An introduction to data integration

Prof. Letizia Tanca

Technologies

for Information Systems

Motivation

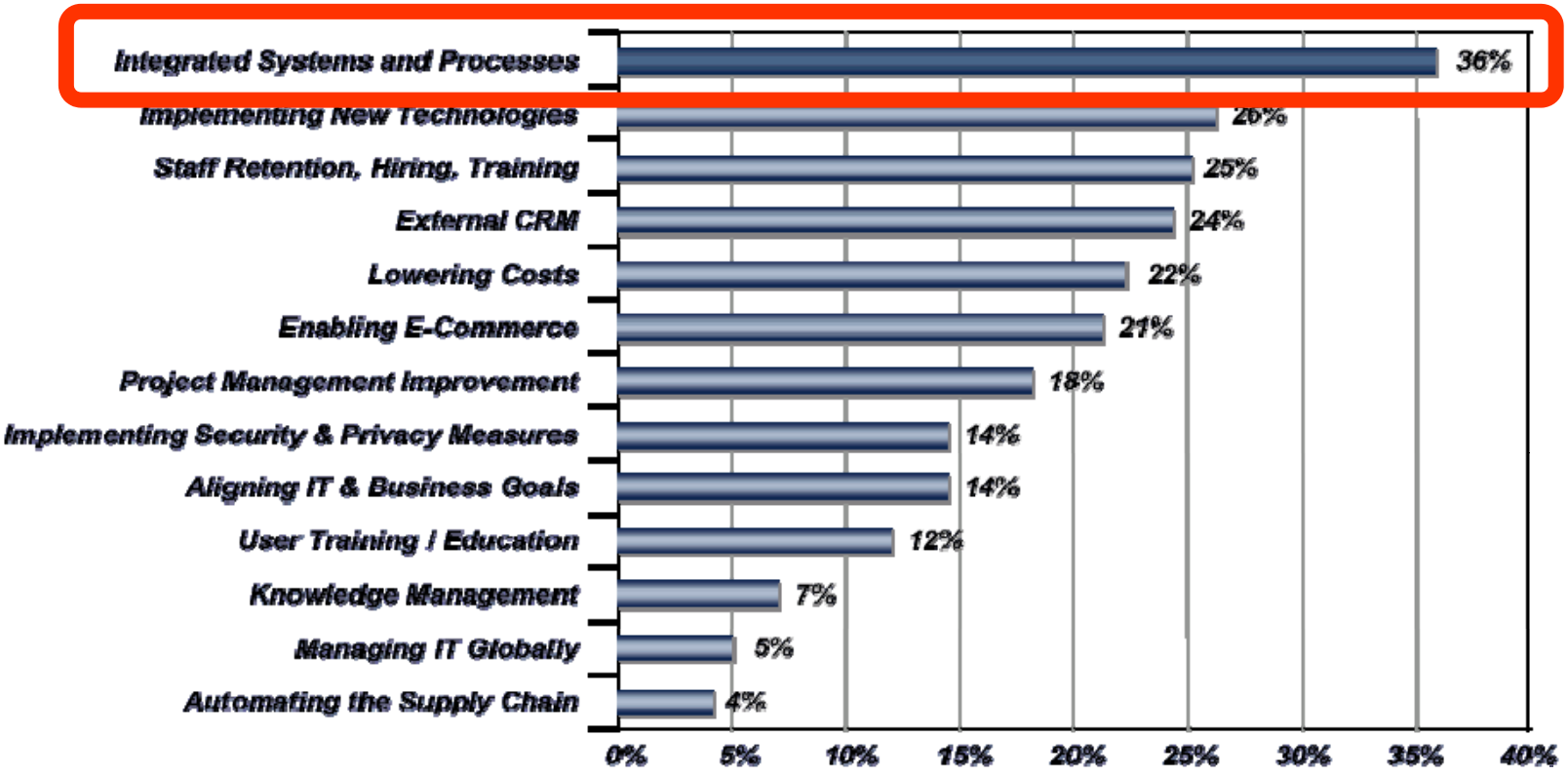
In modern Information Systems there is a growing quest for achieving integration of SW applications, services and information (databases, and others) managed by the organization or set of organizations

Due to the need to:

- reuse legacy applications and databases
- reconcile the different points of views adopted by the different players using the information system

We concentrate on *Data Integration*

Top IT Spending Priorities[1]



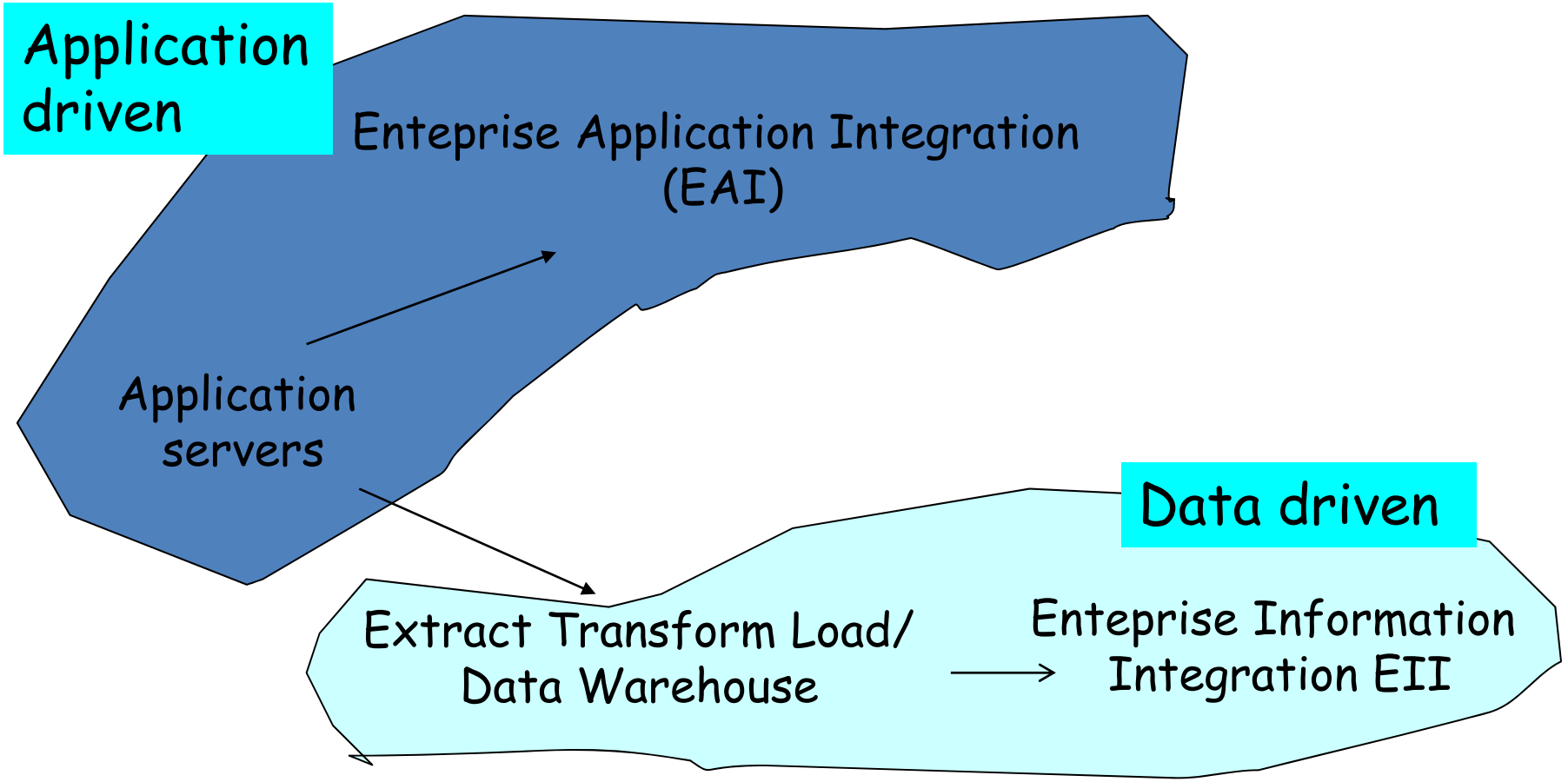
[1]CIO Magazine Survey, February 2002

33% of firms surveyed have EAI projects (Forrester, March 2002 Business Technographics benchmark)

History

- Typical IT assets of a modern company include ERP systems, sales tracking systems, HR systems, etc.
- Over the last twenty years these were implemented based on a client-server computing model where the DBMS runs at the server level, accessed by a collection of applications which run on the client desktop.
- Recently, Client-Server computing has become obsolete: the client level moves inside a web browser, and three-tier architectures are built where the client may be thick or thin.
- In either case, the application is executed on the middle tier (Application Server).

Evolution from Application Servers



Problems in traditional DB architectures

Lot of different types of heterogeneities among several DBs to be used together

1. Different platforms: Technological heterogeinity
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 previuosly developed at the participating DBMS → Schema (or semantic) heterogeneities
5. Errors in data, that result different values for the same info → instance heteorgeneities
6. Dependencies exist among databases, databases and applications, among applications

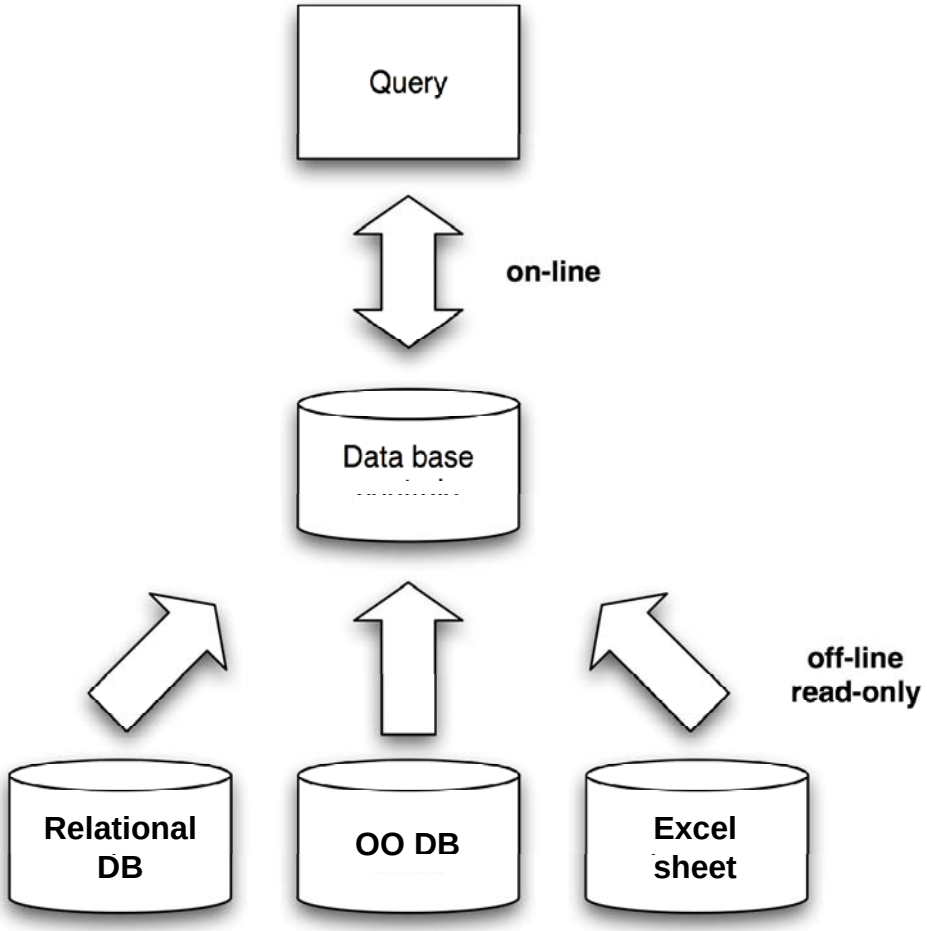
Virtual vs materialized integration

A distinction is made on how the information needed is retrieved:

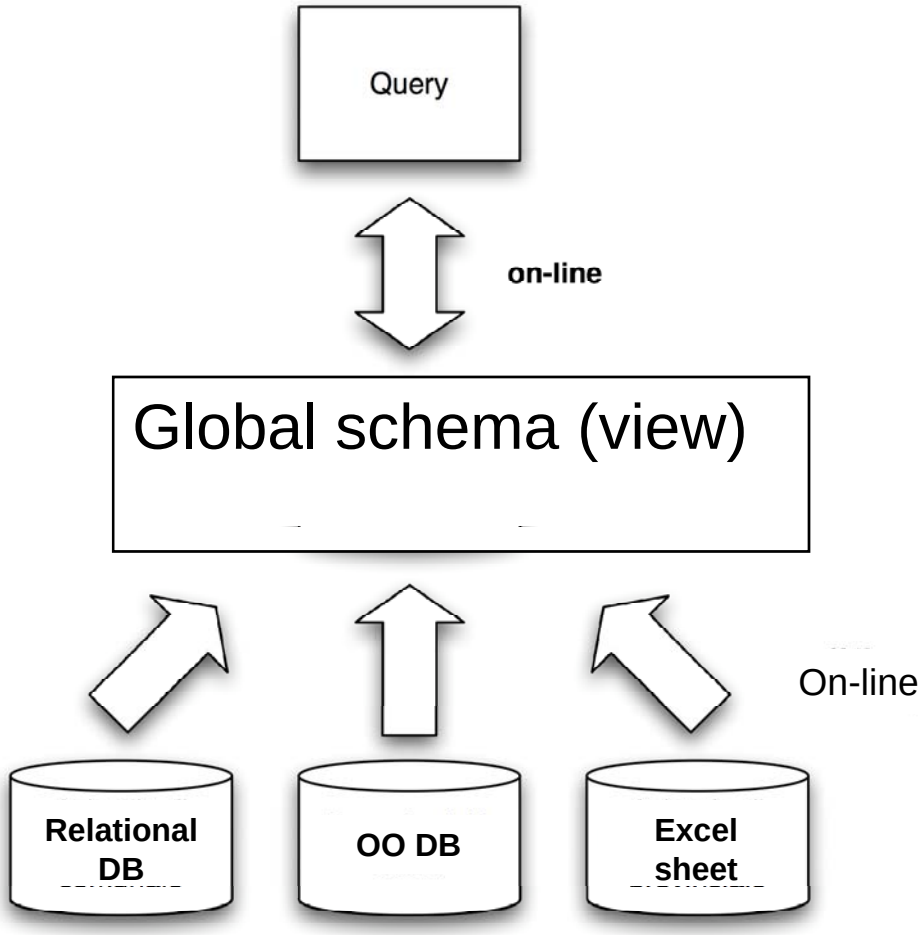
- virtual or
- materialized integration approach

determines how the information is accessed.

Materialized



Virtual


Query
on-line
Global schema (view)
On-line
Relational DB
OO DB
Excel sheet

Relevant Types of Integrated Database Systems

1. Use a materialized data base (data are merged in a new database) → Enterprise-Transform-Load Systems
   → Data Warehouses: Materialized integrated data sources

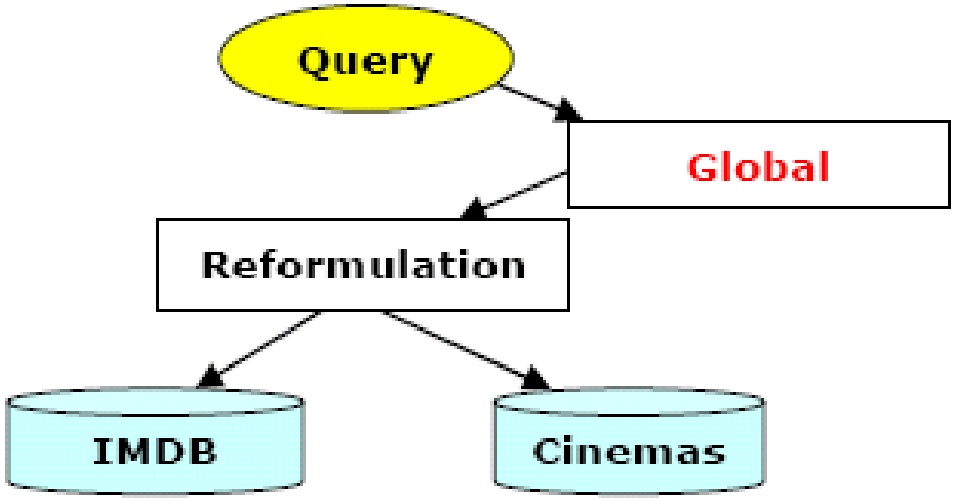
2. Use a virtual non-materialized data base (data remain at sources) → Enteprise Information Integration (EII) (or Data Integration) Systems
   → Several solutions

Materialized integration

- Large common machines 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 mining.
- Not appropriate when need to integrate the *operational* systems (keeping data up-to-date)
- Will be dealt with in the second part of the course

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.



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. After all, 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 simple example (Batini)

Multidatabase schema (e.g. DataJoiner)

Sybase.PUBLICATIONS(PNR, TITLE, AUTHOR, JOURNAL)
Sybase.AUTHORS(ANR, TITLE, NAME, AFFILIATION)
Oracle.PAPERS(NUMBER, TITLE, WRITER, PUBLISHED)
Oracle.WRITER(FIRSTNAME, LASTNAME, NROFPUBLICATIONS)

PUBLICATIONS(PNR, TITLE, AUTHOR, JOURNAL)
AUTHORS(ANR, TITLE, NAME, AFFILIATION)

Database 1 (e.g. Sybase) schema

PAPERS(NUMBER, TITLE, WRITER, PUBLISHED)
WRITER(FIRSTNAME, LASTNAME, NROFPUBLICATIONS)

Database 2 (e.g. Oracle) schema

Query execution in the integrated database

When a DML statement, such as a query, is submitted, the 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.

Views

- Also called external schemata

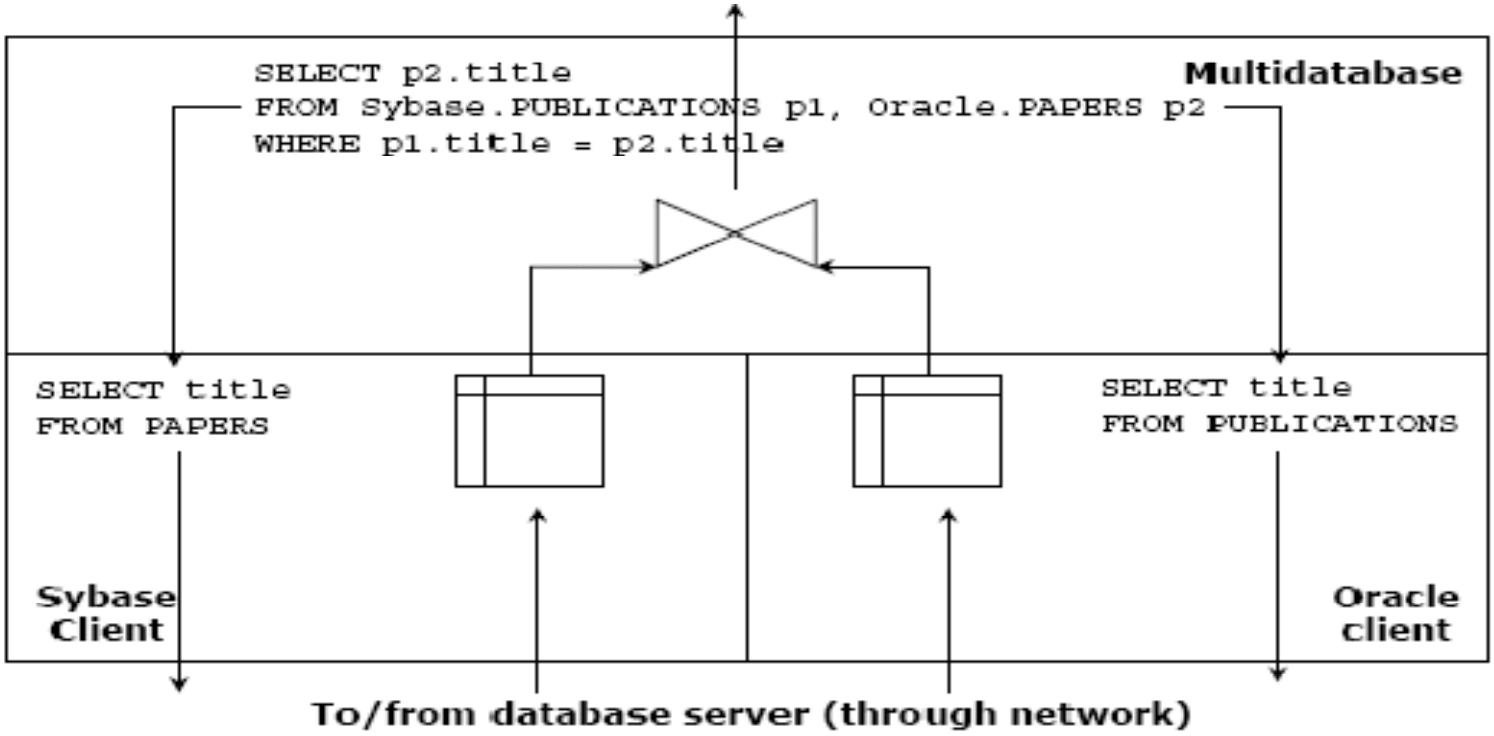
- Sintax:

  `create view` *ViewName* [(*AttList*)] **`as`**
  SQLquery

  [**`with`** [**`local`** | **`cascaded`**] **`check option`**]

Query execution: example

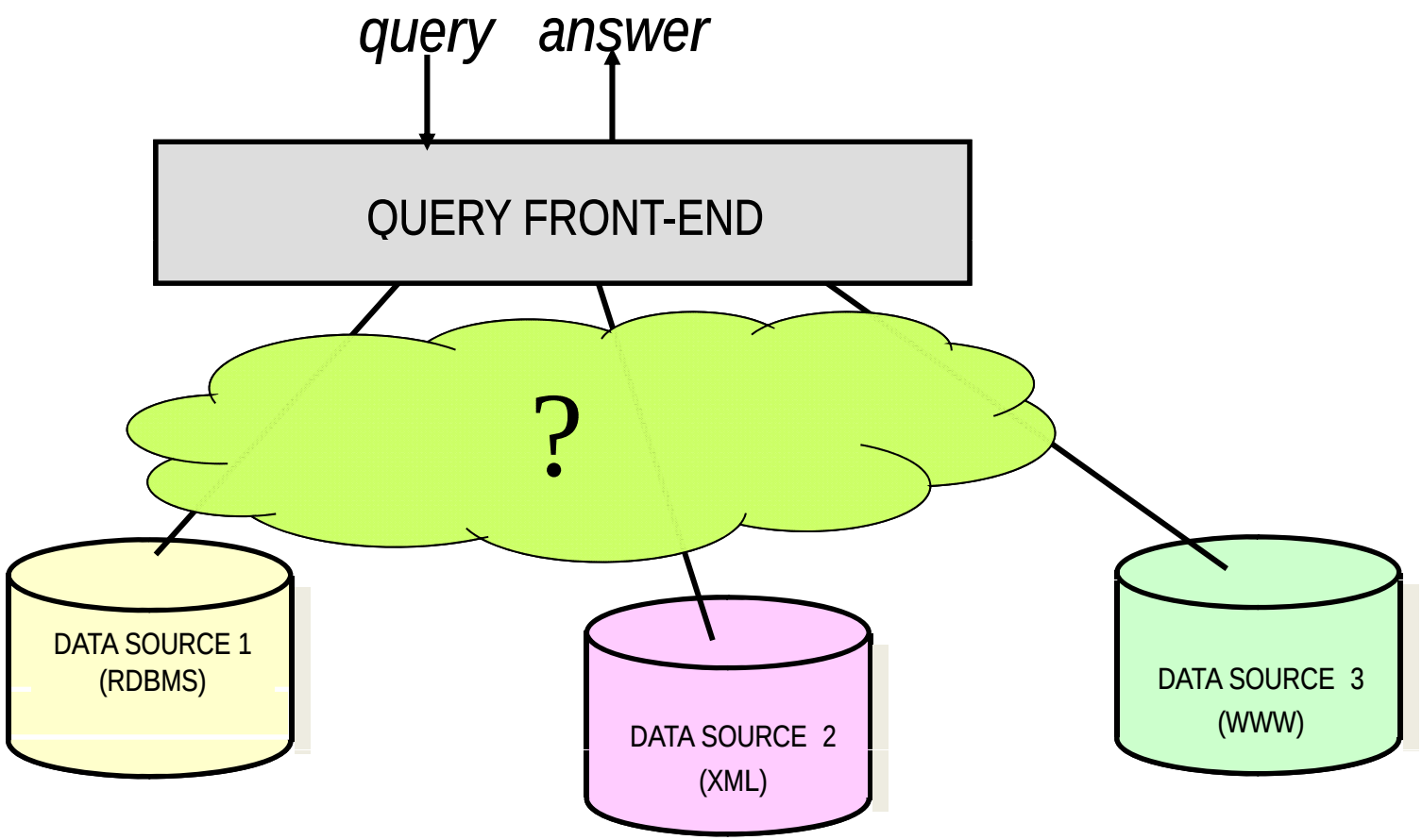
- The join needs to be evaluated at a global level.
- One of the main challenges in developing virtual data integration systems is to find good strategies of how to decompose and evaluate queries against multiple databases in an efficient manner.



The Data Integration problem

- Combining data coming from different data sources, providing the user with a unified vision of the data
- Detecting correspondencies between similar concepts that come from different sources, and conflict solving

The Data Integration problem



More systematically, the data integration problems concern…

- Autonomy:
  - Design, or representation, autonomy: which data, and how
  - Communication autonomy: which services should be provided to the users or to the other DB systems
  - Execution autonomy: which algorithms for query processing and in general for data access

  which causes
- Heterogeneity:
  - Different platforms
  - Different data models
  - Different query languages
  - Different data schemas, i.e., modeling styles (*conflicts*…)
  - Different values for the same info (*inconsistency*)

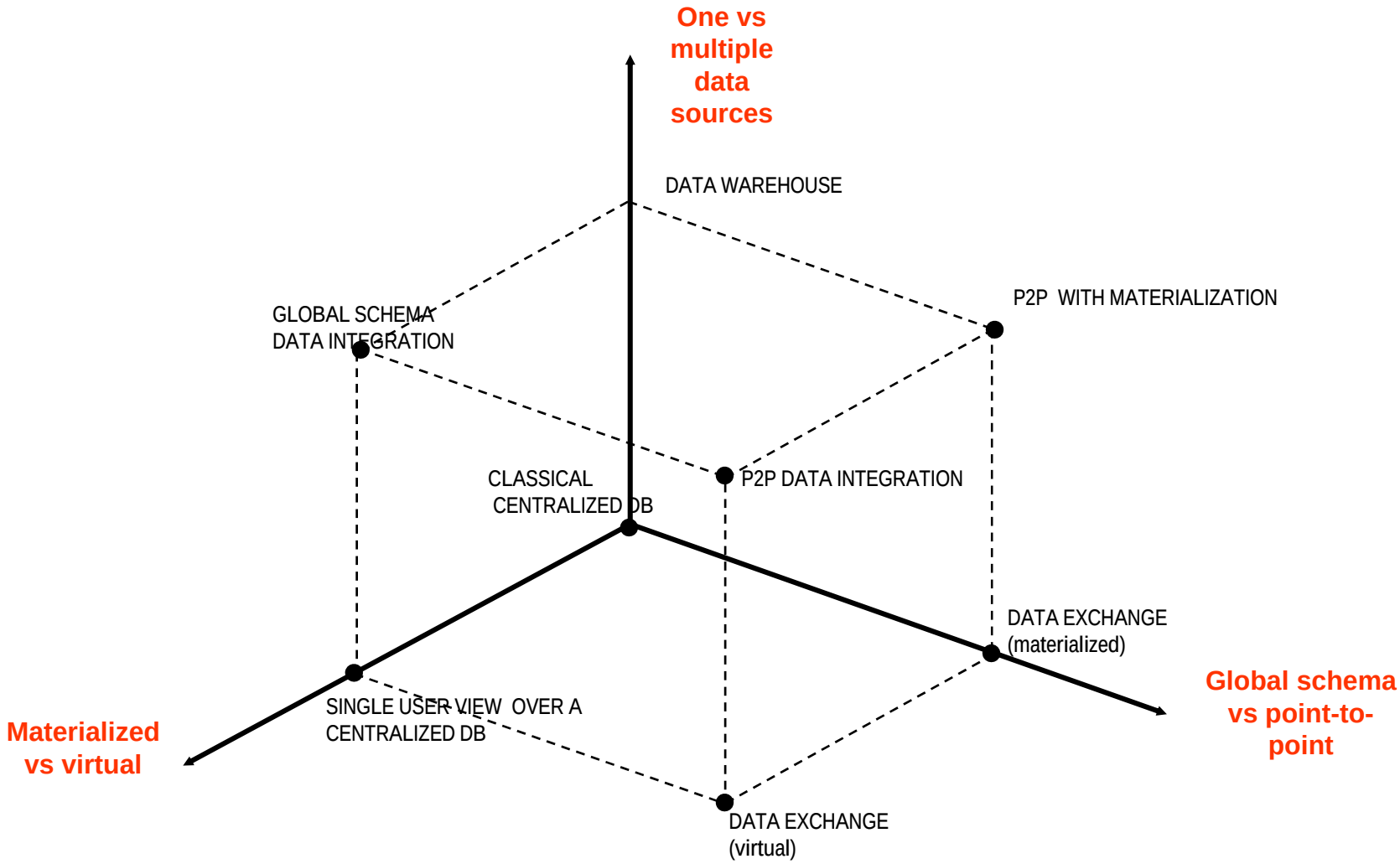
The possible situations

- Also in a unique, centralized DB, there is a problem of integration
- Distributed or federated DB
  - Homogeneous data: ***same*** **data model**
  - Heterogeneous data: ***different*** **data models**
  - Semi-structured data
- The extreme case: data integration for transient, initially unknown data sources

An orthogonal classification

- Centralized architecture: the traditional architecture for centralized, virtual or materialized data integration
- Data exchange: pairwise exchange of data between two data sources
- Peer-to-peer: decentralized, dynamic, data-centric coordination between autonomous organizations

DATA INTEGRATION



The possible solutions

- 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…

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

The typical problem of data design

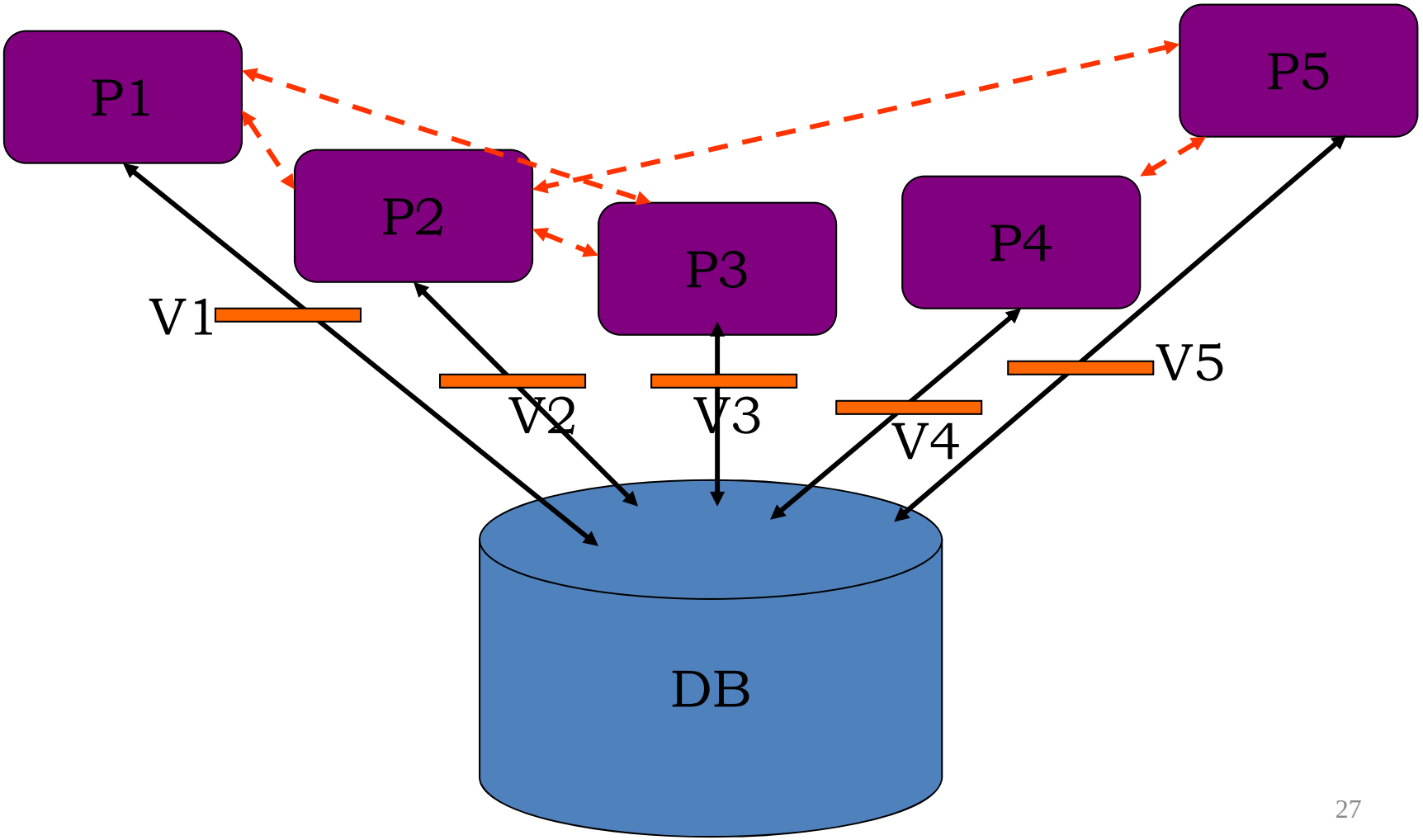
- Given a data model
- Organizing data according to the chosen data model in order to:
  - Avoid inconsistencies
  - Allow *query optimization*
- In a unique DB:
  - only *schema heterogeity*

UNIQUE DB

Each datum, whatever application uses it, only appears once. This ELIMINATES useless REDOUNDANCIES, which would cause:

- Inconsistencies
- Useless memory occupation

An integrated DB



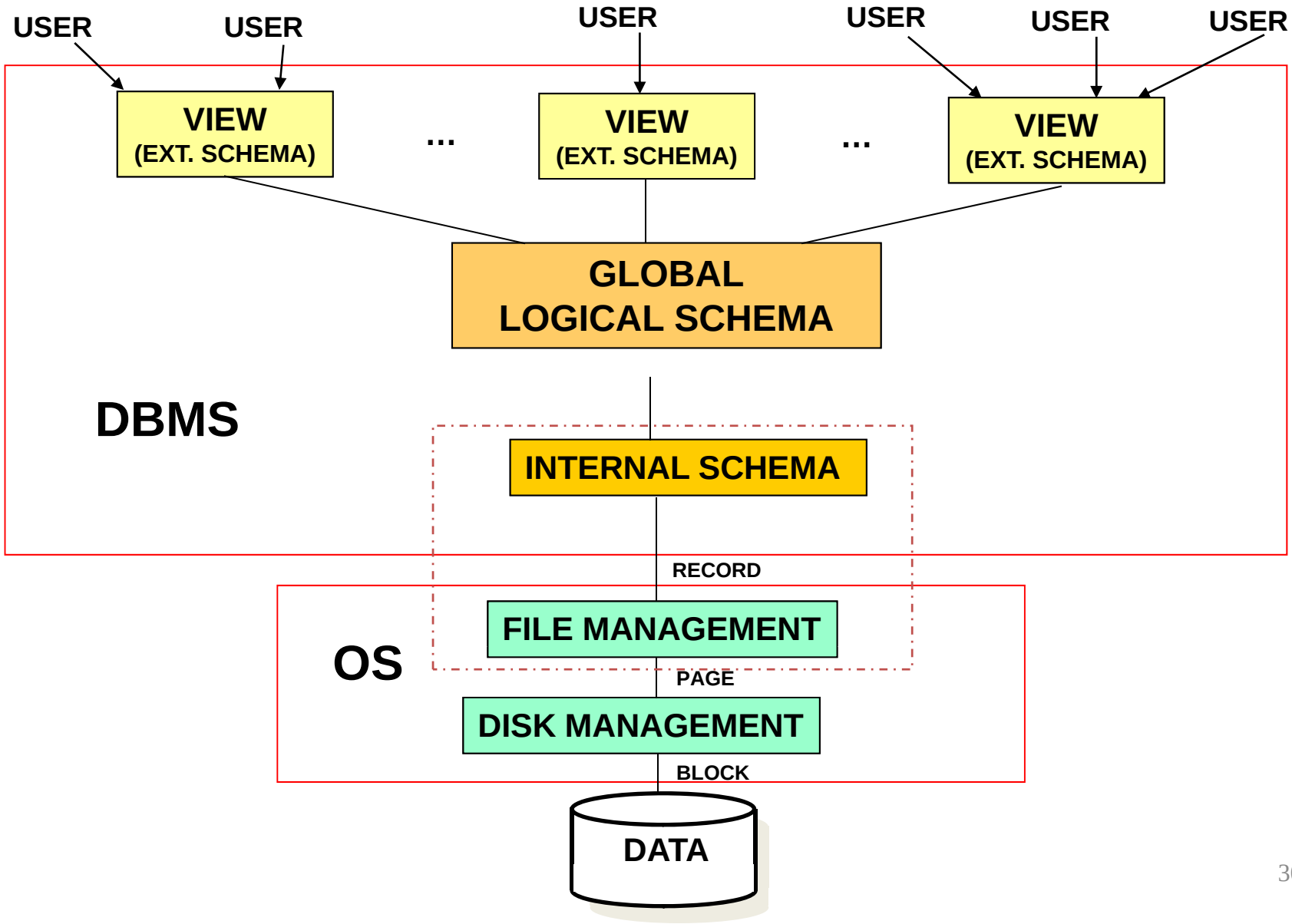
UNIQUE DB : view integration

- 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

Design steps for a unique DB
by view integration
(*mixed* strategy)

1. Subsystem (functionality) identification
2. Design of the skeleton schema
3. Subschema (view) conceptual design
4. View integration and restructuring
5. Conceptual to logical translation (of the global schema, of the single subschemata)
6. Reconciliation of the global logical schema with the single schemata (logical view definition)

Query processing in a centralized DB (ANSI/SPARC architecture)



Point 4 above:
View Integration

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

→ Recall: in a unique DB we can only have *schema heterogeity*

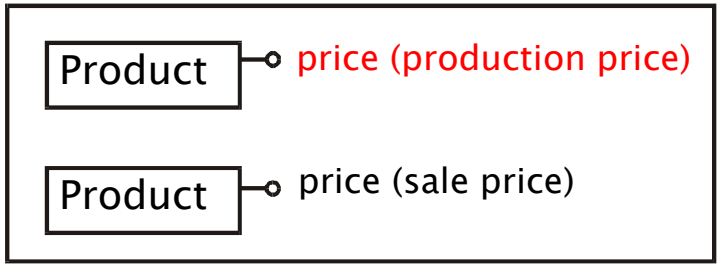
→ This same operation is carried out for each of the integration problems, from the simplest (this one) to the most complex

4.a Related concept identification

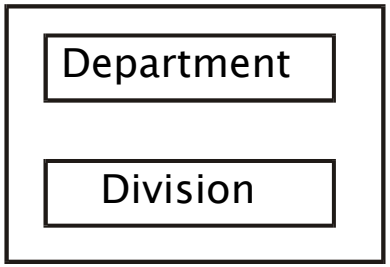
- Ex:
  - employee, clerk
  - exam, course
  - code, num
- Not too difficult if manual
- Very difficult if automatic – this is the extreme case

4.b Conflict analysis

- Name conflicts
  - HOMONYMS



  - SYNONIMS



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)

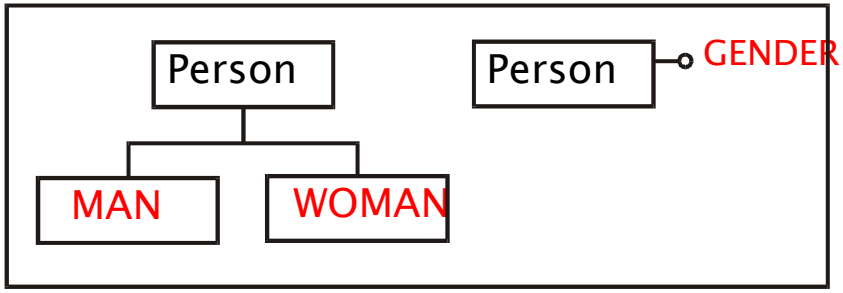
Conflict analysis

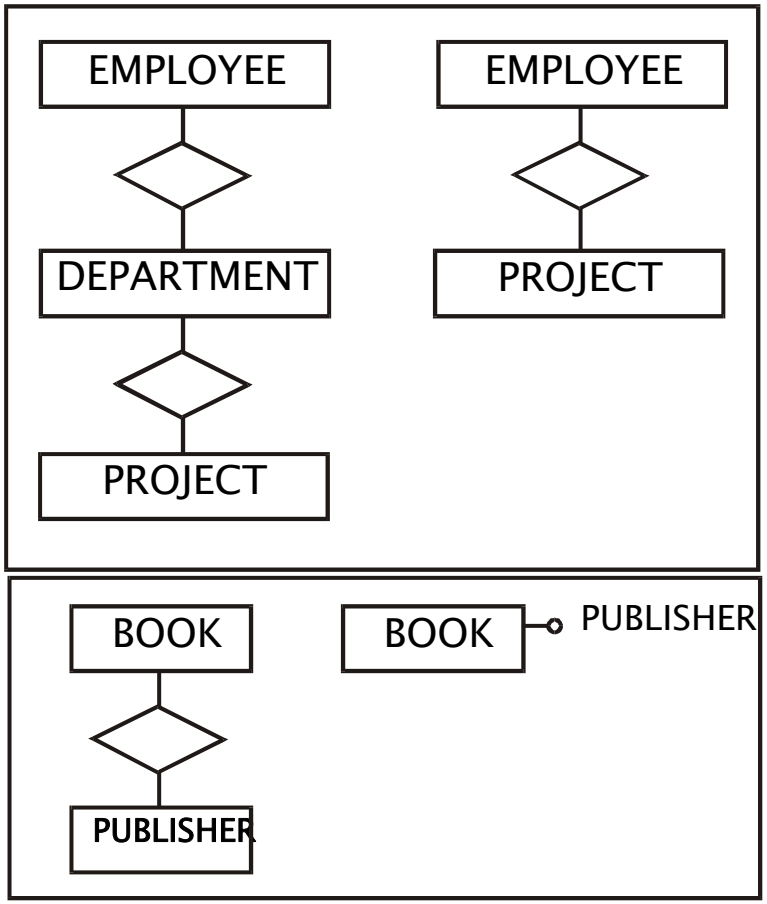
DATA SEMANTICS

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

Conflict analysis

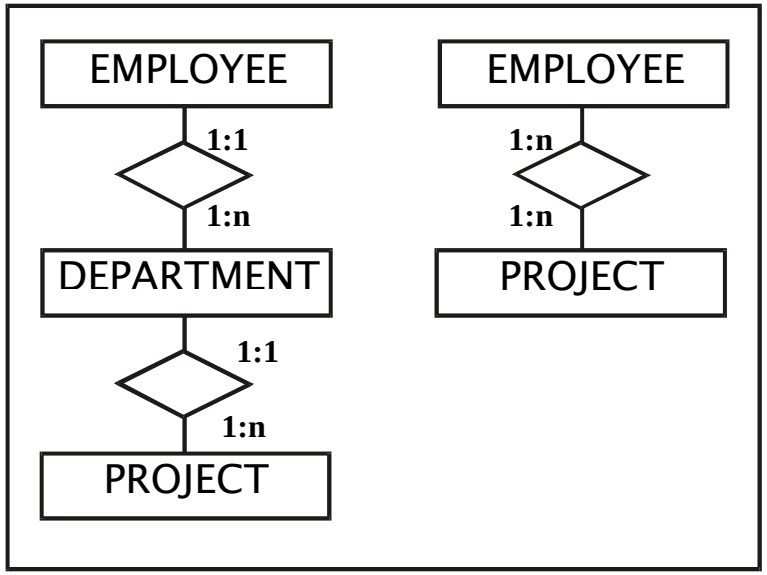
STRUCTURE CONFLICTS





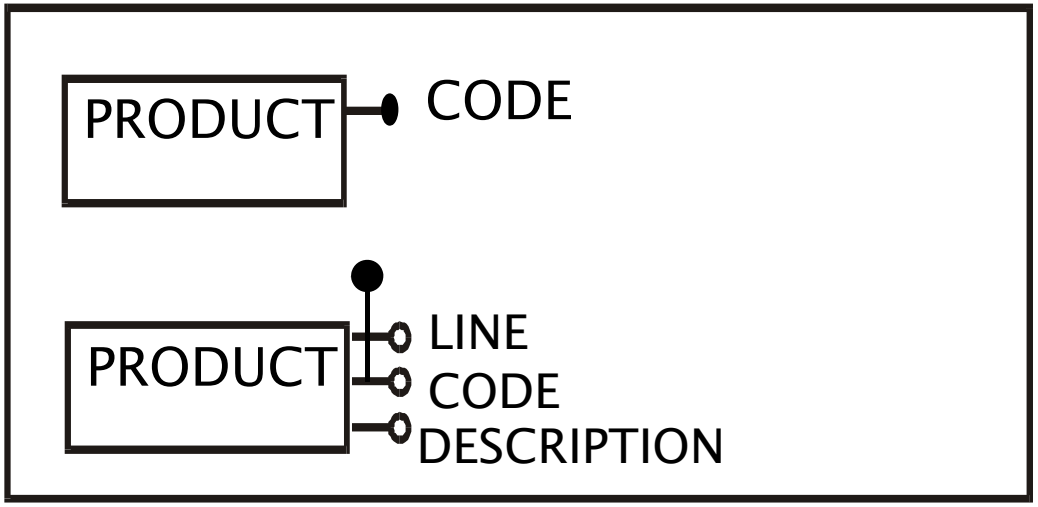
Conflict analysis

- DEPENDENCY (OR CARDINALITY) CONFLICTS



Conflict analysis

- KEY CONFLICTS



4.c Schema Integration

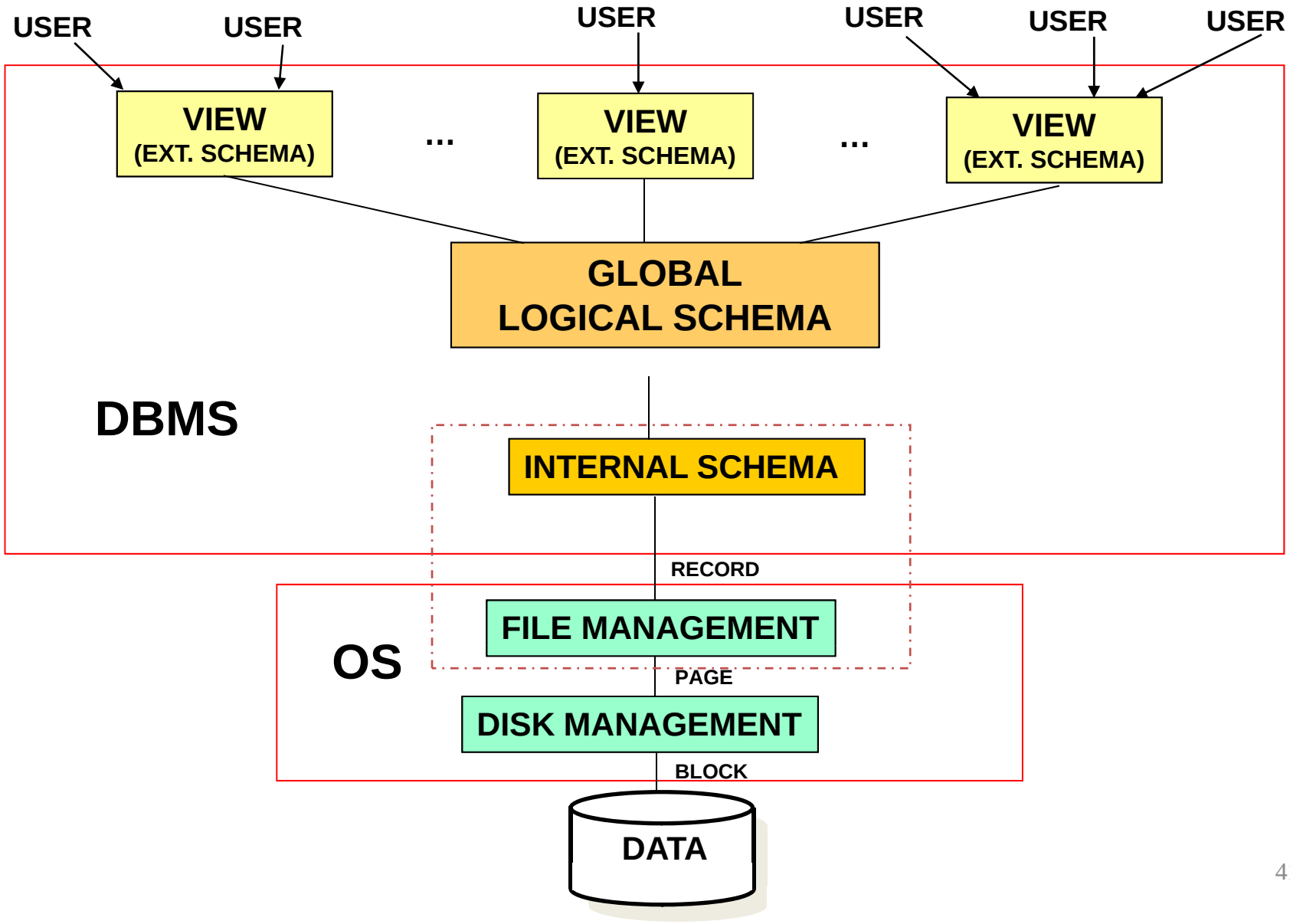
- Conflict resolution
- Production of a new conceptual schema which expresses (as much as possible) the same semantics as the schemata we wanted to integrate
- Production of the transformations between the original schemata and the integrated one: $V_1(DB)$, $V_2(DB)$,… $V_3(DB)$

Exercise

We want to define the database of a ski school having different sites in Italy. Each site has a name, a location, a phone number and an e-mail address. We want to store information about customers, employees and teachers (SSN, name, surname, birth date, address, and phone). For each teacher we store also the technique (cross-country, downhill, snow board). The personnel office has a view over the personnel data, that is, ski teachers and employees.

The school organizes courses in the different locations of the school. The course organization office has a view over these courses. The courses have a code (unique for each site), the starting date, the day of the week in which the course takes place, the time, and the kind of course (cross-country, downhill, snow board), the level, the number of lessons, the cost, the minimal age for participant. For each course, a unique teacher is associated, and the participants.

Query processing in a centralized DB (ANSI/SPARC architecture)



Query processing in a centralized DB

- User issues query on the view → $Q(V_i)$
- Query composition → $Q \circ V_i\ (DB)$
- Answer is in terms of the base relations (global schema) or of the viewed relations (external schema), depending on how sophisticated the system is

Distributed DB

The simplest case of non-centralized DB

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

NON-CENTRALIZED DBs

More heterogeneities:

- 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*

Approaches

- Data conversion (materialization): data are converted and stored into a unique system
  - Multiple copies: redundancy, inconsistency
  - Application rewriting if one system is discarded
- Data exchange: 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: creation of a global schema

Data integration in the Multidatabase

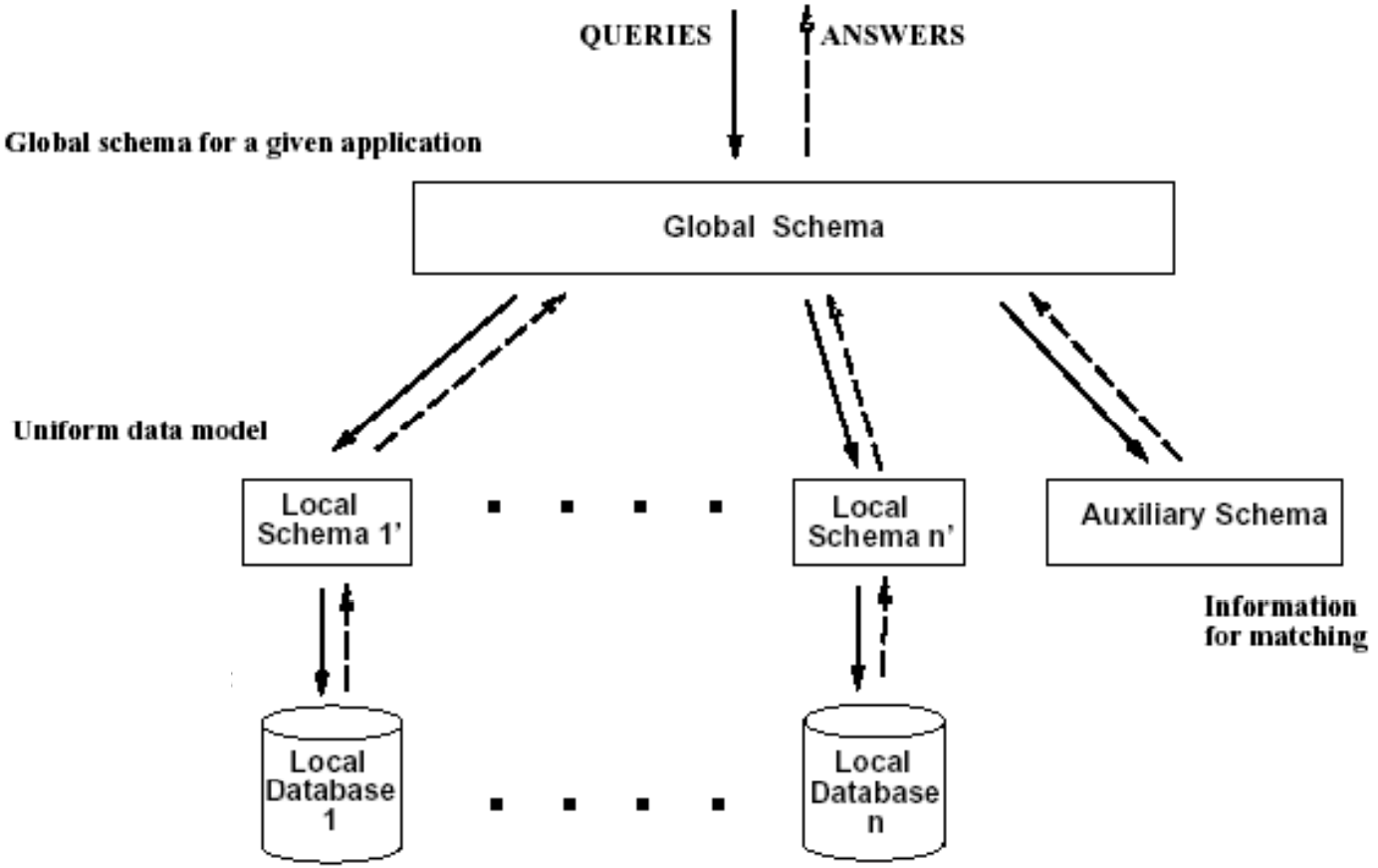
1. Source schema identification (when present)
2. Source schema reverse engineering (data source conceptual schemata)
3. Conceptual schemata integration and restructuring
4. Conceptual to logical translation (of the obtained global schema)
5. Mapping between the global logical schema and the single schemata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

Source schema identification: a first approximation

- 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 homogeneous data model



Global query processing: an example of “unfolding”

A global, nested query. Given R1 in DB1 and R2 in DB2,

```
Select R1.A
From R1
Where R1.B in
    (Select R2.B From R2)
```

In a different notation:

Q(A):-R1(A,B,_,_..), R2(B,_,_...).

Can be decomposed into 2 subqueries

Global query processing: an example (2)

Query over DB2:

```
Select R2.B into X
From R2
```

Query over DB1:

```
Select R1.A
From R1
Where R1.B in X
```

This is an “easy” case, more complex query transformations may be in order, and however…

Questions

- How much data has to be sent “upwards”, for global evaluation
- How much data processing has to be done locally

➢ AN **OPTIMIZATION** PROBLEM

Point 2 above: Source schema reverse engineering

Reverse engineering of the source schemata, and their translation into a conceptual model (e.g. ER)

- CASE tools may partially help us to reconstruct the original conceptual schema
- However, conceptual schemata are more expressive than logical ones, thus we should know the reality of interest to really be able to recover the initial knowledge, lost in the logical design phase

Point 3 above: Conceptual schemata integration and restructuring

As in the centralized DB case

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

Point 4 above: Conceptual-to-logical translation → well-known

Point 5 above: Mapping between the global logical schema and the single source schemata (logical view definition)

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

Mapping between data sources and global (mediated) schema

- 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

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

Mapping between data sources and global schema

- Global schema ***G***
- Source schemata ***S***
- Mapping ***M*** between sources and global schema: a set of assertions

$$q_S \rightarrow q_G$$
$$q_G \rightarrow q_S$$

Intuitively, the first assertion specifies that the concept represented by a view (query) q_S over a source schema S corresponds to the concept specified by q_G over the global schema. Vice-versa for the second assertion.

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 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)
Tipe(TypeCode, Name, Description)

n.b.: 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
```

GAV

- Suppose now we introduce a new source
- The simple view we have just created is to be modified
- In the simplest 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

GAV

- 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
- Query processing is based on unfolding
- Example: one already seen

Query processing in GAV:
Unfolding

```
Select R1.A
From R1
Where R1.B in
    (Select R2.B From R2)
```

Query over DB2:

```
Select R2.B into X
From R2
```

Query over DB1:

```
Select R1.A
From R1
Where R1.B in X
```

Query processing in GAV

QUERY OVER THE GLOBAL SCHEMA

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

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”
```

How do we write the GAV views?

The most useful (and used) integration operators within the relational model

- 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

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

R2

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

More will be said about these “uncertain” values

LAV

A mapping LAV is a set of assertions, one for each element *s* of each source $\mathbb{S}$

$$s \rightarrow q_G$$

Thus the content of each source is characterized in terms of a view $\boldsymbol{q_G}$ 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

LAV

- 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

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

N.B.: Some information is lacking: there is no correspondent value (e.g.Warnings, Notes, etc..). → A difficult job

Query processing in LAV

- Queries are expressed in terms of the global schema

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

- Views specify transformations from global to sources

CREATE VIEW SOURCE1. Version AS

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

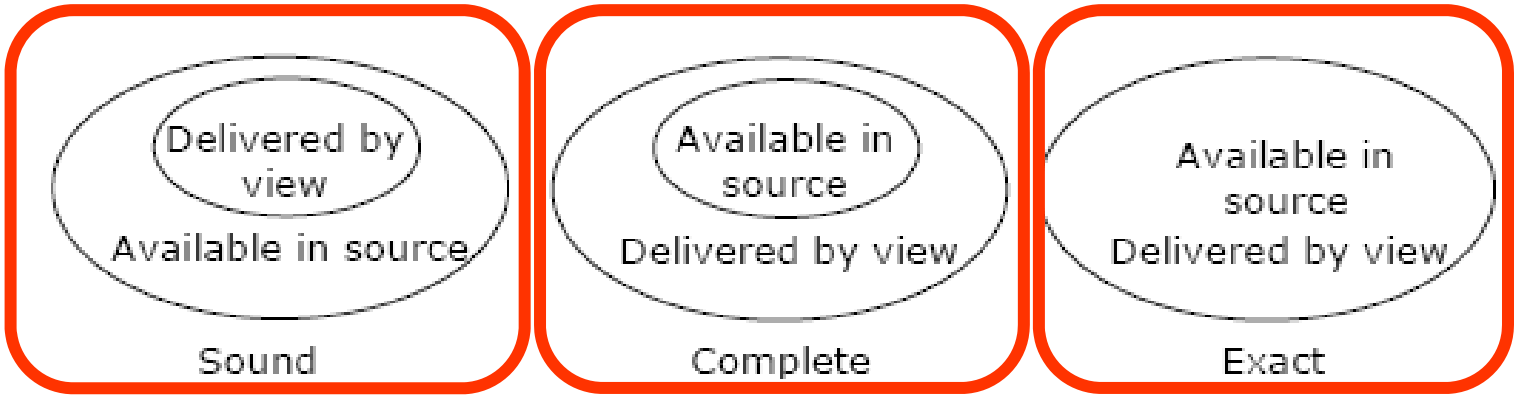
Problem: the mapping, i.e. the view definition, specifies the opposite transformation w.r.t. the one we need!

Sound, exact, and complete views - recall

A view, defined over some data source, is sound when it provides a subset of the available data in the data source that corresponds to the definition.

A view is complete if it provides a superset of the available data in the data source that corresponds to the definition.

A view is exact if it provides all and only data corresponding to the definition.



Observations

- Do GAV and LAV provide exact views?
- Usually we assume so, but it is not so (e.g. in GAV if we define integrity constraints on the global schema)

76

GAV approach

S1 (Name,Age)
S2 (Name,Age)
G (Name, Age)

Create view G as
Select G.Name as S1.Name, G.Age as S1.Age From S1
Union Select G.Name as S2.Name, G.Age as S2.Age From S2

With the following global integrity constraint:
G.Age > 18

S1	Name	Age
	Rossi	17
	Verdi	21

S2	Name	Age
	Verdi	21
	Bianchi	29

Tuples accessible from the data sources

GProf	Name	Age
	Rossi	17
	Verdi	21
	Bianchi	29

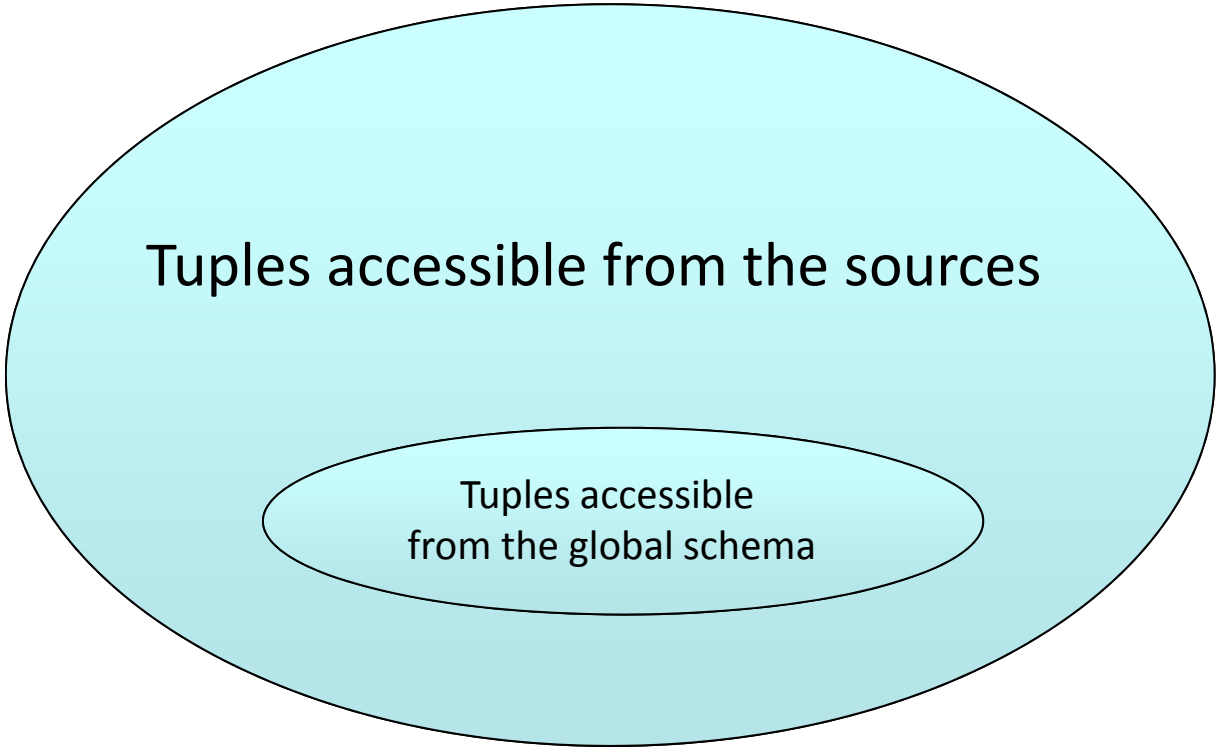
This view is the union of the two data sources but does not satisfy the integrity constraint

Tuples accessible from global schema

GProf	Name	Age
	Verdi	21
	Bianchi	29

The mapping is sound, not complete, thus not exact

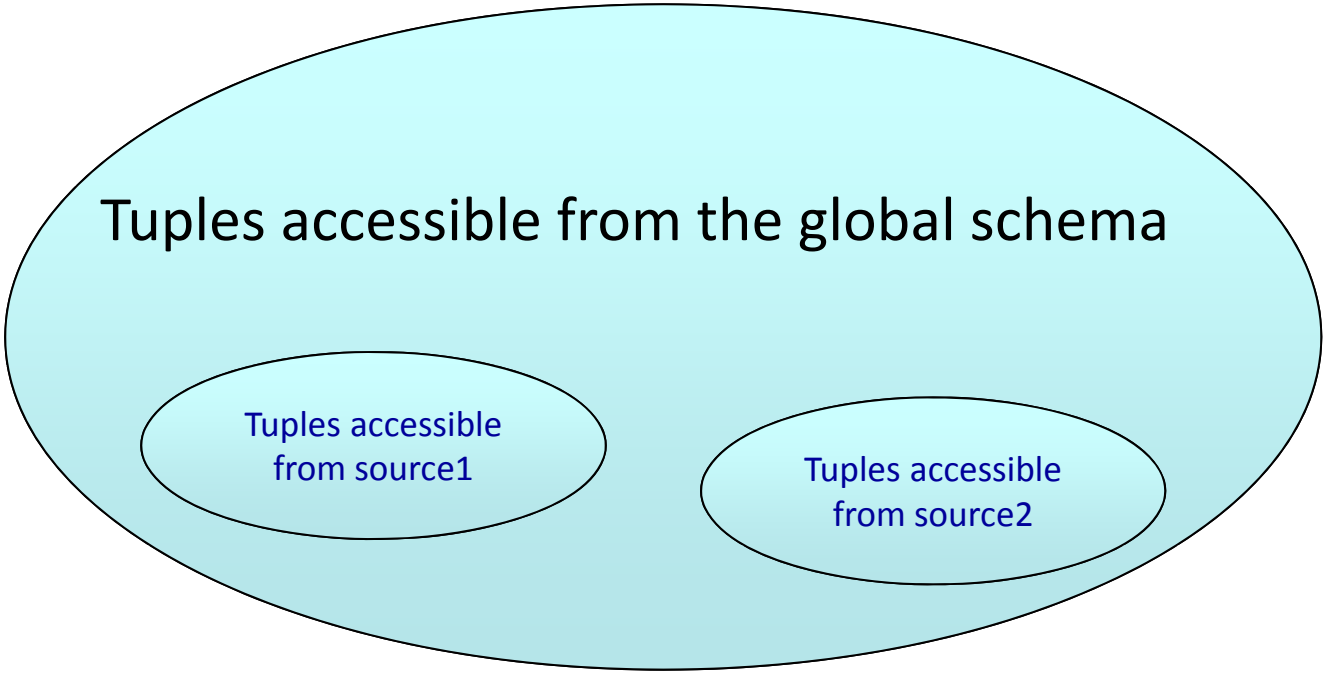
GAV
with integrity constraints



The mapping can be exact or only sound

LAV approach

The global schema can be defined independently of the sources' schemata



The mapping can be exact or only complete

Source heterogeneity

- Schemata (already seen)
- Data Models
- Technology (not our concern here)

Heterogeneous information sources:

UNIVERSITY DB (relational)

```
Department(dept_code,dept_name,budget)
Research_Staff(name,e_mail,dept_code,s_code)
FK dept_code REF Department, s_code REF Session
School_Member(name,school,year,e_mail)
Session(s_code,session_name,length,room_code)
FK room_code REF Room
Room(room_code,seats_number,notes)
```

Heterogeneous information sources:
Tax_Position source (XML)

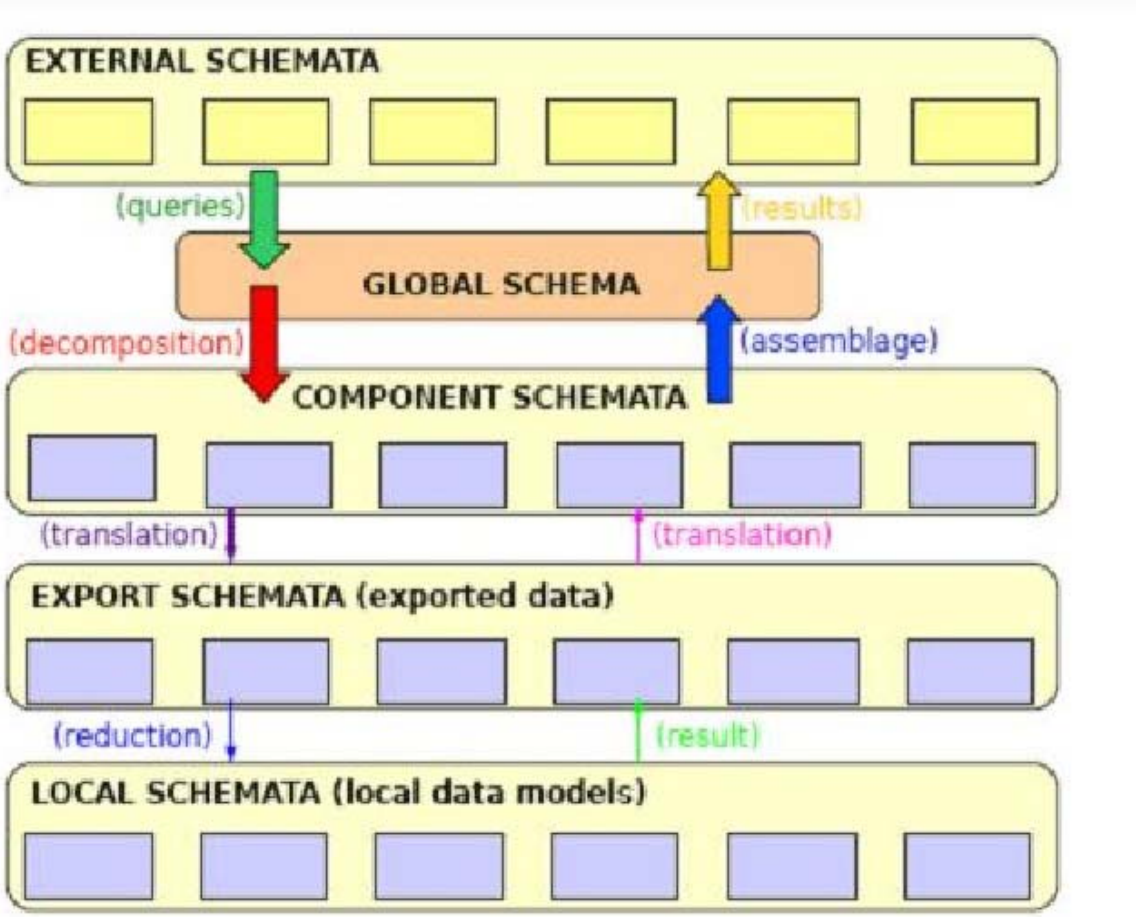
```
<!ELEMENT ListOfStudent (Student*)>
<!ELEMENT Student
(name,s_code,school_name,e_mail,tax_fee)>
<!ELEMENT name (#PCDATA)>
```

Heterogeneous information sources:
Computer_Science source (OO)

```
CS_Person(first_name,last_name)

Professor:CS_Person(belongs_to:Division,rank)

Student:CS_Person(year,takes:set<Course>,rank,e
_mail)

Division(description,address:Location)

Location(city,street,number,country)

Course(course_name,taught_by:Professor)
```

General multidatabase model

(see C. Yu)



Steps

1. Reverse engineering
2. Conceptual schemata integration
3. Choice of the target data model and translation of the global conceptual schema
4. Definition of the language translation
5. Definition of the data views (as usual) according to the chosen paradigm (GAV, LAV,…)

Step 4.

WRAPPERS (translators)

- Convert queries into queries/commands which are understandable for the specific data source
  - they can extend the query possibilities of a data source
- Convert query results from the source's format to a format which is understandable for the application
- We will say more when talking about semi-structured information

The new application context

- A (possibly large) number of data sources
- Time-variant data “form” (e.g. WEB)
- Heterogeneous datasources
- Mobile, transient datasources
- Mobile users
- Different levels of data structure
  - Databases (relational, OO…)
  - Semistructured datasources (XML, HTML, more markups …)
  - Unstructured data (text, multimedia etc…)
- Different terminologies and different operation contexts

Heterogeneous, dynamic systems

In a general setting, we would like to have a uniform, as transparent as possible interface to many, however autonomous and heterogeneous data sources. This interface should:

- Take care of ***finding***, for us, the data sources which are relevant to the issue we are interested in
- ***Interact*** with the single sources
- ***Combine*** results obtained from the single sources

A more dynamic solution: no global schema, but *mediators*

- An interface from the users/applications to the database servers, which only defines communication protocols and formats, does not deal with the abstraction and representation problems existing in today's data and knowledge resources
- The interfaces must take on an active role
- We will refer to the dynamic interface function as mediation. This term includes (Wiederhold):
  - the processing needed to make the interfaces work
  - the knowledge structures that drive the transformations needed to transform data to information
  - any intermediate storage that is needed

Mediators

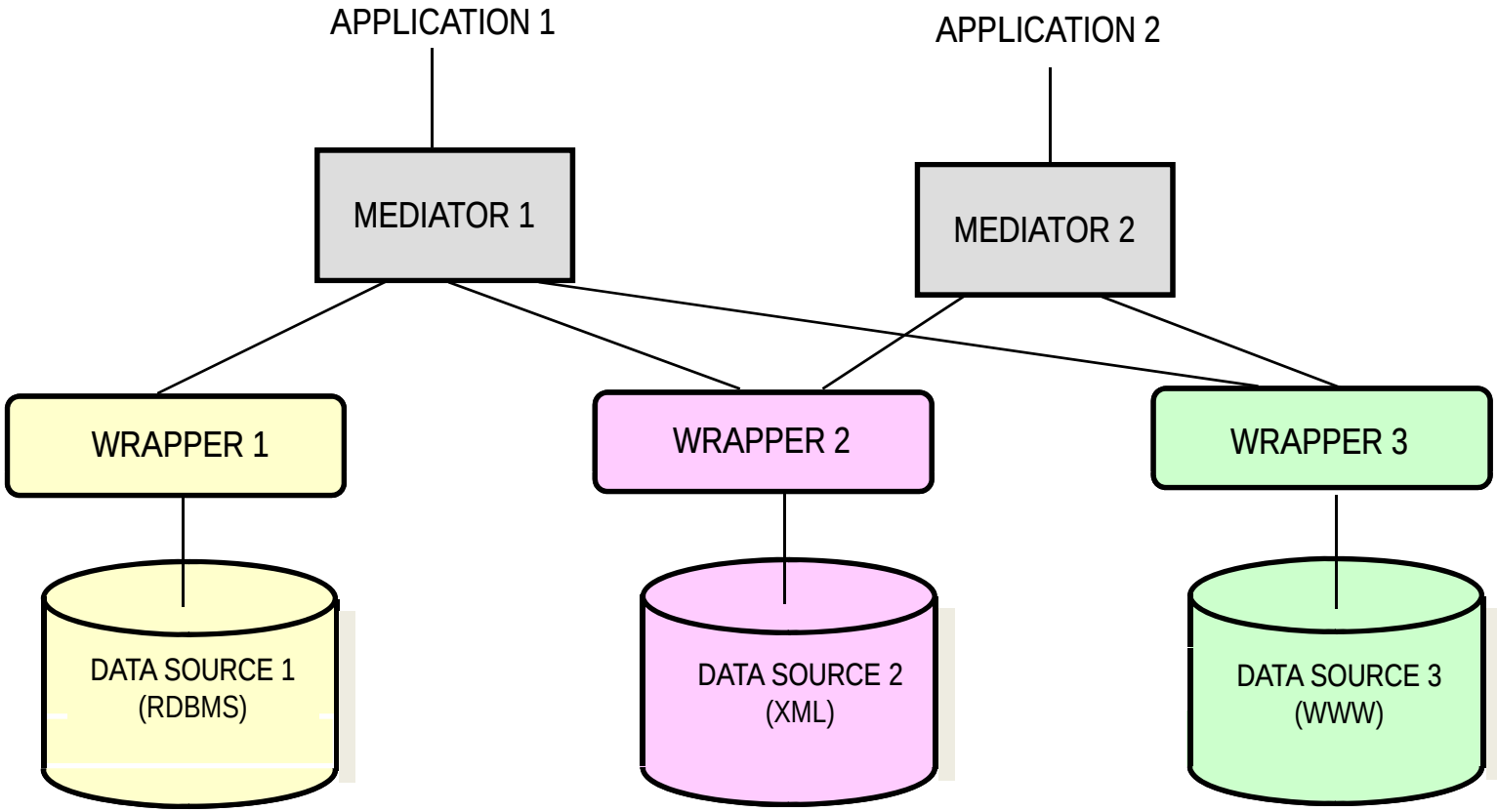
A mediator's main functionality is object fusion:

- group together information about the same real world entity
- remove redundancy among the various data sources
- resolve inconsistencies among the various data sources

Types of mediation functions that have been developed

- Transformation and subsetting of databases using view definitions and object templates
- Methods to access and merge data from multiple databases
- Computations that support abstraction and generalization over underlying data
- Intelligent directories to information bases such as library catalogs, indexing aids and thesaurus structures
- Methods to deal with uncertainty and missing data because of incomplete or mismatched sources

AN EXAMPLE OF ARCHITECTURE WITH MEDIATORS (TSIMMIS)



Mediators

- No unique global schema is required
- Each mediator has its own functioning way
- One mediator may or may not use a global schema
- E.g.: in the Tsimmis project, *dataguide*

P2P data integration

- Several peers
- Each peer with local and external sources
- Queries over one peer
- Answers integrating the peer's own data plus the others

Instance Heterogeneity

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

SSN	NAME	AGE	SALARY
234567891	**Ketty**	**48**	**18k**

SSN	NAME	AGE	SALARY
234567891	**Ketty**	**48**	**25k**

Resolution function

Data 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 → the full salary is the sum of the two
- In general, the correct value may be obtained as a function of the original ones

  (maybe: $0*value_1 + 1*value_2$!!)

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 OJ R2)

Conclusion: integration presents problems along several dimensions

- Source heterogeneity
  - Schemata
  - Models
  - Technology
- Connection failures
- Access limitations
  - Security
  - Privacy
- Final integration model (relationships among sources)
  - Global Schema
  - Mediators
  - Peer-to-peer
  - Data exchange
- Expression of queries and updates
- Integration type: virtual or materialized
- Estraction, cleaning
- Reconciliation
  - possible inconsistencies among data sources

Conclusions

- The integration problem does not lend itself to a unique possible solution
- Approaches we listed are not necessarily mutually exclusive
- The more complex the problem, the less precise we can expect the solution to be
- We have not introduced semistructured data: adding yet more complexity