Data Integration

Monir Ahmad
Adjunct Lecturer, CSE, IIUC

Schema Integration

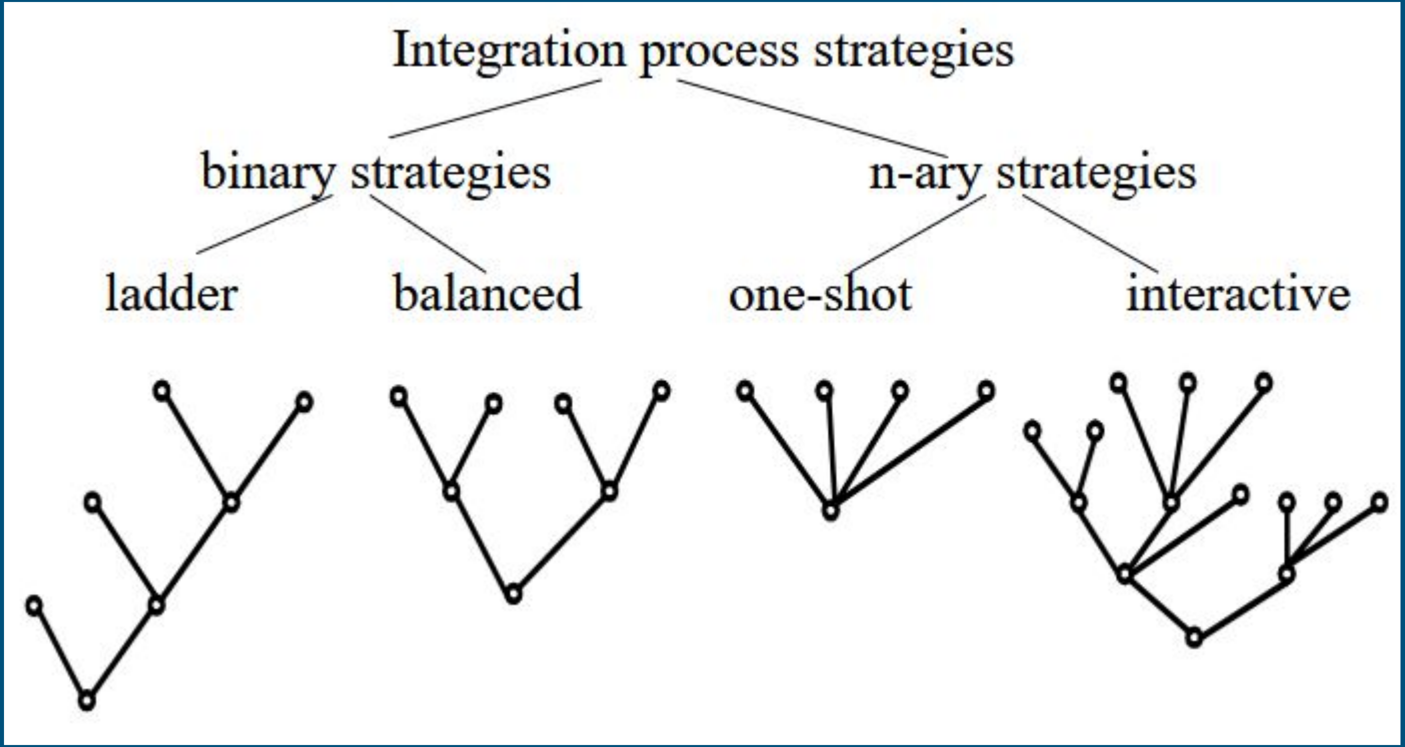
Definition

- Process of combining multiple database schemas from different sources into one unified, global schema.

Why Schema Integration?:

✔ Organizations often have multiple databases (e.g., Sales, HR, Finance).
✔ These databases are independent and may use different structures.
✔ To get a complete view of data, we need to integrate schemas.
✔ It helps in:
-- Data consistency
-- Eliminating duplication
-- Easier data access for decision-making

Integration processing strategies



Steps in integration process

1. Pre-integration

-- Choose integration processing strategies

-- This governs the choice of schemas to be integrated

2. Comparison of the schemas

-- Schemas are analyzed and compared to determine the correspondences among concepts and detect possible conflicts.

Steps in integration process

3. Conforming the schemas

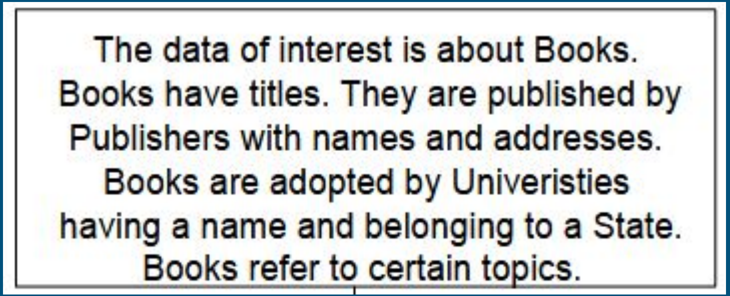
-- Once conflicts are detected, an effort is made to resolve them so that the merging of various schemas is possible.

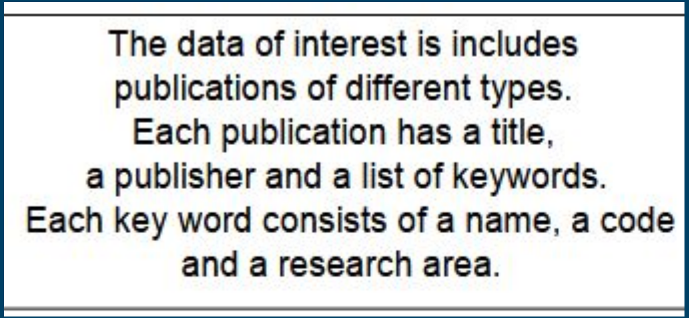
-- Automatic conflict resolution is generally not feasible; interaction with designers is required.

4. Merging and Restructuring

-- The schemas are ready to be superimposed, giving rise to some intermediate integrated schema(s).

Example


The data of interest is about Books.
Books have titles. They are published by
Publishers with names and addresses.
Books are adopted by Univeristies
having a name and belonging to a State.
Books refer to certain topics.


The data of interest is includes
publications of different types.
Each publication has a title,
a publisher and a list of keywords.
Each key word consists of a name, a code
and a research area.

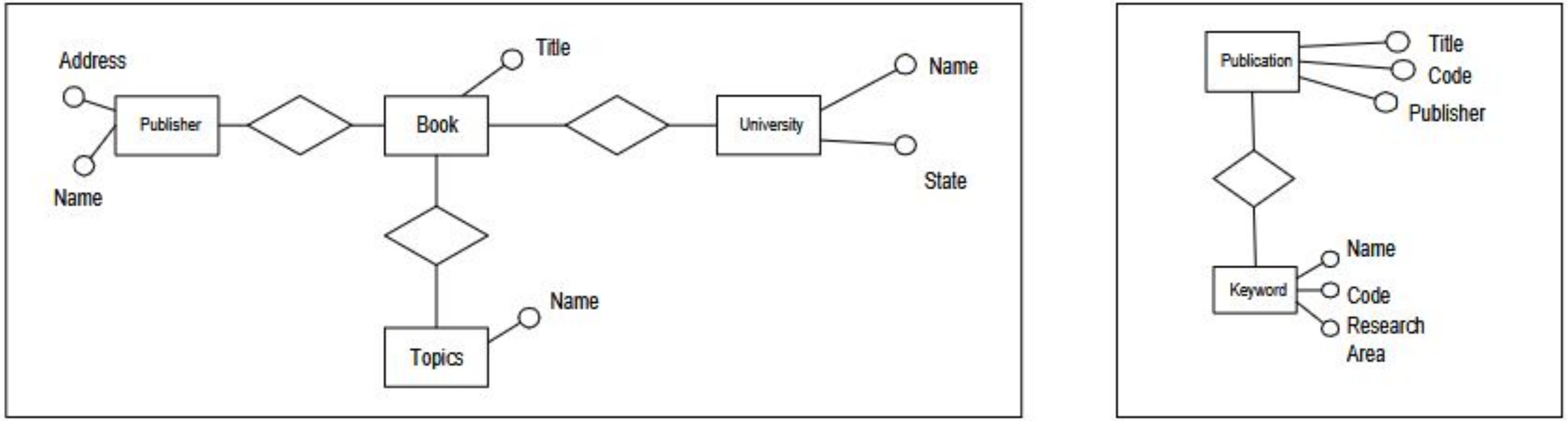


Figure 4a. Original Schemas

Example

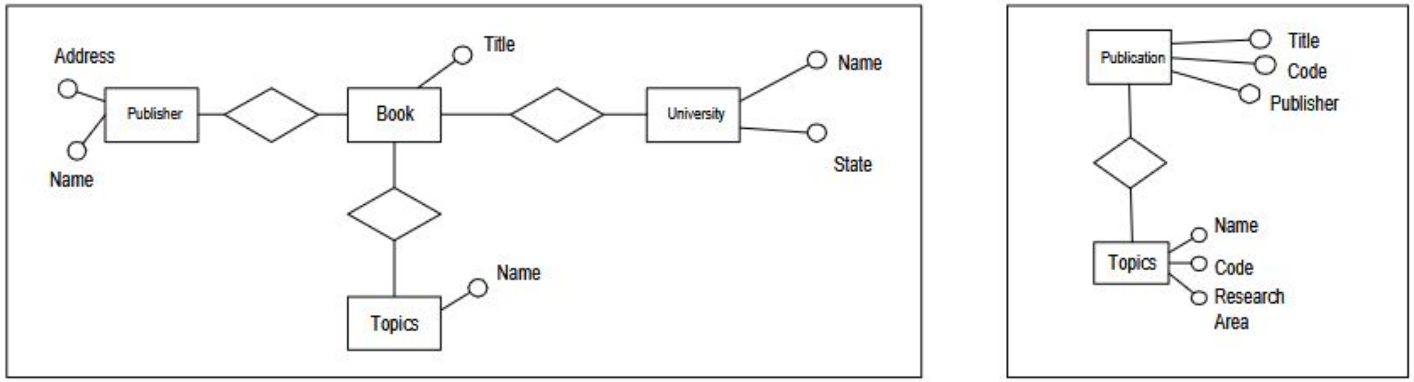


Figure 4b. Choose "Topics" for "Keyword (Schema 2)

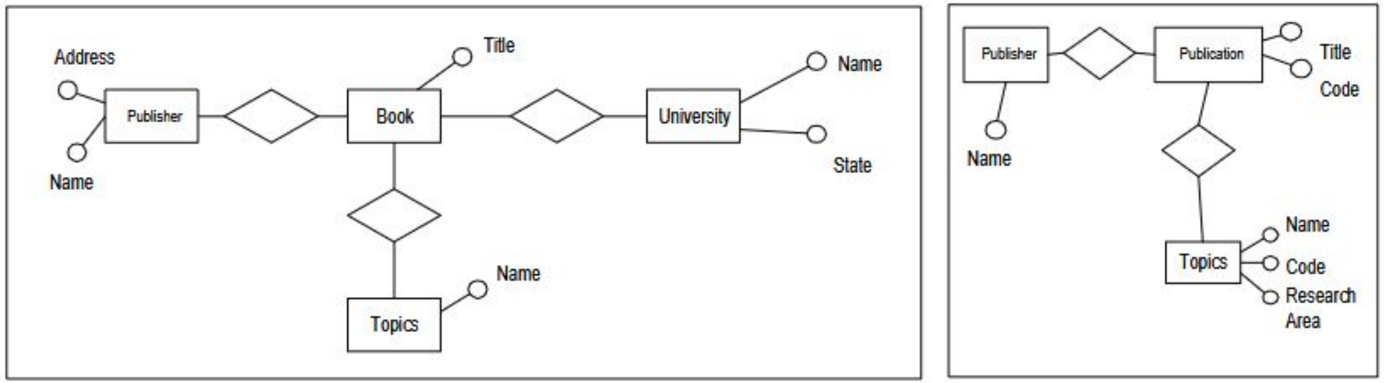


Figure 4c. Make Publisher into an entity (Schema 2)

Example

Figure 4d: Superimposition of schemas

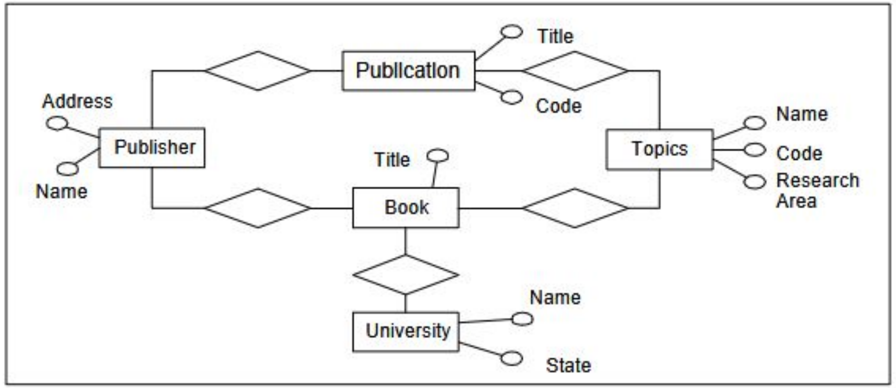
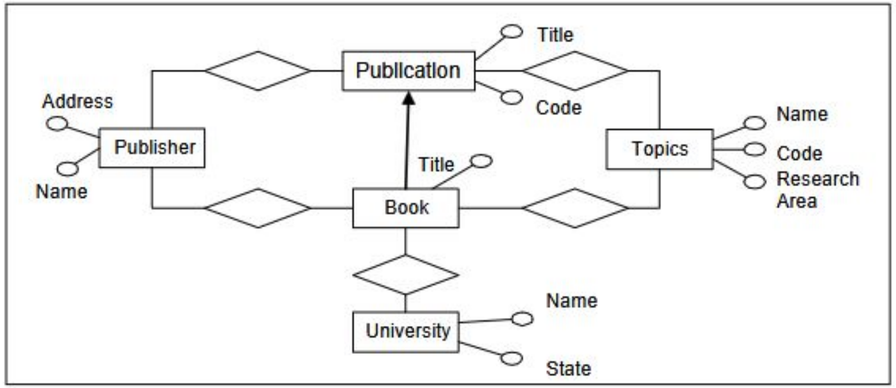


Figure 4e: Creation of a subset relationship.



Example

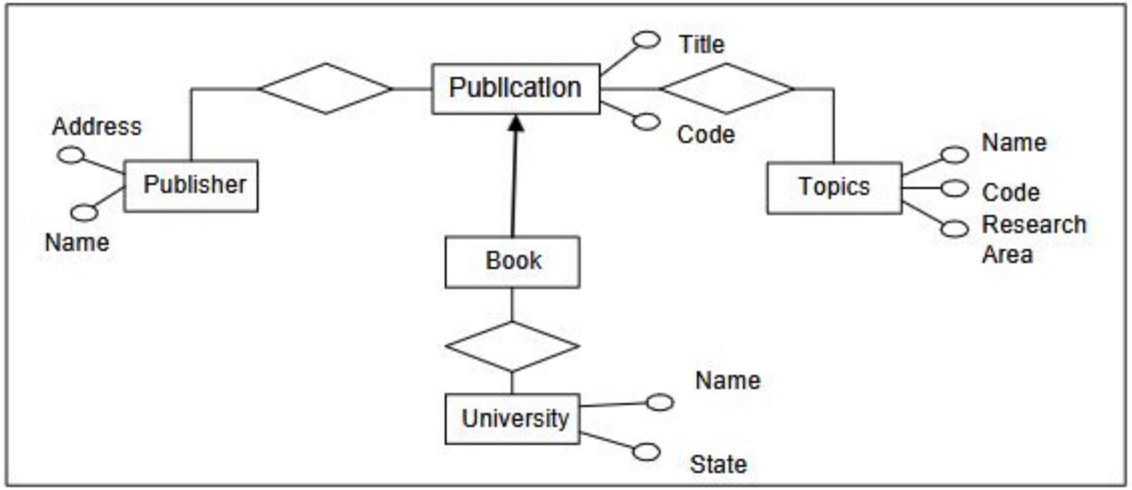


Figure 4f: Drop the properties of Book common to Publication.

Conflicts

Conflicts

- When integrating data from multiple heterogeneous sources, various conflicts arise due to differences in how data is structured, named, or represented.

Types:

✔ Naming Conflicts
✔ Structural Conflicts
✔ Data Type Conflicts
✔ Value Conflicts (Semantic Conflicts)
✔ Key Conflicts (Identifier Conflicts)
✔ Constraint Conflicts

Naming Conflicts

- Different sources use different names for the same concept (or same name for different concepts).

Types:

✔ **Synonyms**:
-- Same concept, different names.
-- Example: EmployeeID vs. StaffNumber

✔ **Homonyms**:
-- Same name, different concepts.
-- Example: Code might mean PostalCode in one source, and ProductCode in another.

Structural Conflicts

- Data is organized differently across sources, even though it represents the same real-world concept.

Types:

✔ **Attribute vs. Entity:**
-A field (attribute) in one schema might be modeled as a separate entity in another.
-Example: Address as a field in one schema, but as a separate table in another.

Structural Conflicts

- Data is organized differently across sources, even though it represents the same real-world concept.

Types:

✔ **Aggregation differences:**

--One source uses detailed data, another uses summarized/aggregated data.

--*Example*: Daily sales vs. monthly sales.

Data Type Conflicts

- The same attribute has different data types in different sources.

- Example:
  --PhoneNumber stored as integer in one source, and as string in another.

  --DateOfBirth stored as DATE in one source, and as STRING (text) in another.

Value Conflicts (Semantic Conflicts)

- The meaning or units of the data differ between sources, causing inconsistencies.

- Example: Price in USD vs. EUR.

Key Conflicts (Identifier Conflicts)

- Different primary keys or identifiers are used for the same entity in different sources.
- Example: An Employee identified by EmployeeID in one system and by SocialSecurityNumber in another.

Constraint Conflicts

- Different constraints are enforced on similar data
- Example: One source may require a NOT NULL constraint on PhoneNumber, another may allow NULL values.

Conflicts-All in One

```
EMPLOYEE (
    EmpID INT PRIMARY KEY,
    FullName VARCHAR(50),
    DOB DATE,
    Gender CHAR(1), -- M/F
    Salary FLOAT, -- in USD
    Address VARCHAR(100),
    DepartmentID INT
)

DEPARTMENT (
    DepartmentID INT PRIMARY KEY,
    DeptName VARCHAR(50)
)
```

Schema S1

```
STAFF (
    StaffNumber VARCHAR(10) PRIMARY KEY,
    Name VARCHAR(60),
    BirthDate TEXT,
    Gender INT, -- 0 = Male, 1 = Female
    Salary DECIMAL(10,2), -- in EUR
    LocationID INT
)

LOCATION (
    LocationID INT PRIMARY KEY,
    LocationAddress VARCHAR(200)
)
```

Schema S2

View-based Data Integration

View

What is view?

- A view is basically a virtual table. It's not a real, physical table storing data.

Creating a view:

```
CREATE VIEW view_name AS
SELECT column1, column2…..
FROM table_name
WHERE condition;
```

View

Key Features of a View:

- **Virtual**: It doesn't hold data itself; it shows data dynamically from the underlying tables.

- **Up-to-date**: Since it’s based on queries, you always see the most current data from the source tables.

- **Security & Simplification:** Views can hide complexity or restrict access, showing only certain columns or rows to users.

View

*“A **view** is like a **live dashboard** on your phone. The data is coming from different apps (or sources), but you only see one clean interface showing you what you need—without copying or moving anything.”*

Definition

- Data Integration: Finding and combining data from different sources
- View-based Data Integration: Integrate sources into a single unified view
- Declarative mapping language: Specify of how each source relates to the unified view
- Three approaches:
    - Global As View (GAV)
    - Local As View (LAV)
    - Global and Local As View (GLAV)

Historical Background

The Problems

- Applications often need data from multiple, heterogeneous sources.
- These sources differ in:
    - Formats (text files, web pages, XML, relational databases).
    - Structures (different schemas).
    - Access methods (web forms, database clients).

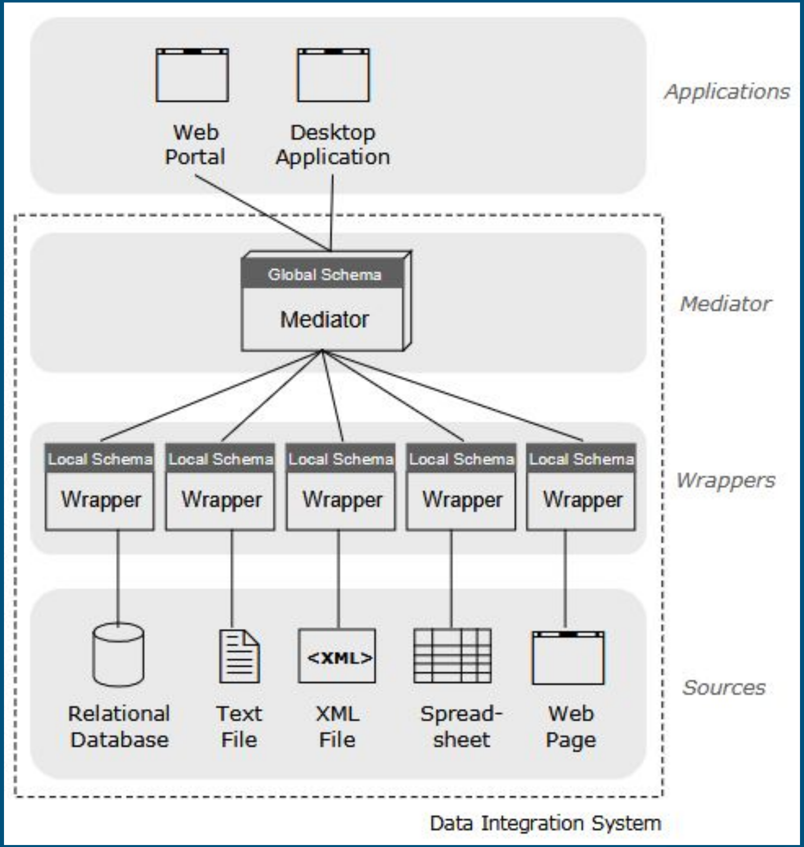
Historical Background

The Solution

- Companies first identified the need to integrate structured data across systems.
- The concept of a single unified view emerged.
- View-Based Integration Systems (VDISs) provided:
    - A single point of access to all data sources.
    - Transparent integration and inconsistency handling by the system.

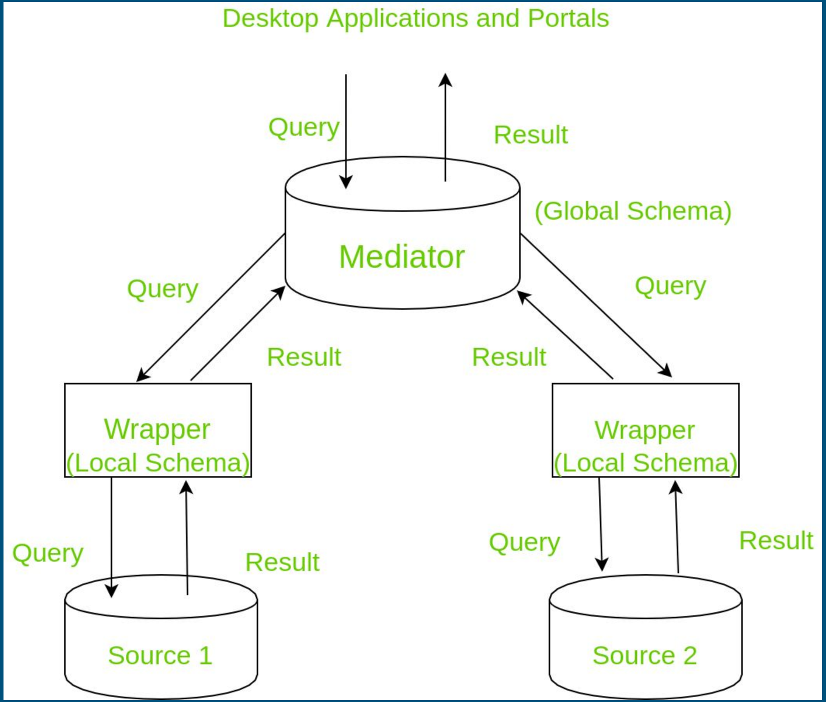
VDIS Architecture

- **Sources:** Different databases, files, web pages.
- **Wrappers:** Convert each source into a common format.
- **Mediator:** Uses mappings to combine the data and present a unified view.
- **Applications/Users:** Query the global view without worrying about where the data comes from.



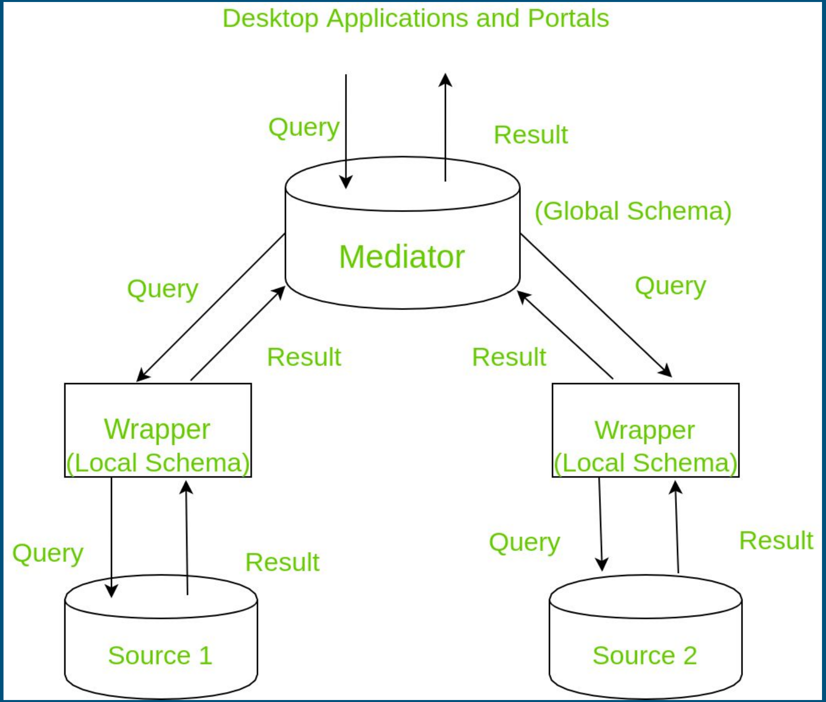
Mediation Process in Data Integration

- **Step 1: User/Application Sends a Query**
  - Desktop Applications or Portals (users/clients) send a query to the Mediator.
  - Users do not need to know where the data is located or how it's structured.
- **Step 2: Mediator Analyzes and Decomposes the Query**
  - The Mediator receives the query and interprets it using the Global Schema.
  - It breaks down the query into sub-queries, deciding which data sources need to be contacted.



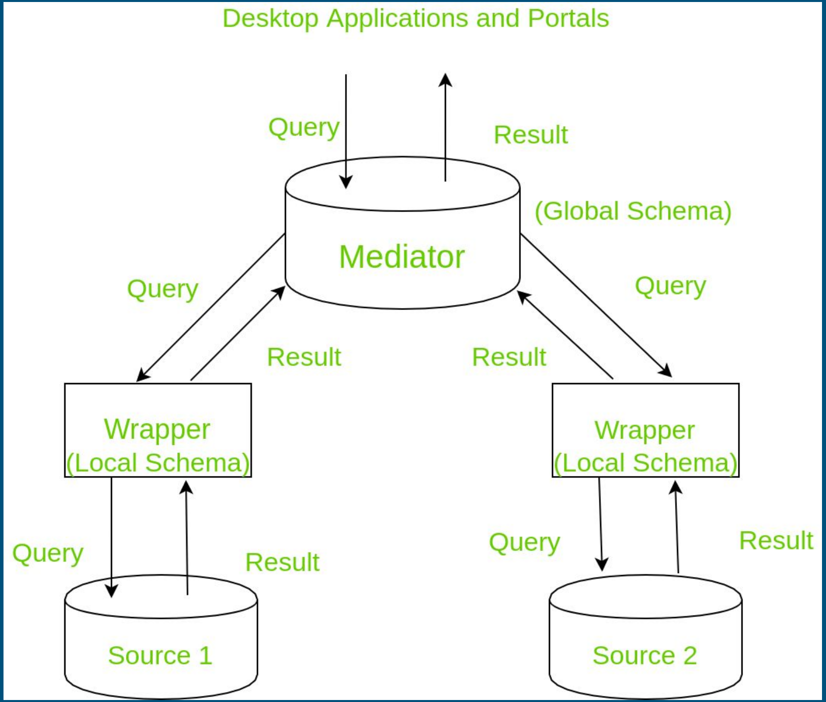
Mediation Process in Data Integration

- **Step 3: Mediator Sends Sub-Queries to Wrappers**
  - The Mediator sends each sub-query to the relevant Wrapper.
  - Each Wrapper handles communication with its corresponding Local Data Source.
- **Step 4: Wrappers Query Their Local Sources**
  - The Wrappers translate the sub-query into the local query language/schema understood by their Source.
  - The query is executed on the local source database (e.g., Source 1, Source 2).



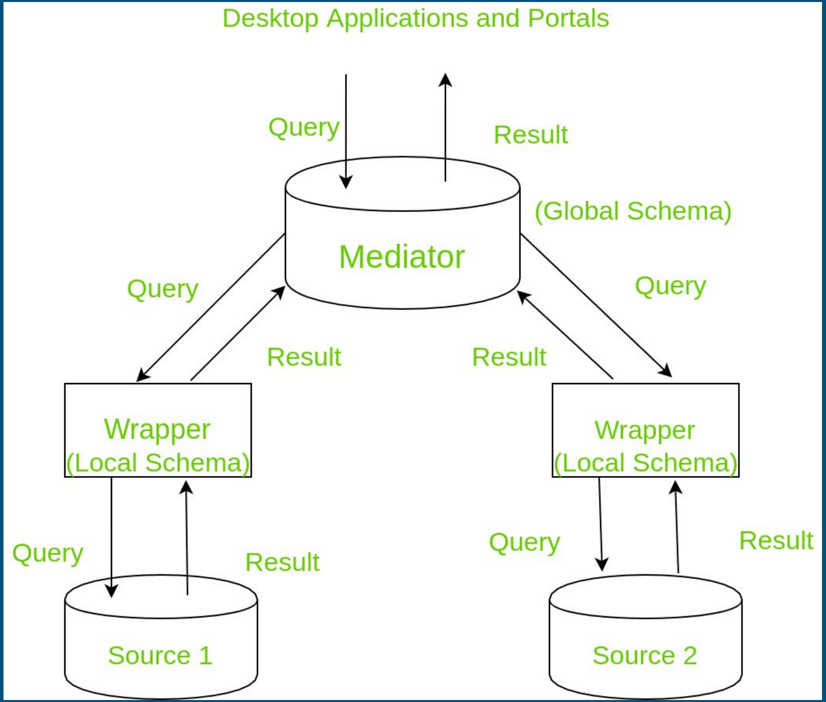
Mediation Process in Data Integration

- **Step 5: Sources Return Results to Wrappers**
  - Each Local Data Source returns its results to the Wrapper.
  - The data might be in a different structure or format, based on the local source.
- **Step 6: Wrappers Send Results Back to the Mediator**
  - The Wrappers reformat or translate the local results to match the Global Schema.
  - They send the processed results back to the Mediator.



Mediation Process in Data Integration

- **Step 7: Mediator Integrates the Results**
  - The Mediator collects results from all Wrappers.
  - It merges, integrates, and resolves inconsistencies between the different data.
  - A single unified result is created.
- **Step 8: Mediator Sends the Final Result to the User**
  - The Mediator returns the final integrated result to the user/application.
  - The user sees one complete answer, as if the data came from a single database



VDIS Categories

VDISs can be categorized according to the following three main axes:

I. Common data model and query language
II. Mapping Language
III. Data storage method

I. Common data model and query language

- The data model and query language that is exposed by the wrappers to the mediator and by the mediator to the applications

- Commonly used data models include the relational, XML and object-oriented data model

II. Mapping Language

- Global As View (GAV)
- Local As View (LAV) and
- Global and Local As View (GLAV)

III. Data storage method

- Where the data are actually stored?
- Two approaches:
    - Materialized Approach (Eager / In-Advance / Data Warehousing)
    - Virtual Approach (Lazy / On-the-Fly / Mediation)

Materialized Approach

What happens?

- The data from all sources is copied and stored in a central repository, often called a data warehouse or materialized global database.

- This data is preloaded and stored in advance, before users run queries.

How it works?

- Data from different sources is collected periodically and integrated into one centralized database.

- Queries are run directly on this centralized, materialized data.

Materialized Approach

Advantages:

- Fast query response time because all the data is already in one place.
- No need to access remote sources in real-time, which avoids delays.

Disadvantages:

- Data may become outdated, since it's only updated at intervals.
- Requires extra storage space and data maintenance efforts.

Virtual Approach

What happens?

- Data stays in the original sources (databases, files, etc.).
- The global database is virtual, meaning there’s no central storage of data.

How it works?

- When a user submits a query to the system, it is translated into sub-queries that are sent to the individual data sources in real-time.
- The system gathers and combines the results on-the-fly to answer the query.

Virtual Approach

Advantages:

- Always up-to-date because data is fetched live from the sources.
- No need for extra storage since data isn't duplicated.

Disadvantages:

- Query performance may be slower, due to network delays and source response times.
- Depends on the availability and performance of the individual sources

Analogy

Materialized Approach:

- Like taking photos of all your books and keeping them in one album. Easy to browse, but not always the latest editions.

Virtual Approach:

- Like calling each library in real time to check if a book is available. Always current, but takes more time.

Materialize Vs Virtual

Aspect	Materialized Approach	Virtual Approach
Data Location	Stored centrally (data warehouse)	Stays in original sources
Data Freshness	May be outdated (periodic updates)	Always up-to-date (real-time access)
Query Speed	Fast (local access)	Slower (remote access in real-time)
Storage Needs	Requires additional storage	No extra storage needed
System Dependence	Independent of sources' availability	Depends on sources being online

Mapping- Global As View

- Global schema is described in terms of the local schemas
- The correspondence between the local schemas and the global schema can be described through a set of mappings of the form:

$$Vi \rightarrow I(Ri)$$

- The view Vi, which retrieves data from the sources, provides the content for the global relation Ri

Mapping- Global As View

- Example: Consider the following two GAV mappings:

  M1 : V1 → I(Book)

  M2 : V2 → I(Book_Price)

- Where,

```
V1(ISBN, title, sug_retail, authorName, 'PH'):-
          PHBook(ISBN, title, authorID, sug_retail, format),
          PHAuthor(authorID, authorName)

V2(ISBN, 'B&N', sug_retail, instock):
          PHBook(ISBN, title, authorID, sug_retail, format),
          BNNewDeliveries(ISBN, title, instock),
```

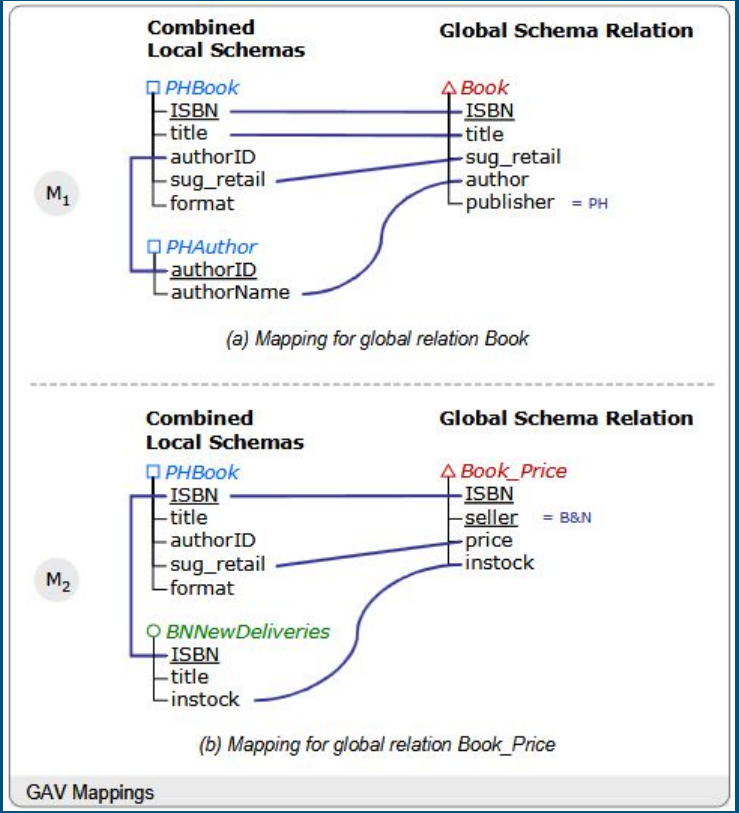
Mapping- Global As View

Mapping 1:

- The Book table in the global schema gets data from PHBook and PHAuthor. The publisher is always 'PH'.

Mapping 2:

- The Book_Price in the global schema gets data from PHBook and BNNewDeliveries. The seller is always 'B&N'



Mapping- Global As View

Advantages:

- **Simple query processing**: Easy to rewrite queries because you know exactly where the data comes from.
- **Efficient performance**: Direct mappings mean faster query translations.
- Widely used in industry (e.g., IBM WebSphere, BEA AquaLogic).

Disadvantages:

- **Tightly coupled to source schemas:** If a source changes or a new source is added, you may have to change the global schema and all mappings.
- Limited flexibility: The global schema can only represent what exists in the sources.

Mapping- Local As View

- Local schema is described in terms of the global schemas
- Local-to-global correspondences can be written in LAV as a set of mappings:

$$I(R_i) \rightarrow U_i$$

- One for every relation Ri in the local schemas, where Ui is a query over the global schema and I the identity query

Mapping- Local As View

- Example: Consider the following two LAV mappings:

  M1’ : I(PHBook$_{condensed}$) → U1

  M2’ : I(BNNewDeliveries) → U2

- Where,

  U1(ISBN, title, author, sug_retail):-

  Book(ISBN, title, sug_retail, author, “PH”),

  U2(ISBN, title, instock):-

  Book(ISBN, title, sug_retail, author, publisher),

  BookPrice(ISBN, “B&N”, sug_retail, instock)

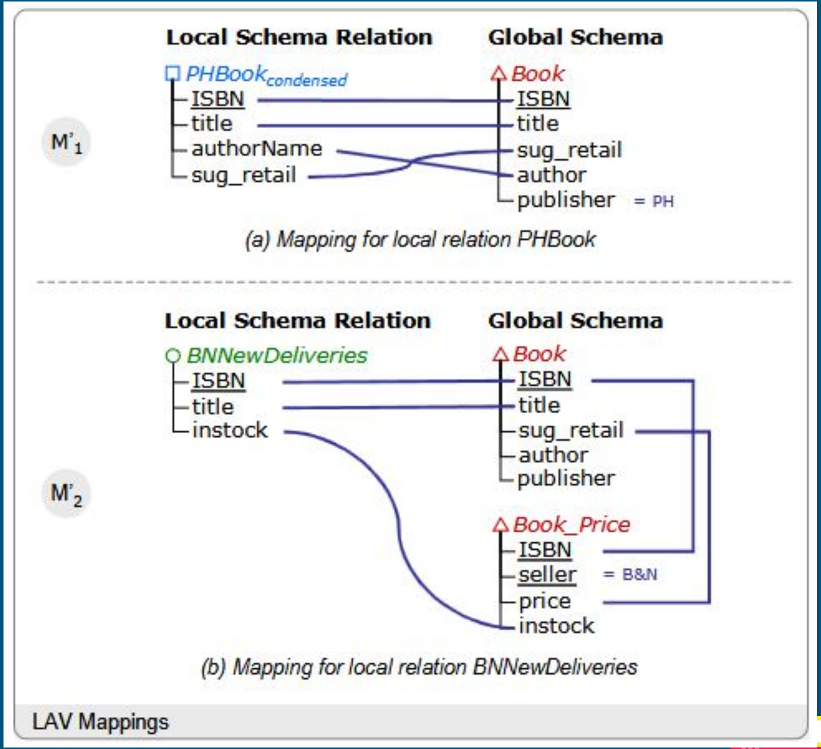
Mapping- Local As View

Mapping 1:

- The PHBook$_{condensed}$ source contains books published by 'PH' that are part of the Book global schema.

Mapping 2:

- BNNewDeliveries contains books that are in the Book global schema and their prices and stock information are available in Book_Price where the seller is 'B&N'



(a) Mapping for local relation PHBook

(b) Mapping for local relation BNNewDeliveries

Mapping- Local As View

Advantages:

- **Flexible & Scalable**: New data sources can be added easily, without changing the global schema.
- **Independent Source Registration**: Each source describes itself, without needing to know about other sources

Disadvantages:

- **Query Answering is Harder:** Figuring out how to rewrite queries to use the sources is more complex.
- **Incomplete Data**: Some sources may not have all the data you need. You work with possible worlds, which can be tricky.

Mapping- Global and Local As View

- Generalization of both GAV and LAV
- Mapping Format:

$$Vi \rightarrow Ui$$

Vi: Query over local schema
Ui: Query over global schema

Mapping- Global and Local As View

Advantages:

- Combines the flexibility of LAV and the simplicity of GAV.
- Supports independent source registration.
- Can express complex mappings not possible with only GAV or LAV.

Disadvantages:

- Query answering is complex, typically requires certain answer semantics.
- Needs advanced query rewriting algorithms