Data Integration

Oscar Romero
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

A Framework to Characterize the Problem

DATA INTEGRATION

The Variety Challenge

- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:

The Variety Challenge

- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:
  - Extract, transform and load data from the sources
  - Integrate the data in a common repository
  - Query and exploit the data

The Variety Challenge

- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:
  Dealing with *external* data as first-class citizen
  - Extract, transform and load data from the sources
  - Integrate the data in a common repository
  - Query and exploit the data

The Variety Challenge

- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:
  Dealing with *external* data as first-class citizen
  - Extract, transform and load data from the sources **analyse any external source; largely automate the process to facilitate incorporating new sources**
  - Integrate the data in a common repository
  - Query and exploit the data

The Variety Challenge

- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:
  Dealing with *external* data as first-class citizen
  - Extract, transform and load data from the sources **analyse any external source; largely automate the process to facilitate incorporating new sources**
  - Integrate the data in a common repository
    ***Right-time* data integration (i.e., automation is a must)**
  - Query and exploit the data

The Variety Challenge

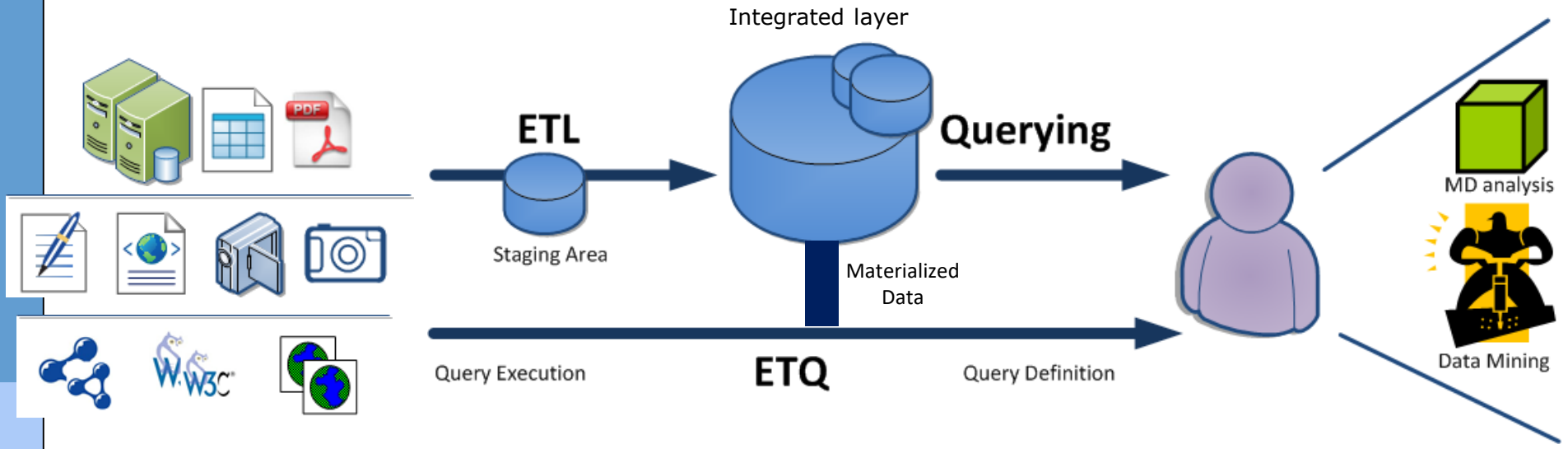
- There is no formal framework to characterize the Variety Challenge
- However, they can easily be reduced to the Data Integration theoretical problem but:

Dealing with *external* data as first-class citizen

  - Extract, transform and load data from the sources **analyse any external source; largely automate the process to facilitate incorporating new sources**
  - Integrate the data in a common repository **_Right-time_ data integration (i.e., automation is a must)**
  - Query and exploit the data **Deterministic Vs. Undeterministic "queries"; Personalisation / recommendations**

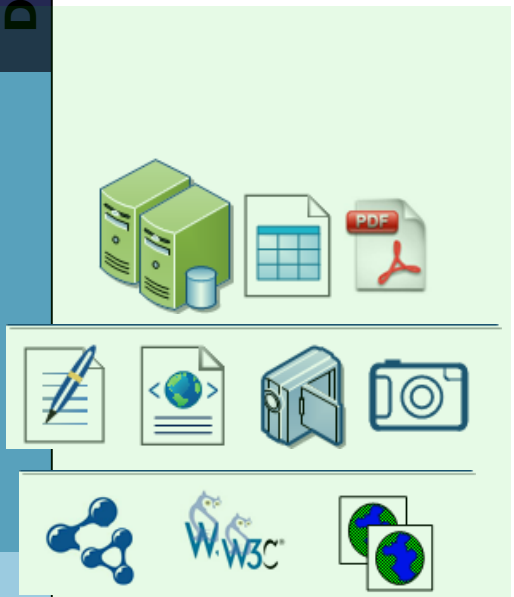
The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:

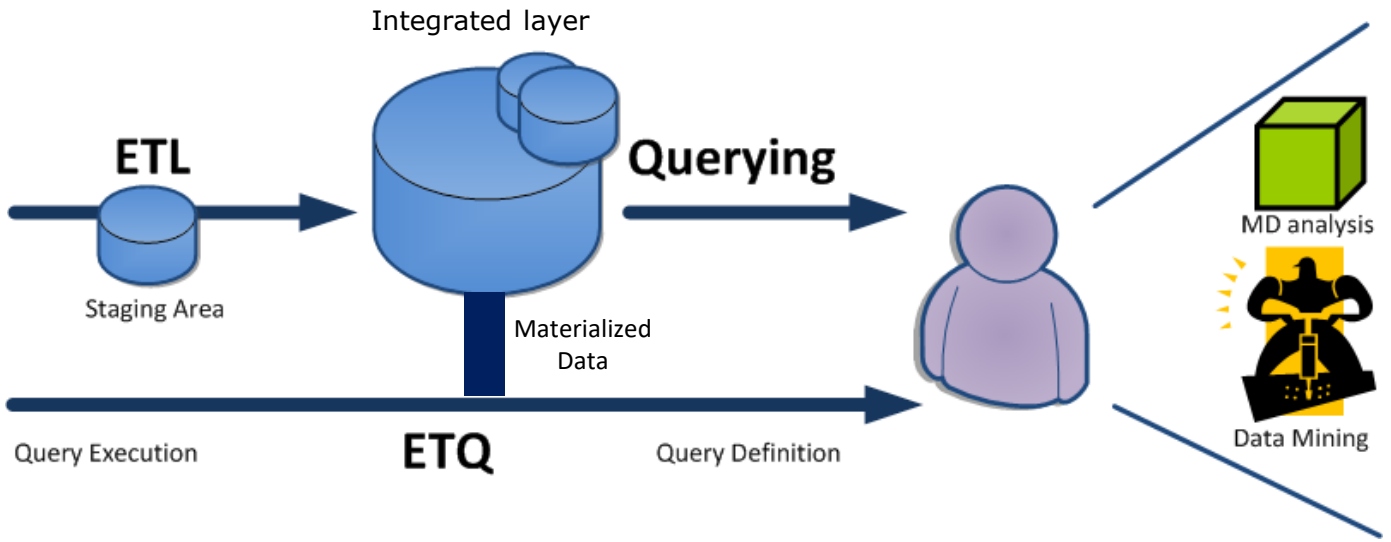


The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:

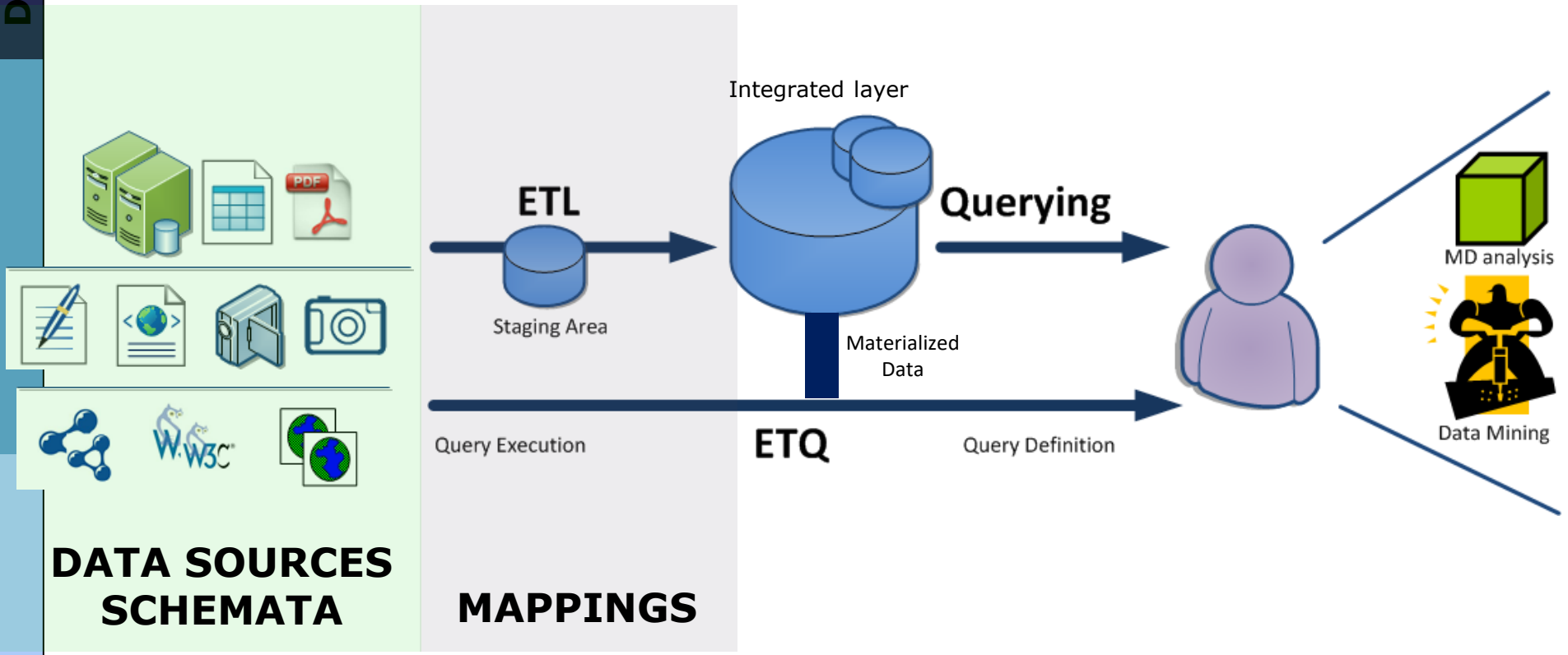


DATA SOURCES SCHEMATA



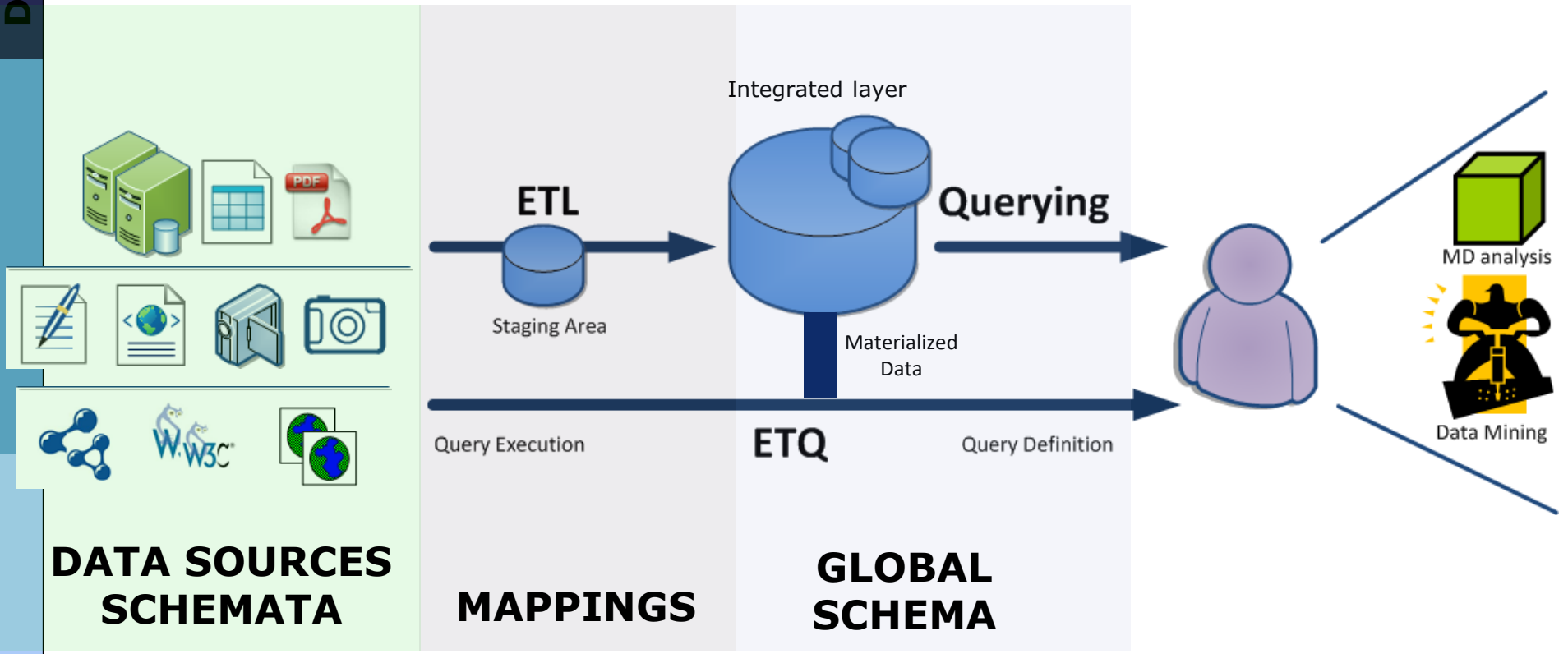
The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:



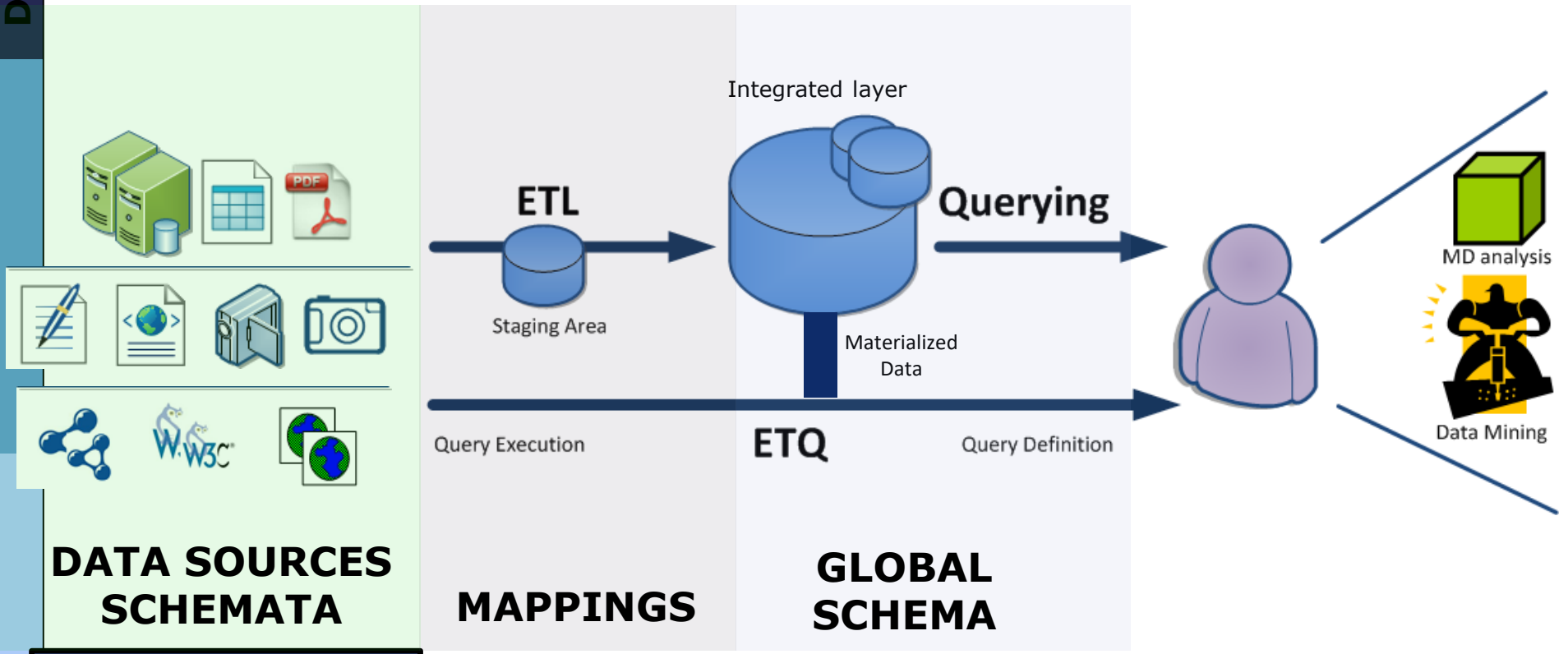
The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:



The Theoretical Problem: Data Integration

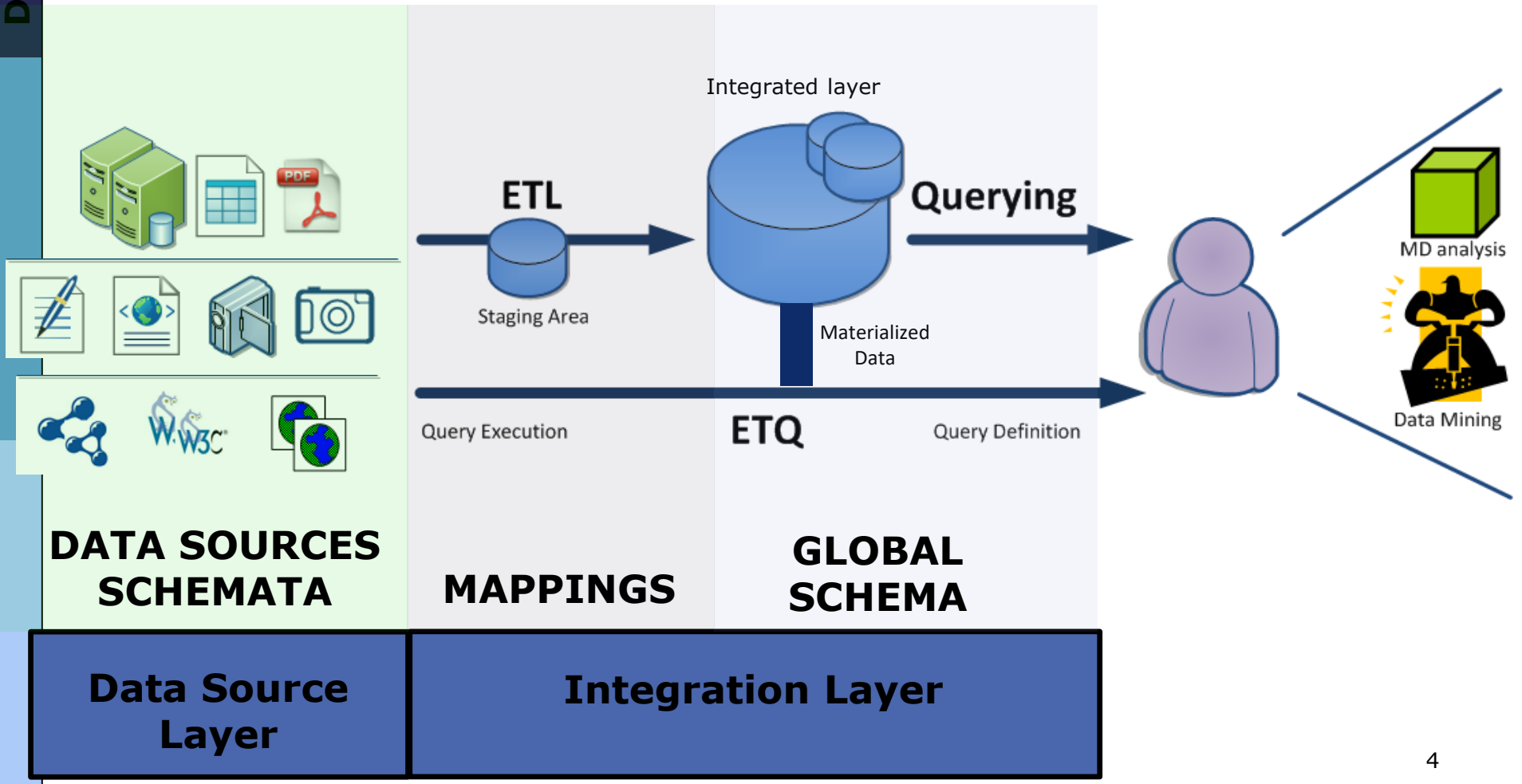
3 MAIN CONSTRUCTS:



Data Source Layer

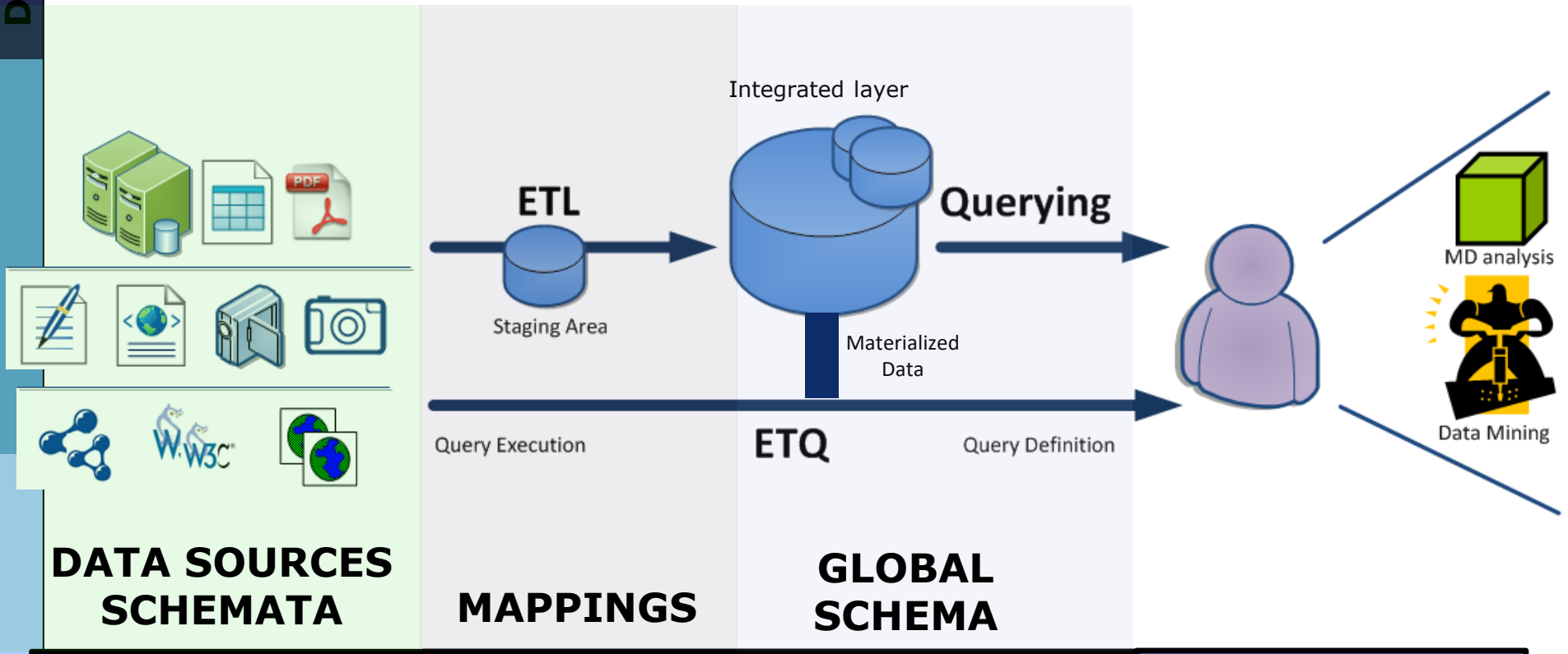
The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:



The Theoretical Problem: Data Integration

3 MAIN CONSTRUCTS:



Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*

Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*

USER FEEDBACK PRODUCT INFO USER WEB BEHAVIOUR

Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*

Twitter API
(JSON)

USER FEEDBACK

In-house DB
(PostgreSQL)

PRODUCT INFO

Web Logs
(Logs)

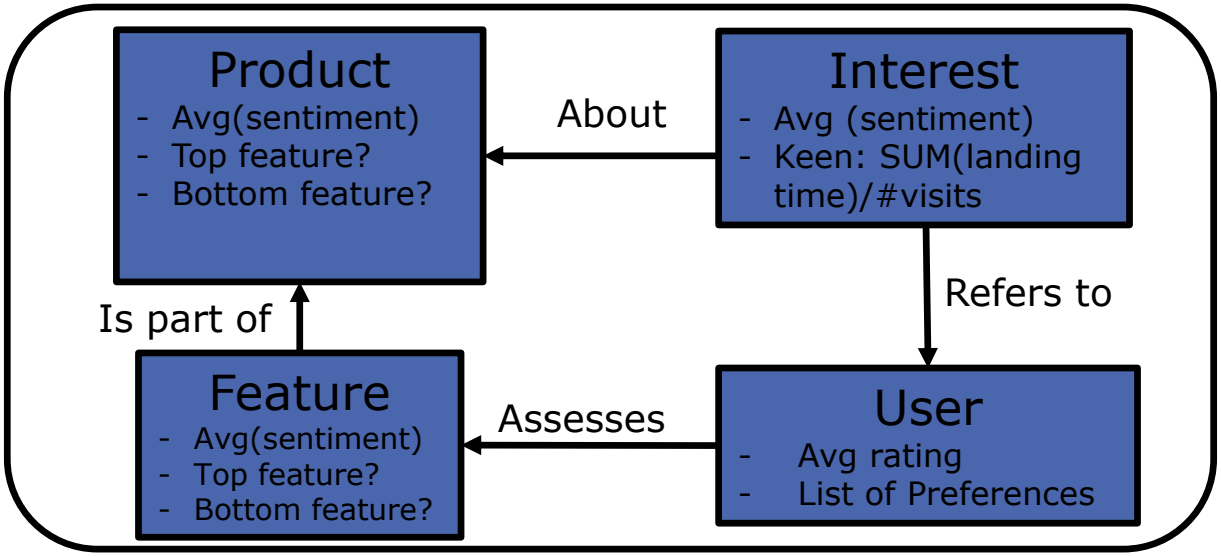
USER WEB
BEHAVIOUR

Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*

Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*



- User
- Tweet
- Date
- Location

Twitter API (JSON)

USER FEEDBACK

- Product
- Product features
- User info

In-house DB (PostgreSQL)

PRODUCT INFO

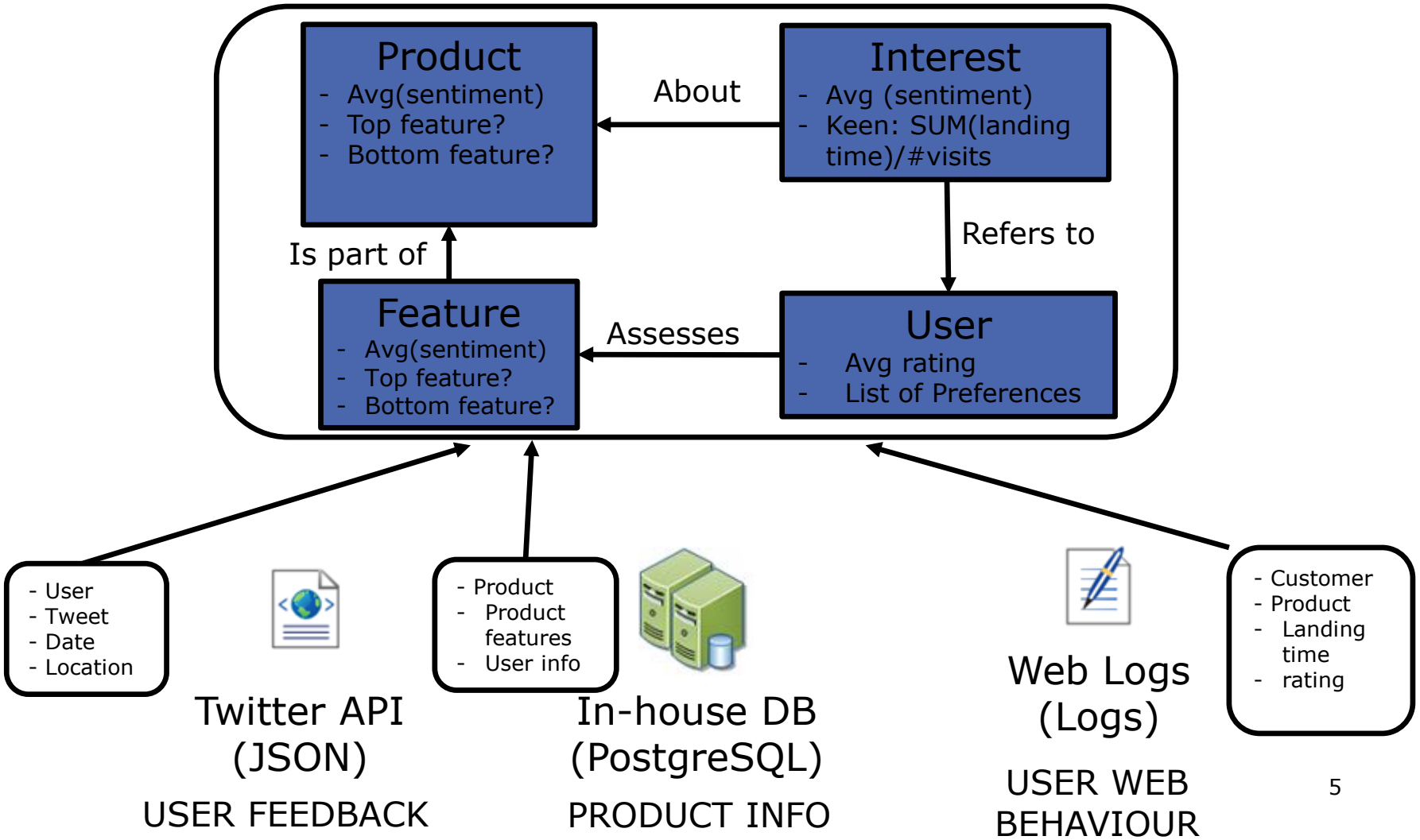
Web Logs (Logs)

USER WEB BEHAVIOUR

- Customer
- Product
- Landing time
- rating

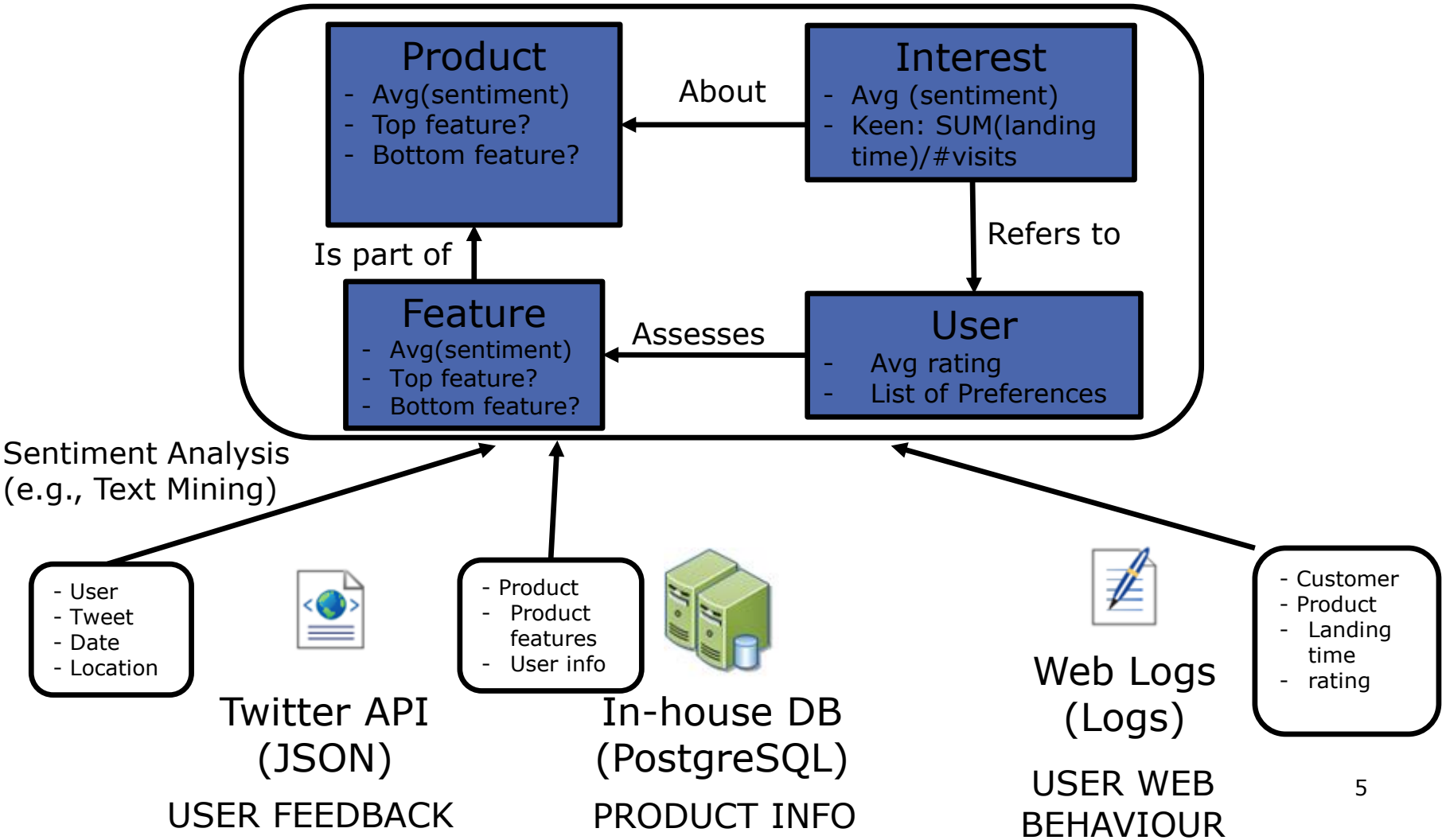
Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*



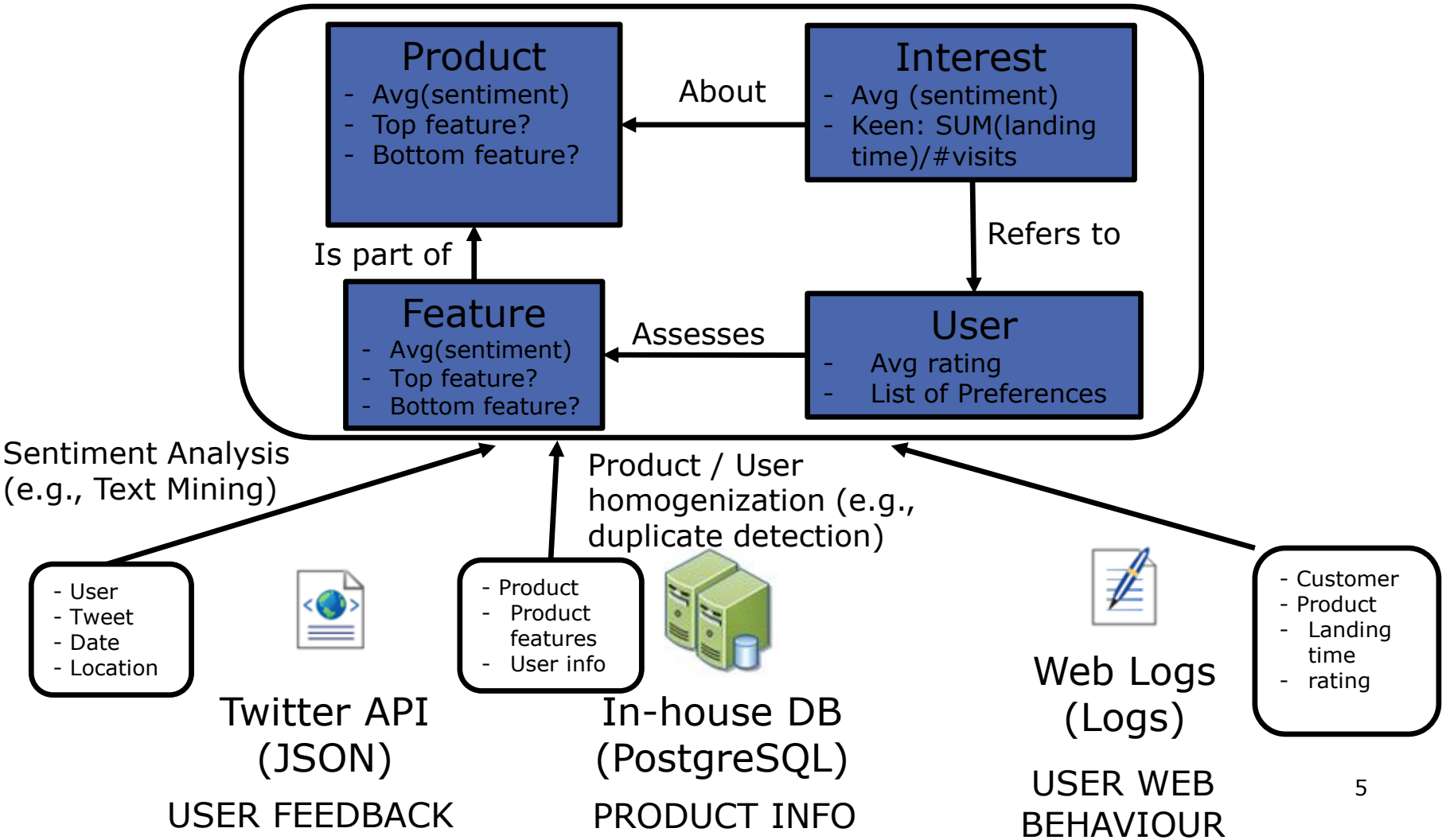
Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*



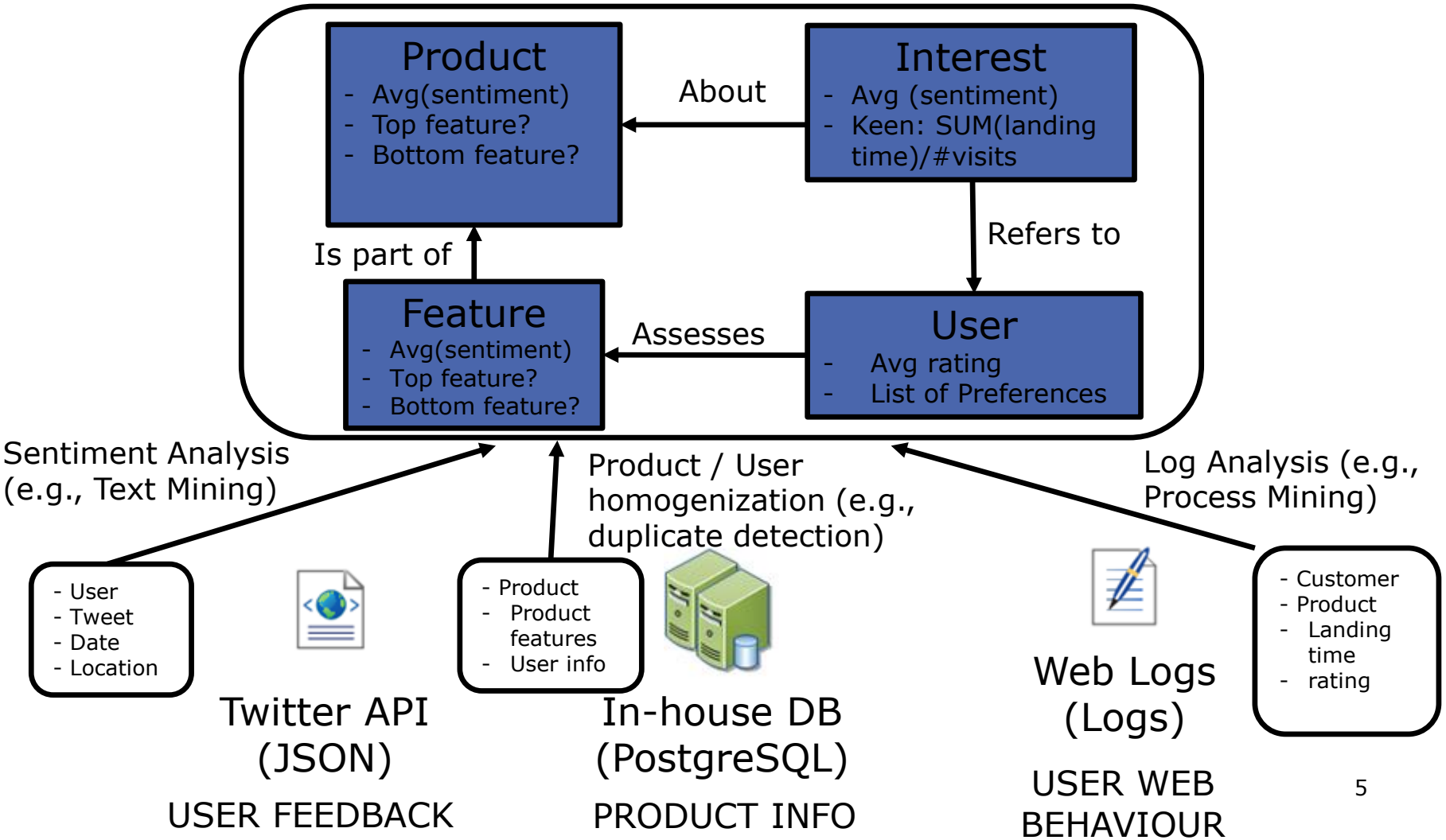
Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*



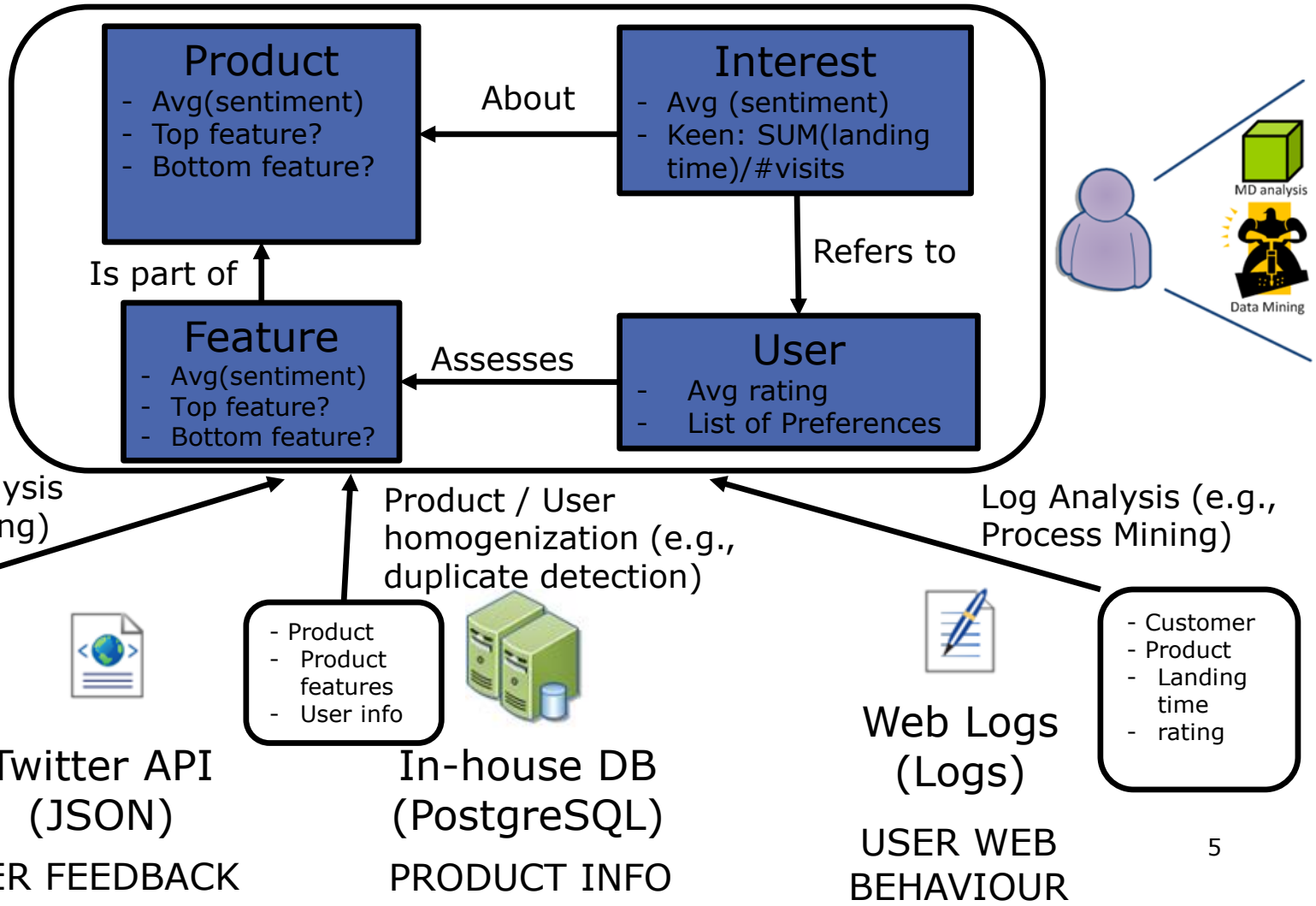
Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*



Data Integration Example

- *A company wants to contextualise their in-house data with opinions of their products given in Twitter and the user behaviour in their website*

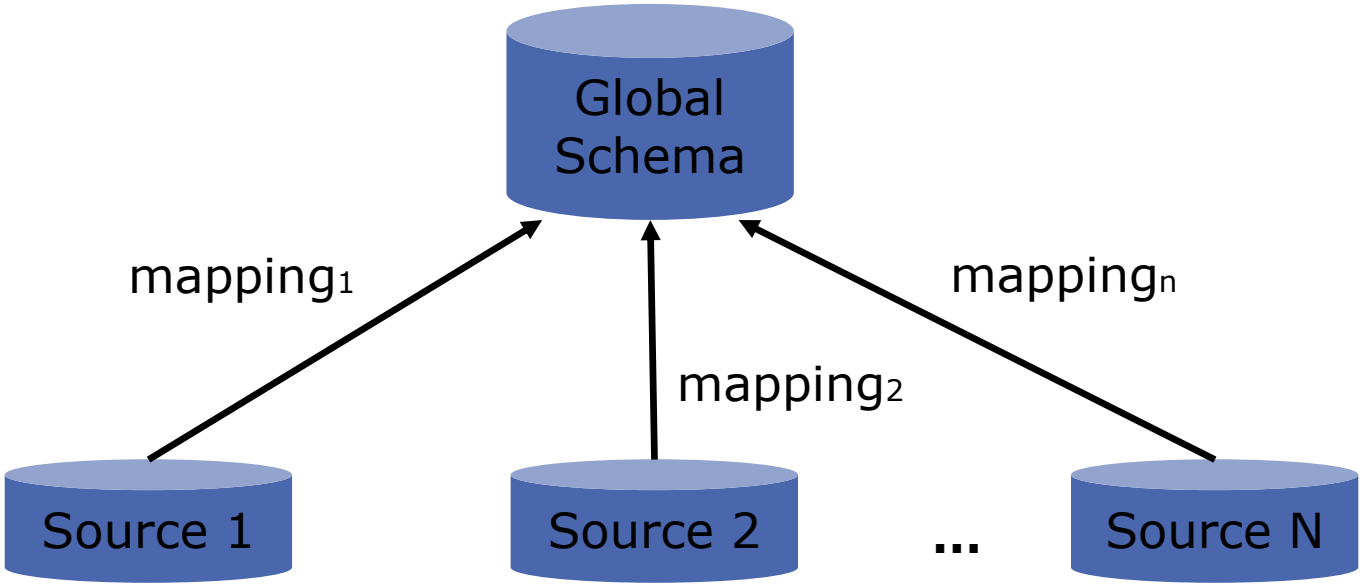


Data Integration Example

- Interest is every event detected in the web relating a user with a product
  - For each pair user-product, if we have identified the Twitter user, we extract the average sentiment by mining his tweets about that product
  - His / her keenness is computed by the ratio total landing time and number of visits
- Assessments are extracted, by means of data extraction techniques, from the user tweets about product features
  - For each feature, from all the assessments, we compute its average sentiment
  - Interest is every event detected in the web relating a user with a product
  - For each pair user-product, if we have identified the Twitter user, we extract the average sentiment
  - by mining his tweets
  - The top 5% features with higher sentiment are top features; similarly for bottom features
- Products compute their average sentiment from the tweets about them or about their features (features are weighted)
  - For each products, its bottom and top feature is identified (min and max sentiment, respectively)
  - Products and product features are deduplicated (in the source, several items may refer to the same product or product feature)
- Users are those from the database, who logged in the web. In addition, we may have info about their Twitter account
  - We trace the users in the web, and for each visit to the Web we generate a log line as follows: <user, product, landing time, rating>
  - The user list of preferences results from normalizing those products in the Web they visited and rated above his rating standard deviation

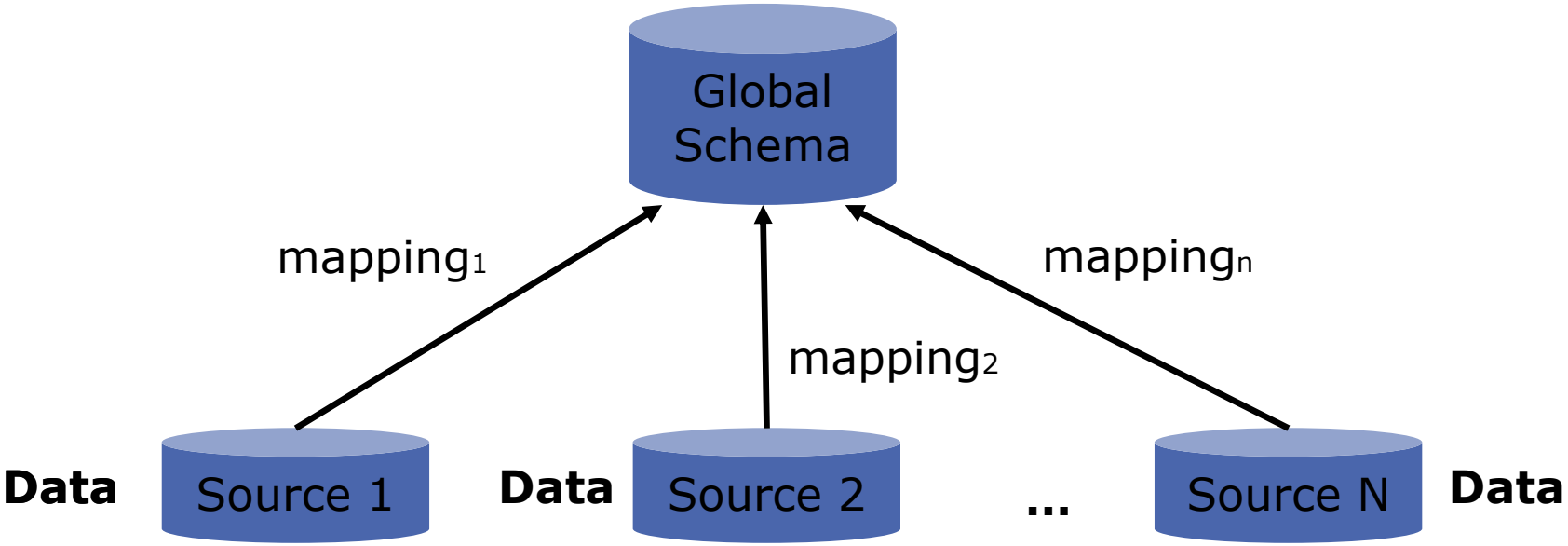
Basics on Data Integration

- Two main decisions to call:
  - Virtual Vs. physical integration
  - Global-As-View Vs. Local-As-View mappings



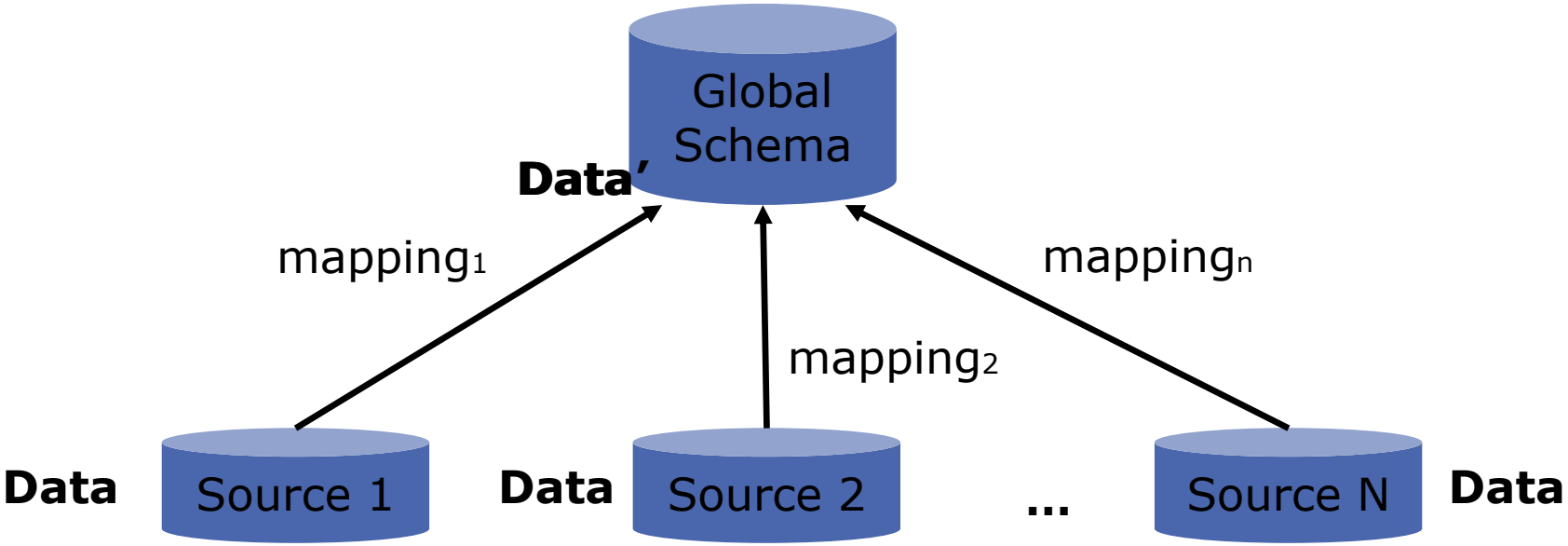
Basics on Data Integration

- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



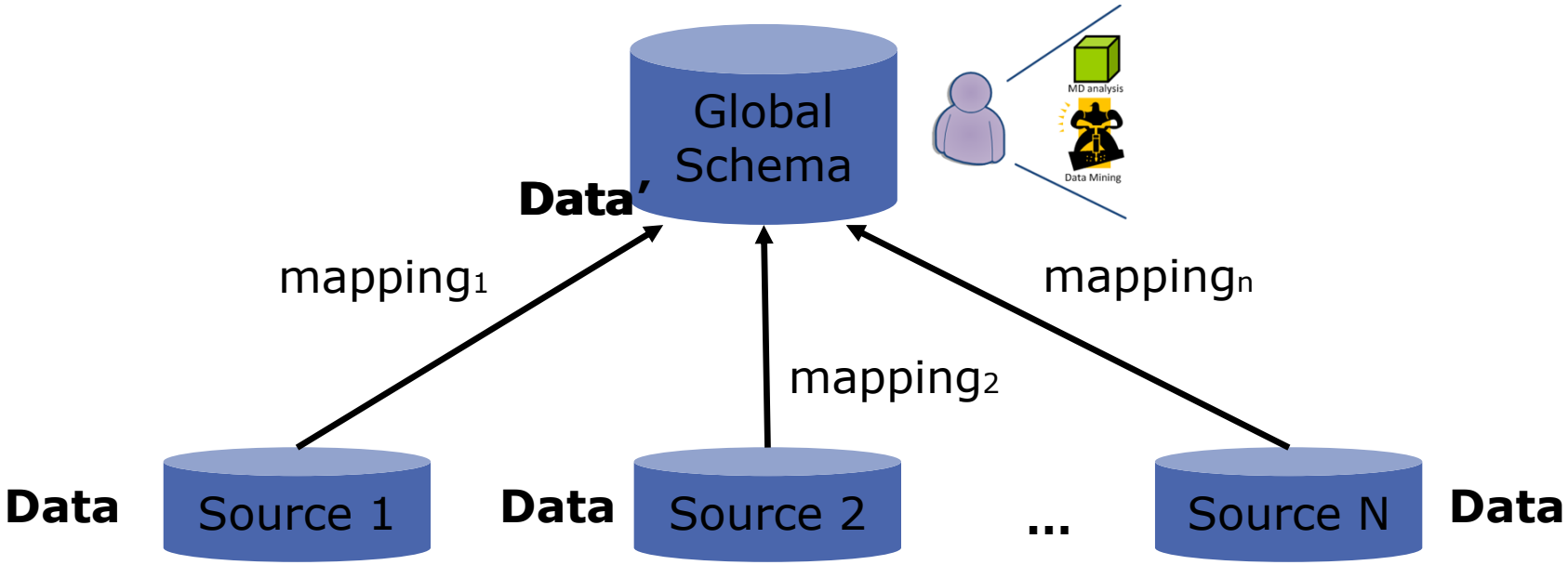
Basics on Data Integration

- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



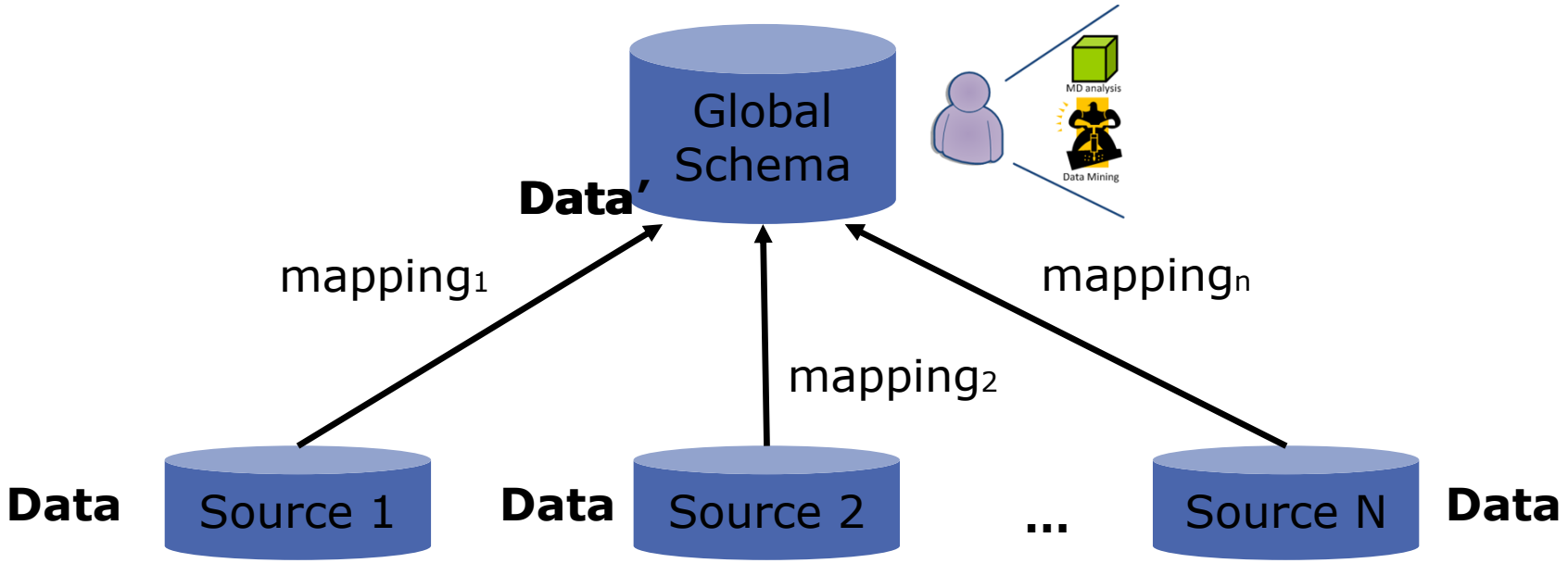
Basics on Data Integration

- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



Basics on Data Integration

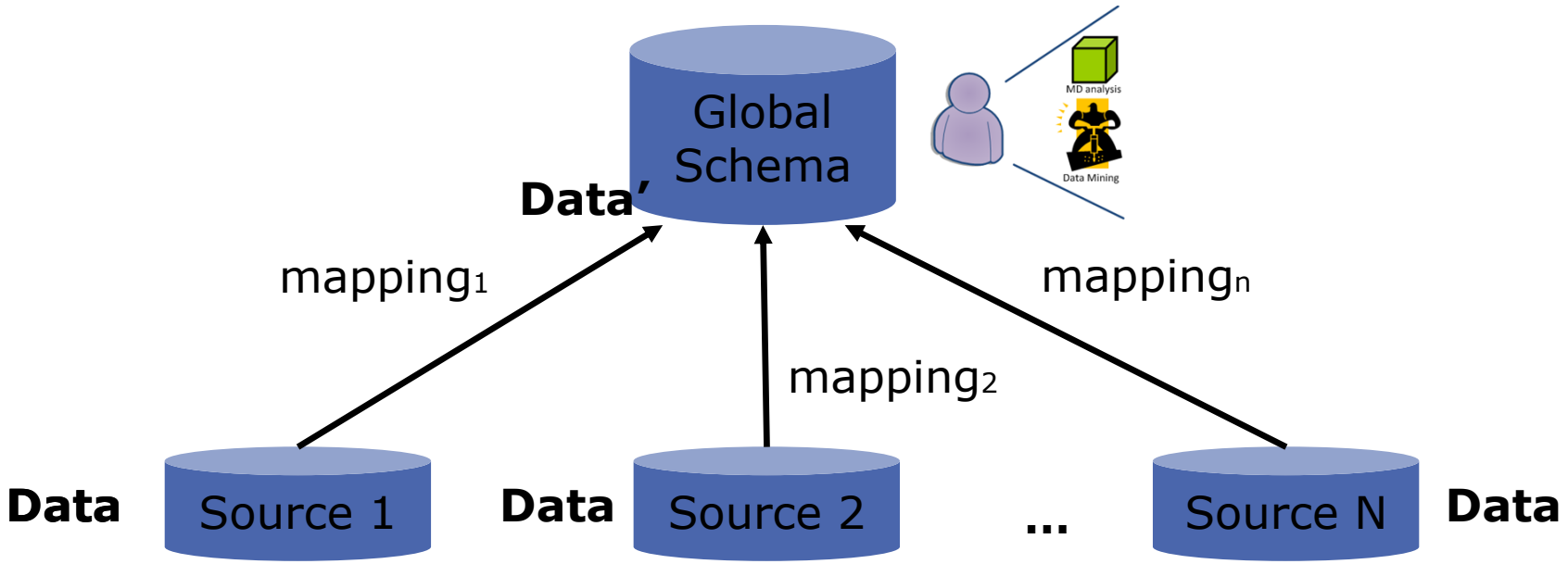
- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



Note that the mappings are created and coded in design time. Thus, they become simpler as the system only need to run them (no need to generate them on-the-fly)

Example: Data Warehousing

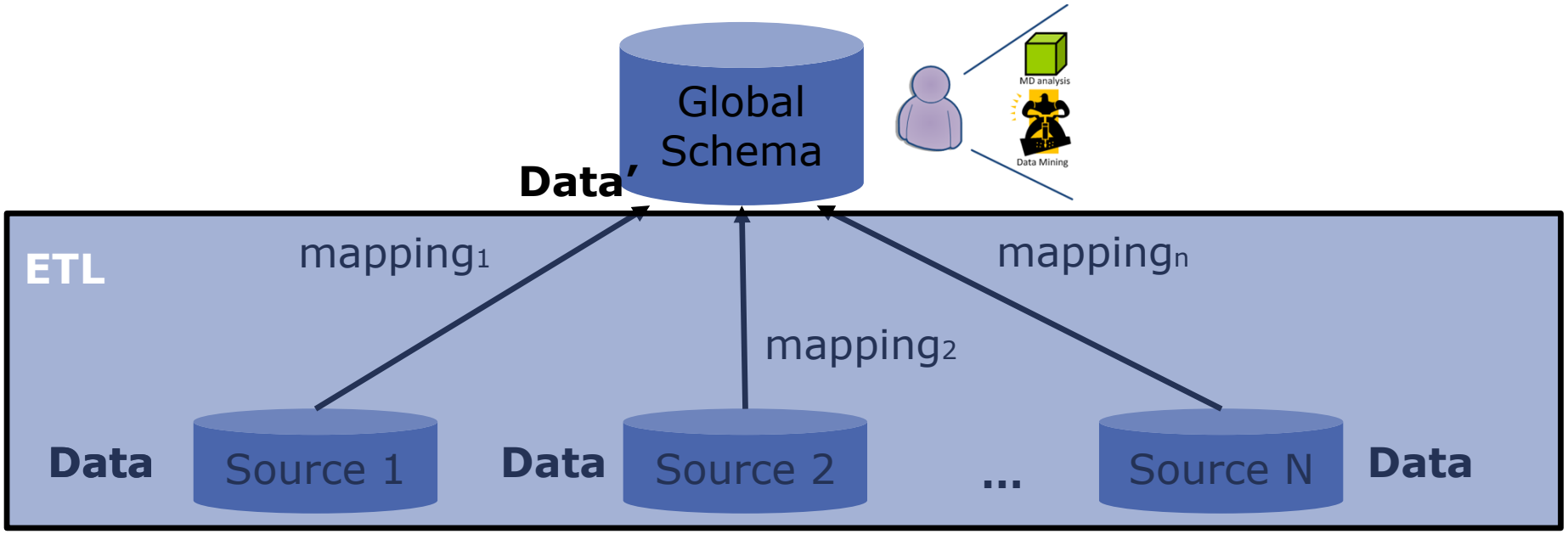
- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



Note that the mappings are created and coded in design time. Thus, they become simpler as the system only need to run them (no need to generate them on-the-fly)

Example: Data Warehousing

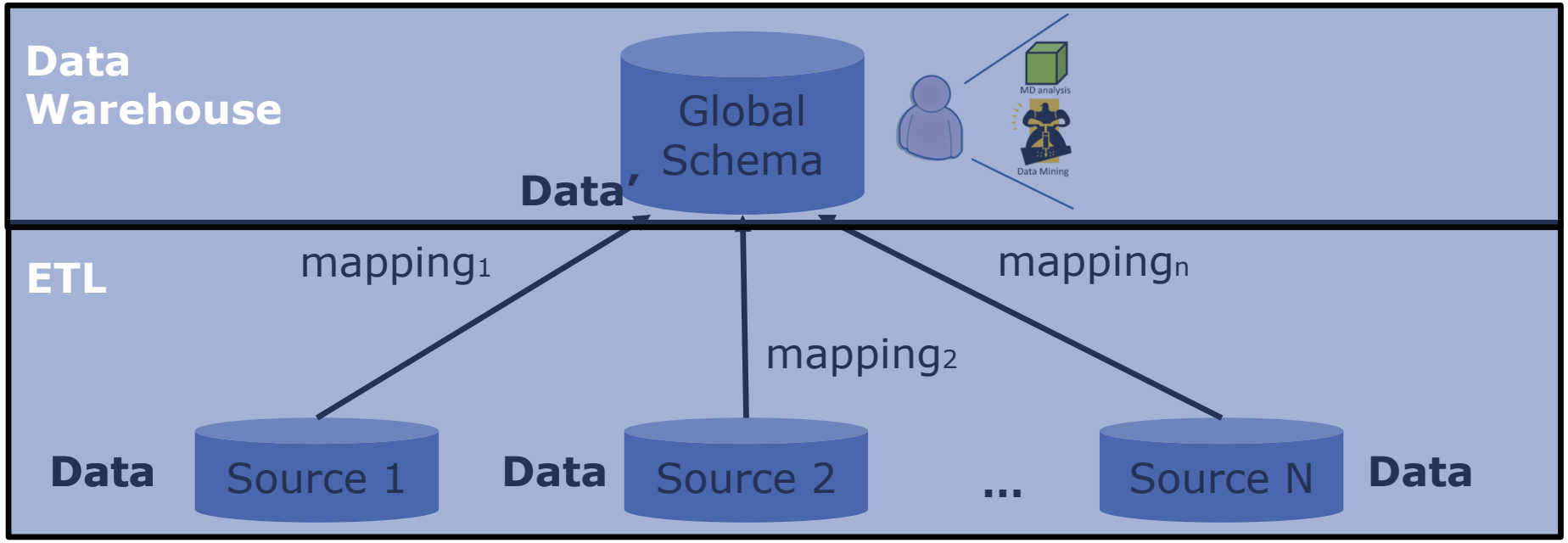
- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



Note that the mappings are created and coded in design time. Thus, they become simpler as the system only need to run them (no need to generate them on-the-fly)

Example: Data Warehousing

- Two main decisions to call:
  - Virtual Vs. **physical** integration
  - Global-As-View Vs. Local-As-View mappings



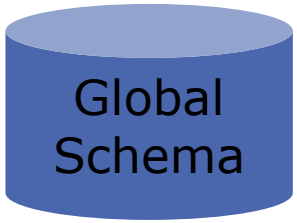
Note that the mappings are created and coded in design time. Thus, they become simpler as the system only need to run them (no need to generate them on-the-fly)

Activity: Physical Integration

- (15') Understand the main constructs behind physical data integration
  - (10') Consider the example introduce:
    - Choose two concepts and one relationship
    - Draw correct ETL flows from the sources to the integrated layer. A correct ETL:
      - relates the target attributes with the corresponding data source attributes
      - Applies a sequence of ETL operators. As a recall, the most typical ETL operators are:
        - Algebraic / relational operators
        - NLP constructs (remove stop words, tokenize, stemming, lemmatize, etc.)
        - Data mining / machine learning algorithms
        - User-defined functions (e.g., parsing, ad-hoc code, etc.)
- (10') Brainstorming

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings

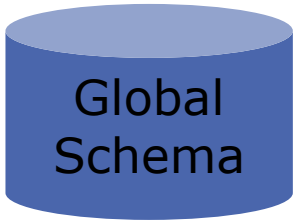


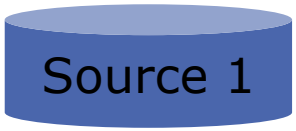
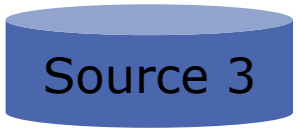
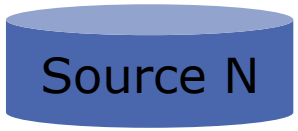
In the mappings, the global schema is defined as a view over the sources

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings

Mapping$_1$
G(User) = S1(User) U S3(Customer)



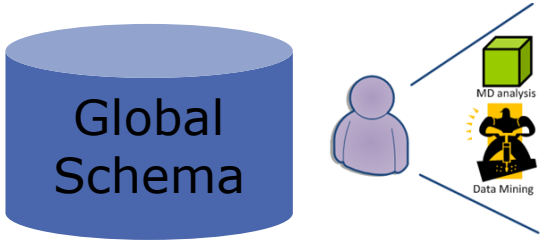
Data  **Data**  **...**  **Data**

In the mappings, the global schema is defined as a view over the sources

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings

Mapping$_1$
G(User) = S1(User) U S3(Customer)



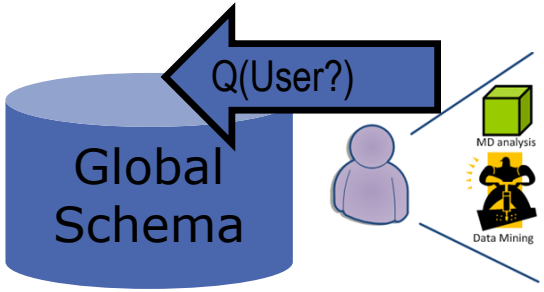
Data Source 1 **Data** Source 3 **...** Source N **Data**

In the mappings, the global schema is defined as a view over the sources

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings

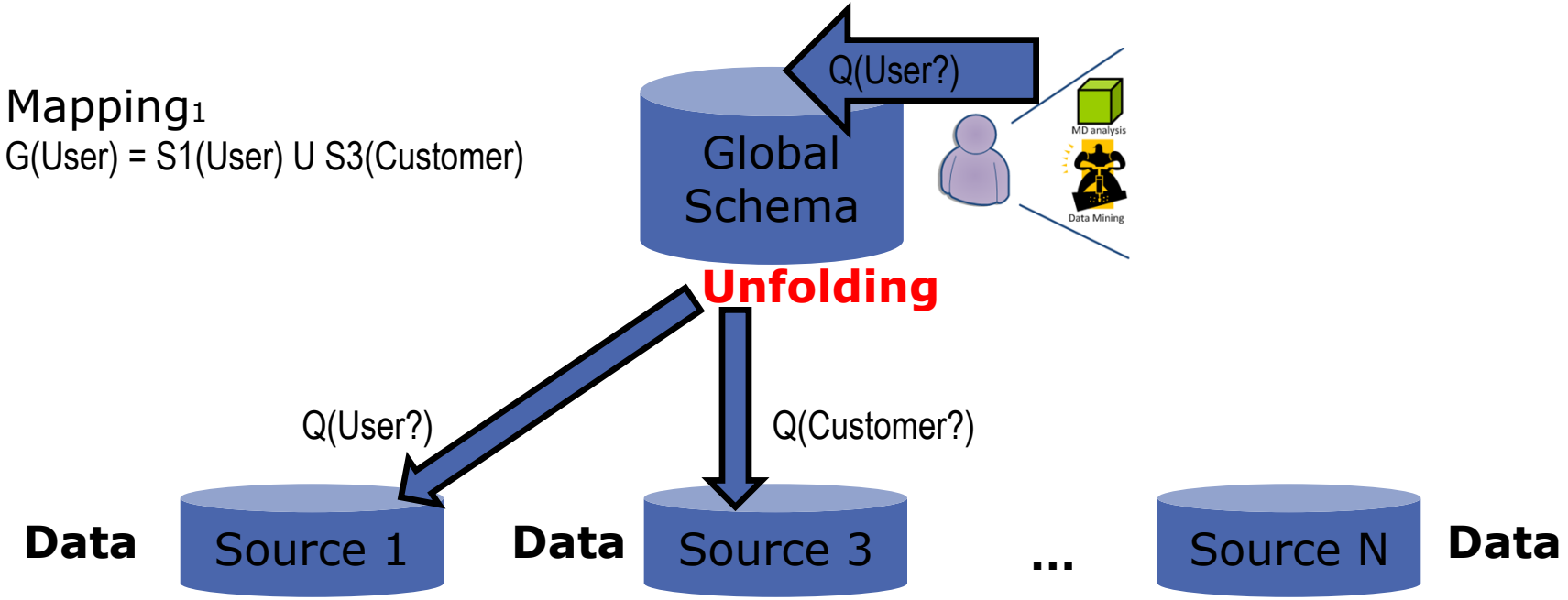
Mapping$_1$
G(User) = S1(User) U S3(Customer)



In the mappings, the global schema is defined as a view over the sources

Basics on Data Integration

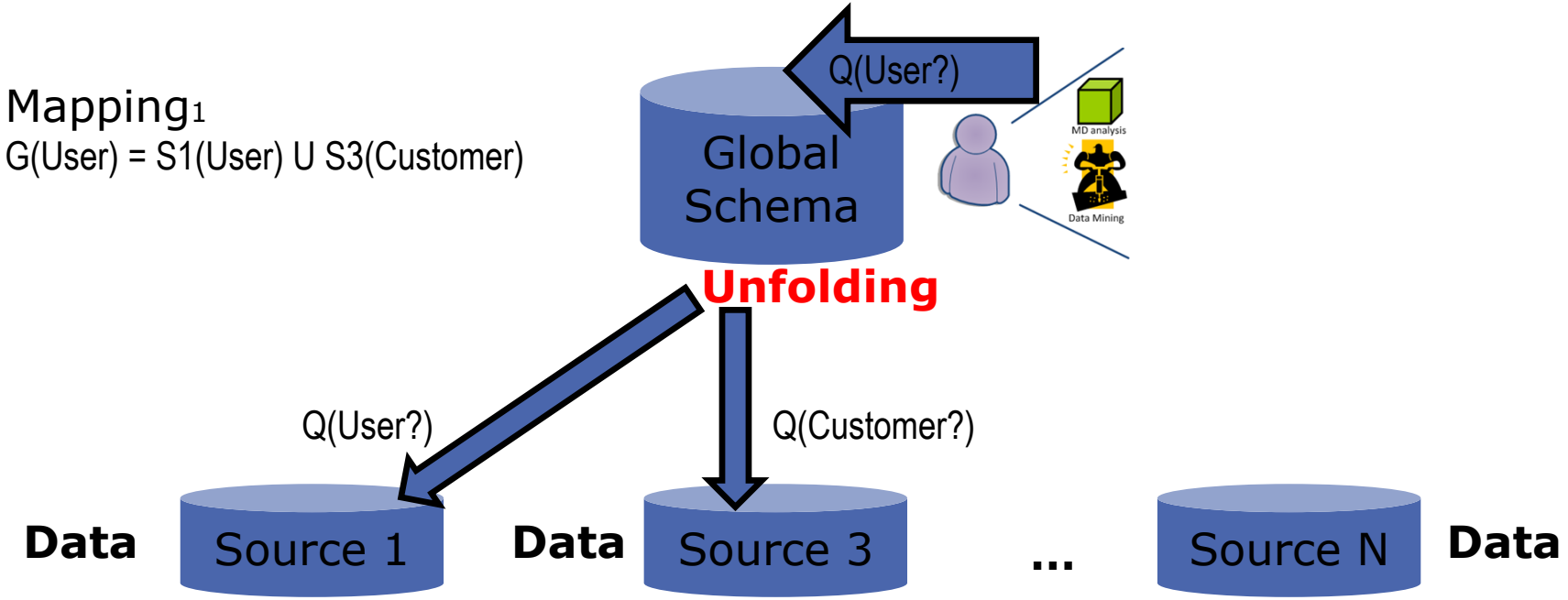
- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings



In the mappings, the global schema is defined as a view over the sources

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - **Global-As-View** Vs. Local-As-View mappings



In the mappings, the global schema is defined as a view over the sources

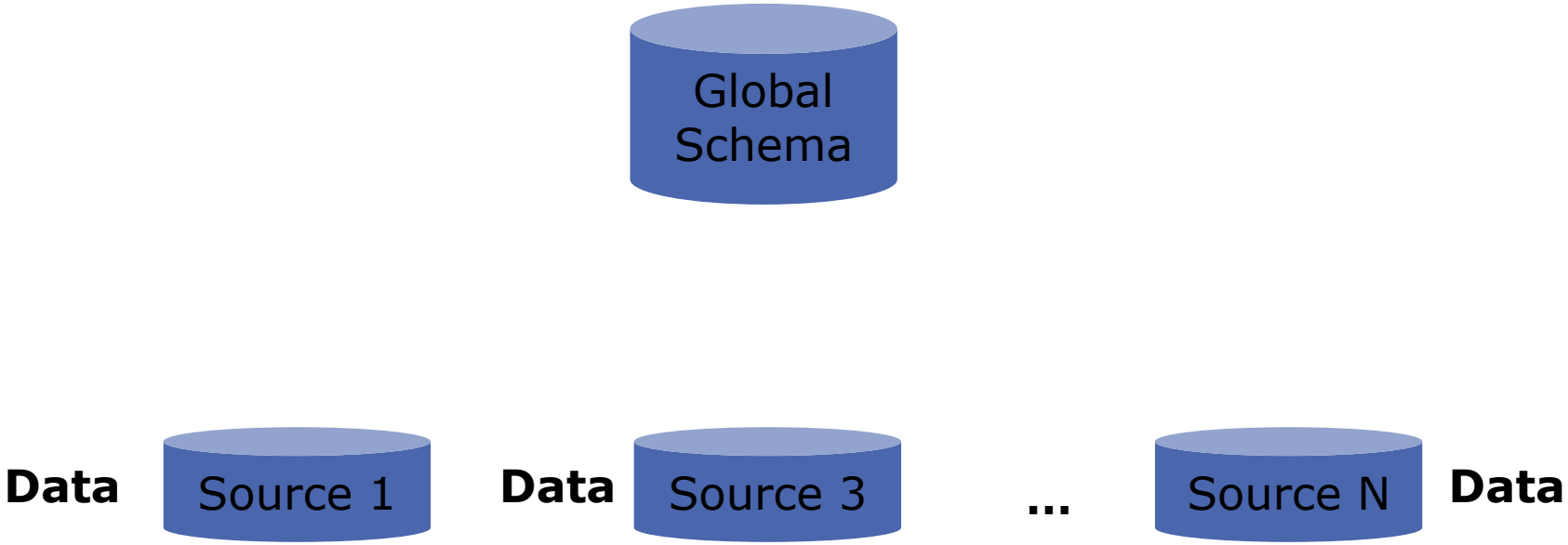
In this case, querying is reduced to unfold the mapping (same theoretical problem as querying views)

Activity: Virtual Integration + GAV

- (5') Understand the main constructs behind virtual data integration
  - (10') Consider your previous :
    - How would the GAV mappings from the target concepts and relationship chosen would look like?
    - Do they resemble the ETL flows? What is similar / what is different?
- (5') Brainstorming

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings

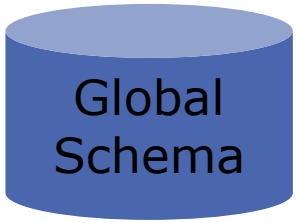


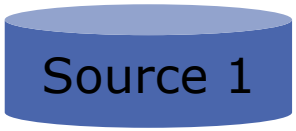
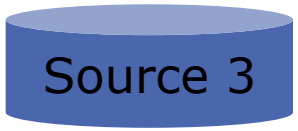
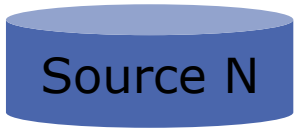
In the mappings, the local schemata are defined as views over the sglobal schema

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings

Mapping$_1$
S1(User) = G(User)
Mapping$_2$
S3(Customer) = G(User)



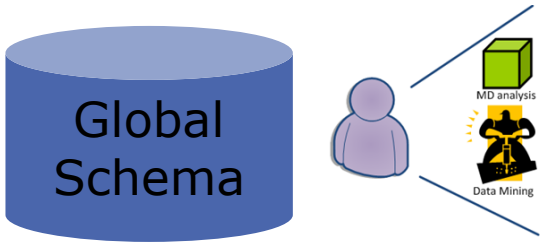
Data  **Data**  **...**  **Data**

In the mappings, the local schemata are defined as views over the sglobal schema

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings

Mapping$_1$
S1(User) = G(User)
Mapping$_2$
S3(Customer) = G(User)



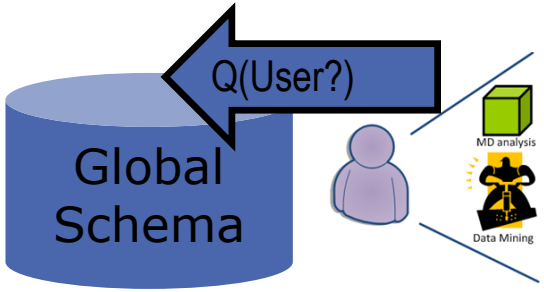
Data Source 1 **Data** Source 3 **...** Source N **Data**

In the mappings, the local schemata are defined as views over the sglobal schema

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings

Mapping$_1$
S1(User) = G(User)
Mapping$_2$
S3(Customer) = G(User)

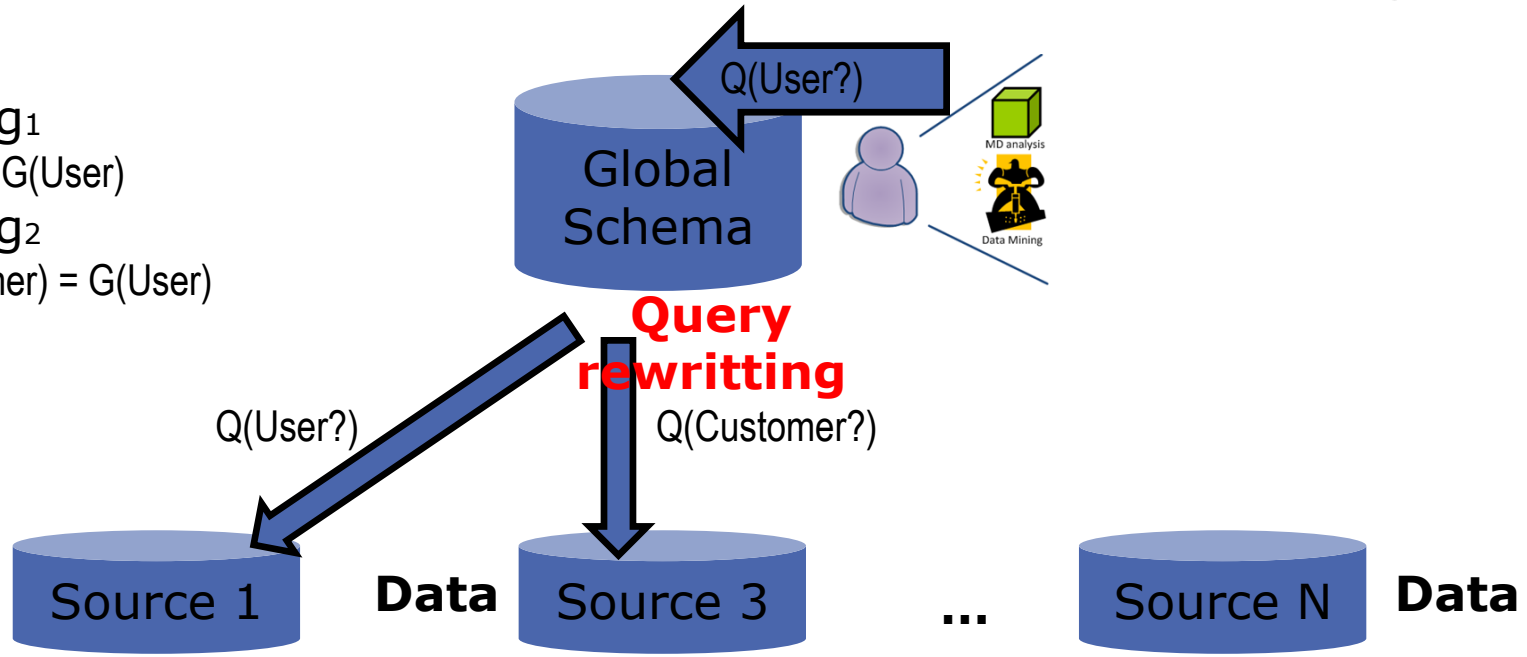


In the mappings, the local schemata are defined as views over the sglobal schema

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings

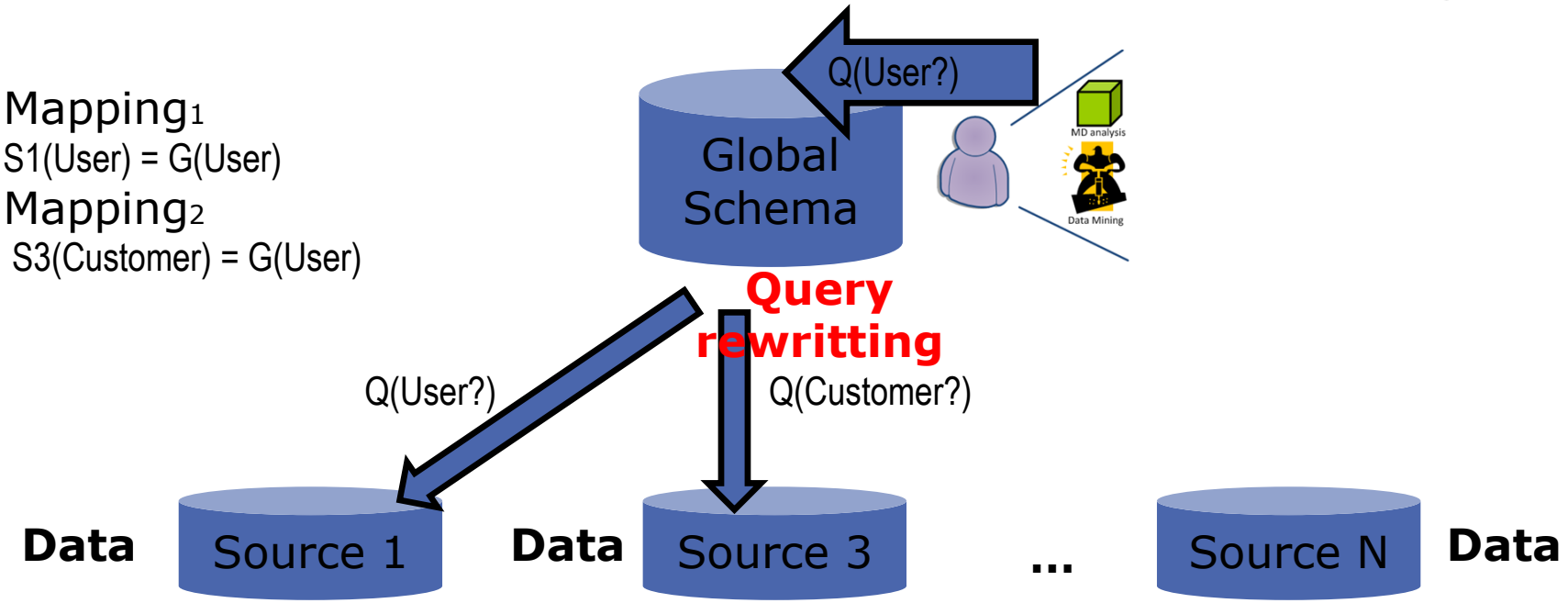
Mapping$_1$
S1(User) = G(User)
Mapping$_2$
S3(Customer) = G(User)



In the mappings, the local schemata are defined as views over the sglobal schema

Basics on Data Integration

- Two main decisions to call:
  - **Virtual** Vs. physical integration
  - Global-As-View Vs. **Local-As-View** mappings



In the mappings, the local schemata are defined as views over the sglobal schema

In this case, query rewriting techniques are needed to transform the query into a set of queries over the sources (logics needed!)

Unleashing the Power of LAV

- Consider the following relational view on top of the Web Logs source:

  (customer, product, landingTime, rating)

- And the following mappings:

(customer) :-> (user)

(product) :-> (product)

(customer c, product p) :-> (product p, interest c/p, user p)

(customer c, product p, SUM(landingTime)/COUNT(c, p) k) :-> (product p, interest c/p, user p, keen k)

- And the following queries:
  - Q(user?) and Q(product?)
  - Interest(product p, user u, keen k)?
  - Interest(user u, product p)?
- How would the query rewriting algorithm look like?

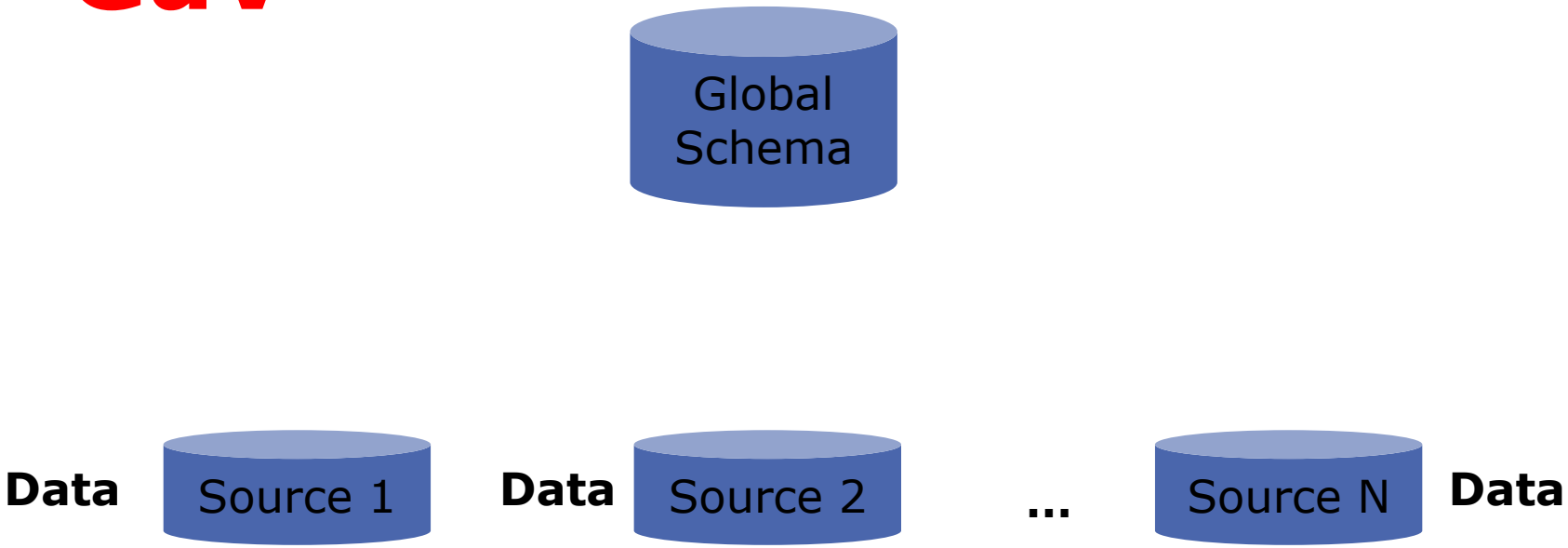
Activity: Virtual Integration + GAV

- (5') Understand the difference between LAV and GAV
  - (10') Consider the previous slide:
    - Considering the corresponding GAV mappings, would you be able to answer the following query?
      - Interest(user u, product p)?
      - Why?
- (5') Brainstorming

Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

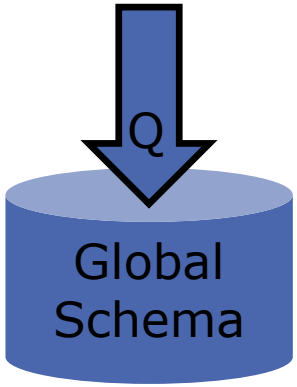
GaV



Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

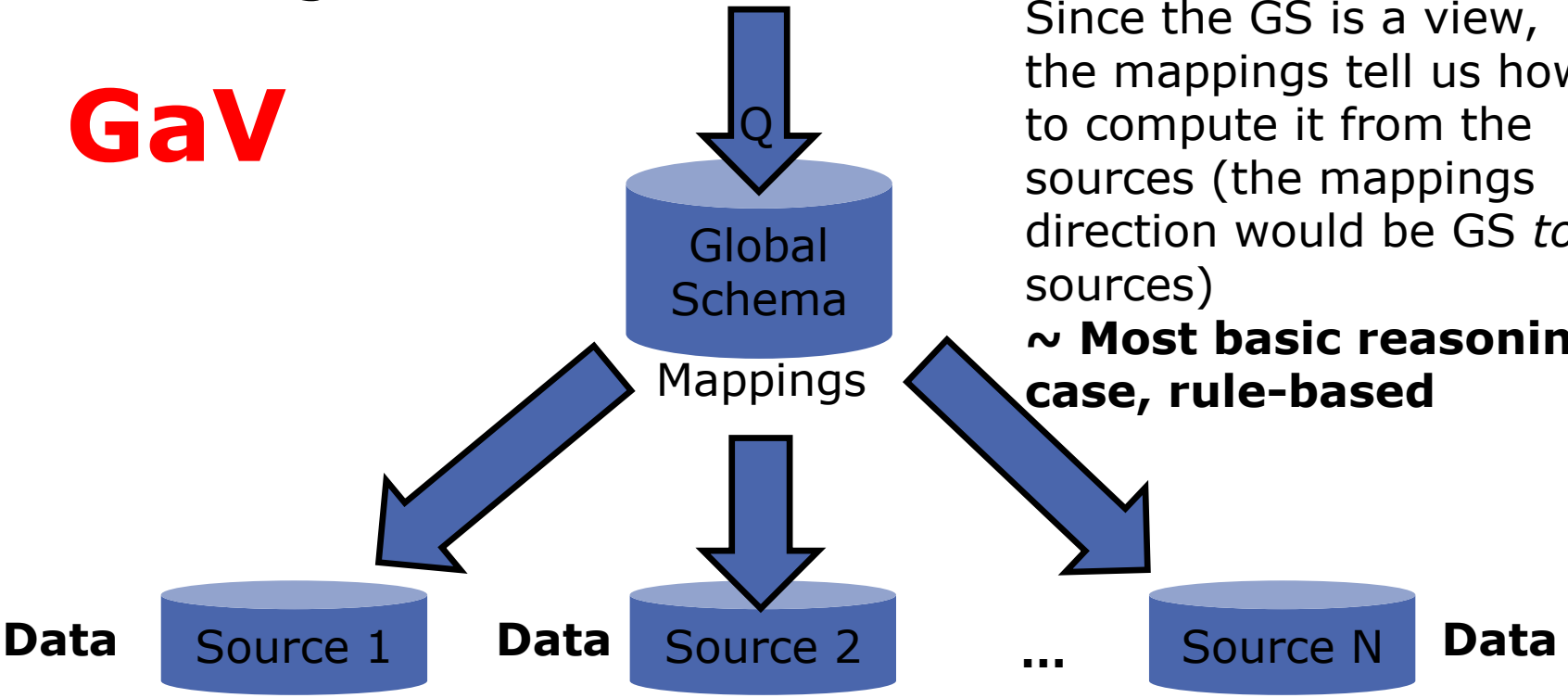
GaV



Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

GaV



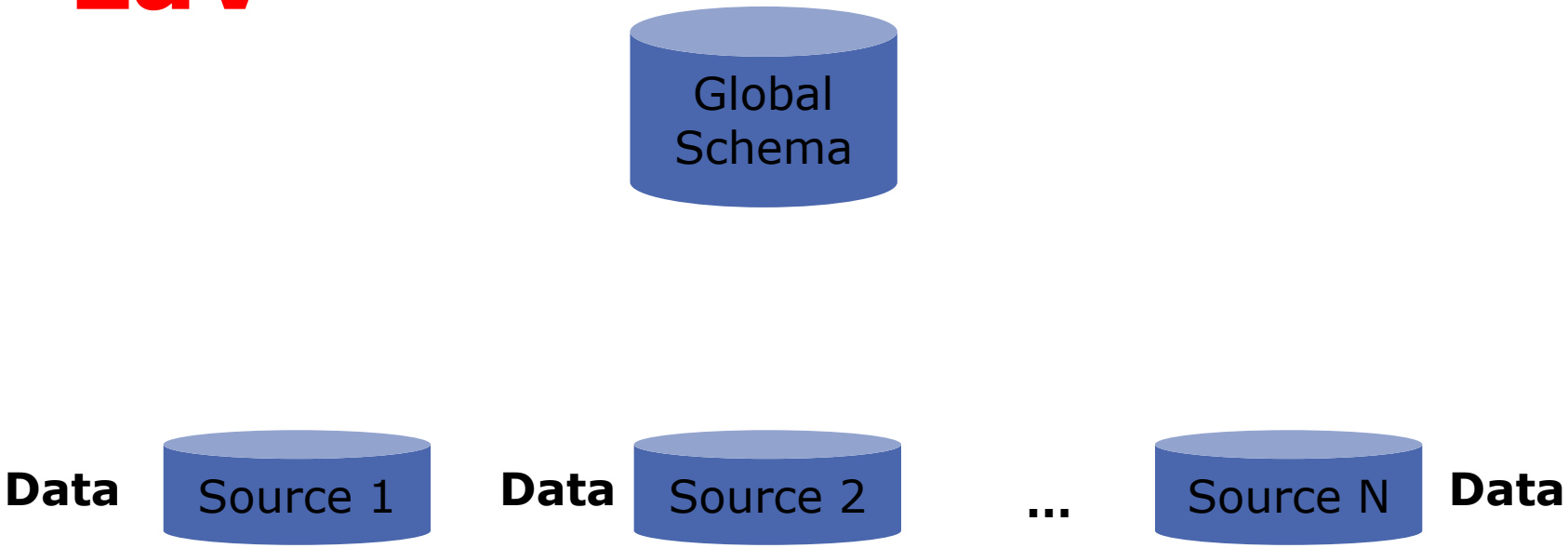
Since the GS is a view, the mappings tell us how to compute it from the sources (the mappings direction would be GS *to* sources)

~ Most basic reasoning case, rule-based

Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

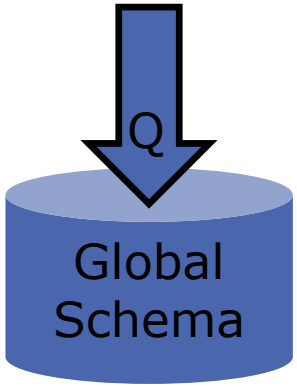
LaV



Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

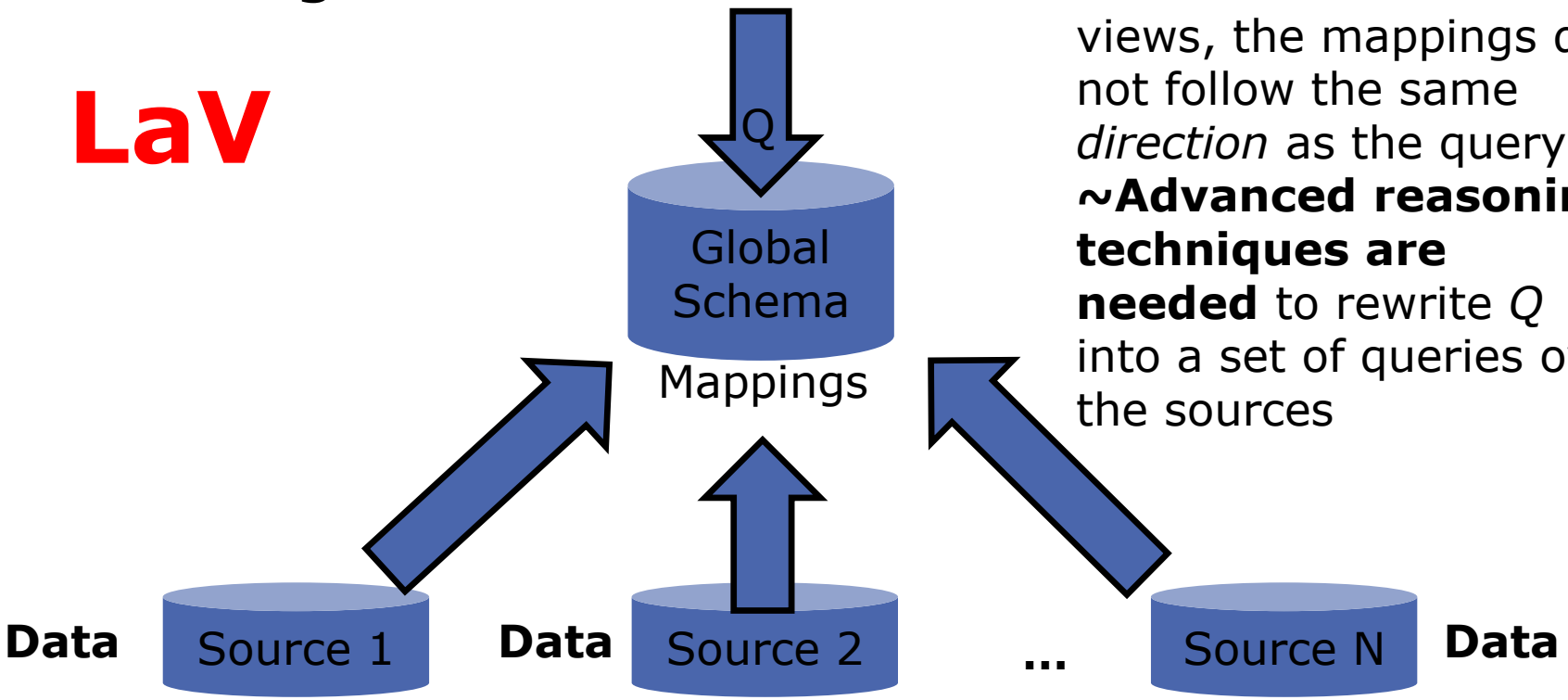
LaV



Basics on Data Integration

- Why LaV is much more complex than GaV in the general case?

LaV



Since the sources are views, the mappings do not follow the same *direction* as the query **~Advanced reasoning techniques are needed** to rewrite *Q* into a set of queries over the sources

Basics on Data Integration

- Realise that the mappings intervene in two main stages:
  - Definition: when a source comes into the system, the mappings between the source and the global schema must be defined
  - Querying: When querying the global schema with a query Q, Q must be rewritten into a set of queries (Q_1, …, Q_n) over the data sources by using the mappings

Activity: Pros and Cons

- (15') Discuss with another peer the pros and cons of physical vs. logical integration / LAV Vs. GAV
  - (10') Understand each of them wrt the following issues:
    - Where does data reside?
    - How do the mappings look like?
    - When querying the global schema, how would query rewriting (i.e., the query execution to fetch data, process it and answer the query) look like?
    - For each case, consider:
      - How difficult is to define the mappings,
      - How difficult is the query rewriting algorithm
    - How easy would be to remove / add a new source?
  - (5') Brainstorming:
    - When is better a physical integration wrt logical integration?
    - When is better LAV than GAV? And the opposite case?

Comparison

		Materialised GS	Virtual GS	
		Static Mappings	**GaV**	**LaV**
Mapping Definition	Mapping Complexity	Very Complex	Low	Very Low
	Maintainability	Very Complex	Medium	Very Low
Querying Time	Query Rewritting	-	Simple (Unfolding)	Very Complex
	Performance	Very High	Low	Very Low

Comparison

		Materialised GS	Virtual GS	
		Static Mappings	GaV	LaV
Mapping Definition	Mapping Complexity	Very plex		Ve w
	Maintainability	Very plex	M n	Ve w
Querying Time	Query Rewritting		Si e (Un ng)	Co ex
	Performance	Ve gh		Ve w

Data Integration in Practice

- Physical Vs. Virtual Integration:
  - Physical Integration
    - AKA data consolidation
    - Querying is reduced to querying a regular database (the GS has been materialised with all the data). Thus, it goes against the autonomy of the sources which are tightly coupled
    - Freshness might be an issue
    - Examples: Data Exchange techniques, Data Warehousing
      - In the Big Data Context: Data Lake
  - Virtual Integration
    - AKA logical integration or data federation
    - Querying requires further processing as data is in the sources. GAV or LAV positioning required
    - However, data freshness is guaranteed as well as the systems autonomy (the data sources coupling is lighter)
    - The approach is less monolithic, and matches the **pay-as-you-go data integration model** blooming with Big Data
    - Examples: Federated databases (aka mediators), multidatabases
      - In the Big Data Context: Polystores, Dataframes

Data Integration in Practice

- Mappings:
  - Most systems use GaV
    - It is easier to query (unfolding)
    - Assume the sources will not vary much (mapping the sources to the GS is expensive)
  - Big Data naturally move towards LaV because:
    - Sources can leave or enter the system much easier
    - Source data is mappable to an arbitrary conjunctive query over the global schema
    - However, querying is really expensive. Two potential ways to improve this are:
      - Reducing the allowed complexity in the mappings
      - Use additional techniques to improve performance (these techniques are also used for GaV approaches)
        - Materialisation of partial results
        - Caching
        - Pre-fetching

Summary

- The Data Integration Framework
- Data Sources Layer
- Integration Layer
  - Physical Vs. Virtual integration
  - GAV Vs. LAV mappings