POLITECNICO
MILANO 1863

Technologies for Information Systems

Cinzia Cappiello
A.A. 2023-2024

These slides are based on Prof. Tanca slides. Thanks!

1

Teachers' info

Technical Lectures: Prof. Cinzia Cappiello
Communication part: Prof. Nicoletta Di Blas
Exercises: Ing. Bavaro Marcello

emails: name.surname@polimi.it

2

Course Overview

For the technical part:

- Analyze the data sources and design a **Data Integration** system

- Design a **Data Warehouse**

From the point of view of Communication, the students will learn how to clearly explain a technological or methodological issue at the correct level of abstraction, considering the common ground with the interlocutor

3

Course Material page - WeBeep

Technologies for Information Systems 2023-2024

Calendar September 2023

Lectures - Cinzia Cappiello

Date	Topic	Recordings' link
14/09/23	Introduction (slides)	

Exercise sessions - Marcello Bavaro

4

Further Information

Prerequisites: Databases I and (possibly) Databases II (the latter can also be attended in the same semester as TIS). For the students who have not attended DB1 there is a set of recorded lectures (Crash Course on Databases) that supplies the main fundamental topics.

The course is completely offered in English.

5

Exam

- The whole exam IS WRITTEN and consists of:
  - design exercises and questions on theoretical topics
  - a Communication Assignment (that must be passed in order to pass the exam), in which the student has to demonstrate the acquired communication skills.

6

Exam details

The exam consists of two main parts:

1.Theoretical Questions

Format: 2 open-ended questions.

Total Points: 10 (5 points each question).

Minimum Required Points: 5.

2.Exercises

Total Points: 22.

Minimum Required Points: 13.

The two parts can be taken in different exam calls, and that result will be retained (if sufficient) for future exam sessions (the entire academic year).

Please note: This is a closed-book examination.

7

Communication assignment

Only for the Communication part of the exam, the mark is a pass/no pass.

Once a student successfully passes the Communication component of the exam, that result will be retained for future exam sessions (the entire academic year). Even if the student does not pass the other exam sections, there is no need to retake the Communication part in subsequent exam calls.

8

Exam enrollment

In the enrollment phase of the exam the student will have the possibility to select the part or the parts that she/he wants to take.

9

WHAT'S NEW IN INFORMATION TECHNOLOGIES?

10

LOTS OF THINGS ARE CHANGED THROUGH THE YEAS

PAST → NOW

14/09/23

INTERNAL DATAS → BIG OVERVIEW, A MORE GENERAL SITUATION

↓ FOCUS ON INTERNAL SITUATION OF THE COMPANY

↓ OPERATIONAL LEVEL

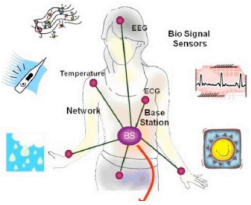
↓ DB BUILT THROUGH CRUD APPROACH

↓ INTEGRATION OF ALL THE DATABASES OF THE COMPANY TO HAVE A FULL OVERVIEW OF THE COMPANY'S SITUATION

The Challenges For Modern Information Systems

- ✓ Pre-history: focused on challenges that occur *within enterprises*
- ✓ The Web era:
  - scaling to a much larger number of sources
  - Handling sources with less structure.
- ✓ Nowadays:
  - large scientific experiments rely on data management for progress.
  - People continuously create data fragments by interacting with services on the Web.
  - Massive use of social media and smart devices
  - User-generated content merges with the Internet of Things
  - Users *as sensors and actuators*

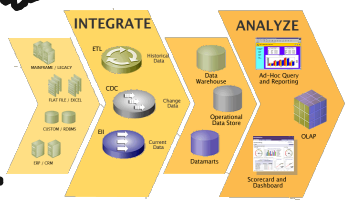
How do we make sense of this mass of data?



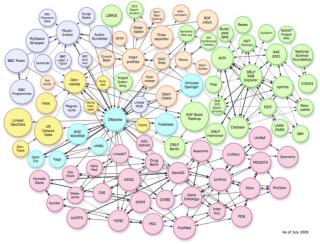
11

The four V's of Big Data

Volume → LOTS OF DATAS AND SOURCES
Velocity
Variety → HETEROGENEITY IN THE TYPE OF DBs
Veracity → QUALITY DATA. MORE INFO ⇒ HIGHER EXPOSURE TO UNCERTAINTY



We should be able to govern:
- data abundance
- data and user dynamicity and mobility
- heterogeneity, data semantics
- incompleteness/uncertainty
- interaction with the real-world



Making sense of all this data:
→ Extract useful knowledge

12

↓ DECISION LEVEL

↓ NEEDS FOR DIFFERENT INFORMATIONS PRODUCED BY PROCESSING DATAS AND BEING ABLE TO FIND THE RIGHT INFORMATION AT THE RIGHT TIME

↓ NEEDS FOR A DASHBOARD, A SYSTEM THAT THE MANAGER (A DATA SCIENTIST) CAN USE TO FIND THE REQUIRED INFORMATIONS

≠ INFORMATION PROVIDED BY ≠ SOURCES

↓ INTEGRATING INTERNAL AND EXTERNAL SOURCES EVEN WITH HETEROGENEOUS FORMAT

The new challenges in the Data Management Area

INTEGRATION HAS ALWAYS BEEN A PROBLEM, BUT NOWADAYS IT IS GETTING WORST

From the Seattle Report on Database Research (*) - what has changed in the last few years:

- **Technological breakthroughs in machine learning (ML) and artificial intelligence** (AI): rise of data science as a discipline that combines elements of data cleaning and transformation, statistical analysis, data visualization, and ML techniques
- **Cloud computing has become mainstream**. The industry now offers on-demand resources that provide on-demand, elastic storage and computation services.
- For cloud-based systems, the industry has converged on **data lakes** as novel systems for data integration
- **Modern data warehousing** query engines
- Society has become more concerned about the state of **data governance**, concentrated in data usage, e.g. quality-aware, privacy-aware, ethical and fair use of data

(*) Communications of the ACM, August 2022, Vol. 65 No. 8, Pages 72-79

DATA LAKES: BIG REPOSITORIES CONTAINING RAW DATAS THAT CAN BE ELABORATE AS WE PREFER ⇒ FLEXIBILITY

13

The new challenges in the Data Management Area (2)

Extraction of (synthetic and useful) knowledge: build environments that mimic the progressive inspecting, observing, surveying activity with which users make decisions.

Massive data analysis and processing: A process of inspecting, cleaning, transforming, and modeling data with the goal of highlighting useful information, suggesting conclusions, and supporting decision making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, in different business, science, and social science domains.

Massive data integration: People and enterprises need to integrate data and the systems that handle those data: Relational DBMSs and their extensions, legacy data and legacy DBMSs, sensors and user-generated content produce structured or unstructured data

Data warehousing: A single, complete and consistent store of data obtained from a variety of different sources made available to end users, so that they can understand and use it in a business context.[Barry Devlin]

DATA WAREHOUSE: WORKING WITH ALREADY PREPARED DATAS ⇒ CLOSED SITUATION, USEFUL WHEN YOU KNOW WHAT TO DO

14

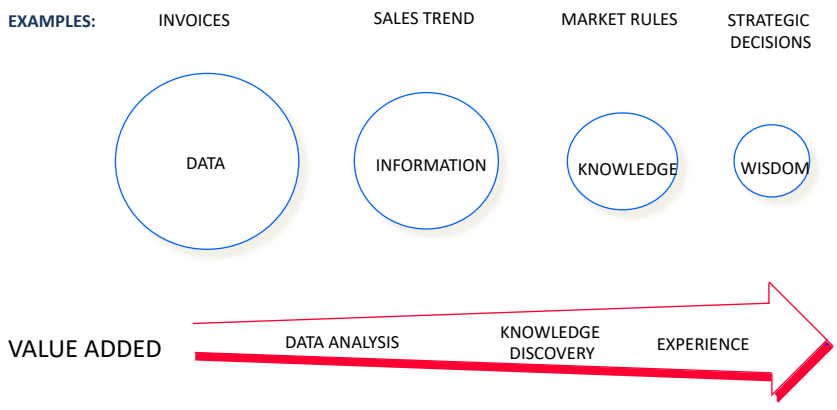
The new challenges in the Data Management Area (3)

Extraction of (synthetic and useful) knowledge:

- *Knowledge representation and reasoning*: using conceptual models and ontologies: formal specifications allowing offering a common vocabulary for automatic knowledge sharing; using reasoning services, which allow some forms of deduction and inference.
- *Personalization and context-awareness*: can eliminate "information noise" reducing the available data only to the part that is appropriate for the current user and context

15

Knowledge Distillation



(courtesy of prof. Fabio A. Schreiber)

16

An introduction to (Big) Data Integration

(as of Oct 5th 2020)

TECHNOLOGIES FOR INFORMATION SYSTEMS

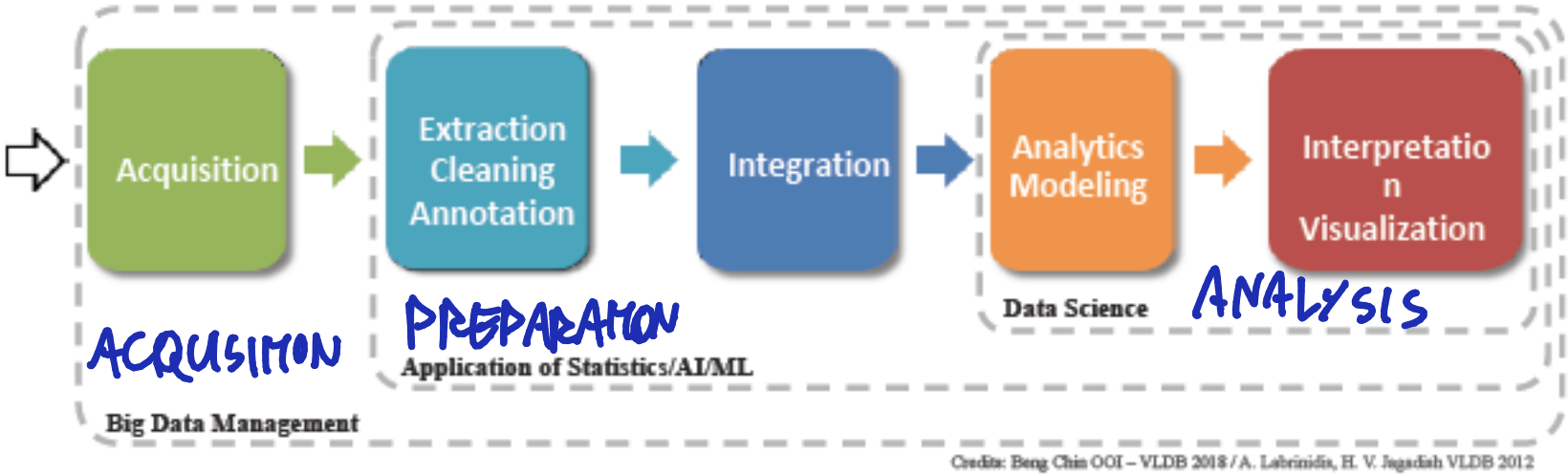
Letizia Tanca

Politecnico di Milano

Motivation:

The reality behind each data analysis task

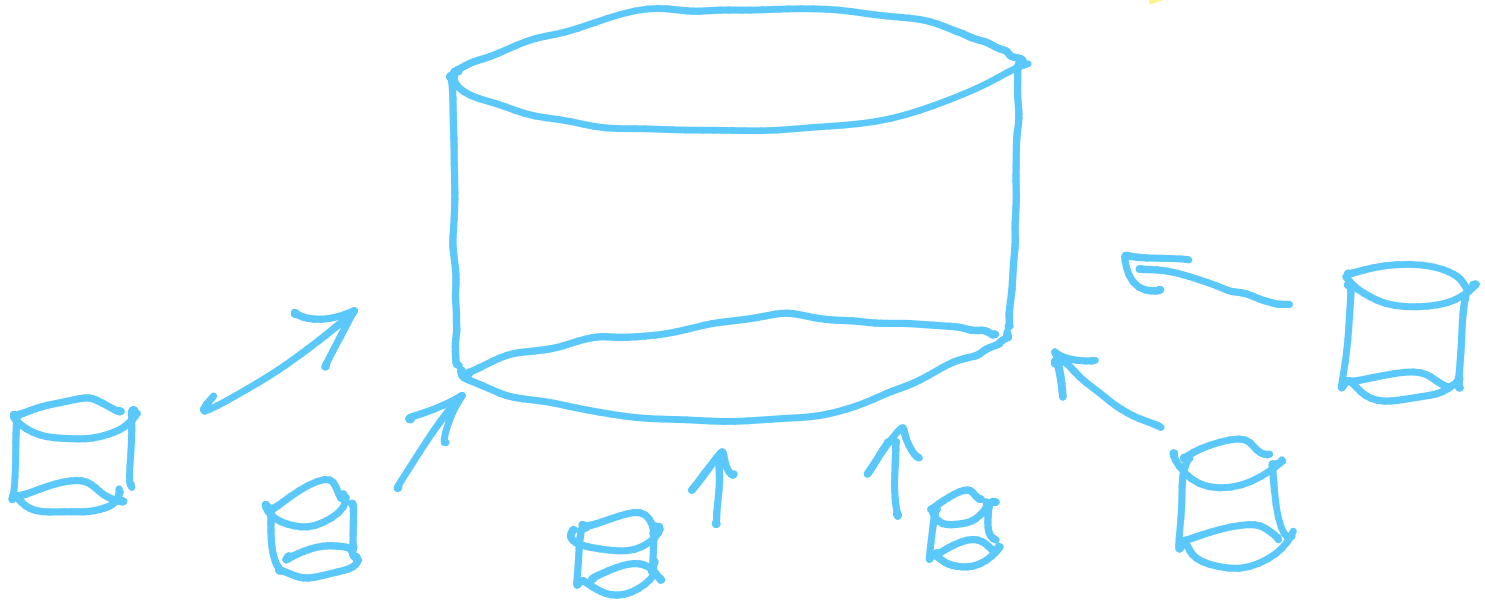
- The actual implementation of the Data Analysis (ML, statistics, Data Mining…) algorithm is usually less than 5% lines of code in a real, non-trivial application
- The main effort (i.e. those 95% LOC) is spent on:
  - Data cleaning & annotation
  - Data extraction, transformation, loading
  - Data integration & pruning
  - Parameters tuning
  - Model training & deployment
  - …



The Data Integration problem is:

Combining data coming from different data sources, providing the user with a unified vision of the data

→ Detecting correspondences between similar concepts that come from different sources, and conflict solving



The four V's of Big data in data integration

- **Volume**: Not only can each data source contain a *huge volume of data*, but also the number of data sources has grown to be in the *millions*.
- **Velocity**: As a direct consequence of the *rate* at which data is being collected and continuously made available, many of the data sources are *very dynamic*.
- **Variety**: Data sources (even in the same domain) are extremely *heterogeneous* both at the schema level, regarding how they structure their data, and at the instance level, regarding how they describe the same real world entity, exhibiting considerable variety even for substantially similar entities.
- **Veracity**: Data sources (even in the same domain) are of widely differing *qualities*, with significant differences in the coverage, accuracy and timeliness of data provided. This is consistent with the observation that "1 in 3 business leaders do not trust the information they use to make decisions."

(Xin Luna Dong, Divesh Srivastava, VLDB2013 tutorial)

Information Systems in the Era of Big data

The Variety dimension:

people and enterprises need to integrate data and the systems that handle those data: relational DBMSs and their extensions, legacy data and legacy DBMSs, sensors and user-generated content produce **heterogeneous**, structured or unstructured data

The Veracity dimension:

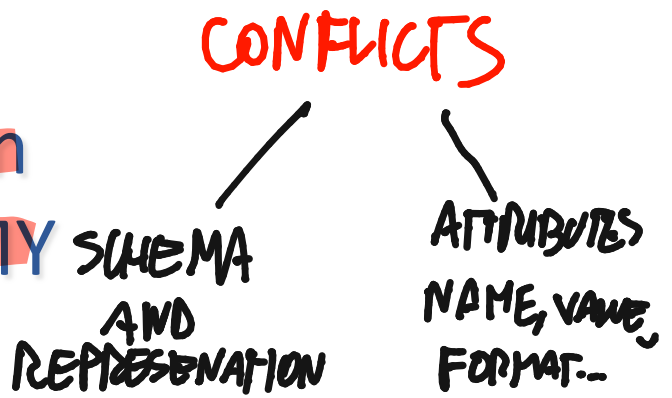
Data Quality is the most general and used term, and represents a number of quality aspects besides veracity:

- Completeness,
- Validity,
- Consistency,
- Timeliness
- Accuracy

Information overload

- The term was used by the “futurist” Alvin Toffler in his book *Future Shock*, already back in 1970
- It refers to the difficulty of *understanding and making decisions* when too much information is available
- This is the main challenge presented by “Big Data”

Heterogeneity derives from various forms of AUTONOMY



- Design (representation) autonomy,
- Communication (querying) autonomy, and
- Execution (algorithmic) autonomy

….. GENERATE HETEROGENEITY,

While there is a need for interoperability among SW applications, services and information (databases, and others) managed by different organizations that need to reuse legacy applications, existing data repositories (e.g. *deep web*), and reconcile the different points of view adopted by the various players using the information.

Data integration approaches interoperability

from a Data Perspective

Interoperability
as proposed by ISO (2002)

Interoperability is associated to (one of) the following tasks:

- Find information and processing tools, when they are needed, independently of physical location.
- Understand and employ the discovered information and tools, no matter what platform supports them, whether local or remote.
- Evolve a processing environment for commercial use without being constrained to a single vendor's offerings.
- Build upon the information and processing infrastructures of others in order to serve niche markets, without fear of being stranded when the supporting infrastructure matures and evolves.

Semantic, Syntactic, Technical, Organizational, Schematic or Structural, and Physical, are the main categories of

INTEROPERABILITY

Interesting in data management

(Big) Data Integration Problems

VARIETY (heterogeneity) among several data collections to be used together

1. Different platforms: Technological heterogeneity
2. Different data models at the participating DBMS → Model heterogeneity
3. Different query languages -> Language heterogeneity
4. Different data schemas and different conceptual representations in DBs previously developed → Schema (semantic) heterogeneity
5. Different values for the same info (due to errors or to different knowledge)→ instance (semantic) heterogeneity

VERACITY: (Main) Data Quality dimensions:

1. Completeness,
2. Validity,
3. Consistency,
4. Timeliness
5. Accuracy

The steps of Data Integration: how is it done?

Schema Reconciliation

Schema reconciliation: mapping the **data structure**

Record Linkage

Record linkage (aka Entity resolution): data matching based on **the same content**

Data Fusion

Data fusion: reconciliation of **non-identical content**

EXAMPLE

STUDENT

ID	NAME	SURNAME	EMAIL
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

S

ID	N	S	PHONE
⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮

① WHAT IS THE UNIFIED VIEW WE ARE LOOKING FOR?

1ST PROBLEM → TABLE 1 PRESENTS "EMAIL" BUT NOT "PHONE"
AND VICEVERSA FOR TABLE 2

POSSIBLE SOLUTION → CONSIDER BOTH OF THEM ACCEPTING POSSIBLE "NULL" VALUES

2ND PROBLEM → LOOK IF $NAME_1 = N_2$, $SURNAME_1 = N_2$

3RD PROBLEM → VERIFY IF ID_1 AND ID_2 HAVE THE SAME MEANING

② FOCUS ON VALUES

STUDENT

ID	NAME	SURNAME	EMAIL
123	PAOLO	ROSSI	PR@G.IT
111	LEONARDO	BIANCHI	LB@H.IT

SAME PERSON?

S

ID	N	S	PHONE
123	PAOLO	ROSSI	3758
76	LEONARDO	BIANCHI	7953

③ HOW TO SOLVE DATA FUSION?

STUDENT

ID	NAME	SURNAME	ADDRESS
123	PAOLO	ROSSI	MONTEMARCIANO

S

ID	N	S	A
123	PAOLO	ROSSI	MILANO

WHAT IF $ADDRESS_1 \neq A_2$ BUT THEY HAVE THE SAME MEANING?

ARCHITECTURES

THE GOAL OF THE FIRST STEP IS TO OBTAIN A BIG TABLE CALLED GLOBAL SCHEMA

Relevant Ways of Integrating Database Systems

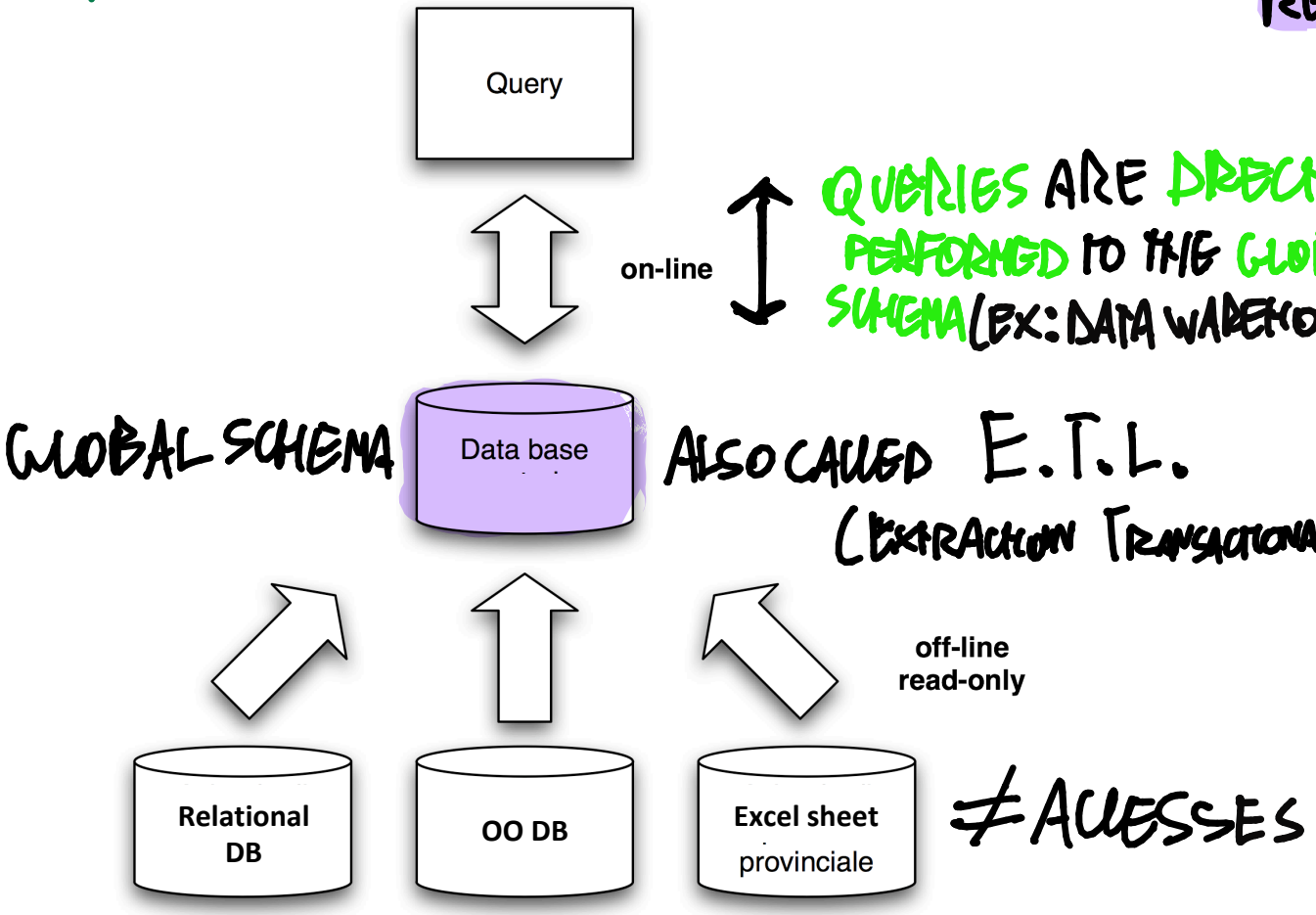
WE HA TWO OPTIONS FOR DIFFERENT ARCHITECTURAL APPROACHES:

1. Use a materialized data base (data are merged in a new database) → Extract-Transform-Load Systems
   → Data Warehouses: Materialized integrated data sources
2. Use a virtual non-materialized data base (data remain at sources) →
   - Enteprise Information Integration (EII) (or Data Integration) Systems (common front-end to the various datasources)
   - Data Exchange (source-to-target)

DOMANDA ESAME LUGLIO 17
GENNAIO 18
LUGLIO 18

Materialized

DATA ARE PHISICALLY STORE SOMEWHERE. THE MATCHED DATAS ARE COMBINED IN ONE SINGLE REPOSITORY



QUERIES ARE DIRECTLY PERFORMED TO THE GLOBAL SCHEMA (EX: DATA WAREHOUSE)

GLOBAL SCHEMA

ALSO CALLED E.T.L. (EXTRACTION TRANSACTIONAL LOADING)

≠ ACCESSES

CON: THE SYSTEM HAS TO BE PERIODICALLY UPDATE

PRO: FASTER RESPONSES

- FASTER
- CLOSED
- NO UPDATES

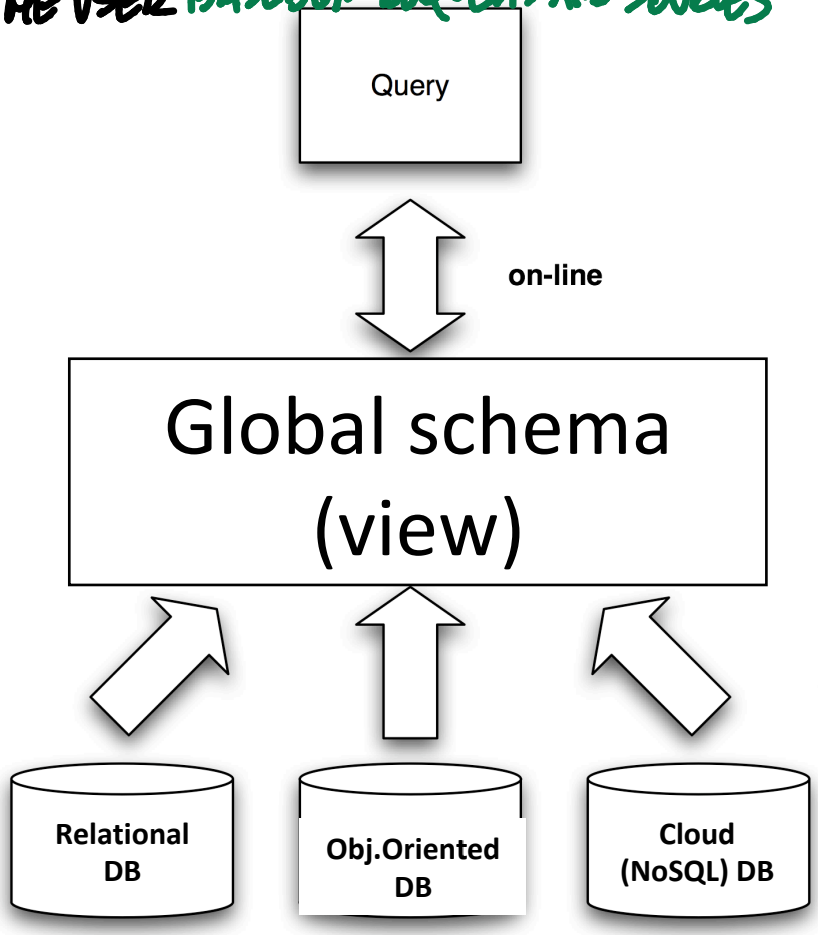
Materialized integration

- Large common stores came to be known as warehouses, and the software to access, scrape, transform, and load data into warehouses, became known as extract, transform, and load (ETL) systems.
- In a dynamic environment, one must perform ETL periodically (say once a day or once a week), thereby building up a history of the enterprise.
- The main purpose of a data warehouse is to allow systematic or ad-hoc data analysis and mining.
- Not appropriate when need to integrate the *operational* systems (keeping data up-to-date)

Virtual

DATA ARE NOT MATERIALIZED IN A PHYSICAL STORE

THE GLOBAL SCHEMA IS DEFINED VIRTUALLY SPLITTING QUERIES MADE BY THE USER BASED ON REQUESTS AND SOURCES

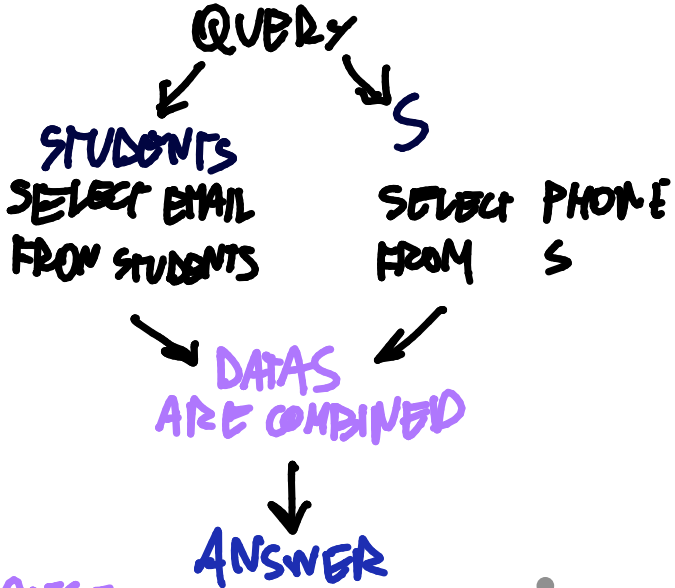


On-line

Given a target schema T, a set S of source schemas, a set M of mappings relating T to each element in S, and a query Q(T) against the target schema, use M to evaluate Q(T) over the set I(S) of source instances (i.e., over the data stored in the sources).

CONSIDERING OUR EXAMPLE (PAG. 19)

FIND BOTH EMAIL AND PHONE



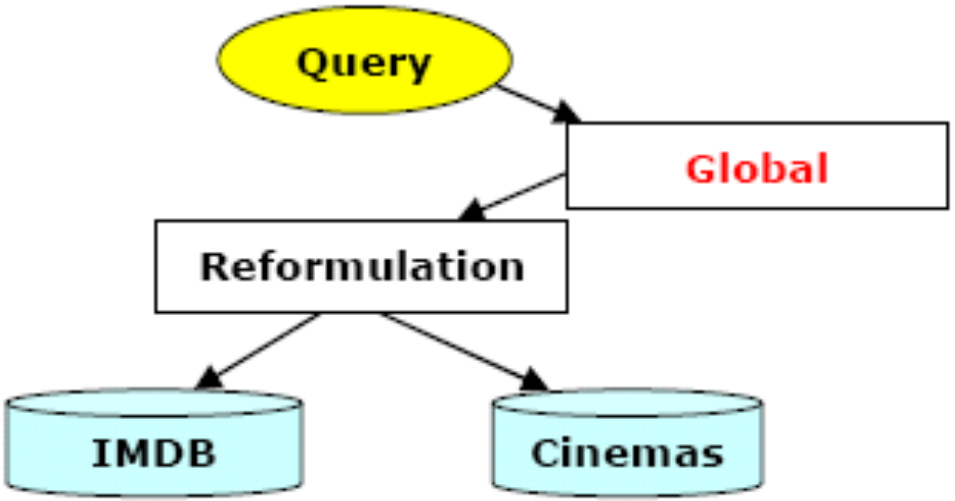
THE INTEGRATION IS FORMED WHEN THE USER SEND A REQUEST

THIS GUARANTEES A HIGH FLEXIBILITY

Virtual integration

The virtual integration approach **leaves the information requested in the local sources.** The virtual approach will always return *a fresh answer to the query*. The query posted to the global schema is reformulated into the formats of the local information system. The information retrieved needs to be combined to answer the query.

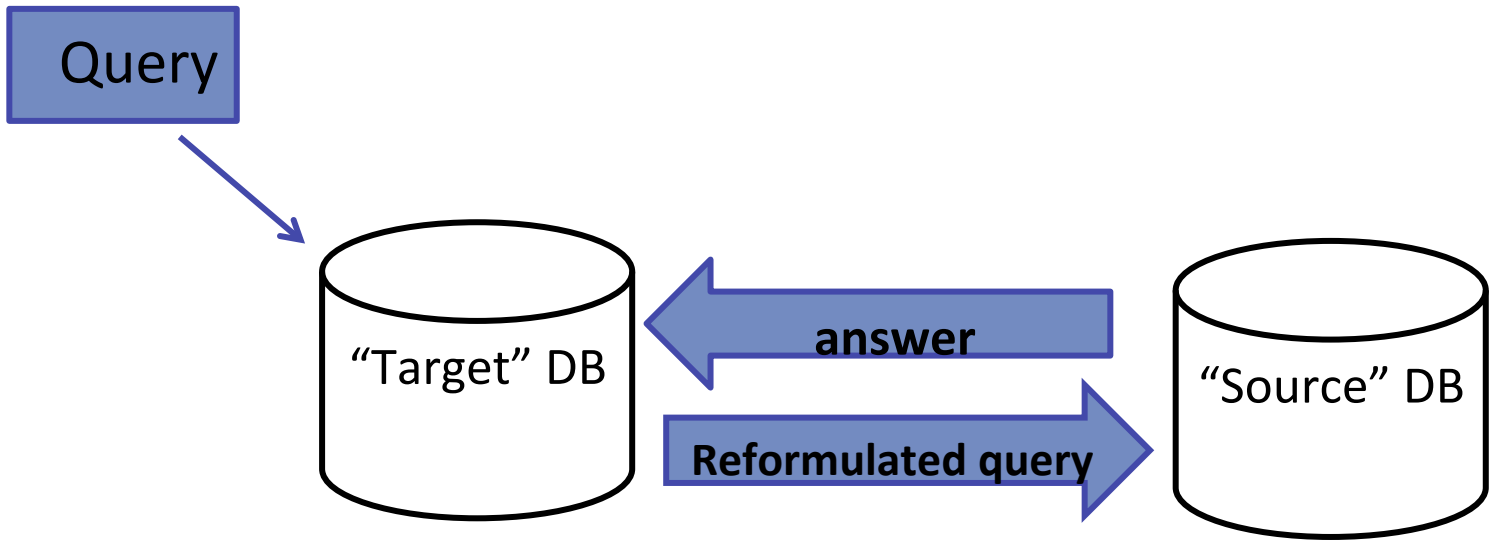
- FLEXIBLE
- UPDATED VALUES, SINCE THE G.S. IS BUILT WHEN THE QUERY IS PERFORMED
- SLOWER RESPONSES



Data Exchange:

virtual, between two sources e.g. IBM CLIO

Given a target schema T, a source schema S, a set of M mappings relating T and S, materialize the target instance I(T) using the source instance I(S) (i.e., the data stored currently stored in the source)



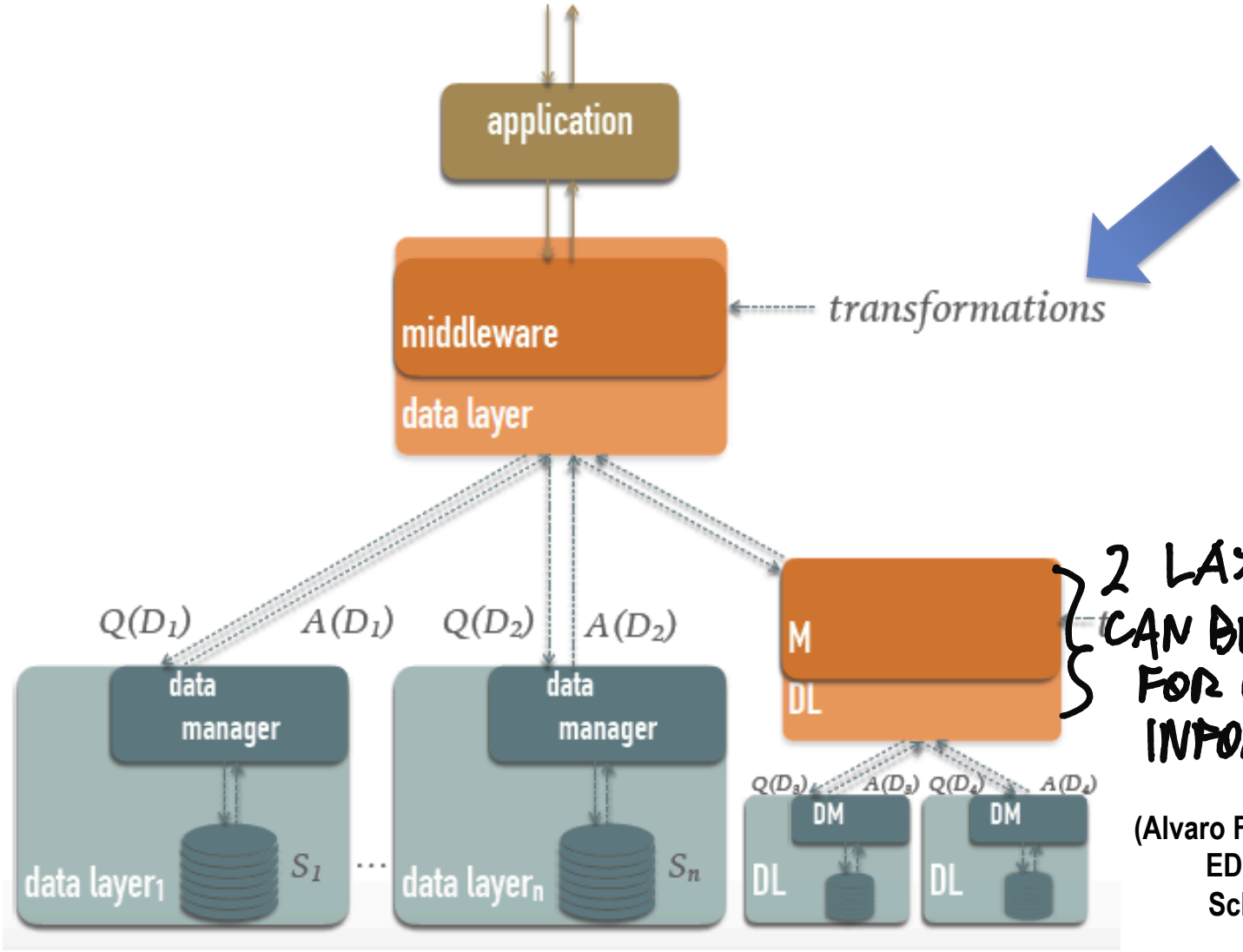
Rationale

- The conventional wisdom is to use data warehousing and ETL products to perform data integration. However, there is a serious flaw in one aspect of this wisdom.

- Suppose one wants to integrate current (operational) data rather than historical information. Consider, for example, an e-commerce web site which wishes to sell hotel rooms over the web. The actual inventory of available hotel rooms exists in 100 or so information systems since Hilton, Hyatt and Marriott all run their own reservation systems.

- Applying ETL and warehousing to this problem will create a copy of hotel availability data, which is quickly out of date. If a web site sells a hotel room, based on this data, it has no way of guaranteeing delivery of the room, because somebody else may have sold the room in the meantime.

A general framework for Data Integration



2 LAYERS THAT CAN BE USEFUL FOR COMPLEX INFORMATIONS

(Alvaro Fernandez, EDBT Summ. School 2017)

Query execution in the integrated database

When a DML statement, such as a query, is submitted, the integration system has to decompose it into queries against the two component databases.

1. determine first which tables are from which database,
2. which predicates apply to only a single database and
3. which predicates apply to tuples from both databases.

The latter ones can only be evaluated over the global database (view), whereas the former ones can be evaluated within the component databases.

The possible situations

- Data integration problems arise even in the simplest situation: unique, centralized DB…

…and it becomes more and more complex

…up to the extreme case of transient, dynamic, initially unknown data sources…

EVEN IN THE SAME COMPANY, EACH AREA OF IT WILL ASK A SPECIFIC PART OF THE DB.
IT IS EASY THAT DIFFERENT DATAS ARE COMMON WITH DIFFERENT PARTS => REDUNDANCE

UNIQUE DB

Each datum, whatever application uses it, must only appear once. This ELIMINATES useless REDUNDANCIES which would cause:

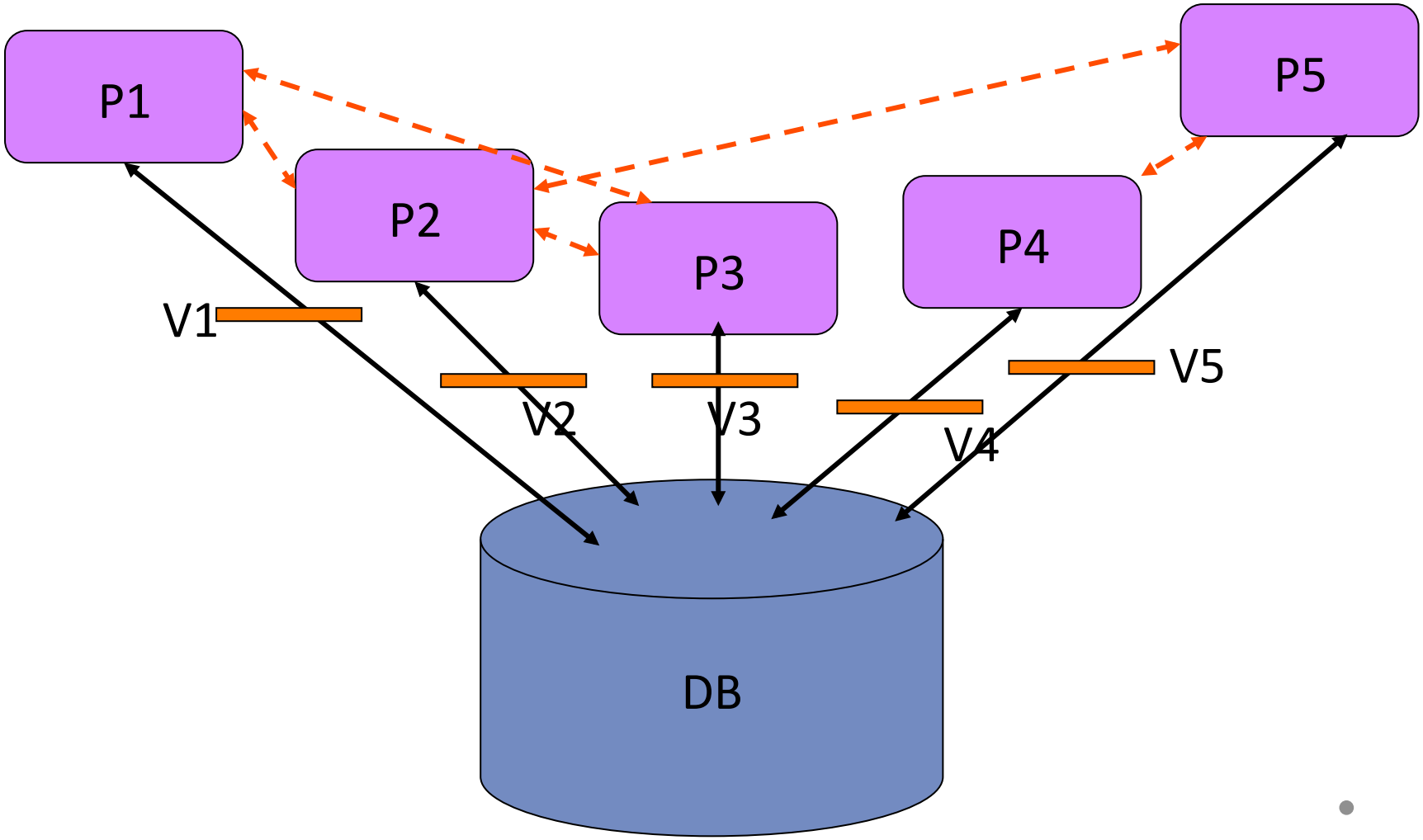
- Inconsistencies
- Useless memory occupation

Each functionality in the company will have its own personalized view

This is achieved by using view definitions

Views allow personalization as well as access control

Recall: an integrated centralized DB



The VIEW: a conceptual tool to express the transformations

VIEWS ARE USED TO SAVE EXECUTION OF QUERIES EVEN AFTER THE APPLICATION IS CLOSED AND THE RESULT IS STORE INTO THE SYSTEM.
THE GENERATED TABLE IS SAVED INTO THE DB

- Example:

  Create view *CourseCampus* **as**
  Select *CourseName, CampusName*
  From *Courses, Rooms*
  Where *Courses.RoomName=Rooms.RoomName*

  GLOBAL SCHEMAS ARE DEFINED WITH VIEWS

  ```
  COURSES(CourseName, Teacher, RoomName)
  ROOMS(RoomName, BuildingNo, CampusName)
  ```

- View schema:

  ```
  CourseCampus(CourseName, CampusName)
  ```

Non-centralized DB's:

Distributed DB

It is the simplest case of non-centralized DB:
NOT a matter of data integration

- Often, data for the same organization
- Integrated a-priori: same design pattern as in the centralized situation, indeed we have homogeneous technology, data model, thus the schema integration problems are as above
- At distribution design time, design decisions on:
  - Fragmentation:
    - Vertical
    - Horizontal
  - Allocation
  - Replication

The Data Integration problem is (recall):

Combining data coming from different data sources, providing the user with a unified vision of the data

→ Detecting correspondences between similar concepts that come from different sources, and conflict solving

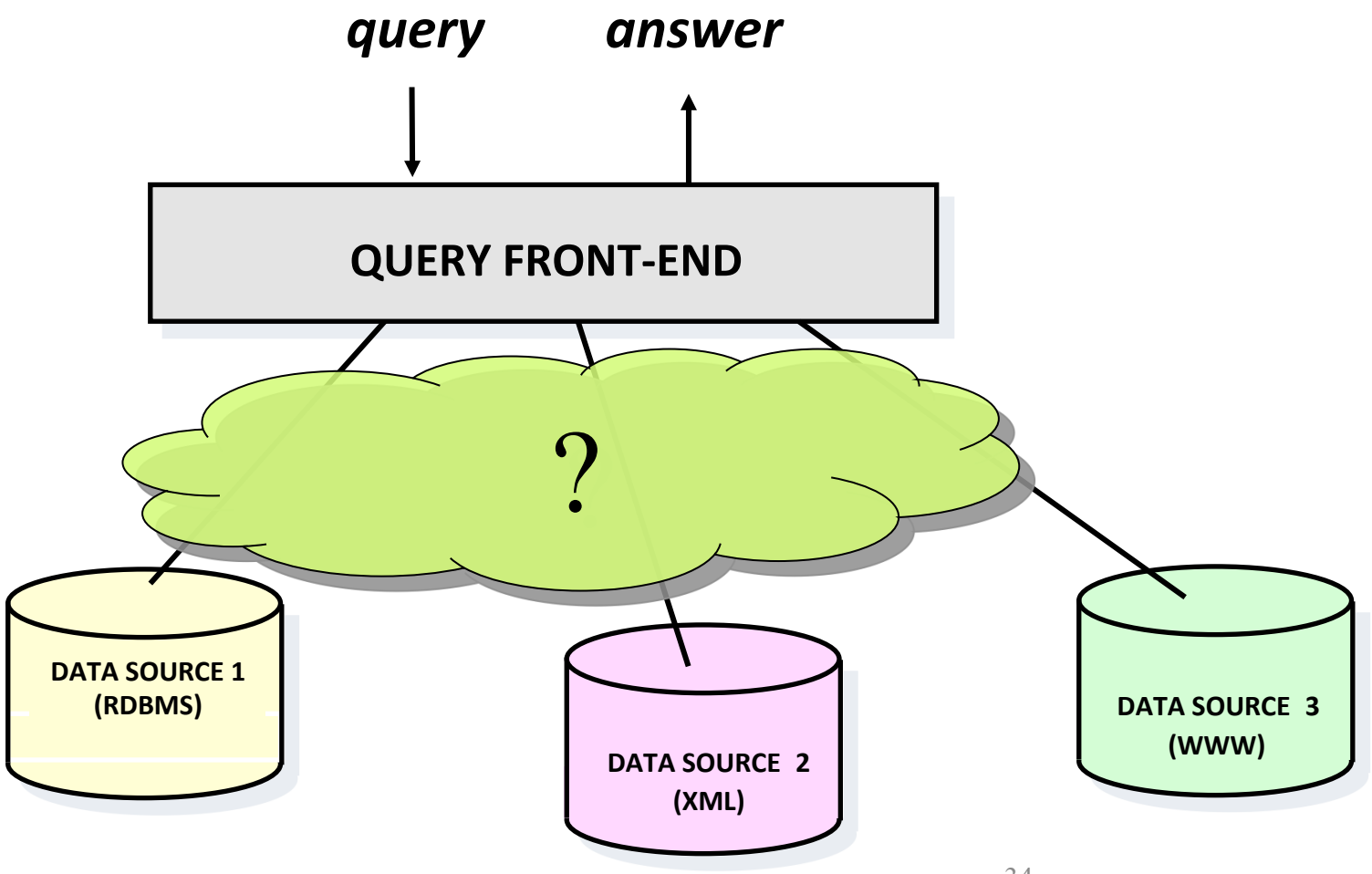
We study data integration for:

- Federated DB
  - Homogeneous data: ***same*** **data model**
  - Heterogeneous data: ***different*** **data models: OO, XML, relational, RDF, Web data etc. (Semi-structured data**), even **free text (unstructured data)**

- Transient, initially unknown data sources

We propose techniques and methods which should be applied, adding complication as the situation becomes more complex

Data Integration with various data sources



The steps of Data Integration: how is it done?

Schema Reconciliation

(If the sources have a schema) **Schema reconciliation**: mapping the **data structure** as in the centralized case

Record Linkage

Record linkage (aka Entity resolution): data matching based on **the same content**

Data Fusion

Data fusion: reconciliation of **non-identical content**

Approaches (recall)

- Data conversion (materialization): data are converted and stored into a unique system
  - Multiple copies: untimeliness, redundancy, inconsistency
  - Application rewriting if one system is discarded
- Data exchange (virtual): creation of gateways between system pairs
  - Only appropriate when only two systems, no support for queries to data coming from multiple systems (e.g. peer-to-peer)
  - Number of gateways increases rapidly
- ✓ Multidatabase (virtual): creation of a global schema

Data integration in the Multidatabase

We must build a system that:

- Supports access to different data sources
- "knows" the contents of these data sources
- Integrates the different data sources by means of a unifying, global schema
- Receives queries expressed in the language of the global schema
- Distributes "rewritten" queries to the sources
- Combines the answers received from the sources to build the final answer

Many heterogeneities (recall):

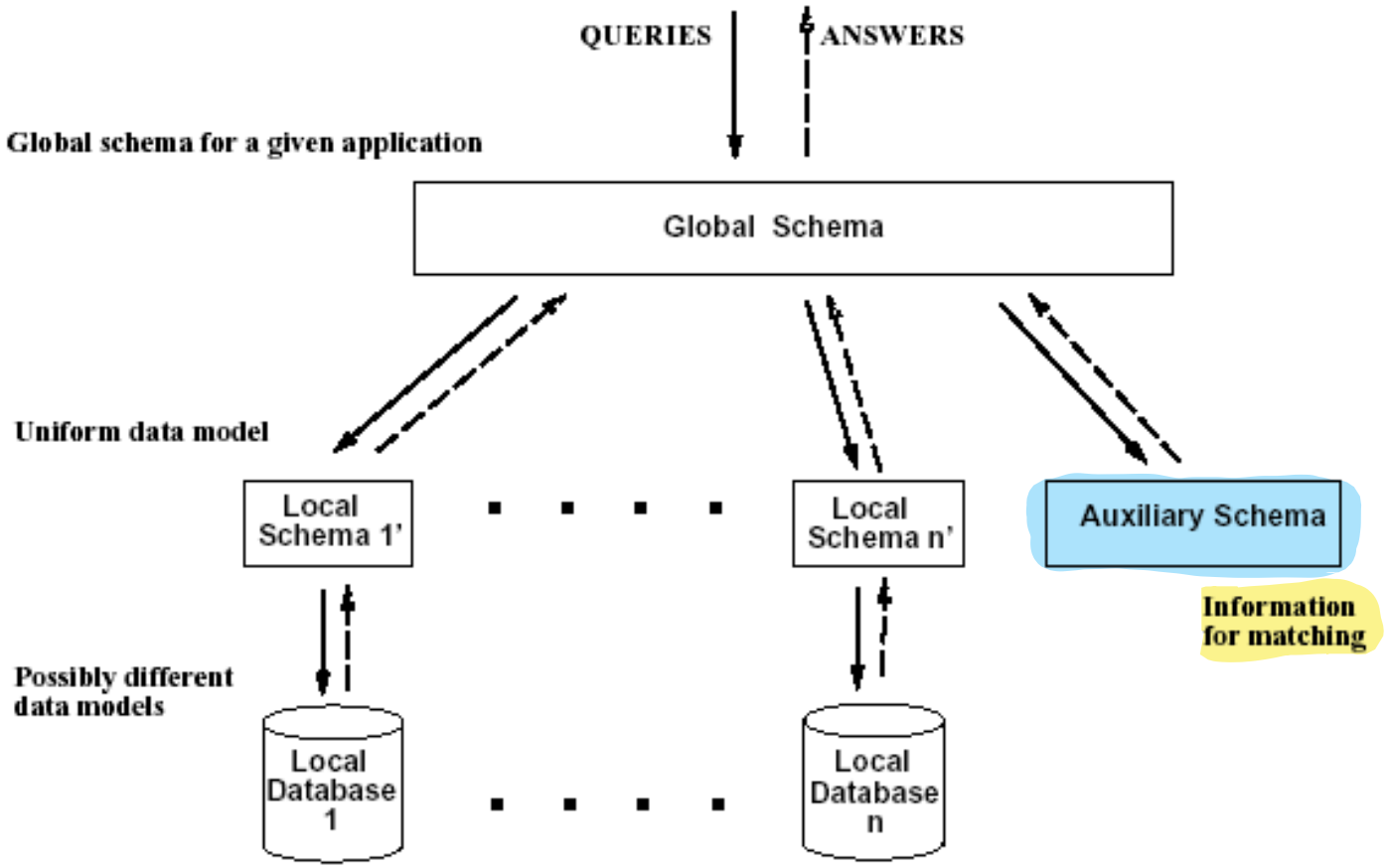
- Same data model, different systems e.g. relational (Oracle, Sybase, DB2…) → *technological heterogeneity*
- Different data models, e.g. hierarchical or network (IMS, Codasyl…), relational, OO → *model heterogeneity*
- Same data model, different query languages (SQL, Quel) → *language heterogeneity*
- Semi- or unstructured data (HTML, XML, multimedia…) → *again model heterogeneity*

A first case

- The data sources have the same data model
- Adoption of a *global schema*
- The global schema will provide a
  - Reconciled
  - Integrated
  - Virtual

  view of the data sources

Architecture with a uniform data model



REVIEW OF DB'S REALIZATION

- CONCEPTUAL DESIGN — USUALLY THE E.R. DIAGRAM
  INDIPENDENT FROM THE LOGICAL MODEL;
  WE CAN TRANSLATE IT IN DIFFERENT MODELS
  (WE WILL GENERALLY USE THE RELATIONAL MODEL)
- LOGICAL DESIGN — TABLES
- PHYSICAL DESIGN — CREATION

Designing data integration in the Multidatabase (with a unique data model)

1. Source schema identification (when present)
2. Source schema reverse engineering (data source conceptual schemata)
3. Conceptual schemata integration and restructuring: conflict resolution and restructuring
4. Conceptual to logical translation (of the obtained global schema)
5. Mapping between the global logical schema and the single schemata (logical view definition)
6. After integration: query-answering through data views

Example

PoliRobots:

REPORT (ID, dateTime, #ofFaults)

REPORTFAULT (reportID, faultID)

FAULT (ID, description, solution, responsibleOperatorID)

UniRobots:

MESSAGE (robotID, dateTime, errorCode) *//robotIDs are R001, R002, ..., R100*

ERROR (code, description, solution, personInCharge, urgency)

1) WE HAVE THE SOURCES ✓

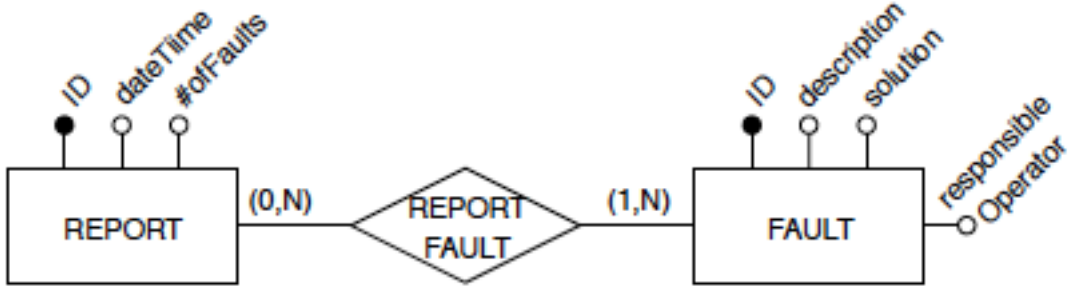
② REVERSE ENGINEERING Example

PoliRobots:

REPORT (ID, dateTime, #ofFaults)

REPORTFAULT (reportID, faultID)

FAULT (ID, description, solution, responsibleOperatorID)

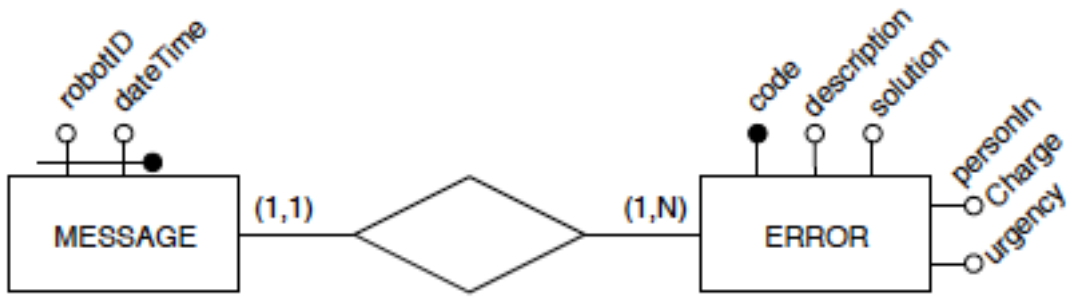


Example

UniRobots:

MESSAGE (robotID, dateTime, errorCode) *//robotIDs are R001, R002, ..., R100*

ERROR (code, description, solution, personInCharge, urgency)



3. Conflict resolution and restructuring

a. Related concept identification
b. Conflict analysis and resolution
c. Conceptual Schema integration

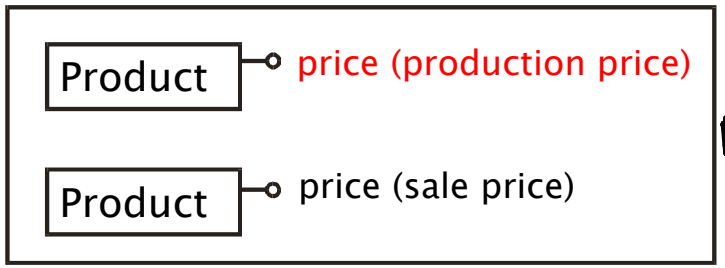
Related Concepts' identification

- Ex:
  - employee, clerk
  - exam, course
  - code, num
- Not too difficult if manual
- Very difficult if automatic – this is the extreme case

Conflict analysis

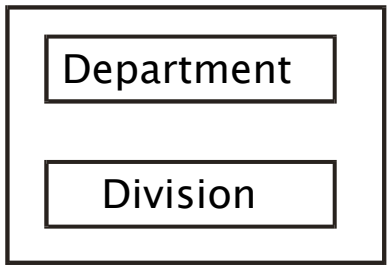
NAME CONFLICTS

- HOMONYMS



SAME NAME
DIFFERENT MEANING

- SYNONIMS



DIFFERENT NAME
SAME MEANING

Conflict analysis

TYPE CONFLICTS

- in a single attribute **(e.g. NUMERIC, ALPHANUMERIC, …)**

  e.g. the attribute "gender":
  - **Male/Female**
  - **M/F**
  - **0/1**
  - **In Italy, it is implicit in the "codice fiscale" (SSN)**

- in an entity type

  different abstractions of the same real world concept produce different sets of properties (attributes)

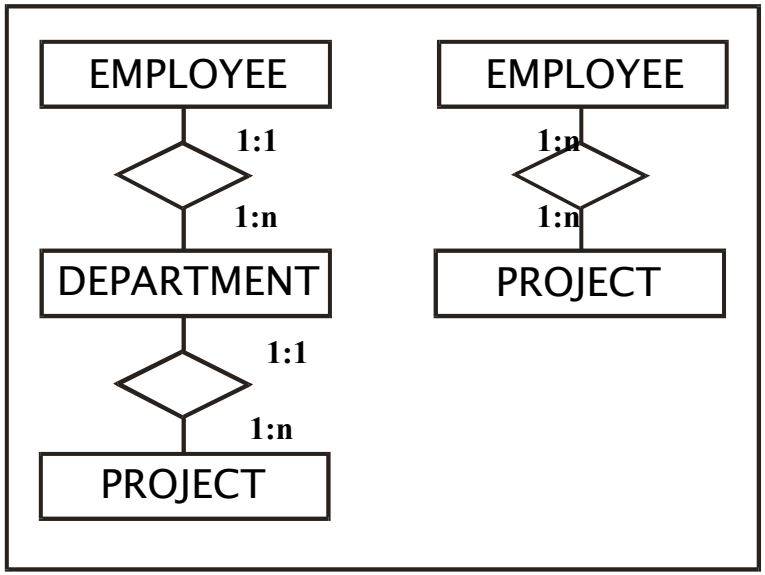
Conflict analysis

DATA SEMANTICS

- different currencies (euros, US dollars, etc.)
- different measure systems (kilos vs pounds, centigrades vs. Farhenheit.)
- different granularities (grams, kilos, etc.)

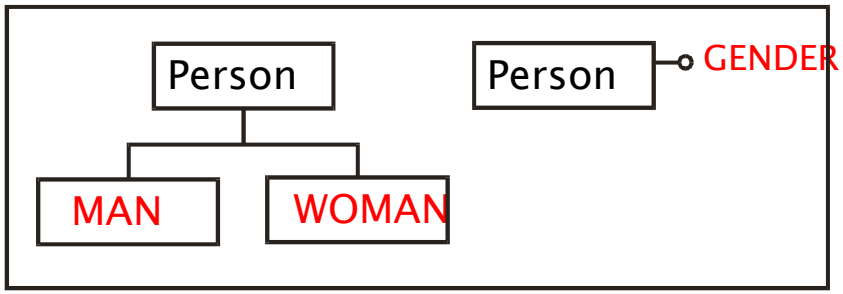
Conflict analysis

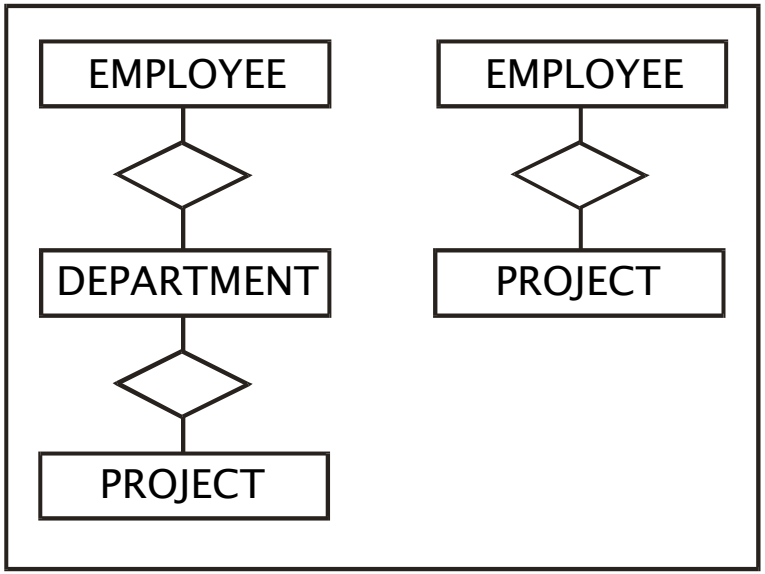
- DEPENDENCY (OR CARDINALITY) CONFLICTS

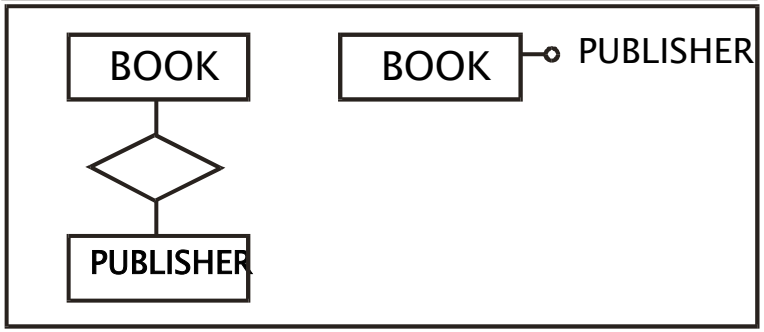


Conflict analysis

STRUCTURE CONFLICTS

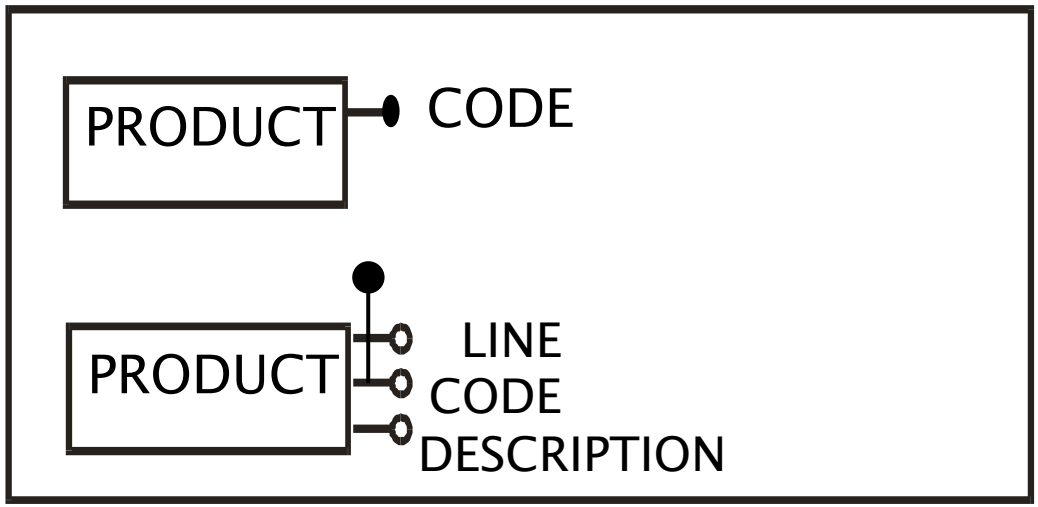




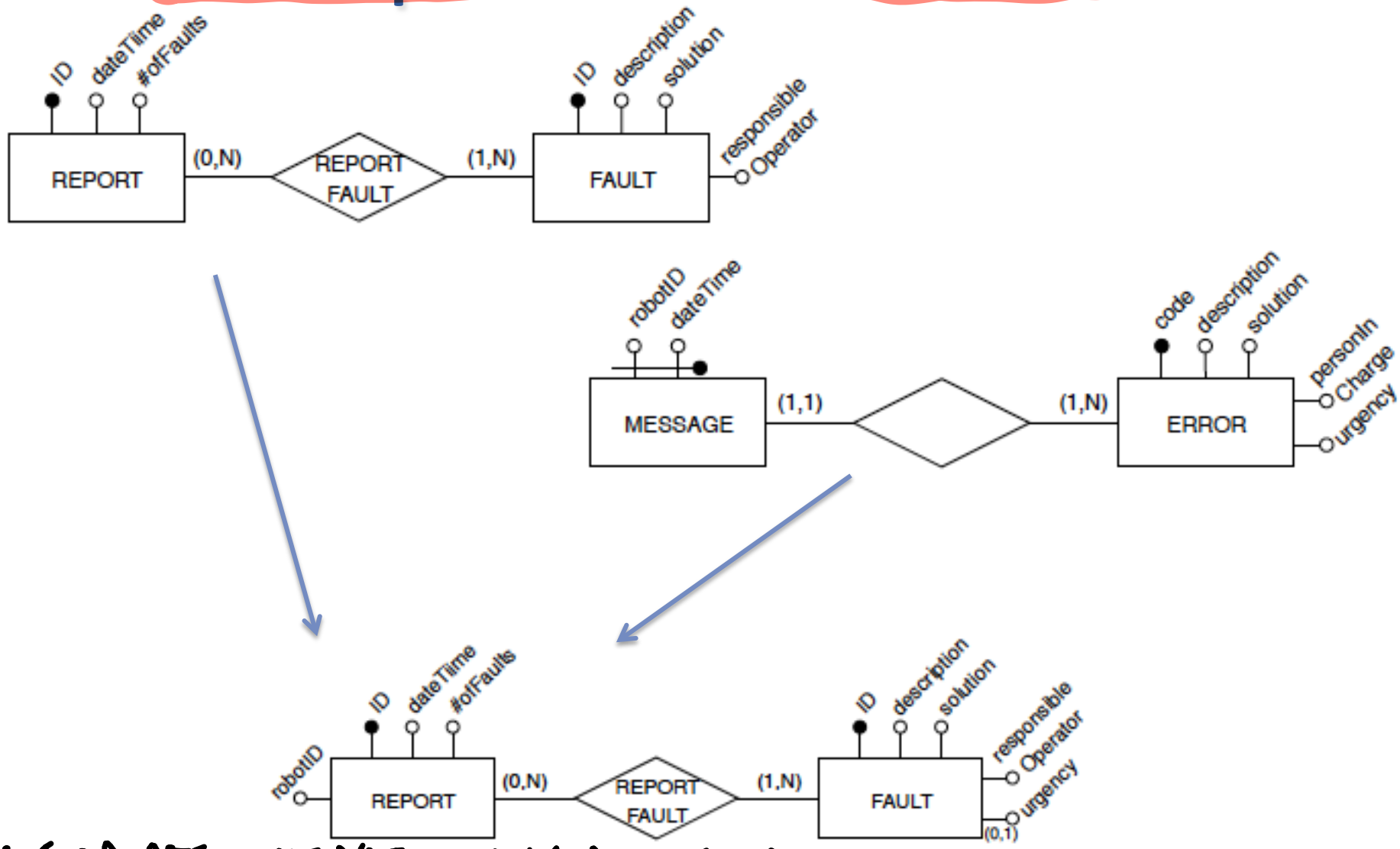


Conflict analysis

- KEY CONFLICTS



Example: UniPoliRobots



NOW WE CREATE SOME VIEWS WHERE WE CAN SOLVE THIS CONFLICTS

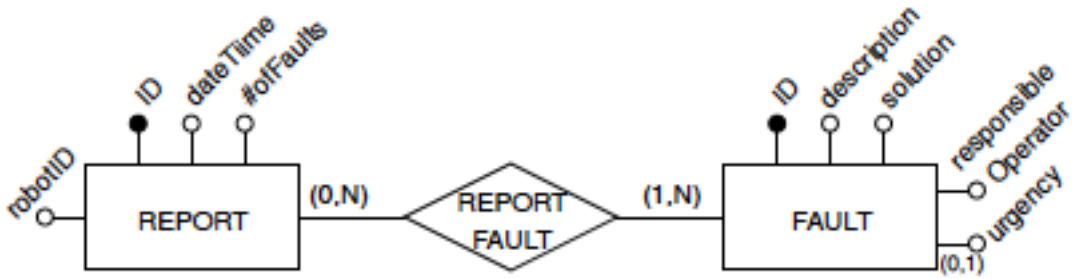
Designing data integration in the Multidatabase (with a unique data model)

1. Source schema identification (when present)
2. Source schema reverse engineering (data source conceptual schemata)
3. Conceptual schemata integration and restructuring: conflict resolution and restructuring

NOW:

1. Conceptual to logical translation (of the obtained global schema)
5. Mapping between the global logical schema and the single schemata (logical view definition)
6. After integration: query-answering through data views

UniPoliRobots



REPORT(ID, RobotID, dateTime, #ofFaults)

FAULT(ID, description, solution, resp_operator, urgency*)

Report_Fault(RID, FID)

```
CREATE VIEW UniPoliRobots.Report (ID, RobotID, dateTime, #ofFaults) AS
(
    SELECT KeyGenReport(ID, 'PoliRobots'), 'R101', dateTime, #of-
    Faults
    FROM PoliRobots.Report

    UNION

    SELECT KeyGenReport(robotID || dateTime, 'UniRobots'), robotID,
    dateTime, 1
    FROM UniRobots.Message
)
```

Mapping between the global logical (mediated) schema and the single source schemata.

In general:

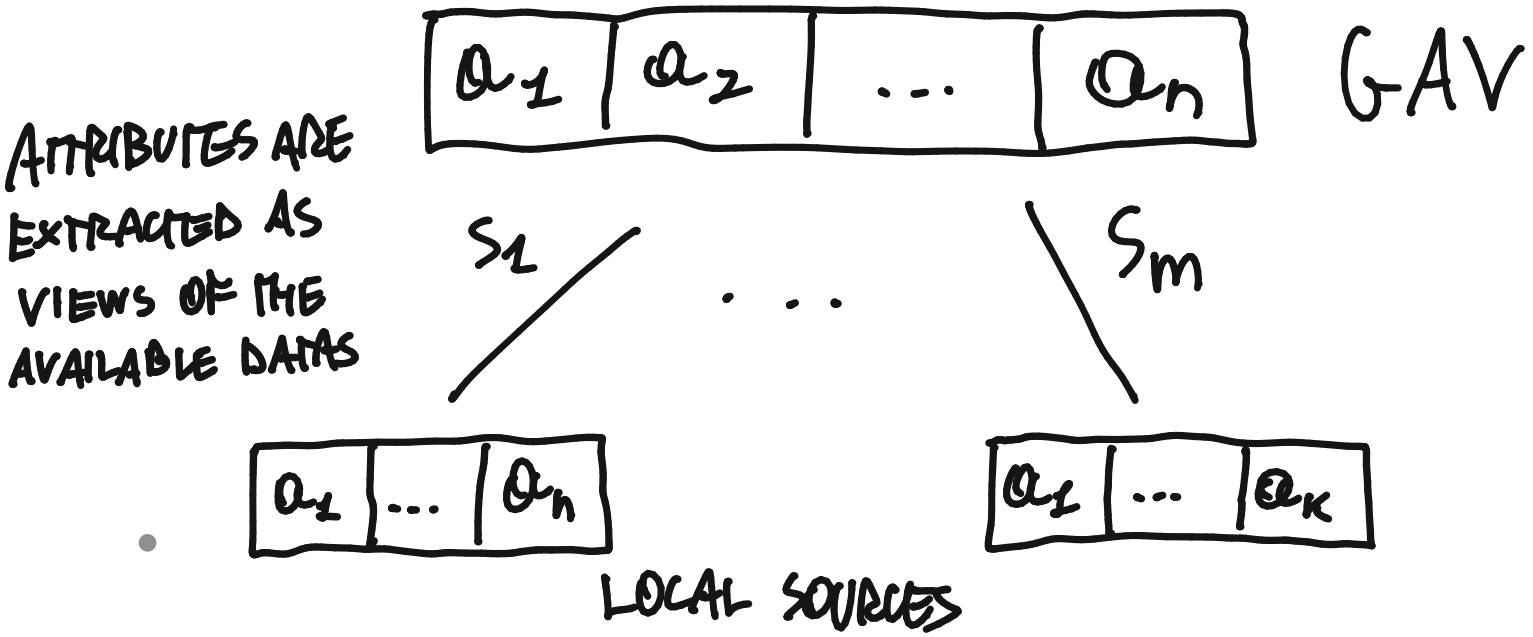
- A data integration system is a triple (G, S, M)
- The query to the integrated system are posed in terms of G and specify which data of the virtual database we are interested in
- The problem is understanding which real data (in the data sources) correspond to those virtual data

Mapping between the global logical (mediated) schema and the single source schemata (logical view definition)

- Two basic approaches
  - **GAV (Global As View)**
  - **LAV (Local As View)**
- The same approach can be used also in case of different data models
- In that case also a **model transformation** is required (we'll see it later)

GAV (Global As View)

- Up to now we supposed that the global schema *be derived* from the integration process of the data source schemata
- Thus the global schema is *expressed in terms of the data source schemata*
- Such approach is called the *Global As View approach*



The other possible ways

LAV (Local As View)

- The global schema has been designed independently of the data source schemata
- The relationship (mapping) between sources and global schema is obtained by defining each data source as a view over the global schema

GLAV (Global and Local As View)

- The relationship (mapping) between sources and global schema is obtained by defining a set of views, some over the global schema and some over the data sources

GAV

- A GAV mapping is a set of assertions, one for each element *g* of ***G***

$$g \rightarrow q_S$$

That is, the mapping specifies g as a query q_S over the data sources. This means that the mapping (view) tells us exactly how the element g is computed.

- OK for stable data sources
- Difficult to extend with a new data source

GAV example

SOURCE 1

Product(Code, Name, Description, Warnings, Notes, CatID)
Category(ID, Name, Description)
Version(ProductCode, VersionCode, Size, Color, Name, Description, Stock, Price)

SOURCE 2

Product(Code, Name, Size, Color, Description, Type, Price, Q.ty)
TYpe(TypeCode, Name, Description)

note: here we do not care about data types…

SOURCE 1

Product(Code, Name, Description, Warnings, Notes, CatID)

Version(ProductCode, VersionCode, Size, Color, Name, Description, Stock, Price)

SOURCE 2

Product(Code, Name, Size, Color, Description, Type, Price, Q.ty)

GLOBAL SCHEMA

```
CREATE VIEW GLOB-PROD AS
SELECT Code AS PCode, VersionCode
    as VCode, Version.Name AS Name,
    Size, Color, Version.Description as
    Description, CatID, Version.Price,
    Stock
FROM SOURCE1.Product,
    SOURCE1.Version
WHERE Code = ProductCode
UNION
SELECT Code AS PCode, null as
    VCode, Name, Size, Color,
    Description,Type as CatID, Price,
    Q.ty AS Stock
FROM SOURCE2.Product
```

Query processing in GAV

QUERY OVER THE GLOBAL SCHEMA

```
SELECT PCode, VCode, Price, Stock
FROM GLOB-PROD
WHERE Size = "V" AND Color = "Red"
```

In this particular case the transformation is easy, since the combination operator is a UNION → push selections through union!!

```
SELECT Code, VersionCode, Version.Price, Stock
FROM SOURCE1.Product, SOURCE1.Version
WHERE Code = ProductCode AND Size = "V" AND Color = "Red"
UNION
SELECT Code, null, Price, Q.ty
FROM SOURCE2.Product
WHERE Size = "V" AND Color = "Red"
```

Further considerations on GAV

- Suppose we introduce a new source
- The simple views we have just created must be modified
- In the simplest case (like in the GLOB_PROD case) we only need to add a union with a new SELECT-FROM-WHERE clause
- This is not true in general, view definitions may be much more complex
- In LAV this problem does not occur

LAV (Local As View)

A mapping LAV is a set of assertions, one for each element *s* of each source S

$$s \rightarrow q_G$$

Thus the content of each source is characterized in terms of a view q_G over the global schema

- OK if the global schema is stable, e.g. based on a domain ontology or an enterprise model
- It favours extensibility
- Query processing much more complex

SOURCE 1

Product(Code, Name, Description, Warnings, Notes, CatID)

Version(ProductCode, VersionCode, Size, Color, Name, Description, Stock, Price)

SOURCE 2

Product(Code, Name, Size, Color, Description, Type, Price, Q.ty)

GLOBAL SCHEMA

GLOB-PROD (PCode, VCode, Name, Size, Color, Description, CatID, Price, Stock)

In this case we have to express the sources as views over the global schema

GLOBAL SCHEMA

GLOB-PROD (PCode, VCode, Name, Size, Color, Description, CatID, Price, Stock)

SOURCE 2

Product (Code, Name, Size, Color, Description, Type, Price, Q.ty)

CREATE VIEW SOURCE2.Product AS
SELECT PCode AS Code, Name, Size, Color, Description, CatID as Type, Price, Stock AS Q.ty
FROM GLOB-PROD

GLOBAL SCHEMA

GLOB-PROD (PCode, VCode, Name, Size, Color, Description, CatID, Price, Stock)

SOURCE 1

Product(Code, Name, Description, Warnings, Notes, CatID)

Version(ProductCode, VersionCode, Size, Color, Name, Description, Stock, Price)

```
CREATE VIEW SOURCE1.Product AS
SELECT Pcode AS Code,
Name?,Description?,
    Warnings ?, Notes?, CatID
FROM GLOB-PROD

CREATE VIEW SOURCE1. Version AS
SELECT Pcode AS ProductCode,
    VCode as VersionCode, Size, Color,
    Name,
Description , Stock, Price)
FROM GLOB-PROD
```

Notes

- Some information is lacking: there is no corresponding value (e.g.Warnings, Notes, etc..).

  → A difficult job

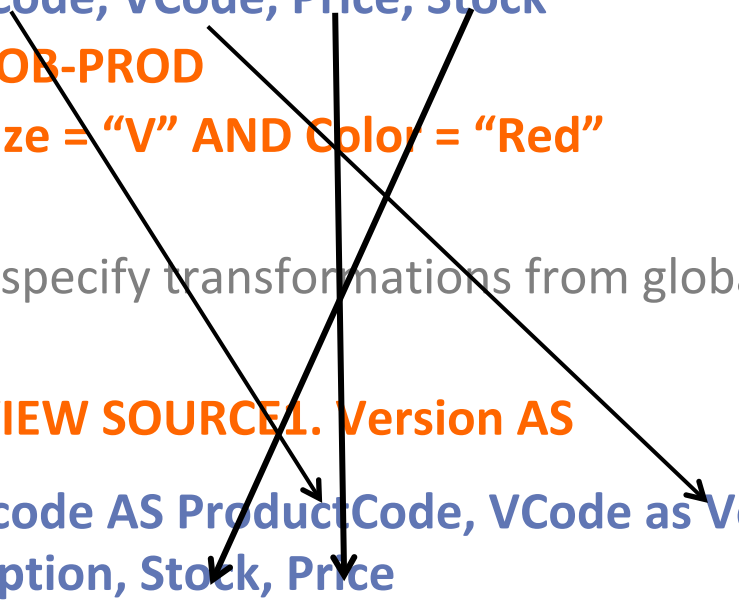
- On the other hand, no query coming from the global schema requires those attributes, since they are not present in the global schema!!!

ALL QUERIES ARE DIRECTED TO THE GLOBAL SCHEMA

Query processing in LAV

- Queries are expressed in terms of the global schema: **REASONING** needed

SELECT PCode, VCode, Price, Stock
FROM GLOB-PROD
WHERE Size = "V" AND Color = "Red"



- Views specify transformations from global to sources, thus:

CREATE VIEW SOURCE1. Version AS

SELECT Pcode AS ProductCode, VCode as VersionCode, Size, Color, Name, Description, Stock, Price
FROM GLOB-PROD

- Do the same for **SOURCE 2.Product**

Query processing in LAV

- Information is in the sources, while the mapping, i.e. the view definition, specifies the opposite transformation w.r.t. the one we need!
- Not possible to obtain answers by a simple, obvious or direct computation of the query definition
- No simple rule or view unfolding for the relations in the body as in GAV
- A *plan* is a rewriting of the query as a set of queries to the sources and a prescription of how to combine their answers
- Both sources are implicitly needed: the system must extract the necessary information from SOURCE1.VERSION and from SOURCE2.PRODUCTS

TO SUM UP

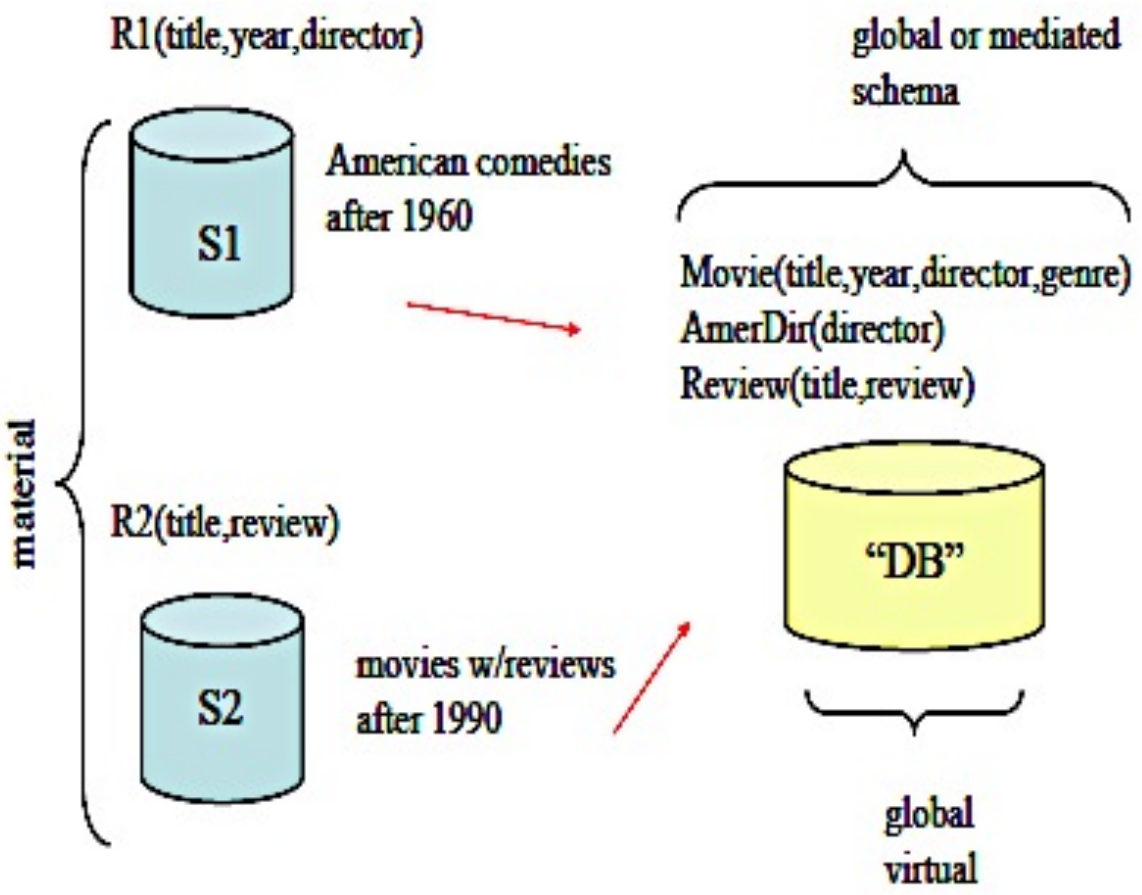
GAV

- Mapping quality depends on how well we have compiled the sources into the global schema through the mapping
- Whenever a source changes or a new one is added, the global schema needs to be reconsidered

LAV

- Mapping quality depends on how well we have characterized the sources
- High modularity and extensibility (if the global schema is well designed, when a source changes or is added, only its definition is to be updated)
- Query processing needs *reasoning*

Example (Bertossi)



Example (Bertossi)

Global schema G:

Movie(Title, Year , Director , Genre), AmerDir (Director),

Review(Title, Rev)

With LAV, the sources S1, S2 have their local relations defined as views.

S1: has V1, containing comedies, filmed after 1960,with American directors and their years:

V1(Title, Year , Director) ← Movie(Title, Year , Director ,Genre),
AmerDir (Director), Genre = comedy,Year ≥ 1960.

S2: has V2, containing movies filmed after 1990 with their reviews, but no directors

V2(Title, Rev) ← Movie(Title, Year, Director, Genre),
Review(Title, Rev), Year ≥ 1990.

Query processing in LAV

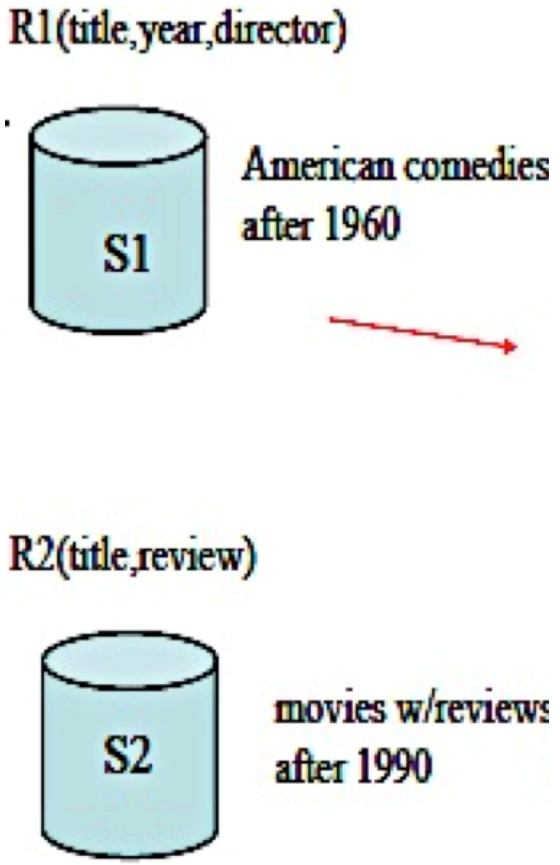
Query over **G**: Comedies with their reviews filmed since 1950 ?

Ans(Title, Rev) ← Movie(Title, Year , Director," comedy"),
Review(Title, Rev), Year ≥ 1950

• The query is rewritten in terms of the views and can be computed:

1. Extract values for Title from V1 (it contains *the comedies*)
2. Extract the tuples from V2 (it contains *the reviews*)
3. At the global level, compute the join via Title

Due to the limited contents of the sources, we only obtain comedies by American directors filmed after 1990 with their reviews



The most useful (and used) integration operators to write relational GAV views

- Union
- Outerunion
- Outerjoin
- Generalization

OPERATORS: EXAMPLES

R1

SSN	NAME	AGE	SALARY
123456789	JOHN	34	30K
234567891	KETTY	27	25K
345678912	WANG	39	32K

SSN	NAME	SALARY	PHONE
234567891	KETTY	20K	1234567
345678912	WANG	22K	2345678
456789123	MARY	34K	3456789

R2

R1 OU R2

SSN	NAME	AGE	SALARY	PHONE
123456789	JOHN	34	30K	NULL
234567891	KETTY	27	25K	NULL
345678912	WANG	39	32K	NULL
234567891	KETTY	NULL	20K	1234567
345678912	WANG	NULL	22K	2345678
456789123	MARY	NULL	34K	3456789

R1 JOIN R2

SSN,NAME

SSN	NAME	AGE	SALARY1	SALARY2	PHONE
234567891	KETTY	27	25K	20K	1234567
345678912	WANG	39	39K	22K	2345678

OPERATORS: EXAMPLES (2)

R1

SSN	NAME	AGE	SALARY
123456789	JOHN	34	30K
234567891	KETTY	27	25K
345678912	WANG	39	32K

R2

SSN	NAME	SALARY	PHONE
234567891	KETTY	20K	1234567
345678912	WANG	22K	2345678
456789123	MARY	34K	3456789

R1 OJ R2

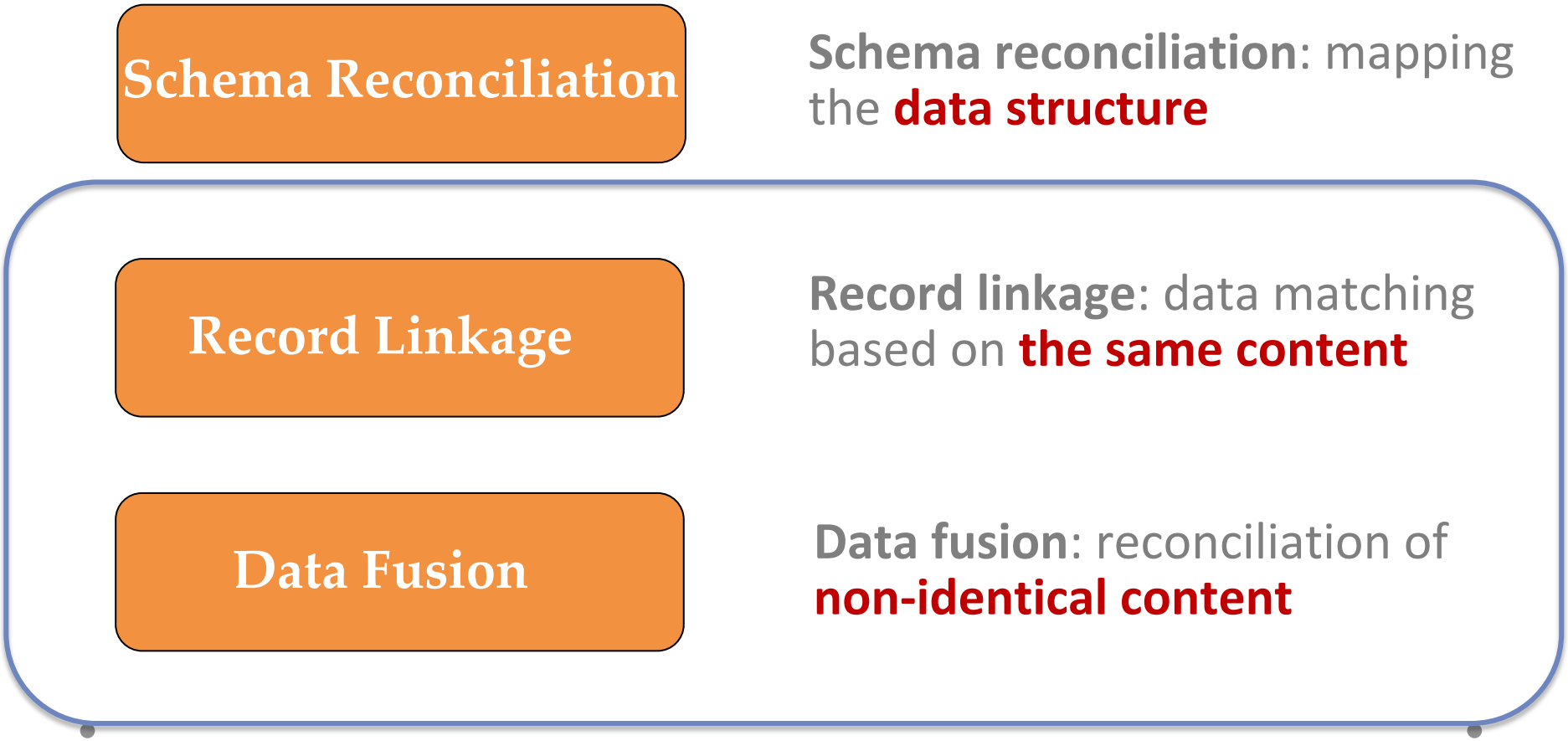
SSN	NAME	AGE	SALARY	PHONE
123456789	JOHN	34	30K	NULL
234567891	KETTY	27	??	1234567
345678912	WANG	39	??	2345678
456789123	MARY	NULL	34K	3456789

R1 Ge R2

SSN	NAME	SALARY
234567891	KETTY	??
345678912	WANG	??
123456789	JOHN	30K
456789123	MARY	34K

What shall we do about these "uncertain" values ?

The steps of Data Integration: how is it done?



Be there a schema or not, we may have inconsistencies in the data

At query processing time, when a real world object is represented by instances in different databases, they may have different values.

- Record Linkage (aka Entity Resolution): finding the info that refer to same real-world entities.
- Data Fusion: once recognized that two items refer to the same entity, how do we reconcile inconsistent information?

We have understood already that data integration often needs finding strings that refer to the same real-world entities:

- "Politecnico di Milano" and "Politecnico Milano"
- "Via ponzio 34" and "Via Ponzio 34/5"

Record Linkage
(aka Entity Resolution)

- Whatever the data model, we have to recognize when two datasets contain the same information
- We refer again to the relational data model because it is the most common case but also because the problem is pretty much the same
- Consider the case of relational databases. Two tuples could be compared as if they were two strings of characters:

Tanca	Letizia	DEIB	Leonardo

Tanca Letizia DEIB Leonardo

String matching vs. tuple or (more generally) structured-data matching

Tanca	Letizia	DEIB	Leonardo

Tanca Letizia DEIB Leonardo

vs.

Tanca Letizia DEIS Lonardo

- If the fields are distinct, it is much easier to spot similarities ! E.g. that a word like Lonardo is a wrong representation of Leonardo, because we know that Leonardo is a campus of Politecnico. The same would happen with DEIB.
- We wouldn't be able to teach this knowledge to the system if the tuple were treated as a single string

Let us start by matching strings

- Typing errors, e.g. Giovanni Rossi vs. Givanni Rossi
- Different representation conventions, e.g. Sept 20th vs. 20/9
- Nicknames or abbreviations, e.g. : Giuseppe vs. Beppe
- String inversion, e.g. : Giovanni Rossi vs Rossi, Giovanni
- Abbreviations, e.g. Politecnico di Milano vs PoliMI

String similarity and similarity measures

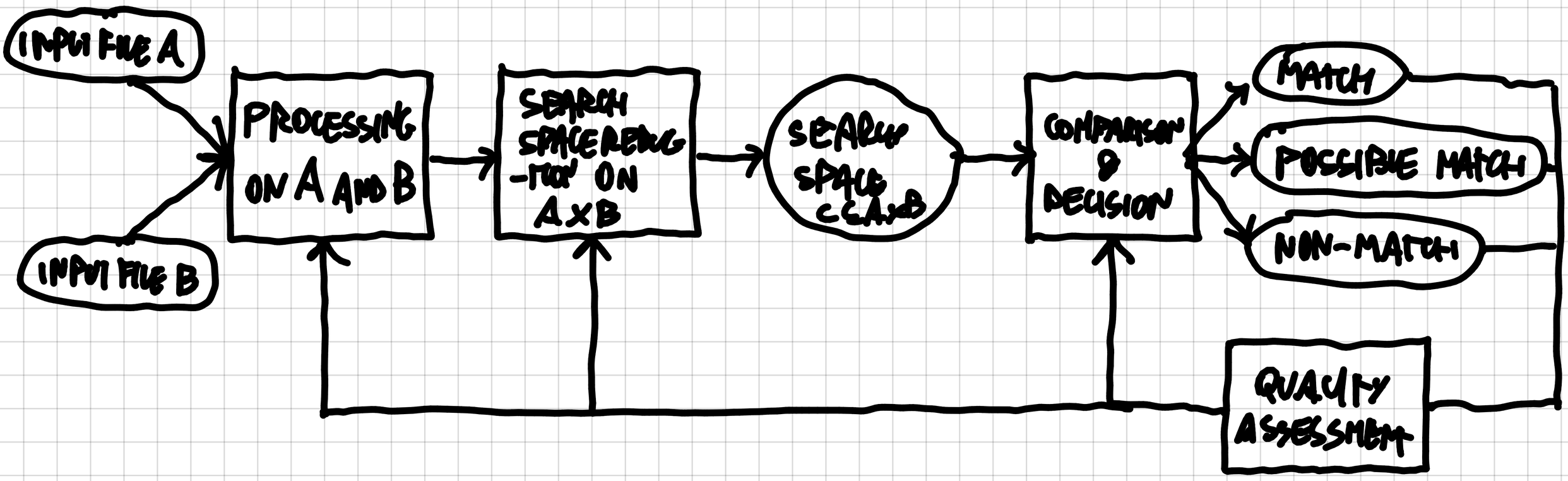
- Given two sets of strings, find all pairs from the two sets that refer to the same real-world entity.
- Each of these pairs is called a match
- In order to find matches, we can use a similarity measure: a value s(x,y) in [0,1]: s(x,y)=1 iff x=y, and the closer s(x,y) is to 0, the lower is the similarity.
- We can decide that x and y match if s(x,y) >= t, with 0 <= t <= 1

  Types of similarity measures:
  - ✓ Sequence-based: edit distance, Needleman-Wunch, affine gap, Smith-Waterman, Jaro, Jaro-Winkler
  - ✓ Set-based: overlap, Jaccard, TF/IDF
  - ✓ Hybrid: generalized Jaccard, soft TF/IDF, Monge-Elkan
  - ✓ Phonetic: Soundex
- Distance measures: they are the opposite, the lower the similarity, the higher the distance

COMPARE EVERYTHING MIGHT TAKE LOTS OF TIME

↓ LOOK FOR A CRITERIA TO MINIMIZE THE NUMBER OF COMPARISONS, BUT INCREASING THE RISK PERCENTAGE

RECORD MATCHING



Edit (or Levenshtein) Distance

- The edit distance is based on the minimal number of operations that are needed to transform string *a* into string *b*:

  character insertion, character deletion, character replacement

- Example:

  a = Politecnico di Milano, *b* = Politecnico Milano

  d(*a*,*b*) = 2, using the following sequence of ops:

  1.Delete the character d of *a*

  2.Delete the character i of *a*

- Similarity: $s(a,b) = 1 - d(a,b) / [\max(\text{length}(a), \text{length}(b))]$
- Example: s(Politecnico di Milano, Politecnico Milano) = 1 – 2 / max(19, 17) = 1 - 0,105... = 0,894...

40

40

Jaro - Winkler

- The **Jaro-Winkler** string comparator counts
  - the **number c of common characters** between two strings (limited to the greatest integer of half the length of the longer string)
  - the number of transpositions that are the number of pairs of common characters that are out of order
    - $S= 1/3*(c/m + c/n + (c-t)/c)$

- Example:
  - Compare two surnames “Barnes” and “Anderson”. Common characters are “arnes”, one transposition “rne” → “ner”
  - S=1/3*(5/6+5/8+(5-1)/5)=0.75

41

Set-based Similarity Measures

- View strings as multisets of tokens
- E.g. the Jaccard measure:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- For strings, this corresponds to dividing the strings into tokens, and computing the measure on the two sets of tokens.
- The tokens are pieces of the strings, like for instance, tokens of lenght 2:

  E.g., for strings pino and pin

  A = {#p, pi, in, no, o#}, B = {#p, pi, in, n#}

  J(pino, pin) = 3/6

SEARCH SPACE REDUCTION

- **BLOCKING**: THE FILE IS PARTITIONED IN EXCLUSIVE BLOCKS AND LIMITING COMPARISONS TO RECORDS WITHIN THE SAME BLOCK. BLOCKING CAN BE IMPLEMENTED BY CHOOSING A BLOCKING KEY AND GROUPING INTO A BLOCK ALL RECORDS THAT HAVE THE SAME VALUES ON THE BLOCKING KEY
- **SORTED NEIGHBORHOOD**: SORTING A FILE AND THEN MOVING A WINDOW OF A FIXED SIZE ON THE FILE, COMPARING ONLY RECORDS WITHIN THE WINDOW
- **PRUNING**: OBJECTIVE OF FIRST REMOVING FROM THE SEARCH SPACE ALL RECORDS THAT CANNOT MATCH EACH OTHER, WITHOUT ACTUALLY COMPARING THEM

Phonetic Similarity Measures

- Match strings based on their *sound*
- Soundex is the most common one. Soundex calculates a four-character code from a word based on the pronunciation and considers two words as similar if their codes are equal.
- Similar sounding letters are assigned the same soundex code.
- Used in in archives, e.g. searching ancestors, since often the way the names of the families evolve according to sound, like for instance Legoff and Legough.
- What is an advantage for the previous use can be a disadvantage for common words, like e.g. “coughing” and “coffin” are pronounced almost the same in English.
- Strongly based on the language, since the way words are pronounced is fundamental, the string “gn” in English and Italian is pronounced in different ways.

American Soundex

- Soundex: it has been defined for problems with names. Names can be spelt in different ways.
- To solve this problem, we need phonetically oriented algorithms which can find similar sounding terms and names.
- Algorithm
  - 1. The first character of the word is retained as the first character of the Soundex code.
  - 2.The following letters are discarded: a,e,i,o,u,h,w, and y.
  - 3.Remaining consonants are given a code number.
  - 4.If consonants having the same code number appear consecutively, the number will only be coded once. (e.g. "B233" becomes "B23")
  - The resulting code is modified so that it becomes exactly four characters long:
    If it is less than 4 characters, zeroes are added to the end (e.g. "B2" becomes "B200")
    If it is more than 4 characters, the code is truncated (e.g. "B2435" becomes "B243")

44

Soundex: code numbers

b, p, f, and v	1
c, s, k, g, j, q, x, z	2
d, t	3
l	4
m,n	5
r	6

45

Efficiency

- Applying s(x,y) to all pairs is quadratic in size of the datasets
- Many solutions have been proposed to choose the pairs of words that most probably match.

Now record matching

Types of record matching

- Rule-based
- Learning- based (supervised or unsupervised)
- Probabilistic

Rule-based Matching

SSN	NAME	AGE	SALARY	POSITION
123456789	JOHN	34	30K	ENGINEER
234567891	KETTY	27	25K	ENGINEER
345678912	WANG	39	32K	MANAGER

Manually written rules that specify when two tuples match. E.g. two tuples refer to the same person if they have the same SSN

SSN	NAME	AGE	SALARY	PHONE
234567891	KETTY	25	20K	1234567
345678912	WANG	38	22K	2345678
456789123	MARY	42	34K	3456789

Many types of rules exist.

E.g.

sim(x,y) =

$0.5s_{SSN}(x,y) + 0.2s_{name}(x,y) + 0.1s_{age}(x,y) + 0.1s_{salary}(x,y) + 0.1s_{position}(x,y)$

Note: possibly using different similarity measures for different attributes !

- Rules are manually specified, it requires a lot of time and effort
- A variant is to learn rules

Learning Matching Rules

- It can be supervised or unsupervised
- Supervised (e.g. classification): learn how to match from training data, then apply it to match new tuple pairs:
  - Learn how to match each attribute of the tuples (the training phase): each (x_i,y_i) is a pair of elements and l_i is a label: "yes" if x_i matches y_i and "no" otherwise
  - Define the weight of each attribute in the final matching of the whole records
  - Apply the learned model to the new tuple pairs
- Supervised learning requires a lot of training data
- Unsupervised learning (typically clustering, based on clusterizing similar values) may solve this problem

Probabilistic Matching

- Model the matching domain using a probability distribution
- Reason with the distribution to make matching decisions
- Key benefits
  - provide a principled framework that can naturally incorporate a variety of domain knowledge
  - can leverage the wealth of probabilistic representation and reasoning techniques already developed in the AI and DB communities
  - provide a frame of reference for comparing and explaining other matching approaches
- Disadvantages
  - computationally expensive
  - often hard to understand and debug matching decisions

Data Fusion

SSN	NAME	AGE	SALARY	POSITION
123456789	JOHN	34	30K	ENGINEER
234567891	KETTY	27	25K	ENGINEER
345678912	WANG	39	32K	MANAGER

SSN	NAME	AGE	SALARY	PHONE
234567891	KETTY	25	20K	1234567
345678912	WANG	38	22K	2345678
456789123	MARY	42	34K	3456789

Once you have understood that some data clearly represent the same entity (same person, in this case), you still have to cope with the problem of what to do when other parts of the info do not match

Resolution function

Inconsistency may depend on different reasons:

- One (or both) of the sources are incorrect
- Each source has a correct but partial view, e.g. databases from different workplaces, e.g.:
  - the full salary is the sum of the two
  - the list of authors of a book is incomplete
  - one of two full names only contains the first letter of the middle name
- Often, the correct value may be obtained as a function of the original ones, e.g. : $value_1 + value_2$, or $1*value_1 + 0*value_2$ or $0{,}5*value_1 + 0{,}5*value_2$)

RESOLUTION FUNCTION: EXAMPLE

SSN	NAME	AGE	SALARY	POSITION
123456789	JOHN	34	30K	ENGINEER
234567891	KETTY	27	25K	ENGINEER
345678912	WANG	39	32K	MANAGER

SSN	NAME	AGE	SALARY	PHONE
234567891	KETTY	25	20K	1234567
345678912	WANG	38	22K	2345678
456789123	MARY	42	34K	3456789

SSN	NAME	AGE	SALARY	POSITION	PHONE
123456789	JOHN	34	30K	ENGINEER	NULL
234567891	KETTY	27	45K	ENGINEER	1234567
345678912	WANG	39	54K	MANAGER	2345678
456789123	MARY	42	34K	NULL	3456789

R=*MAX_AGE*, *SUM_SALARY* (R1 OuterJoin R2)

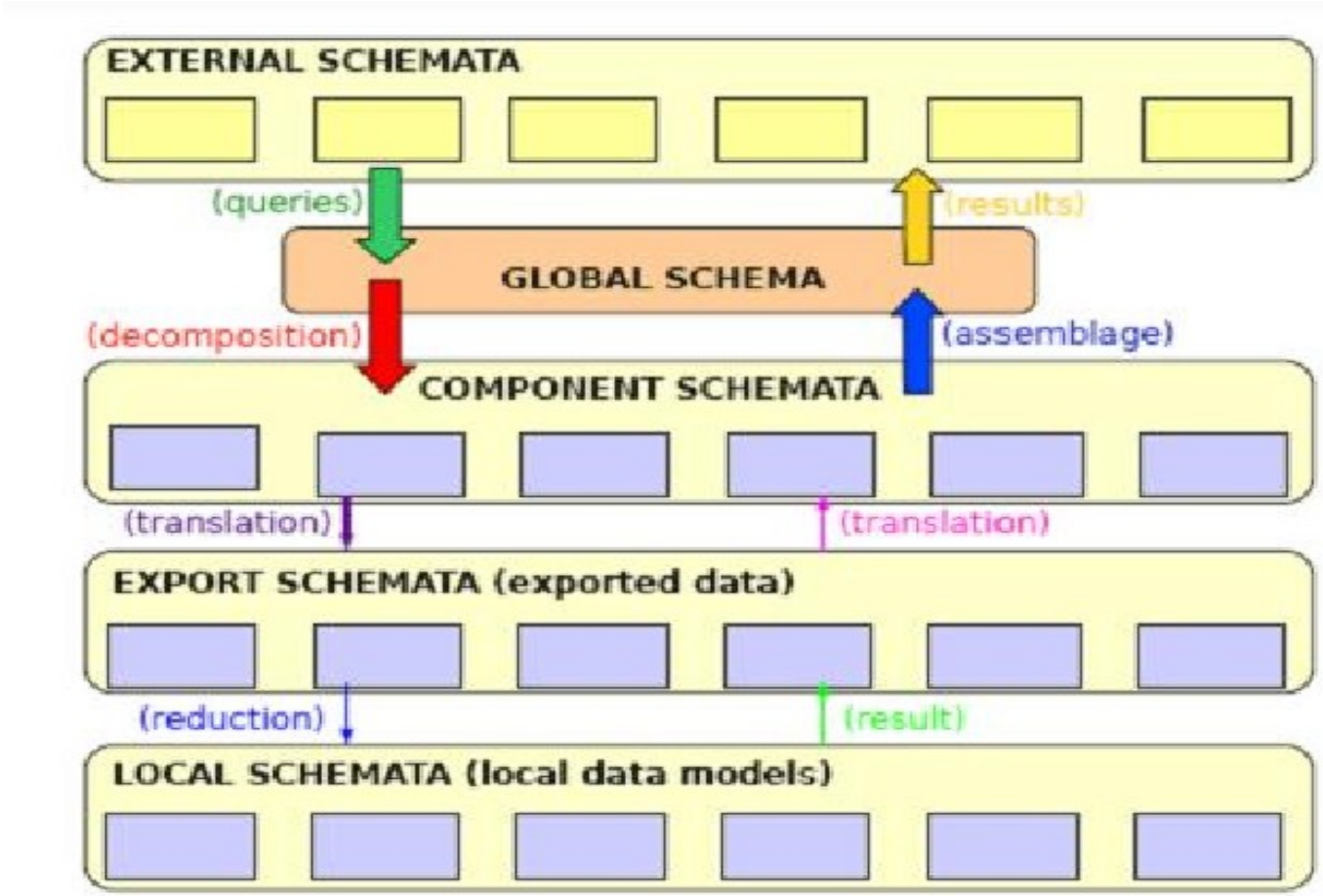
The new application context (recall)

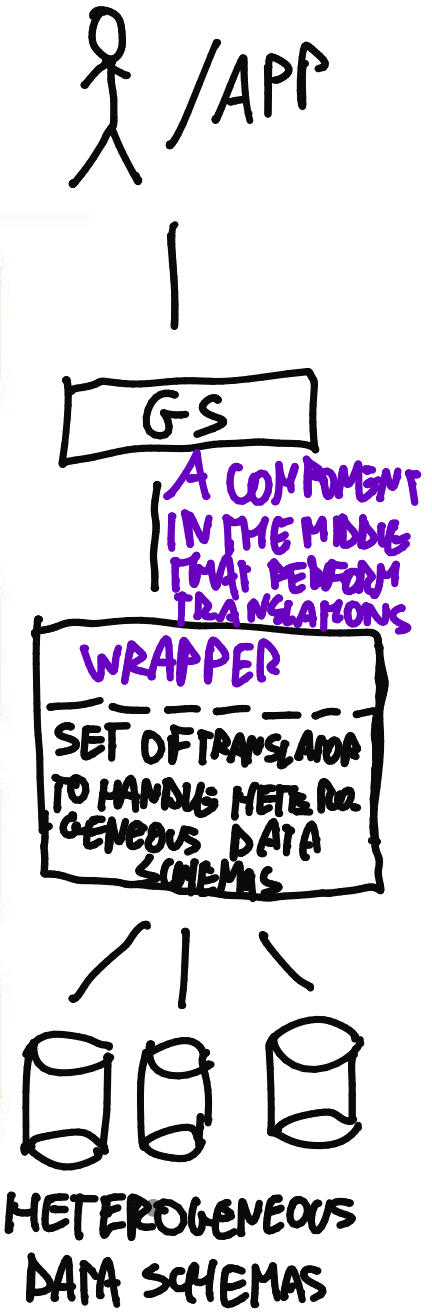
- A (possibly large) number of data sources
- ➢ Heterogeneous data sources
- Different levels of data structure
  - Databases (relational, OO…)
  - Semi-structured data sources (XML, HTML, more markups …)
  - Unstructured data (text, multimedia etc…)
- Different terminologies and different operational contexts
- Time-variant data (e.g. WEB)
- Mobile, transient data sources

Levels of source heterogeneity

- Schemata (already seen)
- Data Models
- Systems (easily overcome if we reconcile data models and schemata)

Data integration in the MULTIDATABASE





A new element in this figure: WRAPPERS (translators)

- They convert queries into queries/commands which are understandable for the specific data source
  - they can even extend the query possibilities of a data source (see e.g. type conversions)
- They convert query results from the source format to a format which is understandable for the application
- Easy to produce in the case of structured data
- If we know exactly the structures e.g. at the base the relational model and of the Object Oriented model, and their query languages, we can translate the queries and the results between the two models
- More difficult is the problem with semistructured data (later)

Design steps

1. Reverse engineering (i.e. production of the conceptual schema)
2. Conceptual schemata integration
3. Choice of the target logical data model and translation of the global conceptual schema
4. Definition of the language translation (wrapping)
5. Definition of the data views (as usual)

Bibliography

- A. Doan, A. Halevy and Z. Ives, Principles of Data Integration, Morgan Kaufmann, 2012
- L. Dong, D. Srivastava, Big Data Integration, Morgan & Claypool Publishers, 2015
- Roberto De Virgilio, Fausto Giunchiglia, Letizia Tanca (Eds.): Semantic Web Information Management – A Model-Based Perspective. Springer 2009, ISBN 978-3-642-04328-4
- M. Lenzerini, Data Integration: A Theoretical Perspective, Proceedings of ACM PODS, pp. 233-246, ACM, 2002, ISBN: 1-58113-507-6
- Clement T. Yu, Weiyi Meng, Principles of Database Query Processing for Advanced Applications , Morgan Kaufmann, 1998, ISBN: 1558604340