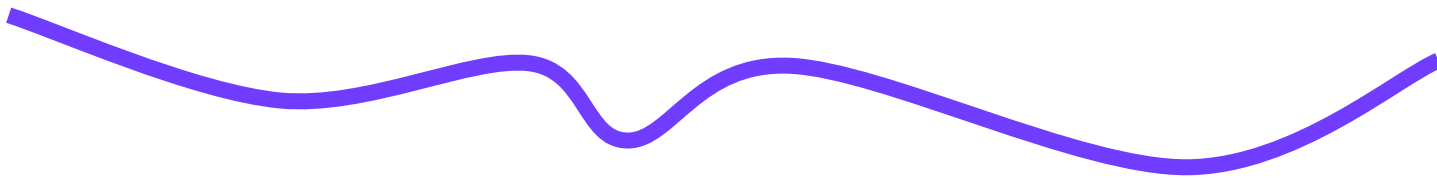




Data Quality & Data Integration

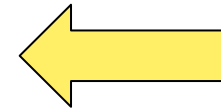
Monica Scannapieco

Department of Computer Engineering
Università di Roma "La Sapienza", Italy



Summary of Previous Lesson

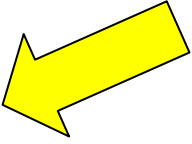
- We have learnt:
 - **what is data quality** in terms of its major dimensions ...
 - ... that are however a subset of larger possible sets that can take into account domain specific features
 - some elements on **how assessing quality of data** in general contexts
 - **data quality is critical in** modern information systems, like **CISs**



In this lesson...

- Specific data quality problems in data integration systems
 - Focus on instance conflicts resolution
- Some techniques solving dq problems in data integration systems

Cooperative Information Systems (CISs) abstracting DISSs

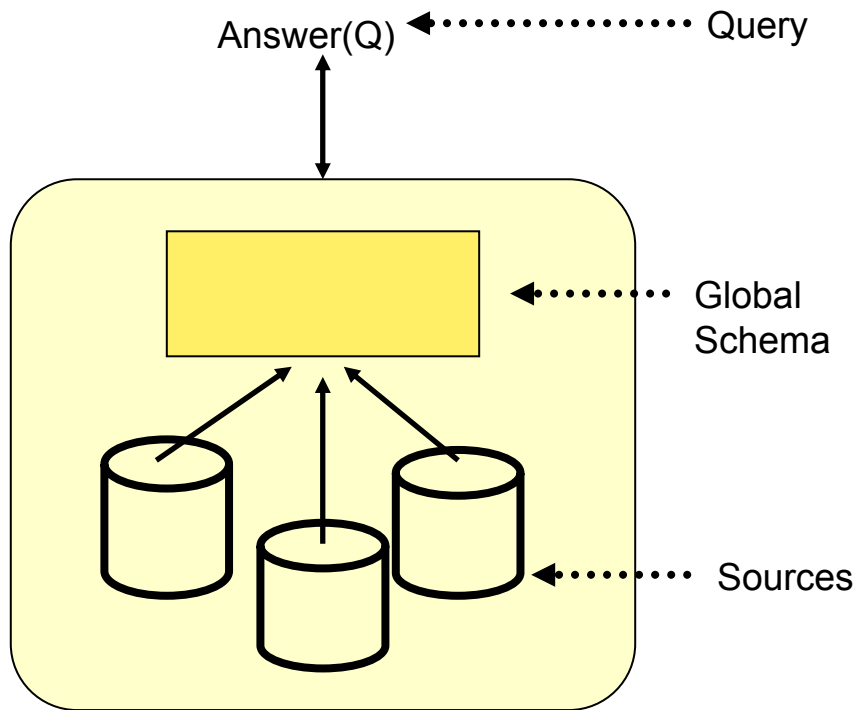
- Different Approaches
 - Data Integration techniques 
 - Agent-based methodologies
 - Process Coordination and Service-based systems

Data Integration

- Data integration is the problem of combining data stored by different sources, and providing the user with a unified view of this data
- Two main approaches to data integration, namely:
 - Materialized data integration where the (unified view of) data is materialized, for instance in a data warehouse;
 - **Virtual data integration** where the unified view is virtual and data resides only at sources

Basics on Data Integration

(Virtual) Data Integration System



- Three elements:
- a global schema
- A source schema, including schemas of all sources
- A mapping between the global schema and the source schema

Cont.

- Sources are **distributed**, **independent** and **heterogeneous** and in general *out of the control* of the data integration system
 - A **mapping** holds the relationships between the sources and the unified view of them
 - When the user **issues** queries, the system carries out the task of **extracting data** from the sources and **reassembling** them into the answer

Approaches to DI

- Two basic approaches have been proposed to specify the mapping
 - **Global-as-View (GAV)** requires that the global schema is expressed in terms of the data sources
 - Easier query processing
 - **Local-as-View (LAV)** requires that each data source is expressed as a view over the global schema
 - New data sources easier integrated
- **GLAV** approach combines the **GAV** and **LAV** approaches and is such that which queries over the sources are put in correspondence with queries over the global schema

Example

- Global View:
 - movie(Title;Year; Director)
 - european(Director)
 - review(Title; Critique)
- Sources:
 - r1(*Title;Year; Director*) since 1960, european directors
 - r2(*Title;Critique*) since 1990
- Query: **Title and critique of movies in 1998**
 - movie(*T; 1998;D*) \wedge review(*T;R*)

Global as View: Example

- Global-as-view: relations in the global view are **views** over the sources
 - $\text{movie}(T; Y; D) \dashrightarrow r1(T; Y; D)$
 - $\text{european}(D) \dashrightarrow r1(T; Y; D)$
 - $\text{review}(T; R) \dashrightarrow r2(T; R)$
- The query
$$\text{mr}(T; R) \dashleftarrow \text{movie}(T; 1998; D) \wedge \text{review}(T; R)$$
is processed by means of unfolding, i.e., by expanding the atoms according to their definitions, until we come up with source relations.
- In this case:
$$r1(T; 1998; D) \wedge r2(T; R)$$

Local as View Example

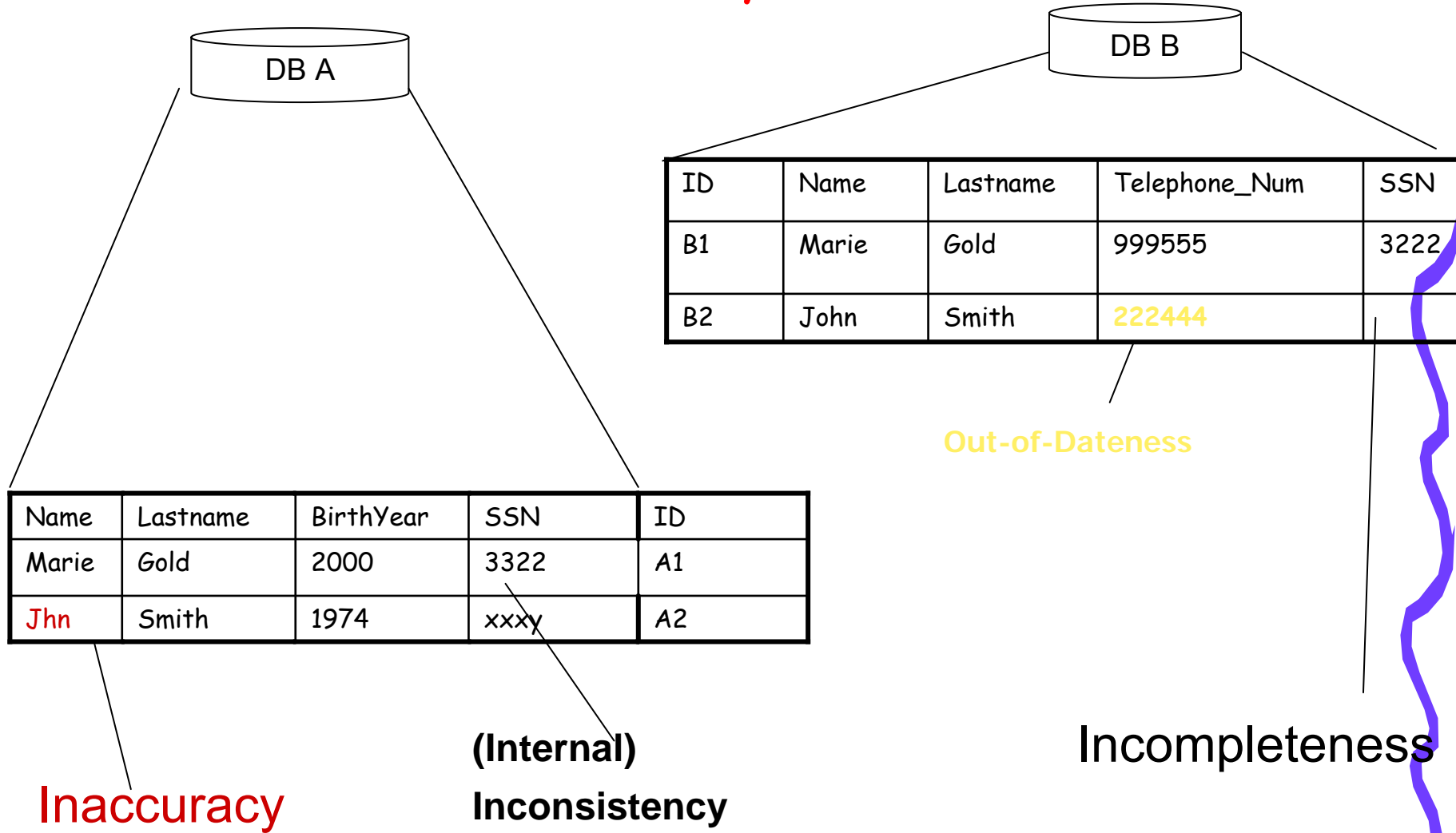
- Local-as-view: relations at the sources are defined as views over the global view
 - $r1(T; Y; D) \leftarrow \text{movie}(T; Y; D) \wedge \text{european}(D) \wedge Y \geq 1960$
 - $r2(T; R) \leftarrow \text{movie}(T; Y; D) \wedge \text{review}(T; R) \wedge Y \geq 1990$
- The query $mr(T; R) \leftarrow \text{movie}(T; 1998; D) \wedge \text{review}(T; R)$ is processed by means of an inference mechanism aiming at re-express the atoms of the global view in terms of atoms at the sources.
- In this case: $r1(T; 1998; D) \wedge r2(T; R)$

Data Quality Issues

Data Replication

- Copies of the same data are stored by different organizations (or within a single organization)
- Problem: data inconsistency!!!

Data Inconsistency



Data Inconsistencies in CISs: Why a problem?

- Organizations do not trust each other if providing inconsistent data → lack of cooperation
- Users of the CISs do not trust organizations providing mutually inconsistent data → lack of data service requests
- Uncontrolled spread of low quality data in the system

The Anatomy of Conflicts in DISs

- Schema level conflicts
- Instance level conflicts

Schema level conflicts - 1

- Longly studied - here a summary
- **Heterogeneity conflicts**, different data models are used
 - e.g. XML and relational
- **Semantic conflicts**, relationship btw model level extensions,
 - e.g. the Person class may have different extensions in different sources (disjoint, partially, overlapping, completely overlapping, including)

Schema level conflicts - 2

- **Description conflicts**, concepts described with different attributes
 - e.g. different formats, different attribute types, different scaling, etc.
- **Structural conflicts**, different design choices within the same model
 - e.g. Address represented in one source as a class and in another as an attribute

Instance level conflicts - 1

- Our focus in the next slides - few research results so far
- On the basis of the model element granularity:
 - attribute conflicts
 - key conflicts (also called entity or tuple conflicts)
 - relationship conflicts

Instance level conflicts - 2

- Given two sources S_1 and S_2 , let A_{1k} and A_{2k} represent the same attribute of a real world object represented as t_1 in S_1 and as t_2 in S_2 .
- **Attribute Conflicts:** An attribute conflict occurs if $t_1.A_{1k} \neq t_2.A_{2k}$
- **Key Conflicts:** Let us suppose that A_{1k} is primary key for t_1 and A_{2k} is primary key for t_2 . A Key conflict occurs iff $t_1.A_{1k} \neq t_2.A_{2k}$ and $t_1.A_{1i} = t_2.A_{2i}$ for all $i \neq k$

Instance level conflicts - 3

- **Relationship Conflict:** Let us suppose that A_{1k} is primary key for t_1 and A_{2k} is primary key for t_2 . Also both S_1 and S_2 are involved in a relationship R with a third source S_3
- A relationship conflict occurs if given t_1 in relationship R with t_{3k} in S_3 , i.e. $t_1 R t_{3k}$, and given t_2 in relation R with t_{3h} in S_3 , i.e. $t_2 R t_{3h}$, then $t_{3k} \neq t_{3h}$.

Instance Level Conflicts - 4

EmployeeID	Name	Surname	Salary	Email
arpa78	John	Smith	2000	smith@abc.it
eugi98	Edward	Monroe	1500	monroe@abc.it
ghjk09	Anthony	Wite	1250	white@abc.it
treg23	Marianne	Collins	1150	collins@abc.it

EmployeeS1

Key
conflict

EmployeeID	Name	Surname	Salary	Email
arpa78	John	Smith	2600	smith@abc.it
eugi98	Edward	Monroe	1500	monroe@abc.it
ghjk09	Anthony	White	1250	white@abc.it
dref43	Marianne	Collins	1150	collins@abc.it

EmployeeS2

Attribute
conflict

Techniques for Instance Conflicts Resolution

- Key conflicts require the application of object identification techniques
 - Record Matching techniques
- Attribute and relationship conflict are solved by Query Time techniques
 - we will see two QT techniques

1st Challenge Conclusions

- Record Matching can be performed to solve inconsistencies by relying onto replication
 - already seen the general idea, no further details here
- Problem: data vary in time, record matching performed in fixed instants, what happens btw two matching instants?

2nd Challenge: Dynamic Inconsistencies Resolution

- **Idea:** let's focus on solving inconsistencies in data that are actually used in the CIS, i.e. **data that are queried!**
- **Solution:** Data integration system that supports query-time inconsistencies resolution

DaQuinCIS

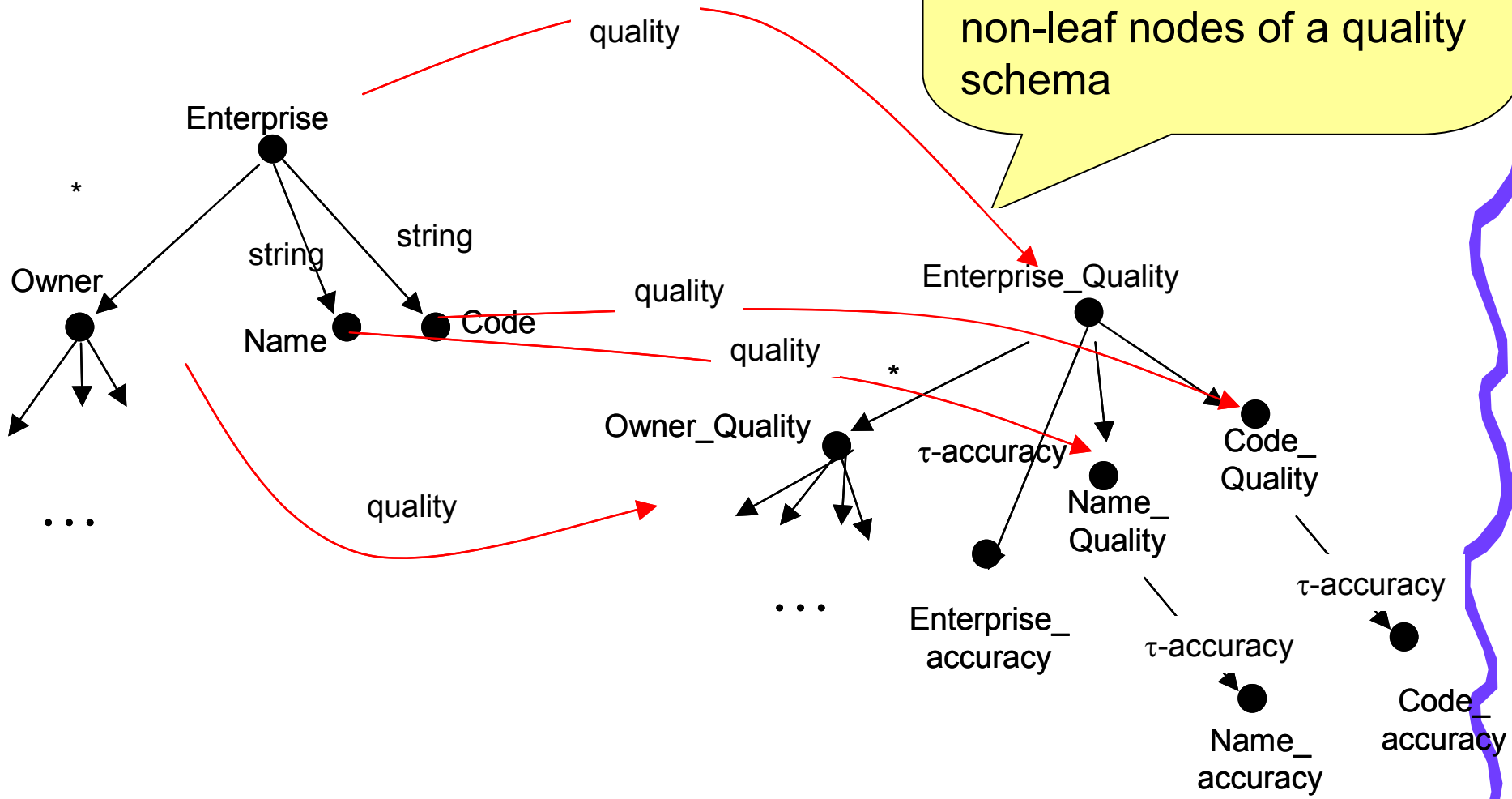
- DaQuinCIS (Data Quality in Cooperative Information Systems) is a
 - System developed at the University of Rome
 - Dealing with query time inconsistencies by solving them on the basis of quality metadata associated to data

D²Q: Data and Data Quality Model

- Graph-based data model, enhancing the semantics of the XML Data Model to represent quality data
- Quality is associated to data in order to:
 - Certify the "*correctness*" (accuracy, consistency, currency) and *completeness* of data
 - Benefits for cooperation
 - Support for instance level reconciliation

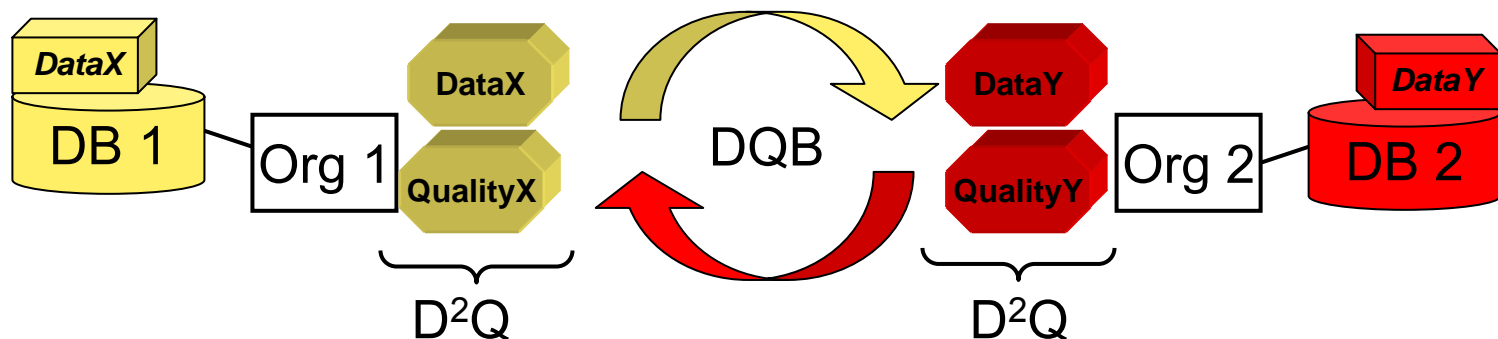
D²Q Schema

Quality Association
Biunivocal functions
among all nodes
of a data schema and all
non-leaf nodes of a quality
schema



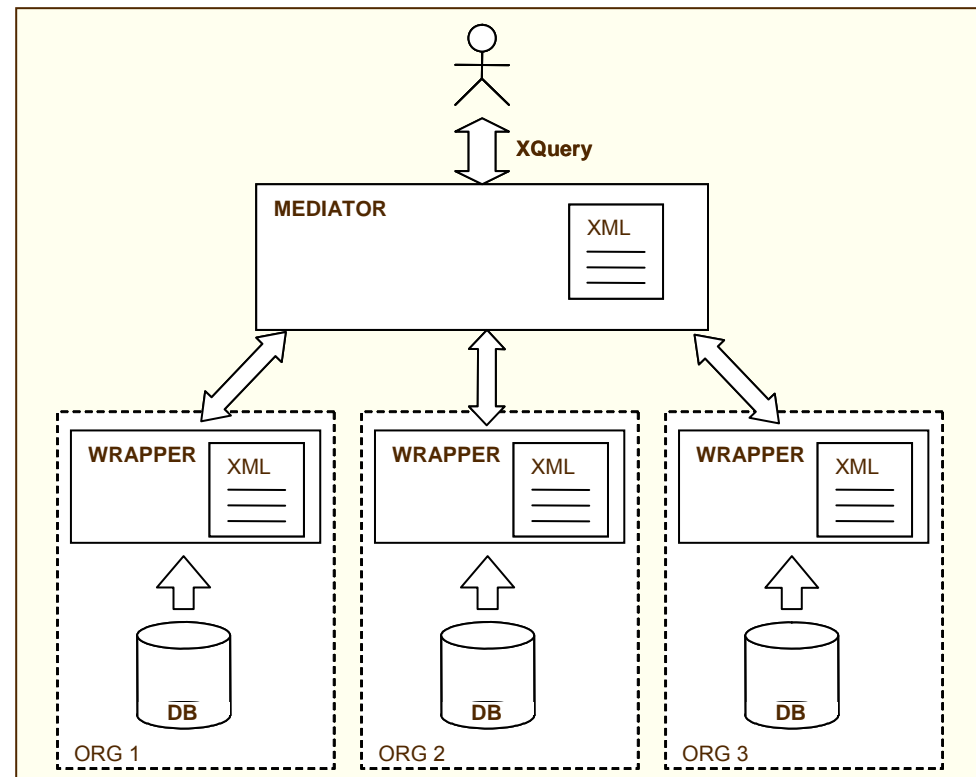
Data Quality Broker ...

- Quality access functionality: accessing data + quality exported according to the D²Q model
- Quality improvement functionality: comparing different copies of the same data available in the whole CIS

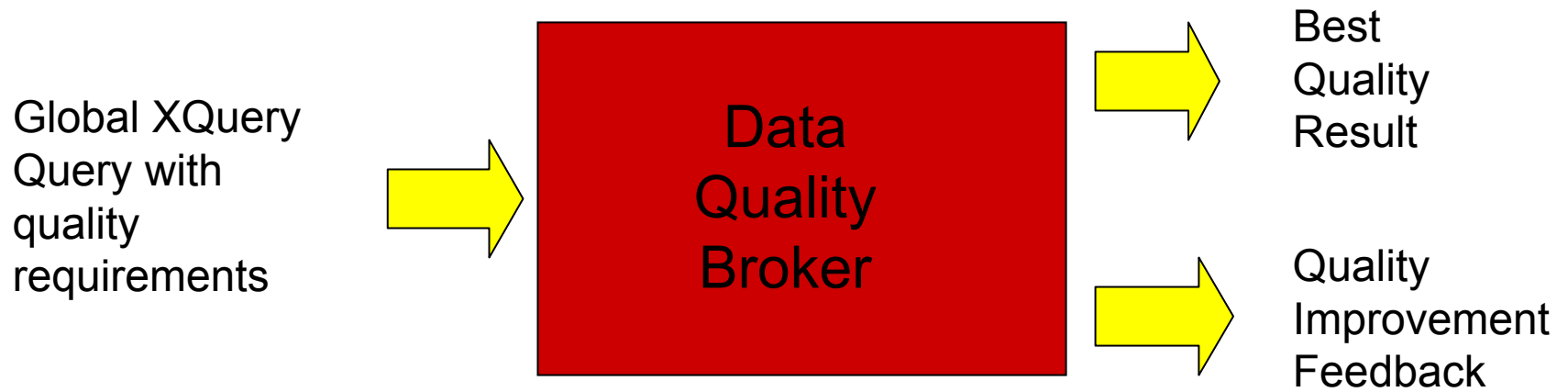


... A Quality-driven Data Integration System

- Wrapper/Mediator or Architecture
- Global and Local views expressed as XML Schemas
- D²Q-compliant
- Global as View (GAV) Mapping



Semantics



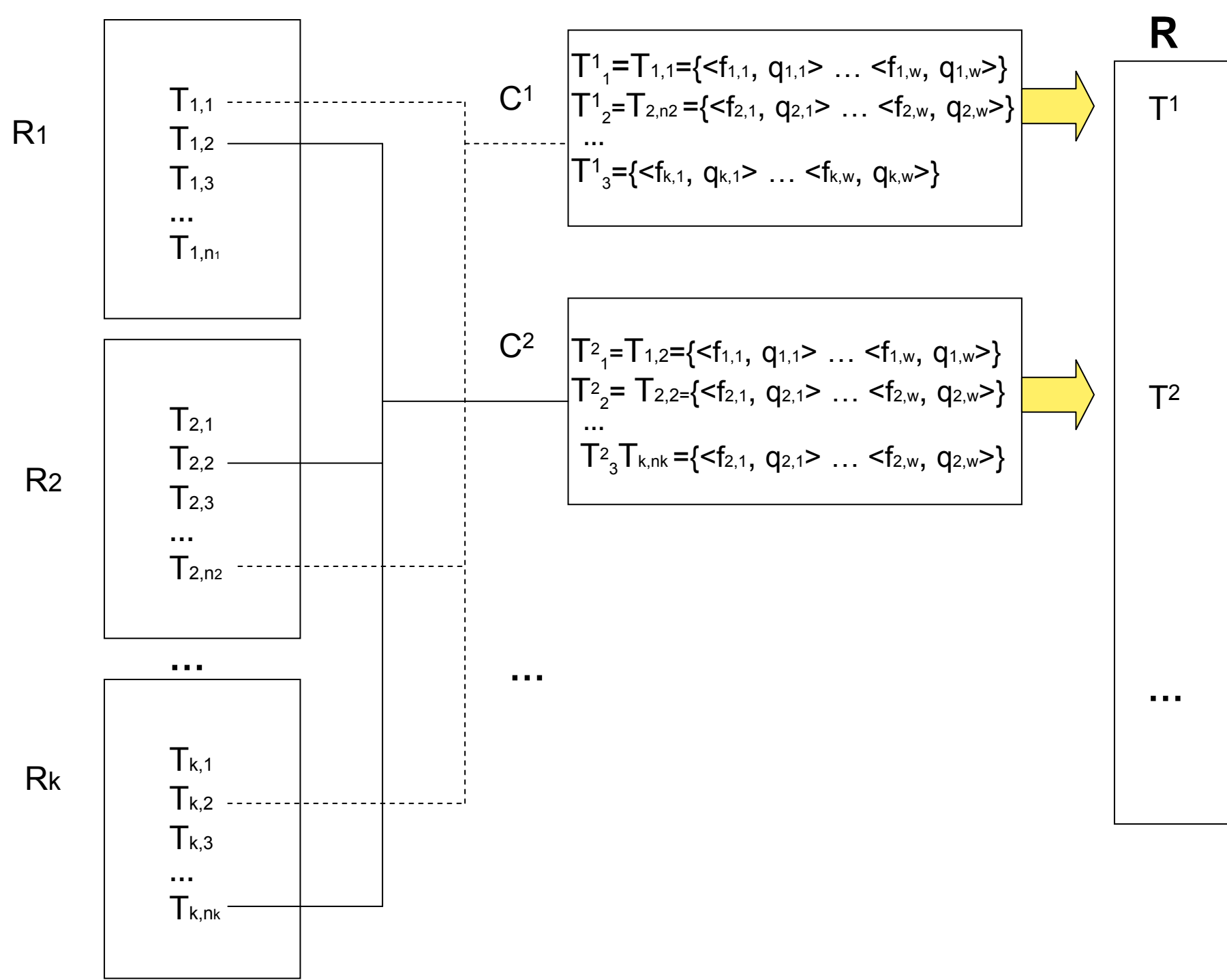
- Best quality copies always available
- Quality Improvement Feature
 - The best quality result is proposed to all data sources that provided lower quality results

Query Processing Steps (1)

1. Given a query Q on a D^2Q global schema
2. Q is unfolded according to a static mapping that retrieves all copies of same data that are available in the CIS
 - Static mapping specified through **path expressions**
 - A path expression allows to locate a concept in a schema
 - XML Schemas are D^2Q compliant

Query Processing Steps (2)

3. The execution of local queries returns a set of results, on which a **run-time matching** is performed
 - The result of this step is the construction of set of clusters composed by tuples referring to same real world objects
 - For each cluster, a best quality representative is either selected or constructed. (**Query Time Inconsistency Resolution**)

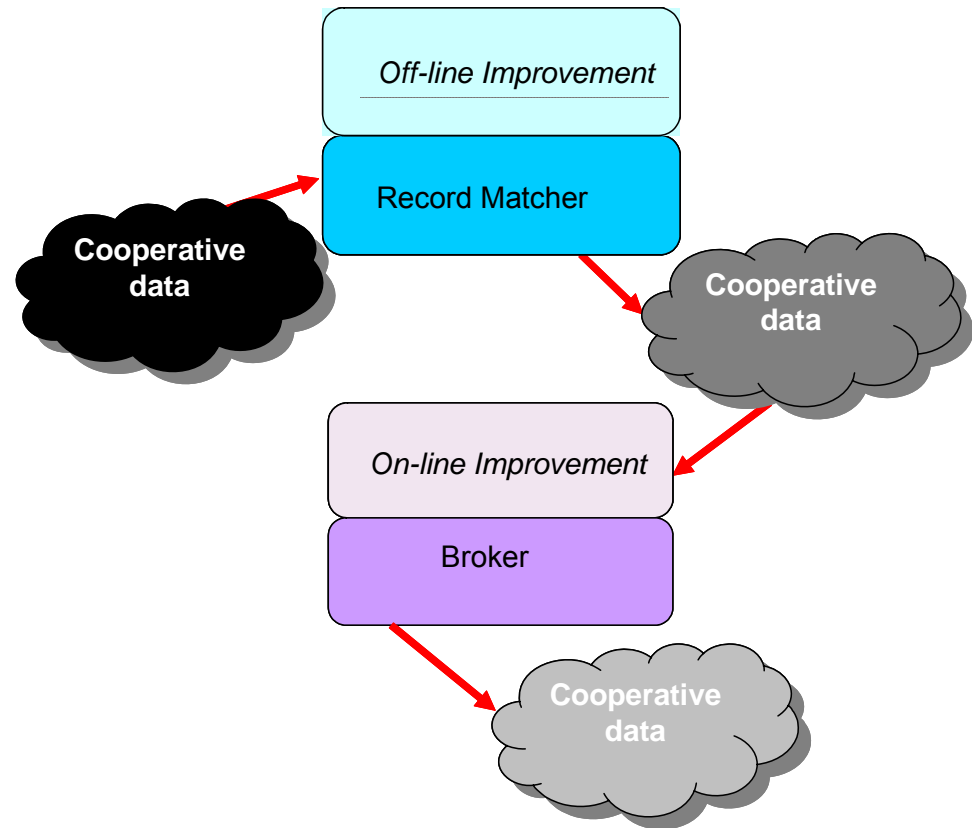


Query Processing Steps (3)

4. The result to be returned is built as follows:
 - (i) if no quality requirement is specified, a **best quality default semantics** is adopted. This means that the result is constructed by selecting/constructing the best quality value
 - (ii) if quality requirements are specified, the result is constructed by checking the satisfiability of the requirements on the whole result
5. The result is also *proposed* to orgs having provided lower quality values - **improvement functionality**

Improvement Strategy

- Off-Line Improvement:
Periodical Record Matching
- On-line Improvement:
Query-time quality improvement



Query Time Consistency Resolution in AURORA

- Aurora is a mediation-based data integration system
- The approach proposes a **Conflict Tolerant (CT) query model** for conflict resolution at a desired degree

Features of the CT Query Model (1)

- Two operators:
 - For attribute conflict resolution, called **Resolve Attribute level Conflict (RAC)**
 - for tuple conflict resolution, called **Resolve Tuple level Conflict (RTC)**

Example: a Global Relation

tupleID	EmployeeID	Name	Surname	Salary	Email
t1	arpa78	John	Smith	2000	smith@abc.it
t2	eugi98	Edward	Monroe	1500	monroe@abc.it
t3	ghjk09	Anthony	Wite	1250	white@abc.it
t4	treg23	Marianne	Collins	1150	collins@abc.it
t5	arpa78	John	Smith	2600	smith@abc.it
t6	eugi98	Edward	Monroe	1500	monroe@abc.it
t7	ghjk09	Anthony	White	1250	white@abc.it
t8	dref43	Marianne	Collins	1150	collins@abc.it

RAC Operator

tupleID	EmployeeID	Name	Surname	Salary	Email
T1	arpa78	John	Smith	2000	smith@abc.it
T2	eugi98	Edward	Monroe	1500	monroe@abc.it
T3	ghjk09	Anthony	White	1250	white@abc.it
T4	treg23	Marianne	Collins	1150	collins@abc.it

RAC(Employee, Salary(MIN), Surname(Longest), EmployeeID(Any))₄₁

RTC Operator

tupleID	EmployeeID	Name	Surname	Salary	Email
T1	Arpa78	John	Smith	2600	smith@abc.it
T2	eugi98	Edward	Monroe	1500	monroe@abc.it
T3	ghjk09	Anthony	Wite	1250	white@abc.it
T4	treg23	Marianne	Collins	1150	collins@abc.it

RTC(Employee,ANY)

Features of the CT Query Model (2)

- Three strategies allowing the user to define the degree of conflicts permitted
- **HighConfidence** allows to specify that on a specific attribute no conflicts is admitted. This means that all values returned by the sources on that attribute must be aligned
- **RandomEvidence** allows to specify that in case of conflicts, a run time function has to select a value to be returned
- **PossibleAtAll** returns all values that correctly answer the query, independently from conflicts

Example

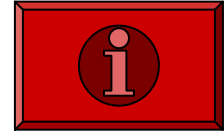
- Q1:

```
SELECT EmployeeID, Name (ANY), Salary[MIN]
FROM Employee
WHERE Salary>1800
with HighConfidence
```

- Q2:

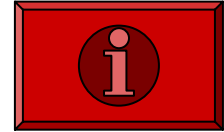
```
SELECT [ANY]EmployeeID, Name, Salary
FROM Employee
WHERE Salary>1800
with RandomEvidence
```

Example



- Both queries select employees with Salary greater than 1800 euros
- If there is a conflict, Q1 selects employees whose Salary's value is greater than 1800 in all sources. Therefore, looking at the global instance the two tuples t1 and t5 are both selected.
- Then applying the resolution function MIN on Salary, the returned tuple will have t1's Salary value.

Example



- Q2 selects a random Salary and if it is greater than 1800, it is returned in the result. Then a tuple resolution function is applied as specified in the selection clause.
- Therefore, looking at the global instance, a random value between t1's salary and t5's salary is selected.

Further Techniques

	Tolerance Strategies	Query Model
SQL-Based Conflict Resolution	NO	SQL
Aurora	High Confidence, RandomEvidence, PossibleAtAll	Ad-hoc Conflict Tolerant Query Model
FusionPlex	No resolution, selective attribute resolution	Extended SQL
DaQuinCIS	NO	Extended XML
FraQL-based Conflict Resolution	NO	Ad-hoc FraQL
OO_{RA}	Thresholds for tolerable and intolerable conflicts	Ad hoc Object Oriented Extension (OO_{RA})

Conclusions

- We have learnt:
 - Data integration systems need to solve instance level inconsistencies
 - Though there are several techniques for schema-level inconsistencies...
 - ...initial results on instance-level inconsistencies
 - Either relying on quality characterization of data, e.g. DaQuinCIS
 - Or relying on declarative specification of conflict resolution functions