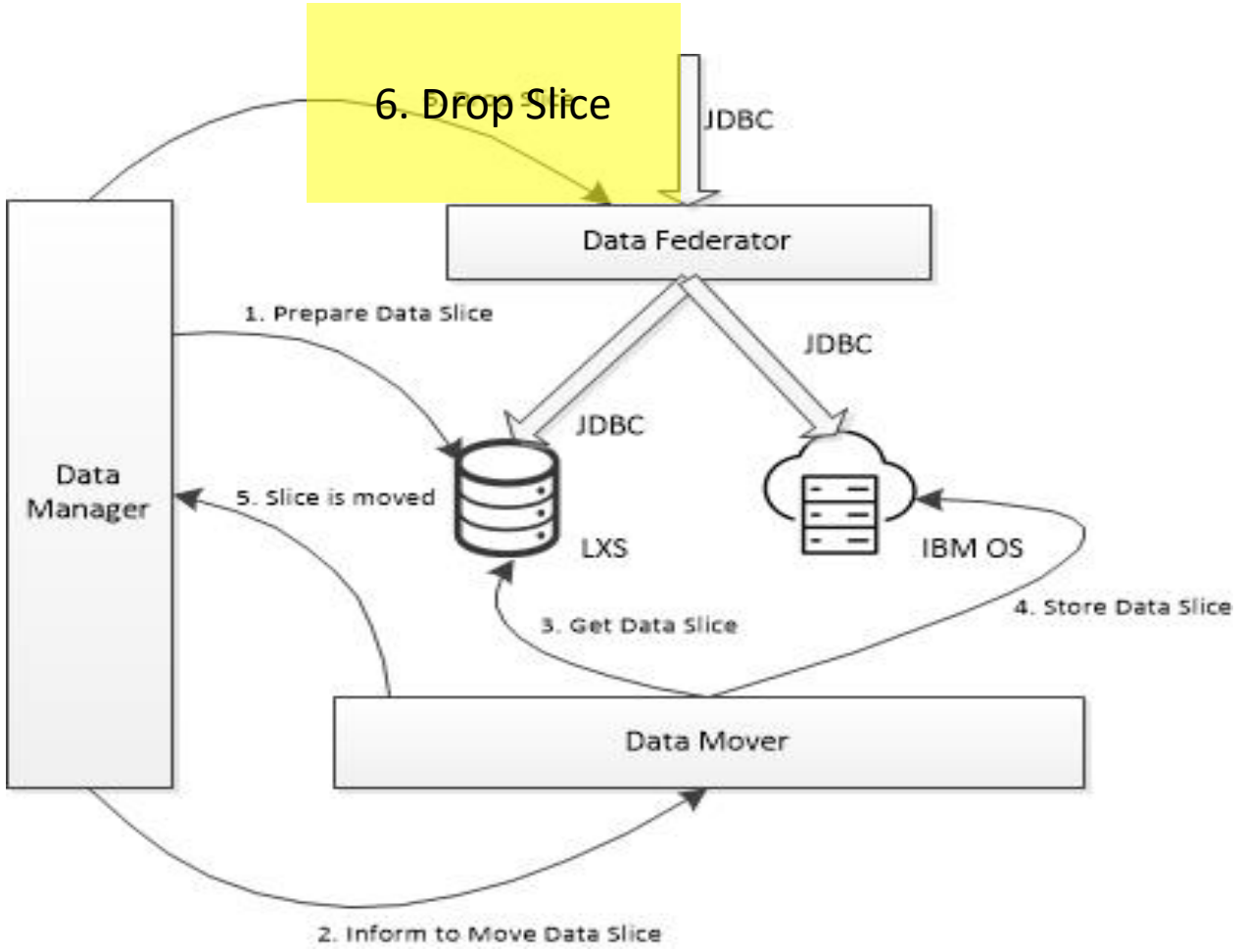
Data Movement High Level

- **Slice is moved** – After data is persisted in Object Store, an ACK is sent to the Data Manager which then informs the federator to adjust the split point and drop the slice from LXS



Data Movement High Level

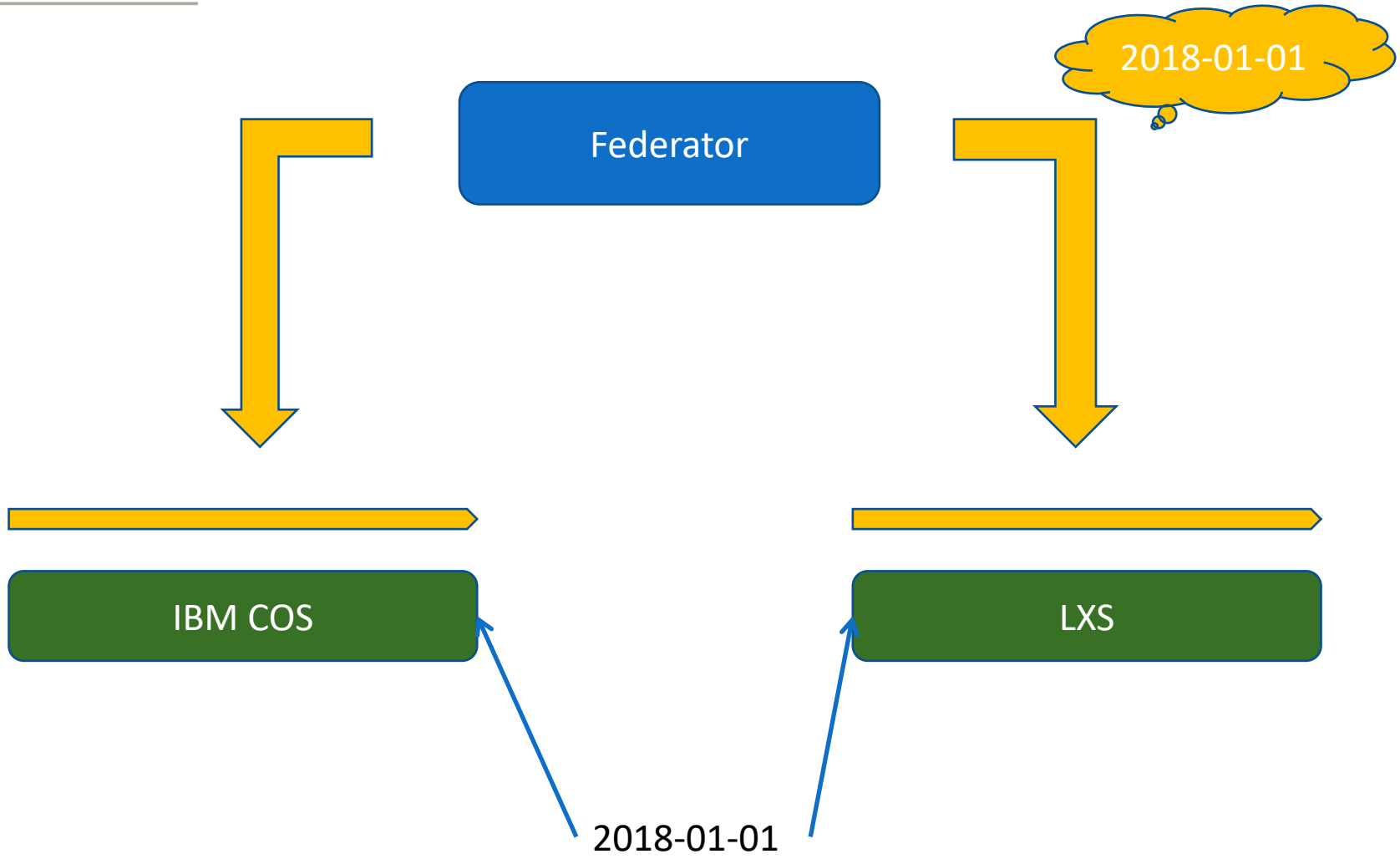
- **Drop Slice –** Federator forwards the split point and requests LXS to drop the slice



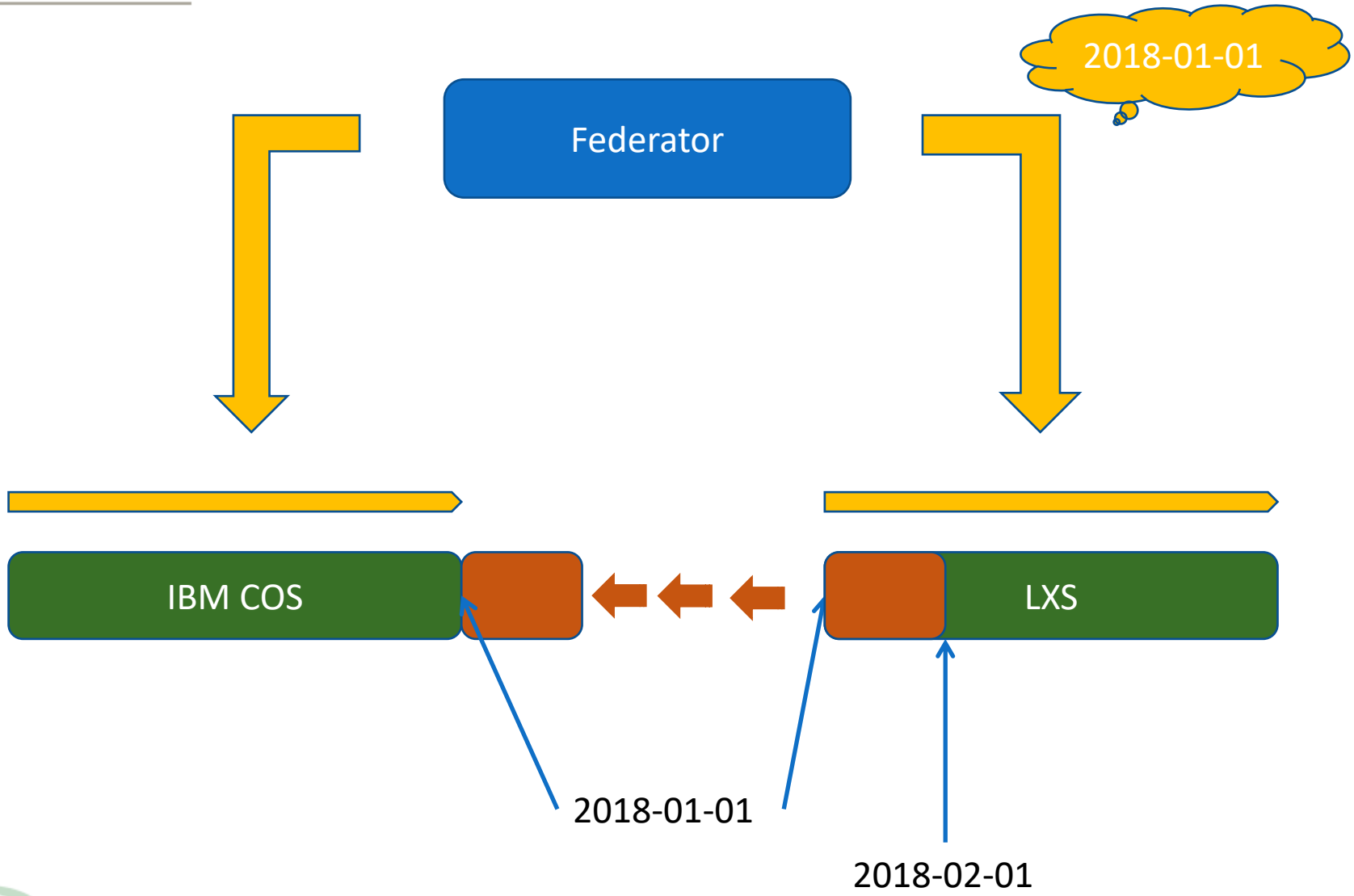
Seamless Analytical Framework

Thank you!!!

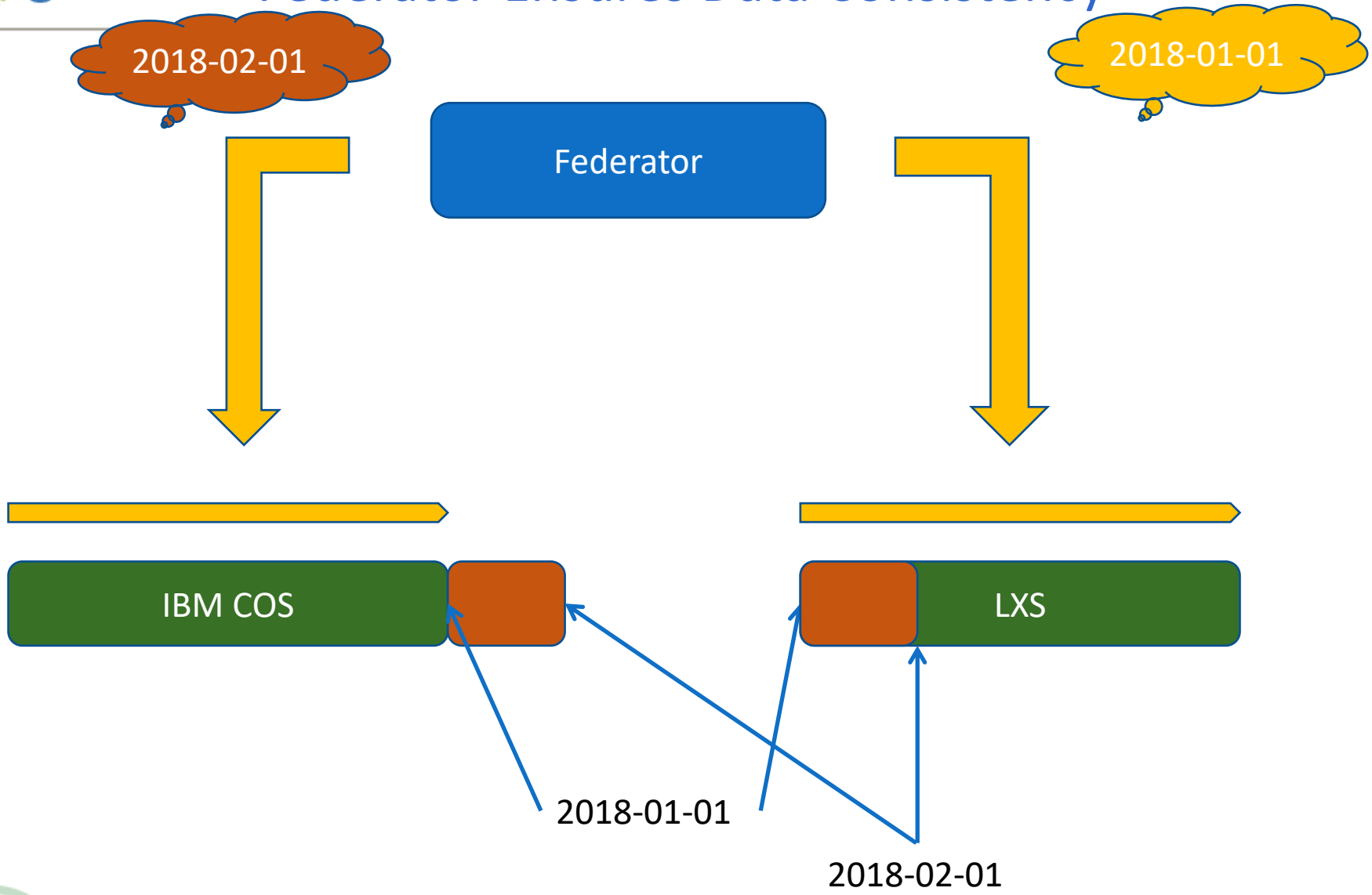
Federator Ensures Data Consistency



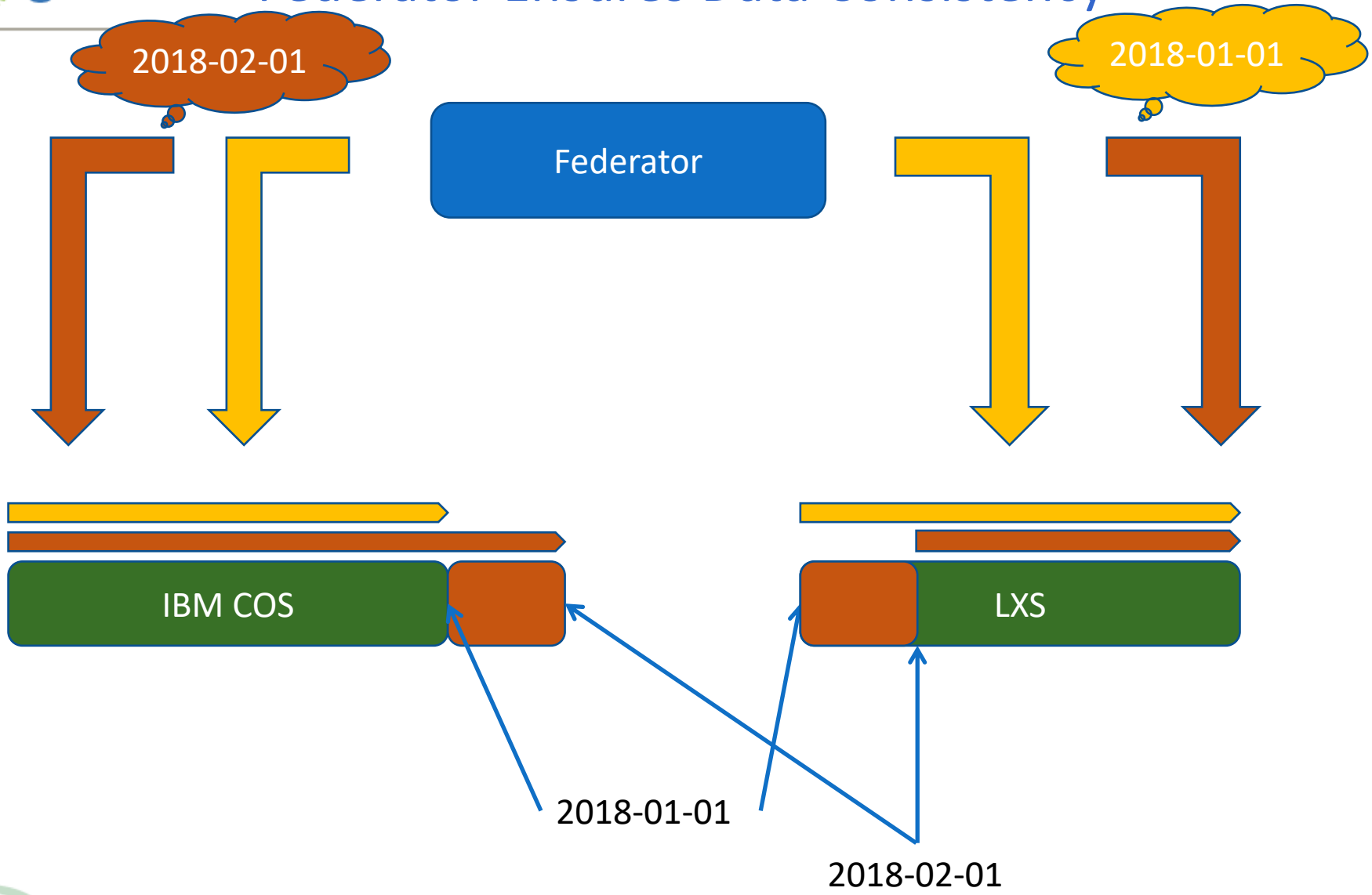
Federator Ensures Data Consistency



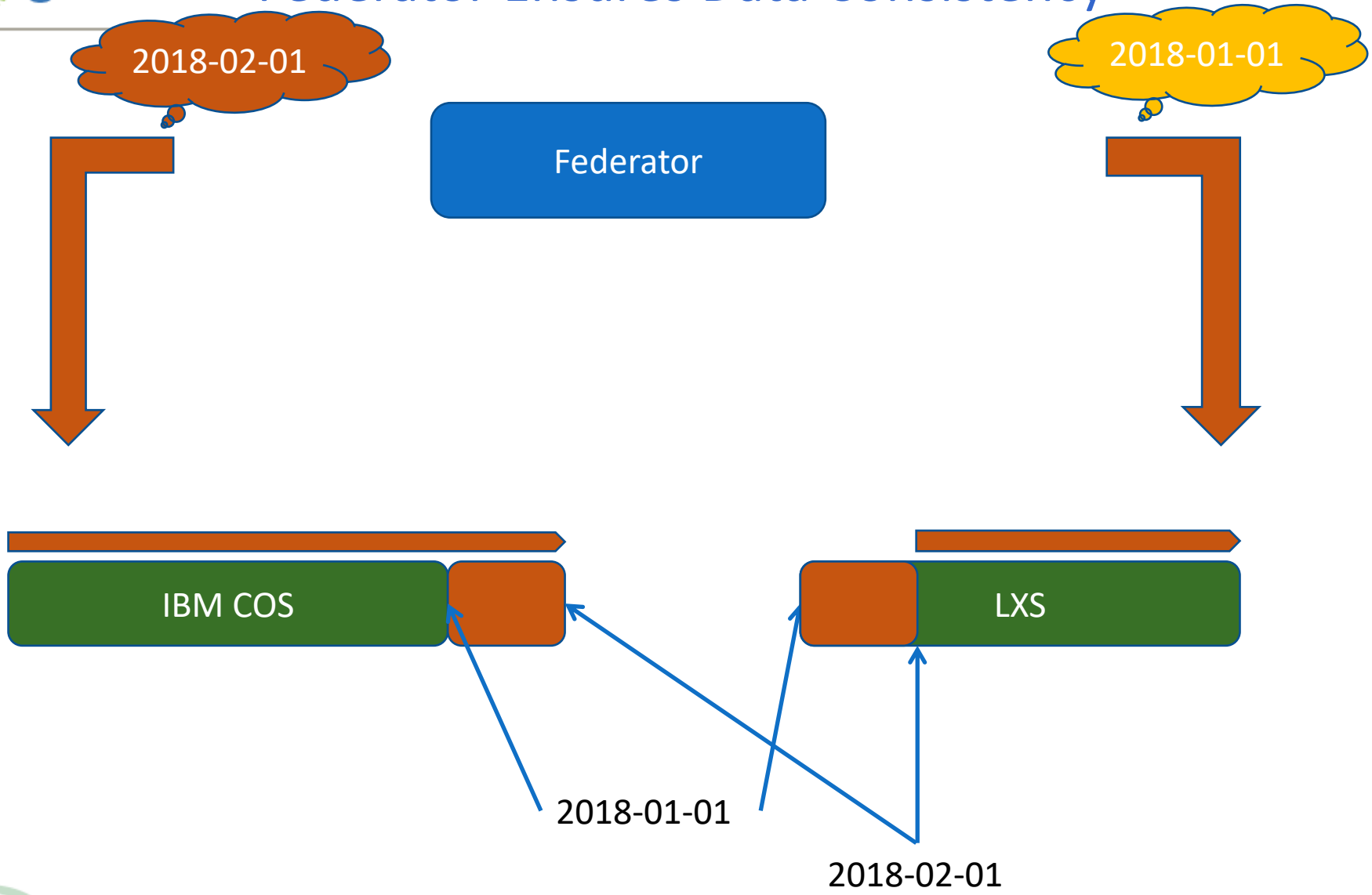
Federator Ensures Data Consistency



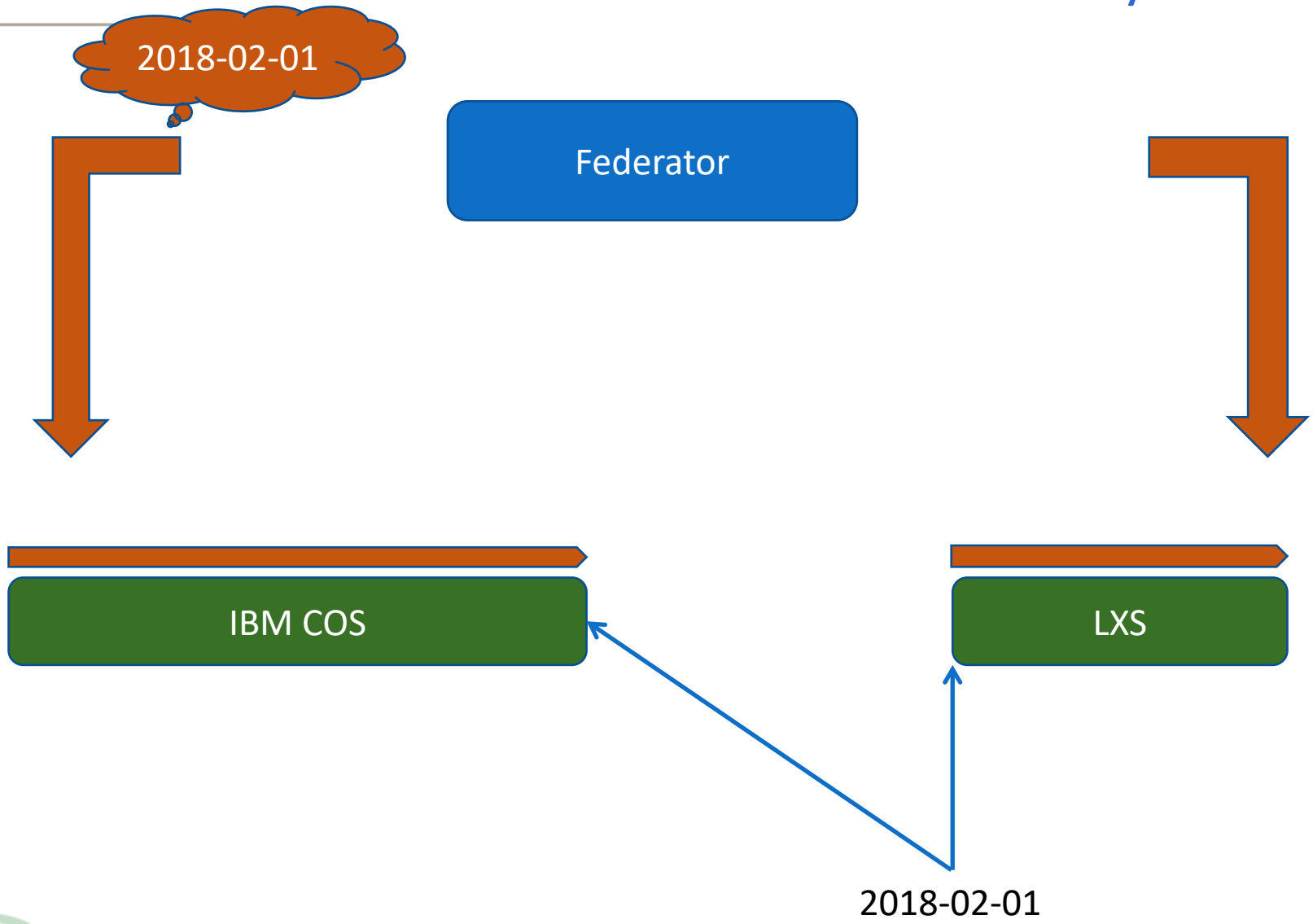
Federator Ensures Data Consistency



Federator Ensures Data Consistency



Federator Ensures Data Consistency



Next Steps

- Support of the JOIN operation on split tables between the stores
  - Both Stores natively support JOIN operations
  - Query Federator can push down the operation and merge results
  - Strategies to take into account data locality to reduce amount of data to be sent
  - JOIN can be translated as the union of 4 separate JOINS
  - Bind Join can be considered for optimize the execution