DATA INTEGRATION

CS561-SPRING 2014
WPI, MOHAMED ELTABAKH

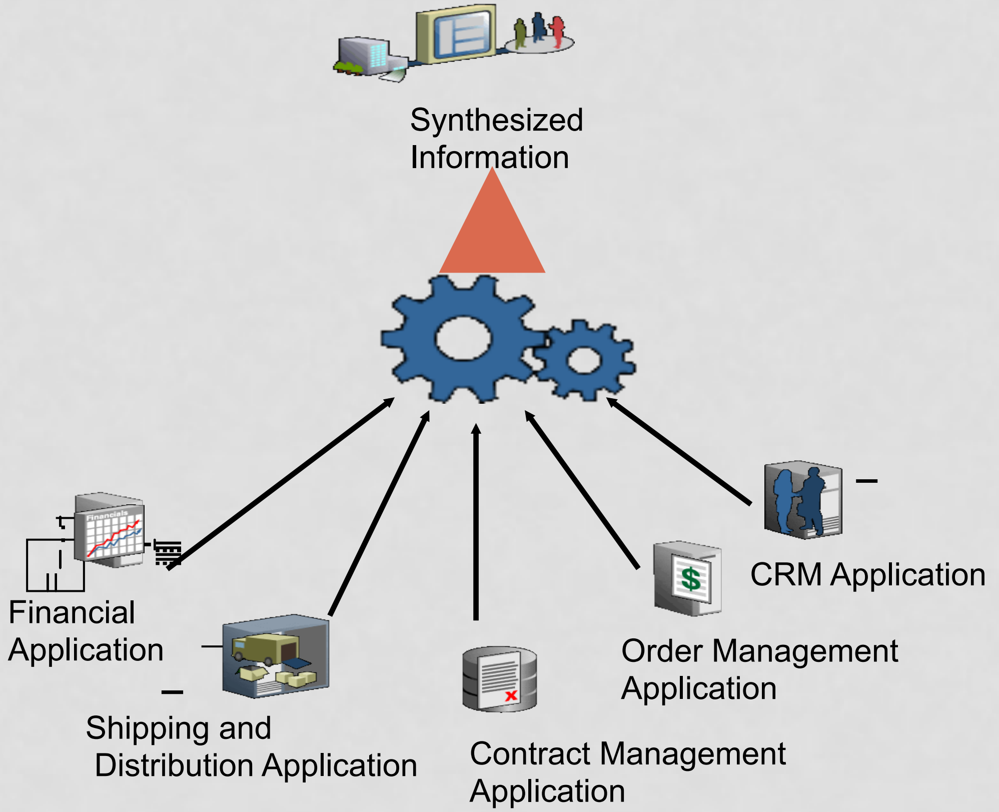
DATA INTEGRATION

- **Motivation**
  - Many databases and sources of data that need to be integrated to work together
  - Almost all applications have many sources of data

- **Data Integration**
  - Is the process of integrating data from multiple sources and probably have a single view over all these sources
    - And answering queries using the combined information
  - Integration can be ***physical*** or ***virtual***
    - ***Physical:*** Coping the data to warehouse
    - ***Virtual:*** Keep the data only at the sources

DATA INTEGRATION

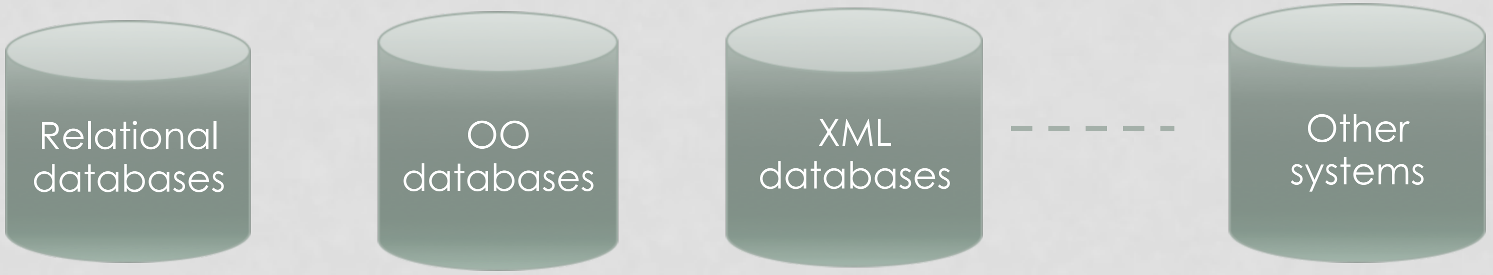
- Data integration is also valid within a single organization
  - Integrating data from different departments or sectors



HETEROGENEITY PROBLEMS

- The main problem is the ***heterogeneity*** among the data sources

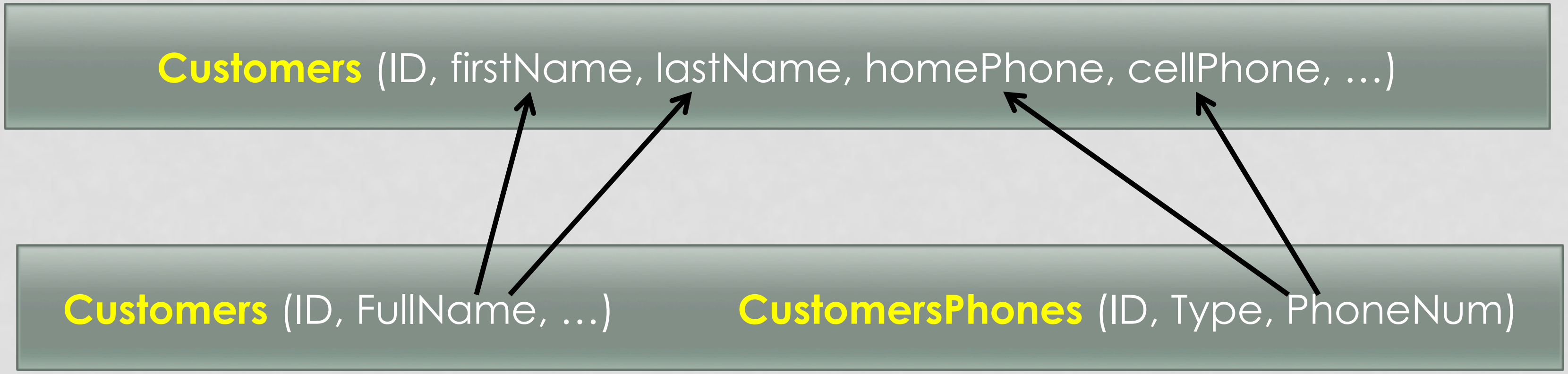
- **Source Type Heterogeneity**
  - Systems storing the data can be different



HETEROGENEITY PROBLEMS

- **Communication Heterogeneity**
  - Some systems have web interface others do not
  - Some systems allow direct query language others offer APIs

- **Schema Heterogeneity**
  - The structure of the tables storing the data can be different (even if storing the same data)



Customers (ID, firstName, lastName, homePhone, cellPhone, …)

Customers (ID, FullName, …)

CustomersPhones (ID, Type, PhoneNum)

HETEROGENEITY PROBLEMS

- **Data Type Heterogeneity**
  - Storing the same data (and values) but with different data types
  - E.g., Storing the phone number as ***String*** or as ***Number***
  - E.g., Storing the name as ***fixed length*** or ***variable length***

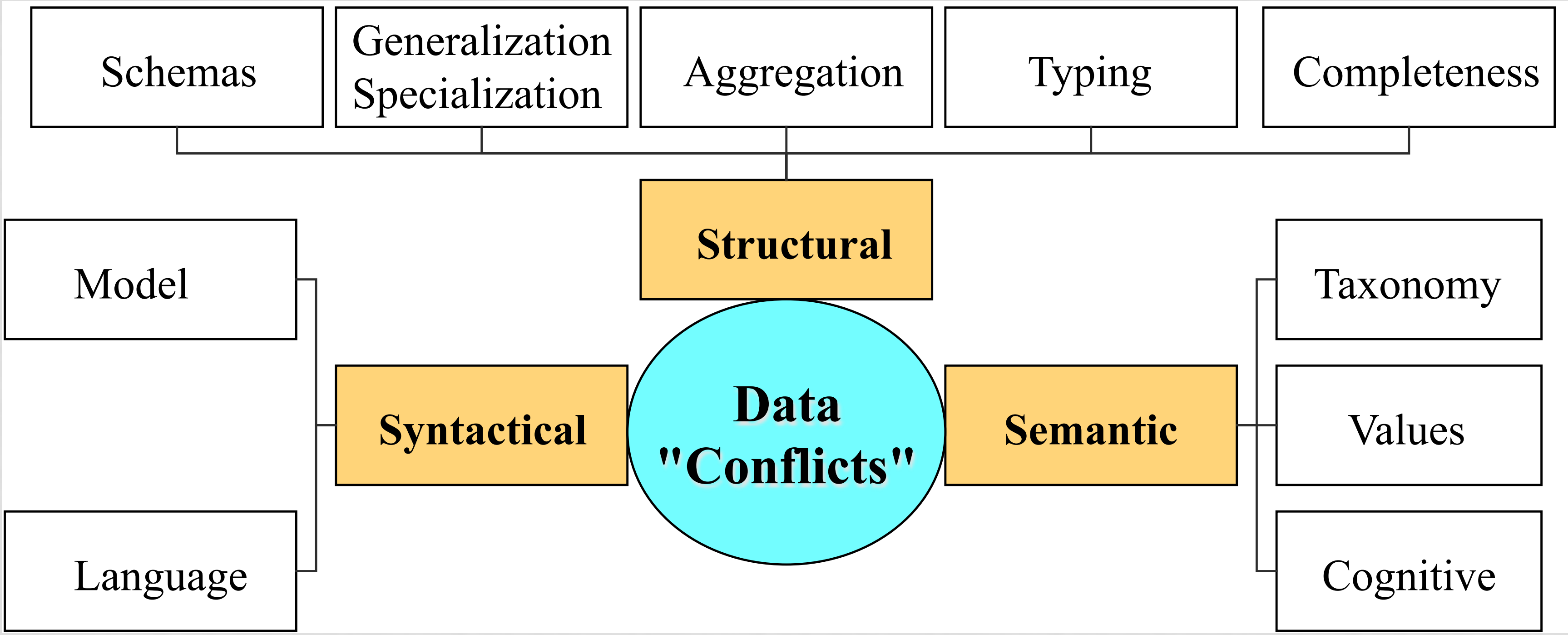
- **Value Heterogeneity**
  - Same logical values stored in different ways
  - E.g., ‘Prof’, ‘Prof.’, ‘Professor’
  - E.g., ‘Right’, ‘R’, ‘1’ ……… ‘Left’, ‘L’, ‘-1’

HETEROGENEITY PROBLEMS

- **Semantic Heterogeneity**
  - Same values in different sources can mean different things
  - E.g., Column ‘Title’ in one database means ‘Job Title’ while in another database it means ‘Person Title’

Data integration has to deal with all such issues and more

REASONS OF HETEROGENEITY



Schemas

Generalization Specialization

Aggregation

Typing

Completeness

Structural

Model

Language

Syntactical

Data "Conflicts"

Semantic

Taxonomy

Values

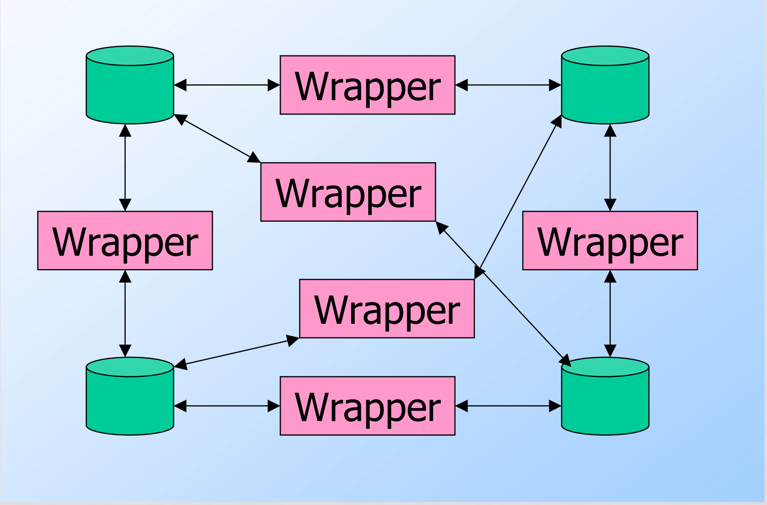
Cognitive

MODELS OF DATA INTEGRATION

- **Federated Databases**
- **Data Warehousing**
- **Mediation**

1- FEDERATED DATABASES

- Simplest architecture
- Every pair of sources can build their own mapping and transformation
- Source X needs to communicate with source Y → build a mapping between X and Y
  - Does not have to be between all sources (on demand)



Advantages

1- if many sources and only very few are communicating

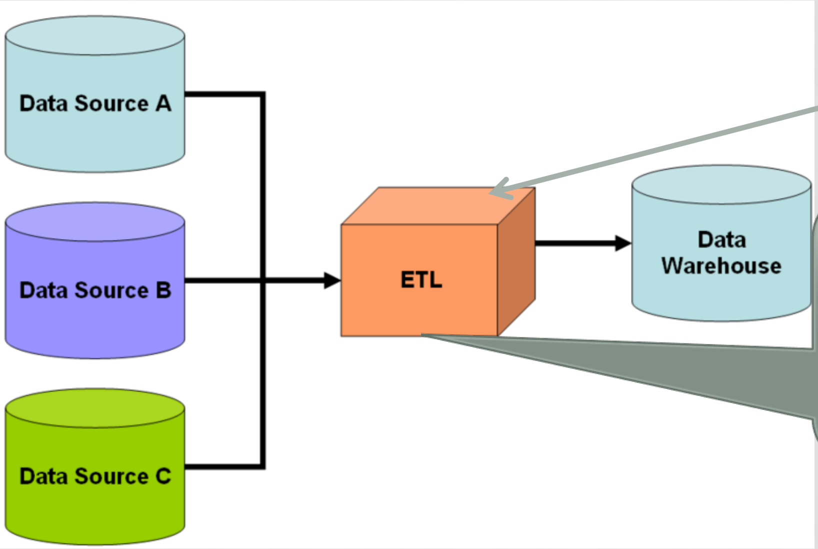
Disadvantages

1- if most sources are communicating (n^2 mappings)

2- If sources are dynamic (need to change many mappings)

2- DATA WAREHOUSING

- Very common approach
- Data from multiple sources are ***copied and stored*** in a warehouse
  - Data is materialized in the warehouse
- Users can then query the warehouse database only

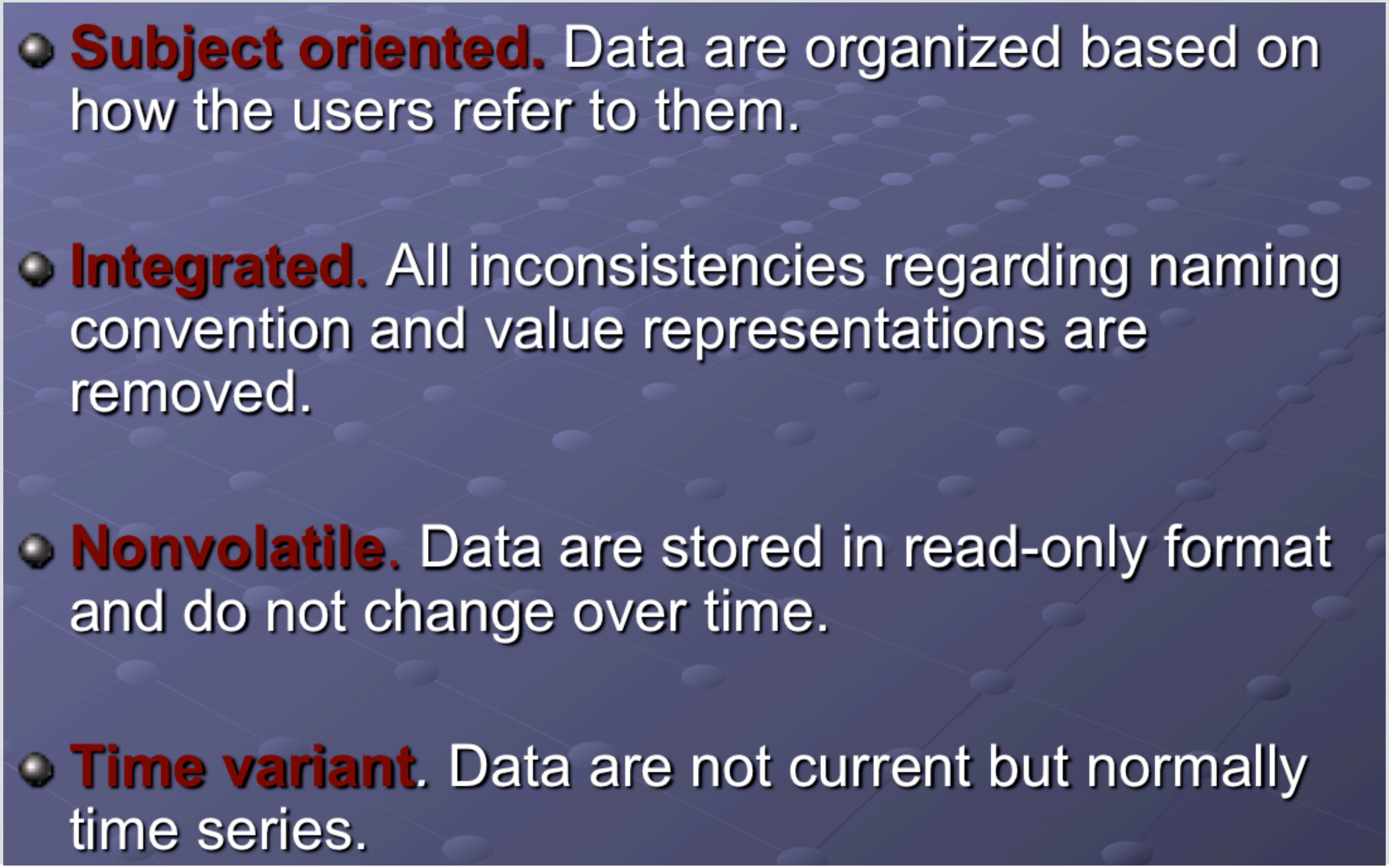


ETL: Extract-Transform-Load process

- ETL is totally performed outside the warehouse

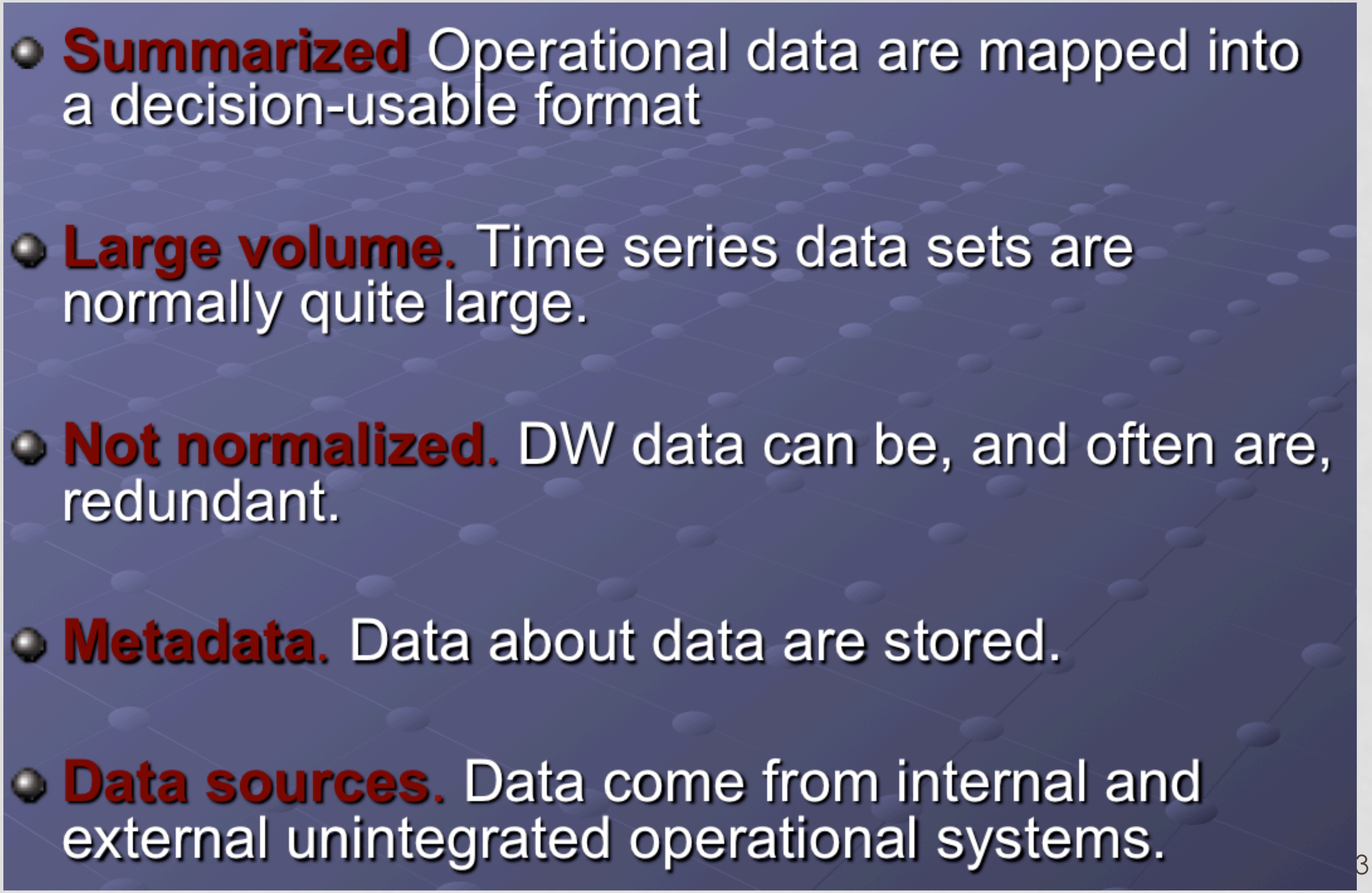
- Warehouse only stores the data

CHARACTERISTICS OF DW (I)



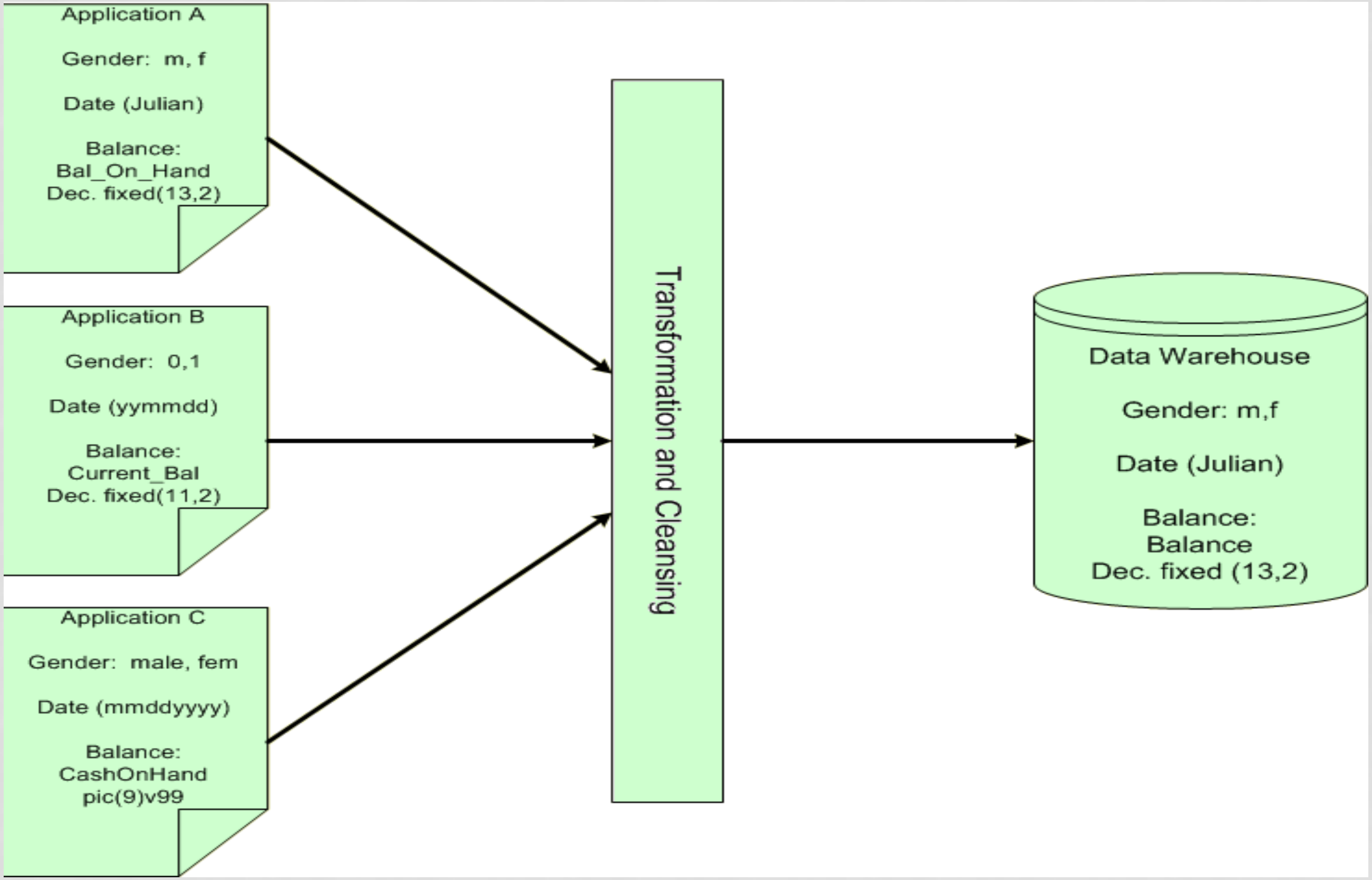
- **Subject oriented.** Data are organized based on how the users refer to them.
- **Integrated.** All inconsistencies regarding naming convention and value representations are removed.
- **Nonvolatile.** Data are stored in read-only format and do not change over time.
- **Time variant.** Data are not current but normally time series.

CHARACTERISTICS OF DW (II)



- **Summarized** Operational data are mapped into a decision-usable format
- **Large volume.** Time series data sets are normally quite large.
- **Not normalized.** DW data can be, and often are, redundant.
- **Metadata.** Data about data are stored.
- **Data sources.** Data come from internal and external unintegrated operational systems.

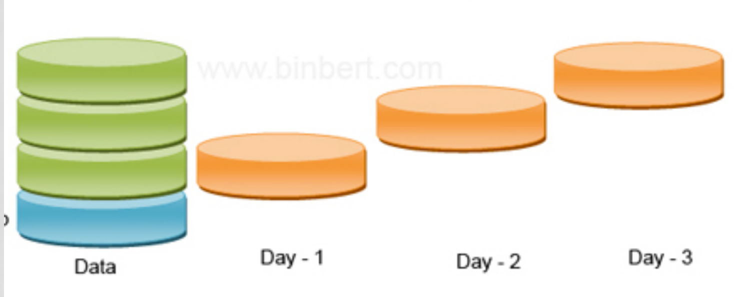
ETL PROCESSING



DW: SYNCHRONIZATION

- How to synchronize the data between the sources and the warehouse???

Complete Rebuild



Incremental Update

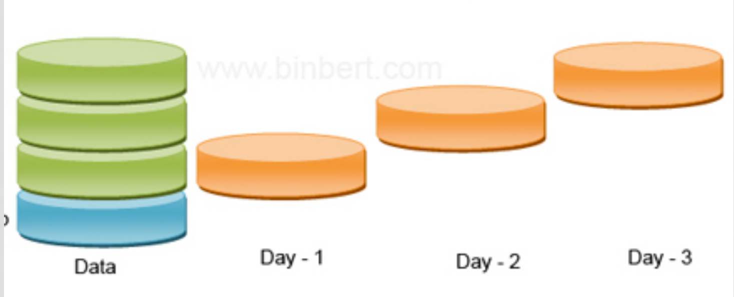
In both approaches the warehouse is not up-to-date at all times

DW: SYNCHRONIZATION

Complete Rebuild

- Periodically re-build the warehouse from the sources (e.g., every night or every week)
- (+) The procedure is easy
- (-) Expensive and time consuming

DW: SYNCHRONIZATION



Incremental Update

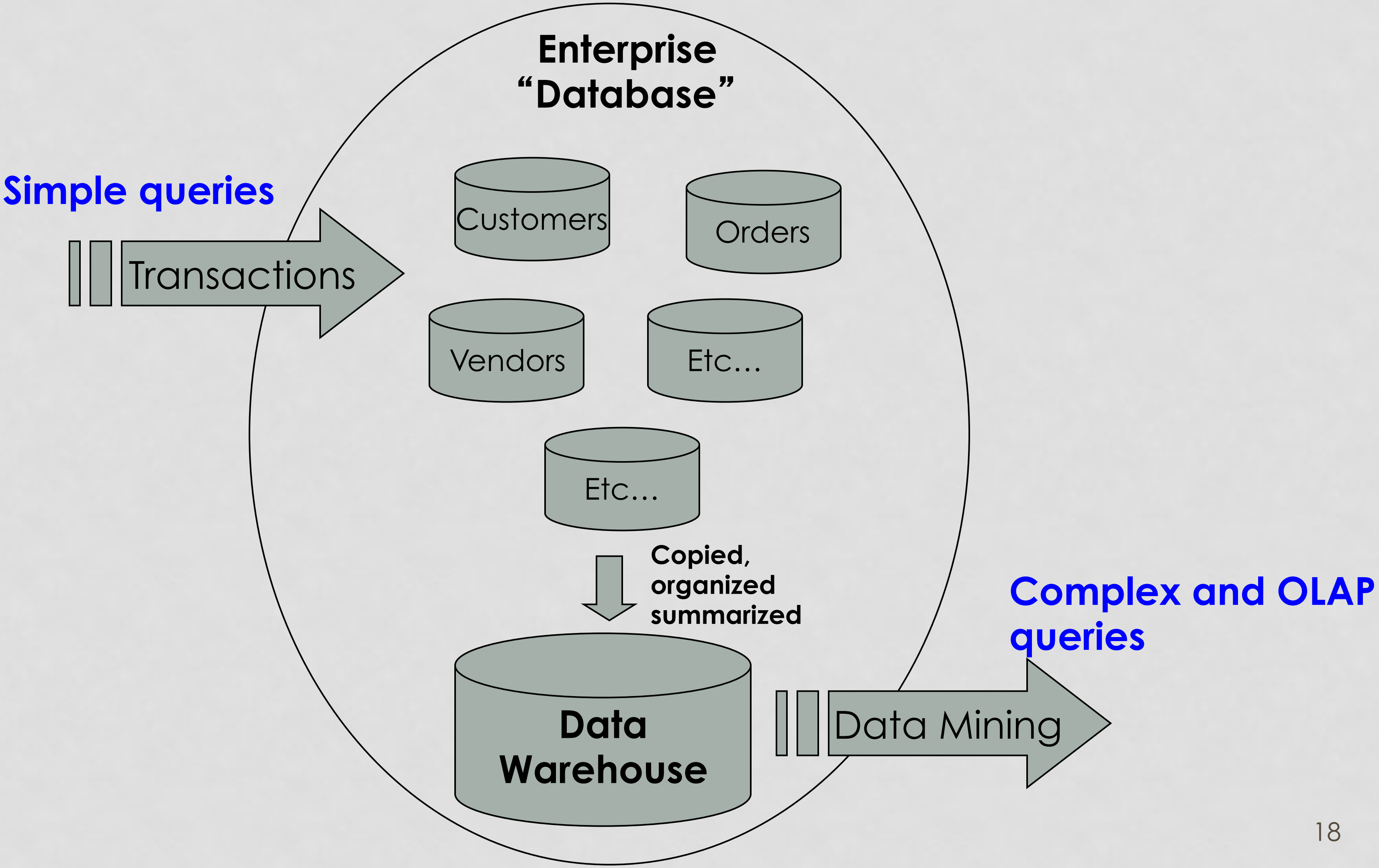
- **Periodically update the warehouse based on the changes in the sources**

- **(+) Less expensive and efficient**

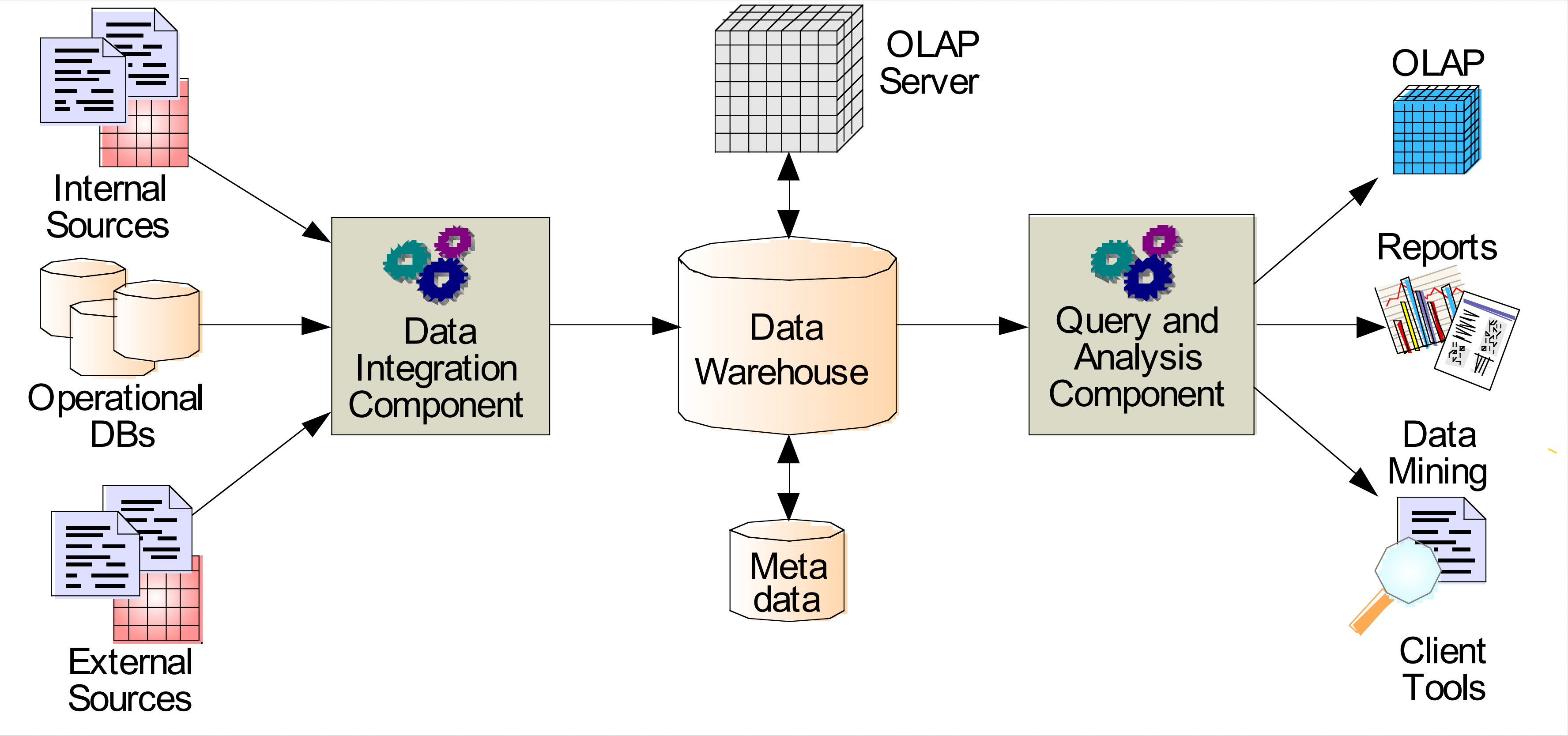
- **(-) More complex to perform incremental update**

- **(-) Requires sources to keep track of their updates**

DATA WAREHOUSING



TRADITIONAL DW ARCHITECTURE

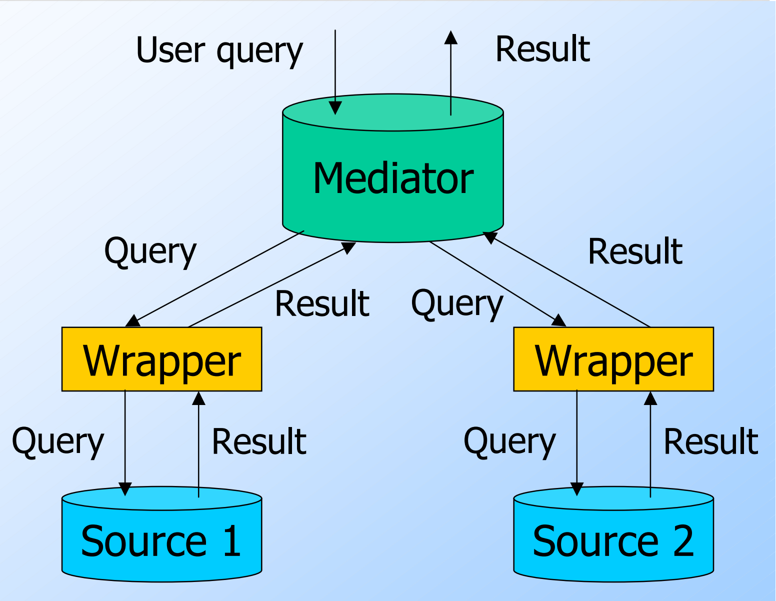


3- MEDIATION

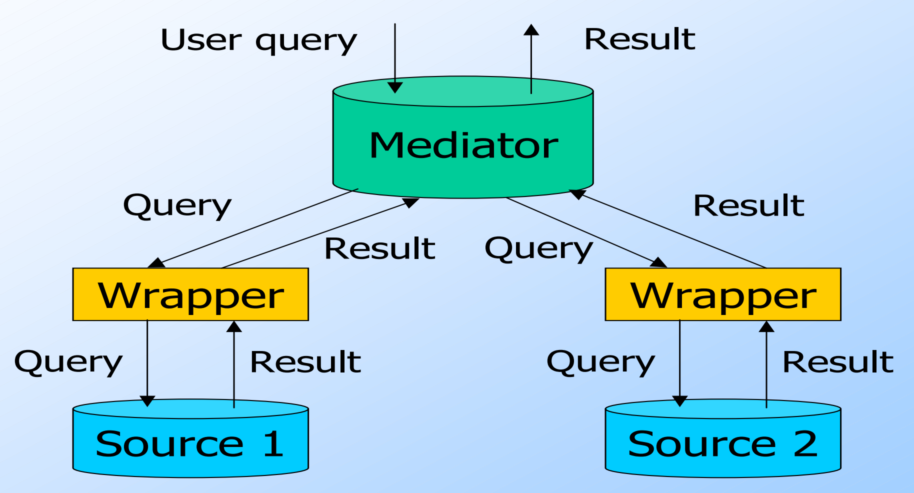
- Mediator is a virtual view over the data (it does not store any data)
  - Data is stored only at the sources

- Mediator has a virtual schema that combines all schemas from the sources

- The mapping takes place at query time
  - This is unlike warehousing where mapping takes place at upload time



MEDIATION: DATA MAPPING



Given a user query

- Query is mapped to multiple other queries
- Each query (or set of queries) are sent to the sources
- Sources evaluate the queries and return the results
- Results are merged (combined) together and passed to the end-user

MEDIATION: EXAMPLE

- Mediator Schema

Cust (ID, firstName, LastName, …)
CustPhones (ID, Type, PhoneNum, …)

- Source 1 Schema

Customers (ID, firstName, lastName, homePhone, cellPhone, …)

- Source 2 Schema

Customers (ID, FullName, …)
CustomersPhones (ID, Type, PhoneNum)

What if we need, first name, last name, and cell phone of customer ID =100?

MEDIATION: EXAMPLE

- Mediator Schema

Cust (ID, FirstName, LastName, …)
CustPhones (ID, Type, PhoneNum, …)

```
Select C.FirstName, C.LastName, P.PhoneNum
From Cust C, CustPhones P
Where C.ID = P.ID
And C.ID = 100
And P.Type = “celll”;
```

Map to source 1

```
Select firstName, lastName, cellPhone
From Customers
Where C.ID = 100;
```

- Source 1 Schema

Customers (ID, firstName, lastName, homePhone, cellPhone, …)

MEDIATION: EXAMPLE

- Mediator Schema

Cust (ID, FirstName, LastName, …)
CustPhones (ID, Type, PhoneNum, …)

```
Select C.FirstName, C.LastName, P.PhoneNum
From Cust C, CustPhones P
Where C.ID = P.ID
And C.ID = 100
And P.Type = “celll”;
```

Map to source 2

Function that returns the first name

```
Select First(C.FullName), Last(C.FullName),
            P.PhoneNum
From Customers C, CustomersPhones P
Where C.ID = P.ID
And C.ID = 100
And P.Type = “celll”;
```

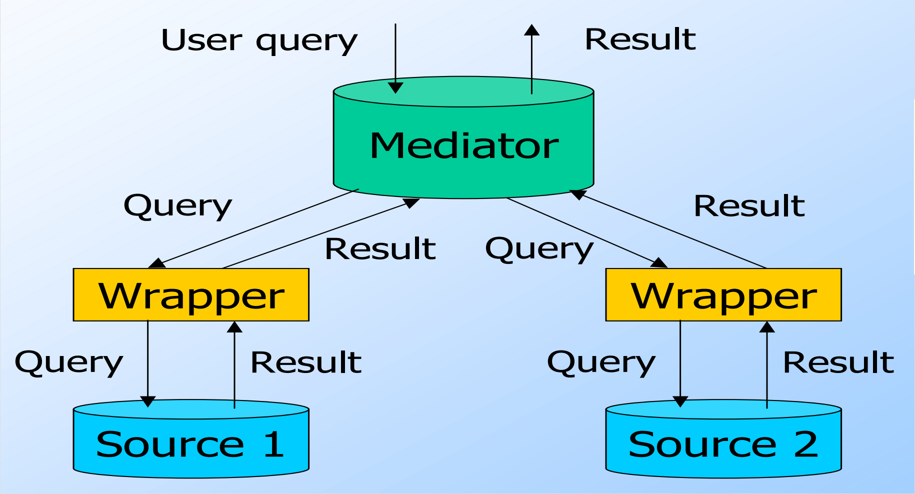
- Source 2 Schema

Customers (ID, FullName, …)
CustomersPhones (ID, Type, PhoneNum)

MEDIATION: WRAPPERS

- Usually ***wrappers*** are the components that perform the mapping of queries

- One approach is to use ***templates with parameters***
  - If the mediator query matches a template, then replace the parameters and execute the query
  - If no template is found, return empty results



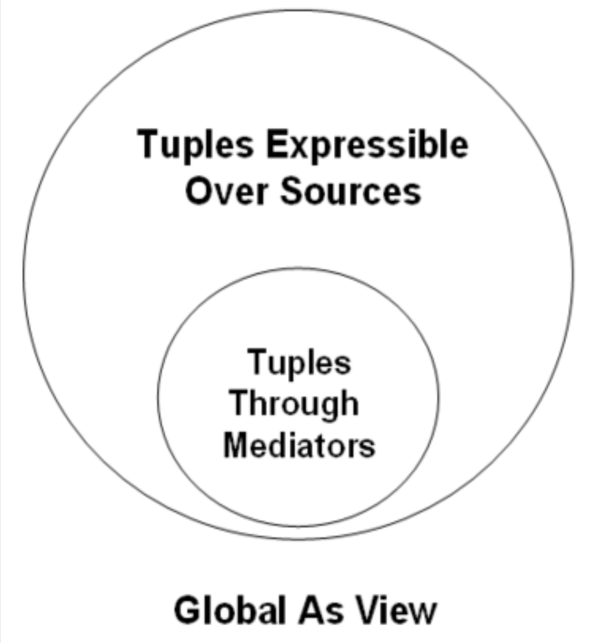
Designing these template is a complex process because they need to be flexible and represent many queries

MEDIATOR TYPES

- **Global As View (GAV)**
- **Local As View (LAV)**

GLOBAL AS VIEW (GAV)

- Mediator schema acts as a view over the source schemas
- Rules that map a mediator query to source queries
- Like regular views, what we see through the mediator is a subset of the available world

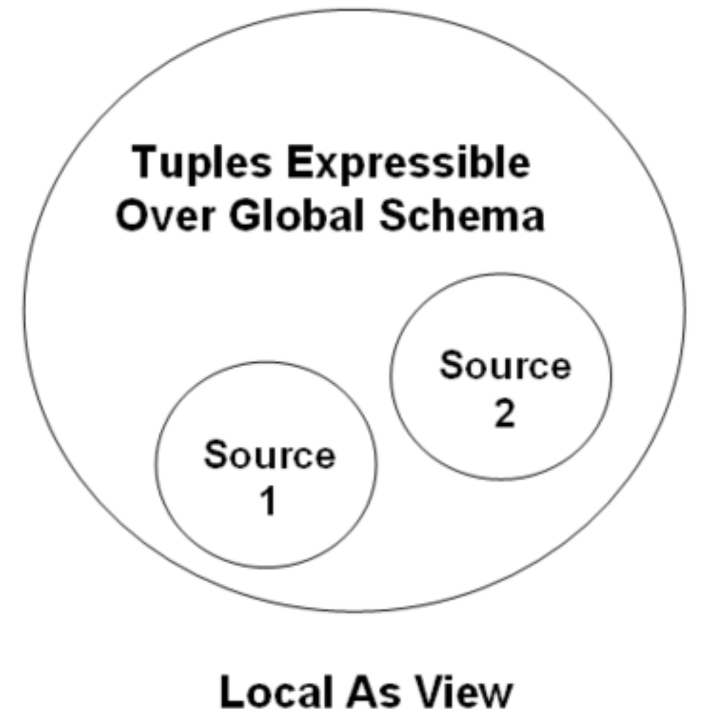


-- Limited view over the data

-- Cannot integrate/combine data from multiple sources to create new data beyond each source

LOCAL AS VIEW

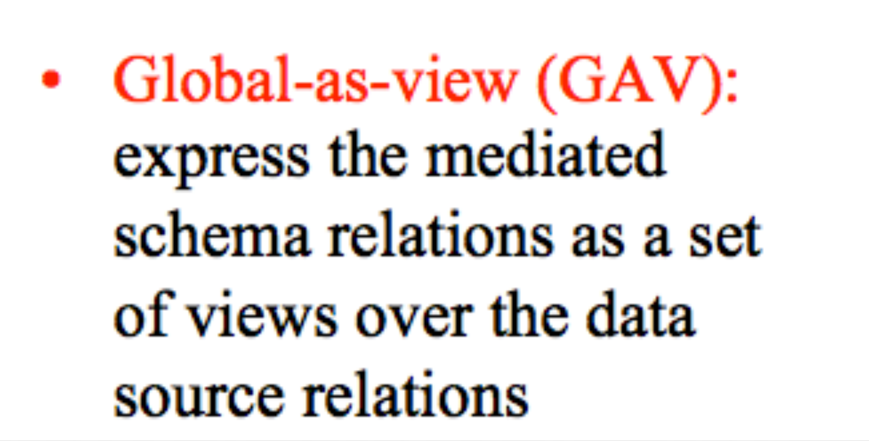
- Sources are defined in terms of the global schema using expression
- Every source provides expressions on how it can generate pieces of the global schema
- Mediator can combine these expressions to find all possible ways to answer a query



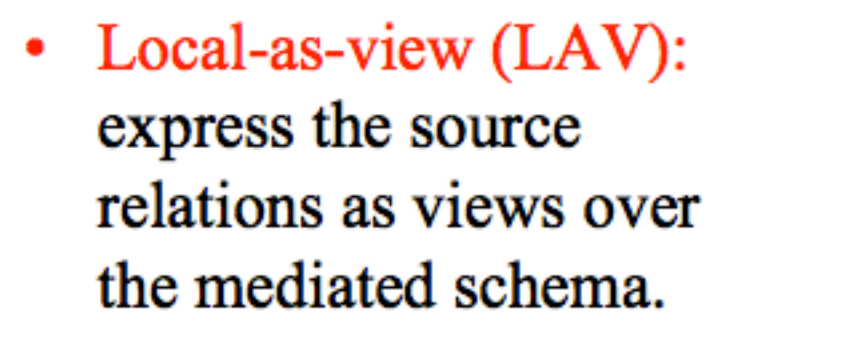
-- Covers more data beyond each source individually

-- more complex than GAV

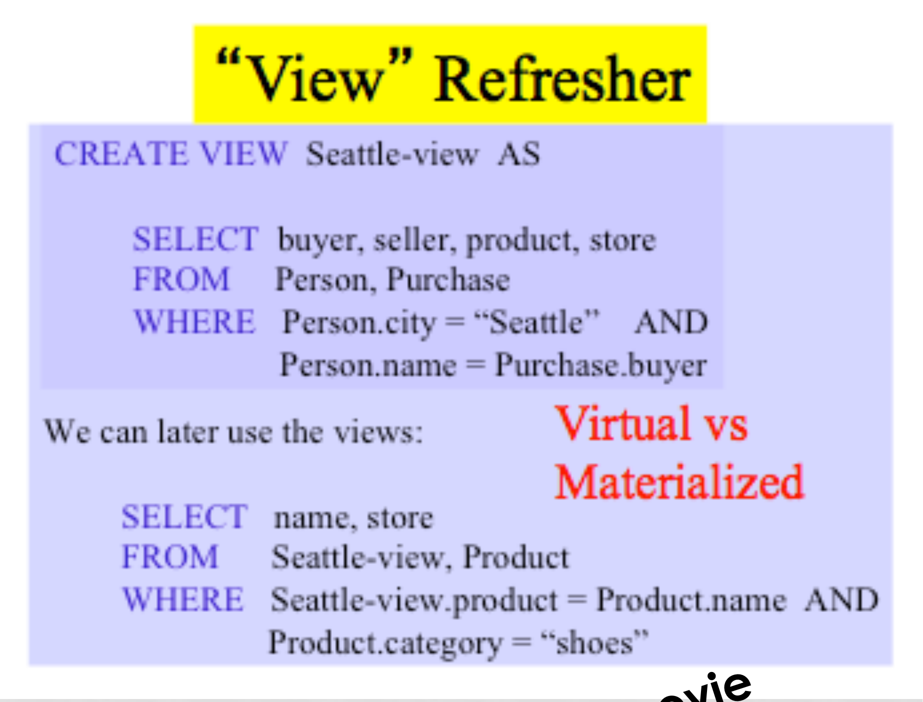
APPROACHES FOR RELATING SOURCE & MEDIATOR SCHEMAS



- **Global-as-view (GAV):** express the mediated schema relations as a set of views over the data source relations



- **Local-as-view (LAV):** express the source relations as views over the mediated schema.



"View" Refresher

```
CREATE VIEW  Seattle-view  AS

        SELECT  buyer, seller, product, store
        FROM    Person, Purchase
        WHERE   Person.city = "Seattle"   AND
                Person.name = Purchase.buyer
```

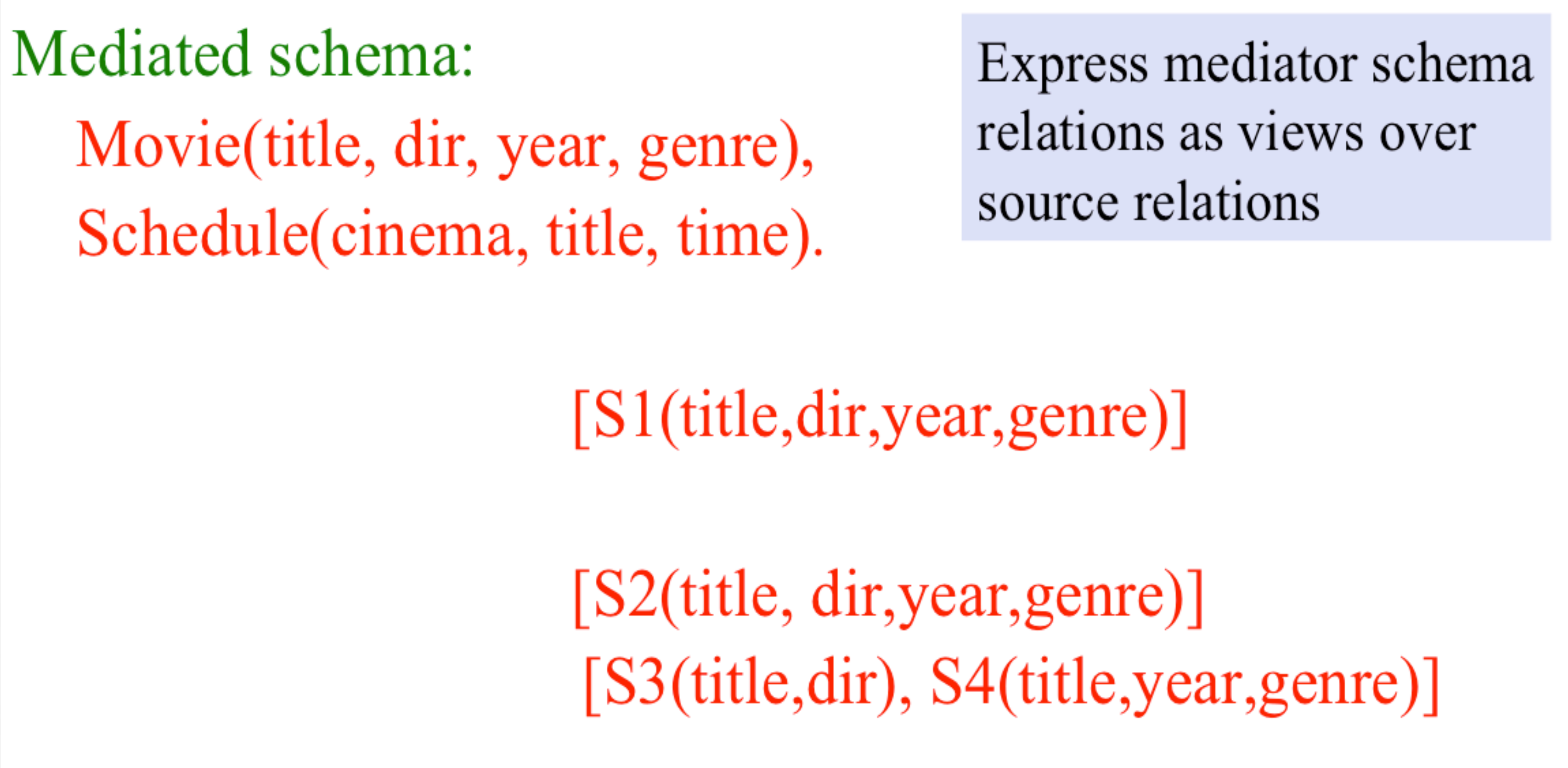
We can later use the views:

Virtual vs Materialized

```
        SELECT  name, store
        FROM    Seattle-view, Product
        WHERE   Seattle-view.product = Product.name  AND
                Product.category = "shoes"
```

Let' s compare them in a movie Database integration scenario..

GLOBAL AS VIEW (GAV)



Mediated schema:

Movie(title, dir, year, genre),

Schedule(cinema, title, time).

Express mediator schema relations as views over source relations

[S1(title,dir,year,genre)]

[S2(title, dir,year,genre)]

[S3(title,dir), S4(title,year,genre)]

GLOBAL AS VIEW (GAV)

Mediated schema:

Movie(title, dir, year, genre),

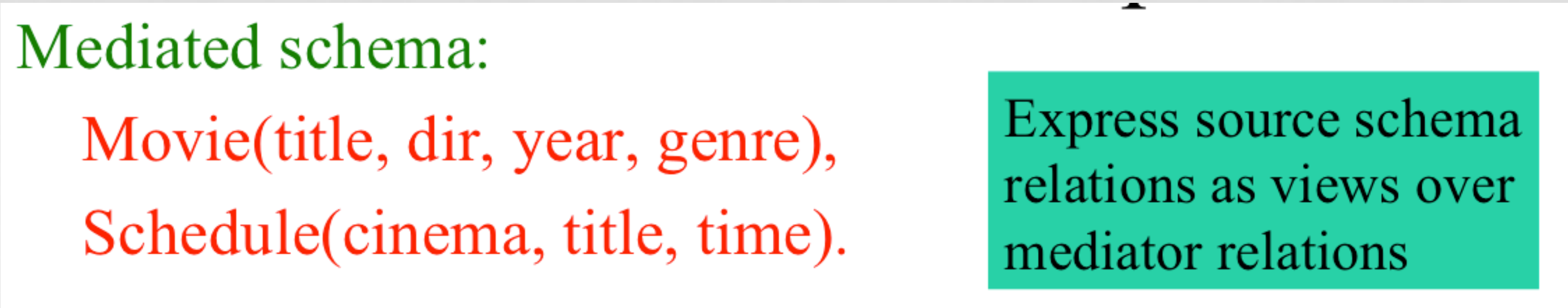
Schedule(cinema, title, time).

Express mediator schema relations as views over source relations

```
Create View Movie AS
  select *  from S1      [S1(title,dir,year,genre)]
  union
  select  * from S2      [S2(title, dir,year,genre)]
  union                   [S3(title,dir), S4(title,year,genre)]
  select S3.title, S3.dir, S4.year, S4.genre
  from  S3, S4
  where S3.title=S4.title
```

Mediator schema relations are Virtual views on source relations

LOCAL AS VIEW (LAV)



Mediated schema:

Movie(title, dir, year, genre),

Schedule(cinema, title, time).

Express source schema relations as views over mediator relations

Create Source S1 AS

select * from Movie

Create Source S3 AS

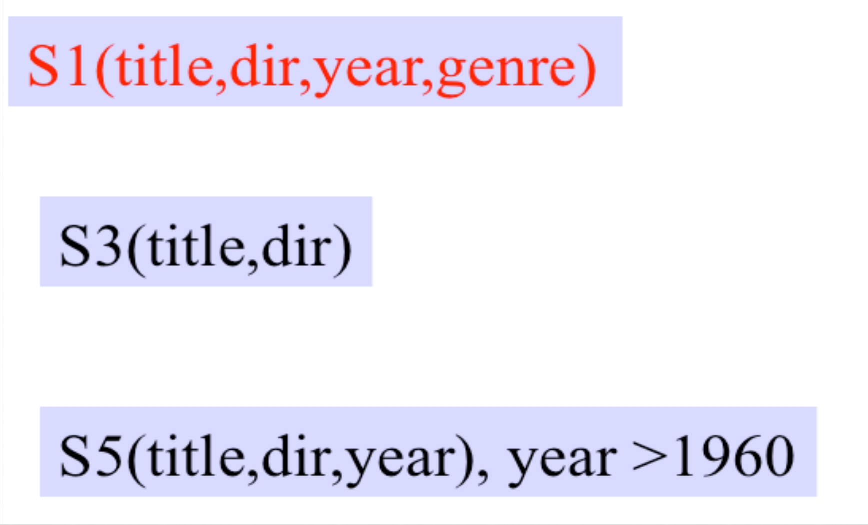
select title, dir from Movie

Create Source S5 AS

select title, dir, year

from Movie

where year > 1960



S1(title,dir,year,genre)

S3(title,dir)

S5(title,dir,year), year >1960

Sources are "materialized views" of mediator schema

GLOBAL (GOV) VS. LOCAL (LOV)

Mediated schema:

Movie(title, dir, year, genre),
Schedule(cinema, title, time).

Source S4: S4(cinema, genre)

GoV

```
Create View Movie AS
   select NULL, NULL, NULL, genre
   from S4
 Create View Schedule AS
    select cinema, NULL, NULL
    from S4.
```

But what if we want to find which cinemas are playing comedies?

Lossy mediation

LoV

```
Create Source S4
   select cinema, genre
   from Movie m, Schedule s
   where m.title=s.title
```

Now if we want to find which cinemas are playing comedies, there is hope!

GAV vs. LAV

- Not modular
  - Addition of new sources changes the mediated schema
- Can be awkward to write mediated schema without loss of information
- Query reformulation easy
  - *reduces to view unfolding (polynomial)*
  - Can build hierarchies of mediated schemas
- Best when
  - Few, stable, data sources
  - well-known to the mediator (e.g. corporate integration)

Modular--adding new sources is easy

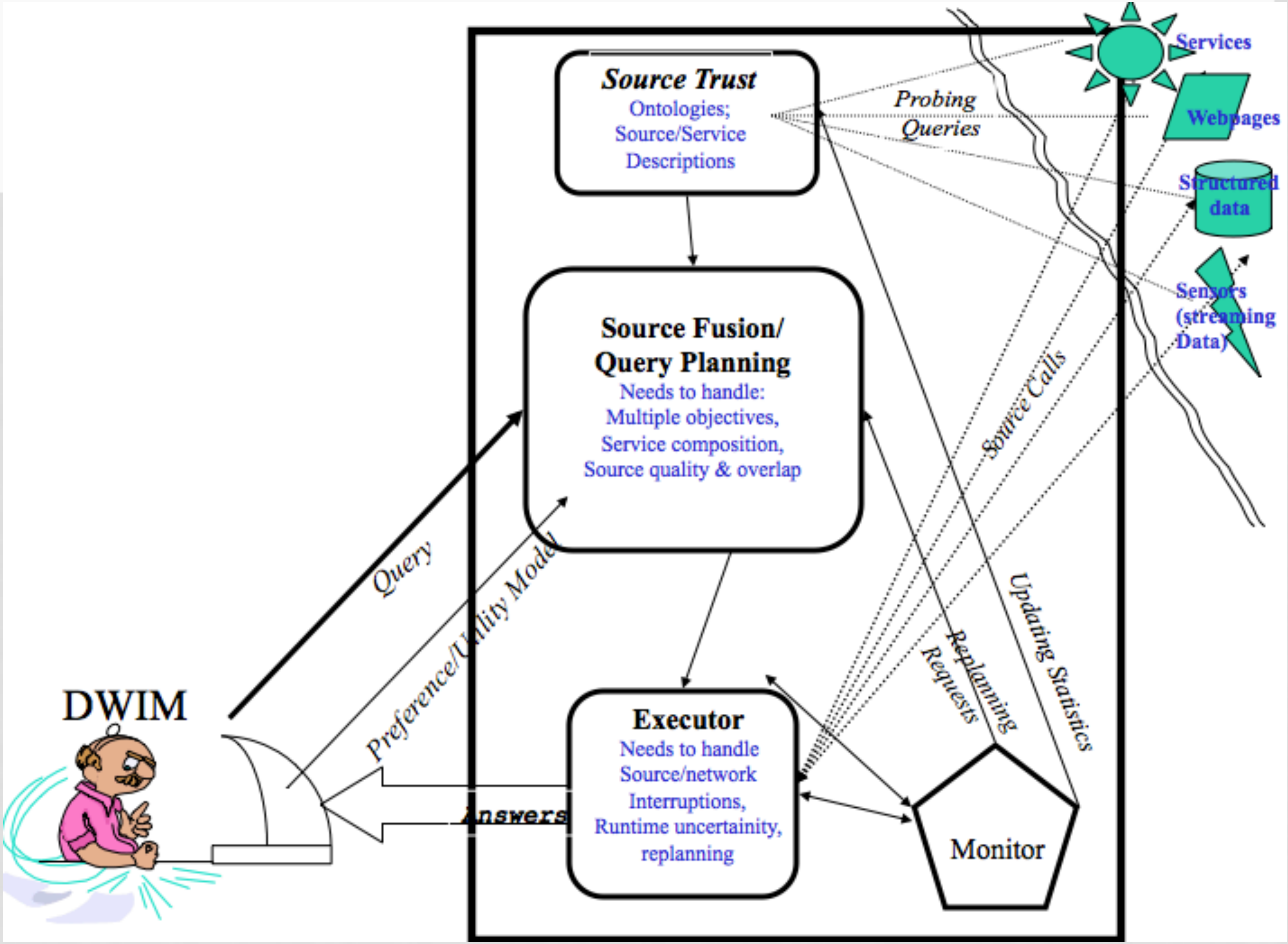
Very flexible--power of the entire query language available to describe sources

Reformulation is hard

- Involves answering queries only using views (can be intractable—see below)

Best when

- Many, relatively unknown data sources
- possibility of addition/ deletion of sources



DWIM

Query

Preference/Utility Model

Answers

Source Trust

Ontologies;
Source/Service
Descriptions

**Source Fusion/
Query Planning**

Needs to handle:
Multiple objectives,
Service composition,
Source quality & overlap

Executor

Needs to handle
Source/network
Interruptions,
Runtime uncertainity,
replanning

Monitor

Probing Queries

Source Calls

Replanning Requests

Updating Statistics

Services

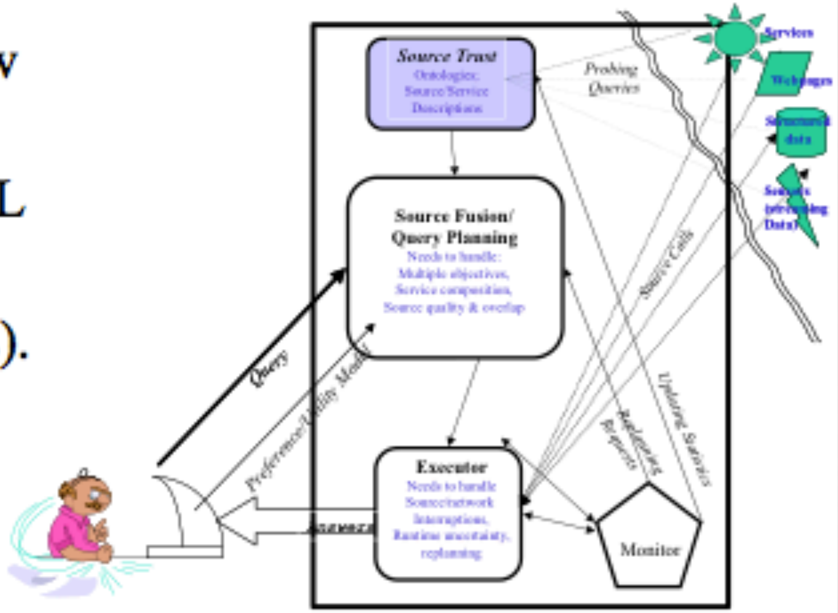
Webpages

Structured data

Sensors (streaming Data)

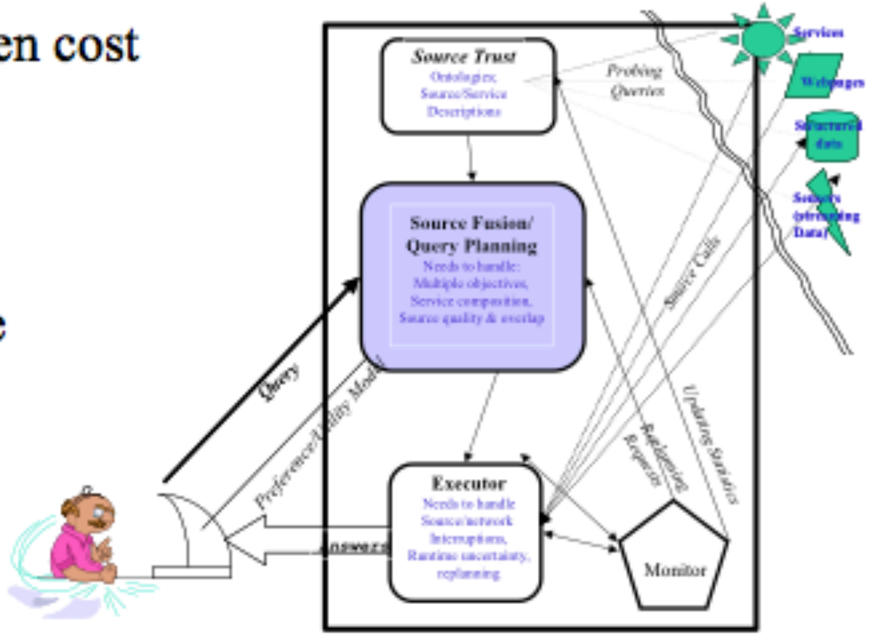
Source Descriptions

- Contains all meta-information about the sources:
  - Logical source contents (books, new cars).
  - Source capabilities (can answer SQL queries)
  - Source completeness (has *all* books).
  - Physical properties of source and network.
  - Statistics about the data (like in an RDBMS)
  - Source reliability
  - Mirror sources
  - Update frequency.



Source Fusion/Query Planning

- Accepts user query and generates a plan for accessing sources to answer the query
  - Needs to handle tradeoffs between cost and coverage
  - Needs to handle source access limitations
  - Needs to reason about the source quality/reputation



Source Trust
Ontologies;
Source/Service
Descriptions

Probing
Queries

Services

Webpages

Structured
data

Sensors
(streaming
Data)

Source Fusion/
Query Planning
Needs to handle:
Multiple objectives,
Service composition,
Source quality & overlap

Source Calls

Query

Preference/Utility Model

Executor
Needs to handle
Source/network
Interruptions,
Runtime uncertainty,
replanning

Answers

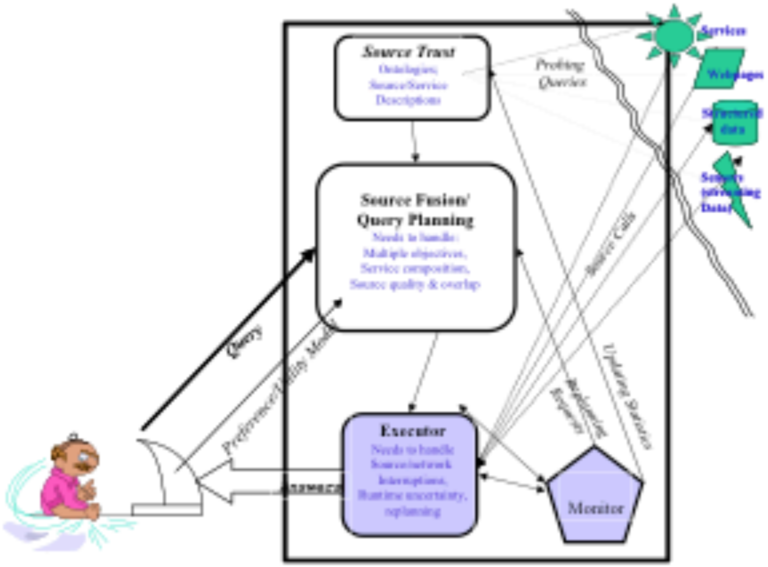
Replanning
Requests

Updating Statistics

Monitor

Monitoring/Execution

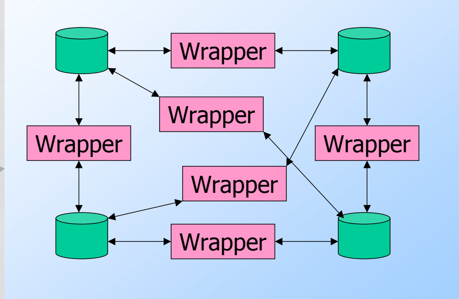
- Takes the query plan and executes it on the sources
  - Needs to handle source latency
  - Needs to handle transient/short-term network outages
  - Needs to handle source access limitations
  - May need to re-schedule or re-plan

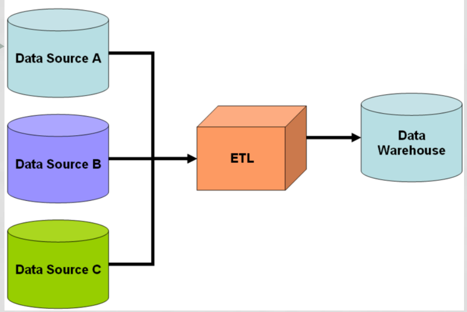


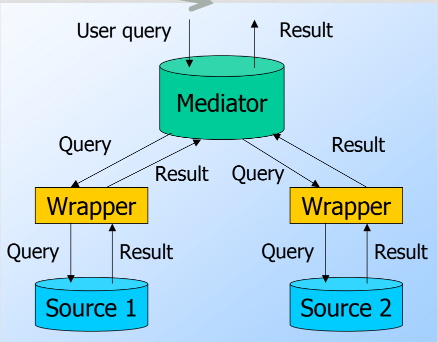
WHAT WE COVERED SO FAR …

- **Data integration** is the process of integrating data from multiple sources And answering queries using the combined information

- **Models of Data Integration**
  - **Federated Database**
  - **Data Warehouse**
  - **Mediators**
    - Global As View (GAV)
    - Local As View (LAV)







ENTITY RESOLUTION

- Data coming from different sources may be different even if representing the same objects

- ***Entity resolution*** is the process of:
  - Figuring out which records represent the same thing
  - Linking relevant records together

(John William, 252 Star rd., MA, 01609, 508-543-2222)

(John Will., 252 Star road, MA, 01609, 508-543-2222)

All of these are the same objects but they are not identical

(John William, 252 Star rd., Massachusetts , 01609-3321, 508-543-2222)

(John William, 252 Star rd., MA, 01609, (508)543-2222)

If structure is different, it becomes even harder

REASONS OF MISMATCHING

- **Misspelling**
  - “Smith”, “Smeth”, “Snith”
- **Variant names, synonyms, and abbreviations**
  - “St.”, “St”, “Street”…..“Prof”, “Professor”….”car”, “vehicle”
- **Different systems**
  - “Chin Le”, “Le, Chin”… “10/02/2000”, “10-02-2000”, “02-10-2000”
- **Different domains**
  - “YES/NO”, “1/0”, “T/F”

MECHANISMS FOR ENTITY RESOLUTION

- **Edit Distance**
  - Compare string fields using edit distance function
  - Can assign different weights to different fields

- **Normalization & Ontology**
  - Using a dictionary, replace all abbreviations with a standard forms
  - Ontology helps in synonyms

- **Clustering and Partitioning**
  - Run a clustering-based algorithm over the returned records
  - Tuples belonging to the same cluster can be further tested for matching

MERGING SIMILAR RECORDS

- **How to merge similar records???**

- In some cases, e.g., misspelling synonyms , it is possible to merge results

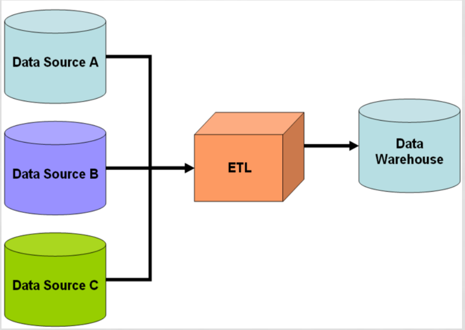
- In other cases, e.g., conflicts, there is no easy way to find the correct values
  - Report all the results we have

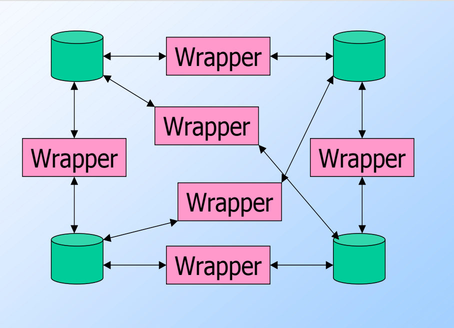
ID	Name	Address	phone
100	Susan Williams	123 Oak St.	818-457-1245
100	Susan Will.	456 Maple St.	818-457-1245

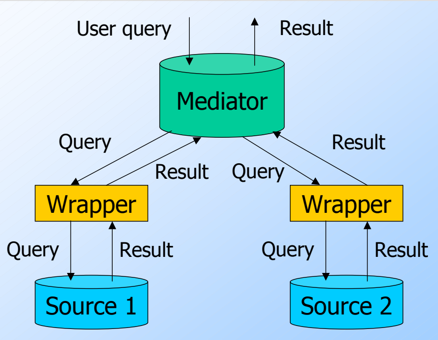
ID	Name	Address	phone
100	Susan Williams	{123 Oak St., 456 Maple St.}	818-457-1245

AUTOMATED DATA INTEGRATION

- **Data integration requires a lot of manual effort**
  - **Data warehouse** → designing and implementing the ETL module
  - **Mediators** → designing and implementing the wrappers
  - **Federated database** → designing and implementing the mapping modules (wrappers)



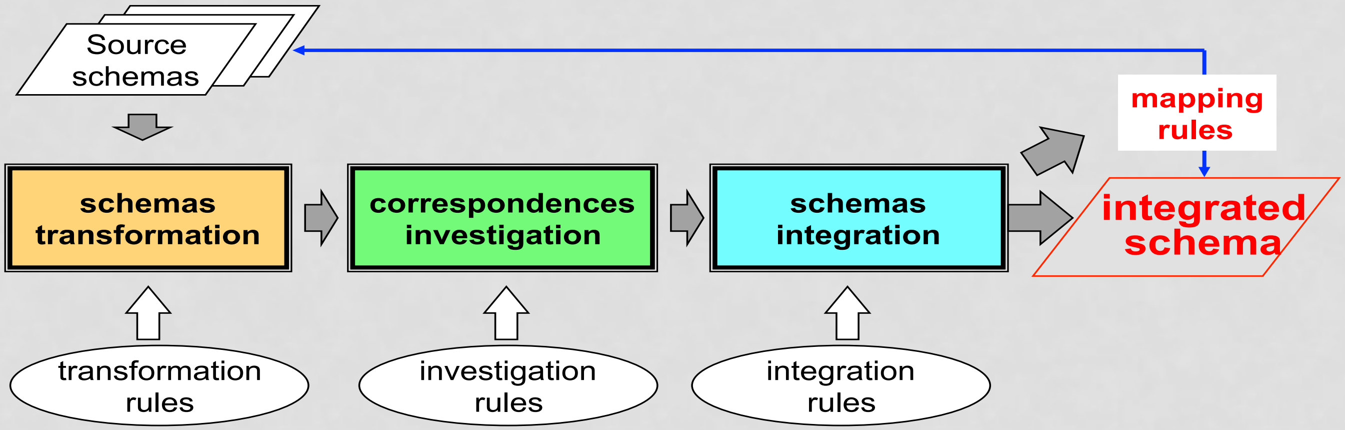




Can we automate this process ???

WORK IN PROGRESS + RECENT RESEARCH

A Generic Framework for Integration



Consider several database schemas for different bookstores

- How to match their schemas automatically ← **schema matching techniques**
- How to find matching records ← **record linkage techniques**
- How to find errors, synonyms, etc. and correct them ← **data cleansing techniques**