Data Integration

Maurizio Lenzerini, Diego Calvanese, Giuseppe De Giacomo

Dipartimento di Informatica e Sistemistica

Università di Roma "La Sapienza"

Slide 0

Dottorato in Ingegneria Informatica
Dicembre, 2000 — Roma

Data integration

Slide 1

Deals with the problem of providing a uniform access to a collection of data stored in **multiple**, **autonomous** and **heterogeneous** sources:

Basic problem in:

- management of distributed information system
- data warehousing
- data re-engineering
- enterprise knowledge management
- querying multiple sources on the web
- e-commerce, e-business, e-government, e-$\cdots$
- integration of data from distributed scientific experiments

Integration vs Cooperation

Slide 2

- Cooperation

  Distributed cooperative applications are **not** necessarily data integration applications

- Integration
  - There **must** be a notion of global view (maybe implicit, not necessarily unique)
  - Distribution is **not** just at the implementation level (as in distributed databases)

Architecture for data integration

Slide 3

At the technological level:

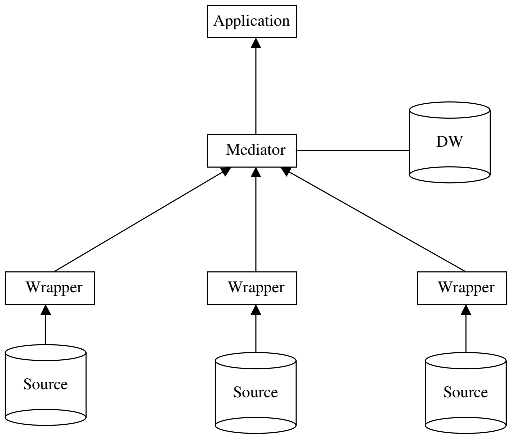
- Tools for cooperation and connectivity (es. ODBC, JDBC, RPC, CORBA, CGI, Servlet, ecc.)

At the conceptual/design level

- queries are expressed over a **global view** (mediated schema, enterprise model, ...)
- **wrappers** access the sources (provide a view in a uniform data model of the data stored in the sources)
- **mediators** combine and reconcile multiple answers coming from wrappers and/or other mediators

Slide 4

Architecture for data integration



Slide 5

Problems in data integration

1. Heterogeinity of sources
2. Relation between the global view and the sources, and between sources
3. Limitations in the mechanisms for accessing the sources
4. Materialized vs virtual integration
5. How to process queries expressed on the global view
6. Data extraction, cleaning and reconciliation
7. How to process updates expressed on the global view

Slide 6

Heterogeinity of the sources

- Sources adopt different models and systems for storing data
- The global view should provide an appropriate abstraction of all the data residing at the sources
- Which language to use for specifying the global view (should it be semistructured?)
- Since the global views should mediate among different representations of overlapping worlds, the language should provide flexible and powerful representation mechanisms

Slide 7

Relation between the global view and the sources

- How do we **discover** relations between the global view and the source, and between sources?
- We assume that such relation should be defined declaratively: this is the **modeling problem**
- Crucial aspect: there must be a glue in the data integration system
- Is the global view defined in terms of the sources (this approach is called **global-as-view**, or **GAV**)?
- Are the sources defined in terms of the global view (this approach is called **local-as-view**, or **LAV**)?
- A mixed approach ?

Limitations in accessing the sources

Slide 8

- Sources are autonomuos
- Typically, we do not have complete freedom in using the sources
- Both in the local-as-view and in the global-as-view approach, sources may presents some limitations on the types of accesses they support
- A typical example is a web source accessible through a form where one of the fields must necessarily be filled in by the user

Materialized vs virtual integration

Slide 9

- Are data moved from the sources and persistently stored in the data integration systems?
- **Virtual integration**: data remain at the sources
- **Materialized integration**: data at the sources are replicated in the integration system (typical of **Data Warehousing**)
- In the materialized approach, since data at sources change, the problem of refreshment is important
- When to propagate updates, how to refresh, how to take into account the data quality requirements on the global view, ...

How to process queries

Slide 10

- Each query is expressed in terms of the global view, and the associated mediator reformulates the query in terms of a set of queries at the sources
- The crucial step is deciding an (efficient query plan, i.e., how to decompose the query into a set of subqueries to the sources, based on the meaning of the sources wrt the global view
- The computed subqueries are then shipped to the sources, and the results are assembled into the final answer (virtual approach), or into the materialized view (materialized approach)
- This process heavily depends on the approach adopted for modeling (global-as-view or local-as-view), and on the presence of limitations in accessing the sources

Data extraction, cleaning and reconciliation

Slide 11

- Many issues arise when considering the integration at the extensional/instance level
- Interpretation and merging of the data provided by the sources: convert/reconcile/combine data coming from different sources
- The complexity of this step is due to several problems:
  - possible mismatches between data referring to the same real world object
  - possible errors in the data stored in the sources
  - possible inconsistencies between values representing the properties of the real world objects in different sources

How to process updates expressed on the global view

Slide 12

- Integration can be **read-only** or **read-and-write**
- In the read-and-write approach, the problem is that updates (not only queries) should be expressed over the global view, and reformulated in terms of updates at the sources
- This the problem of view update
- A crucial aspect is that an update at the global view may be in general translated in several ways in terms of updates at the sources

This course

Slide 13

- The modeling problem (**Maurizio Lenzerini**)
- The query processing problem (**Giuseppe De Giacomo**)
- Semistructured data and data integration (**Diego Calvanese**)
- Selected problems (**Students**)

The modeling problem: fundamental questions

Slide 14

A word of caution

Data modeling (in data integration) is an art. Theoretical frameworks can help humans, not replace them

Questions:

- How do we model the global view (structured vs semistructured)
- How do we model the sources (conceptual and logical level)
- How do we represent possible limitations in accessing the sources
- How do we model the relationship between the global view and the sources (both in the global-as-view and in the local-as-view approach)
- How do we verify the quality of the modeling process

Global-as-view vs local-as-view – Example

Slide 15

Global view: $\mathsf{movie}(\mathit{Title}, \mathit{Year}, \mathit{Director})$
$\mathsf{european}(\mathit{Director})$
$\mathsf{review}(\mathit{Title}, \mathit{Critique})$

Source 1: $\mathsf{r}_1(\mathit{Title}, \mathit{Year}, \mathit{Director})$ since 1960, european directors

Source 2: $\mathsf{r}_2(\mathit{Title}, \mathit{Critique})$ since 1990

Query: Title and critique of movies in 1998

$$\mathsf{mr}(T, R) \leftarrow \mathsf{movie}(T, 1998, D) \wedge \mathsf{review}(T, R)$$

Global-as-view – Example

Slide 16

Global View: $\mathsf{movie}(\mathit{Title}, \mathit{Year}, \mathit{Director})$
$\mathsf{european}(\mathit{Director})$
$\mathsf{review}(\mathit{Title}, \mathit{Critique})$

Global-as-view: relations in the global view are views over the sources

$$\begin{array}{lcl} \mathsf{movie}(T, Y, D) & \leftarrow & \mathsf{r}_1(T, Y, D) \\ \mathsf{european}(D) & \leftarrow & \mathsf{r}_1(T, Y, D) \\ \mathsf{review}(T, R) & \leftarrow & \mathsf{r}_2(T, R) \end{array}$$

The query $\mathsf{mr}(T, R) \leftarrow \mathsf{movie}(T, 1998, D) \wedge \mathsf{review}(T, R)$ 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: $\mathsf{r}_1(T, 1998, D) \wedge \mathsf{r}_2(T, R)$

Local-as-view – Example

Slide 17

Global view: $\mathsf{movie}(\mathit{Title}, \mathit{Year}, \mathit{Director})$
$\mathsf{european}(\mathit{Director})$
$\mathsf{review}(\mathit{Title}, \mathit{Critique})$

Local-as-view: relations at the sources are are defined as views over the global view

$$\begin{array}{lcl} \mathsf{r}_1(T, Y, D) & \leftarrow & \mathsf{movie}(T, Y, D) \wedge \mathsf{european}(D) \wedge Y \geq 1960 \\ \mathsf{r}_2(T, R) & \leftarrow & \mathsf{movie}(T, Y, D) \wedge \mathsf{review}(T, R) \wedge Y \geq 1990 \end{array}$$

The $\mathsf{mr}(T, R) \leftarrow \mathsf{movie}(T, 1998, D) \wedge \mathsf{review}(T, R)$ 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:

$$\mathsf{r}_1(T, 1998, D) \wedge \mathsf{r}_2(T, R)$$

Note that in this case the resulting query is not equivalent to the original query

Global-as-view and local-as-view – Comparison

Slide 18

Local-as-view: (Information Manifold, DWQ, Picsel)

- High modularity and reusability (when a source changes, only its definition is affected)
- Relationships betwen sources can be inferred
- Query processing difficult (**query reformulation** at run time)

Global-as-view: (Carnot, SIMS, Tsimmis, ...)

- Whenever a source changes or a new one is added, the global view needs to be reconsidered
- Query processing typically easier (unfolding)
- Need to understand the relationships between the sources (**query reformulation** at design time)

The modeling problem: formal framework

Slide 19

The integrated database (DB) $\mathcal{DB}$ is simply a set of structures (relations, in the relational model), one for each symbol in an alphabet $\mathcal{A}^{\mathcal{G}}$

- the structure of the global view is specified in the schema language $\mathcal{L}_{\mathcal{G}}$ over $\mathcal{A}^{\mathcal{G}}$
- (a) in GAV each structure in the global view is modeled as a view over the source structures, expressed in the view language $\mathcal{L}_{\mathcal{V}}$ over the alphabet of the source structures $\mathcal{A}^{\mathcal{S}}$

  (b) in LAV each source structure is modeled as a view over the global view, expressed in the view language $\mathcal{L}_{\mathcal{V}}$ over $\mathcal{A}^{\mathcal{G}}$
- queries are issued over the global view, and are expressed in the query language $\mathcal{L}_{\mathcal{Q}}$ over $\mathcal{A}^{\mathcal{G}}$

The modeling problem: formal framework

Slide 20

- The global view $\mathcal{G}$ is specified as a set of constraints in $\mathcal{L}_\mathcal{G}$, and associated to each source structure we have a set (its extension, denote by the function *ext*)
- We have a set $\mathcal{V}$ of views $V_1, \ldots, V_m$, and for each view we have a definition $def(V_i)$ in terms of a query in the language $\mathcal{L}_\mathcal{V}$. For each view, we have a specification $as(V_i)$ of which *assumption* to adopt V_i, i.e., how to interpret $ext(V_i)$ wrt the tuples satisfying V_i
  (a) In GAV, we have a view V_i for each structure in $\mathcal{G}$, with definition $def(V_i)$ expressed over $\mathcal{A}^\mathcal{S}$
  (b) In LAV, we have a view V_i for each source structure, with definition $def(V_i)$ over $\mathcal{A}^\mathcal{G}$
- A query Q is expressed in the language $\mathcal{L}_Q$ over $\mathcal{A}^\mathcal{G}$.

The modeling problem: formal framework

Slide 21

The specification $as(V_i)$ determines how accurate is the extension of the the view with respect to the specification $def(V_i)$

- **Sound Views**: a database $\mathcal{DB}$ is *coherent with the sound view* V_i, if $ext(V_i) \subseteq ans(def(V_i), \mathcal{DB})$
- **Complete Views**: a database $\mathcal{DB}$ is *coherent with the complete view* V_i, if $ext(V_i) \supseteq ans(def(V_i), \mathcal{DB})$
- **Exact Views**: a database $\mathcal{DB}$ is *coherent with the exact view* V_i, if $ext(V_i) = ans(def(V_i), \mathcal{DB})$

The modeling problem: formal framework in GAV

Slide 22

In GAV, we interpret the views as exact. Therefore, given:

- the specification (or, schema) $\mathcal{G}$ of the global view
- the view definitions
- the extensions of the source structures

the semantics is the database (if it exists) over $\mathcal{A}^\mathcal{G}$ coherent with $\mathcal{G}$ and the view definitions

Note: It may happen that no database exists that is coherent with $\mathcal{G}$ and the view definitions

The modeling problem: formal framework in LAV

Slide 23

In LAV, given:

- the specification $\mathcal{G}$ of the global view,
- the set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$, with, for each V_i,
  - its definition $def(V_i)$
  - its extension $ext(V_i)$, which is the extension of the corresponding source structure,
  - the specification $as(V_i)$ of whether it is sound, complete, or exact

the semantics is the set of databases $\mathcal{DB}$ such that:

- $\mathcal{DB}$ satisfies $\mathcal{G}$
- $\mathcal{DB}$ is coherent with $V_1, \ldots, V_m$

The modeling problem: formal framework

Slide 24

Suppose we have the extensions of the source structures. Let $\mathcal{G}$ be the specification (or, schema) of the global view, Q a query of arity n, and $\vec{\mathbf{d}} = (d_1, \ldots, d_n)$ a tuple of constants

- In GAV, the semantics of query processing is: $\vec{\mathbf{d}} \in ans(Q, \mathcal{G}, \mathcal{V})$ iff $(d_1, \ldots, d_n) \in ans(Q, \mathcal{DB})$, for the $\mathcal{DB}$ (if it exists) that satisfies $\mathcal{G}$
- In LAV, the semantics of query processing is the defined as follows. Given the specification $as(V_i)$ (for views $V_1, \ldots, V_m$) of whether the view V_i is sound, complete, or exact, $\vec{\mathbf{d}} \in ans(Q, \mathcal{G}, \mathcal{V})$ iff $(d_1, \ldots, d_n) \in ans(Q, \mathcal{DB})$, for each $\mathcal{DB}$ such that:
  - $\mathcal{DB}$ satisfies $\mathcal{G}$
  - $\mathcal{DB}$ is coherent with $V_1, \ldots, V_m$

The modeling problem: the last frontier

Slide 25

A **more general method** for specifying the relations between the global view and the sources is based on assertions of the form (called generalized interschema assertions):

$$Q_{\mathcal{S}} \subseteq Q_{\mathcal{G}}, \qquad Q_{\mathcal{S}} \supseteq Q_{\mathcal{G}}$$

where $Q_{\mathcal{S}}$ is a query over $\mathcal{A}^{\mathcal{S}}$, and $Q_{\mathcal{G}}$ is a query (of the same arity of $Q_{\mathcal{S}}$) over $\mathcal{A}^{\mathcal{G}}$, and $\subseteq$, $\supseteq$ denote containment between sets

- We obtain LAV if we allow only simple queries in $Q_{\mathcal{S}}$ (sound views are modeled through $\subseteq$, complete views through $\supseteq$)
- We obtain GAV if we allow only assertions of the first form, with simple queries in $Q_{\mathcal{G}}$
- We can associate access limitations to symbols in $\mathcal{A}^{\mathcal{S}}$

Example of generalized interschema assertions

Slide 26

Global view:	$PP(Person, Project)$,	$PF(Project, Field)$
Source 1:	$PT(Person, Topic)$	
Source 2:	$RG(Researcher, Grant)$,	$GP(Grant, Project)$

Assertions:

$$PT(P,T) \subseteq PP(P,Pr) \land PF(Pr,T)$$
$$RG(R,G) \land GP(G,P) \subseteq PP(R,P)$$

What about GAV or LAV ?

The modeling problem: the last frontier

Slide 27

Given

- the specification $\mathcal{G}$ of the global view
- the set $\mathcal{T}$ of generalized interschema assertions
- the extensions for the symbols in $\mathcal{A}^{\mathcal{S}}$ (source structures)

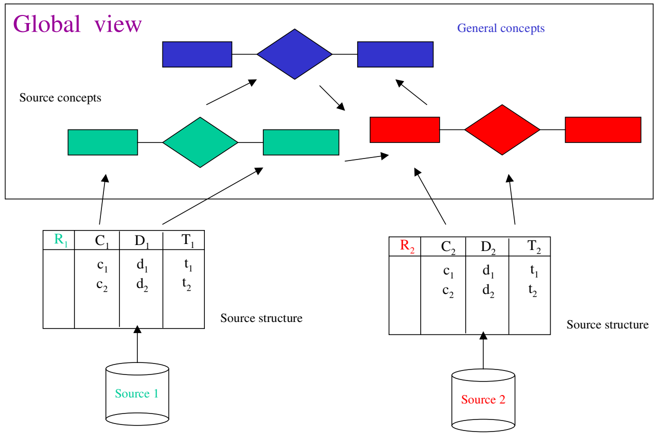
The semantics is the set of databases $\mathcal{DB}$ over $\mathcal{A}^{\mathcal{G}}$ that satisfy $\mathcal{G}$, and are coherent with the generalized interschema assertions $\mathcal{T}$

As usual, $(d_1, \ldots, d_n) \in ans(Q, \mathcal{G}, \mathcal{A})$ iff $(d_1, \ldots, d_n) \in ans(Q, \mathcal{DB})$, for each $\mathcal{DB}$ such that:

- $\mathcal{DB}$ satisfies $\mathcal{G}$
- $\mathcal{DB}$ is coherent with $\mathcal{T}$

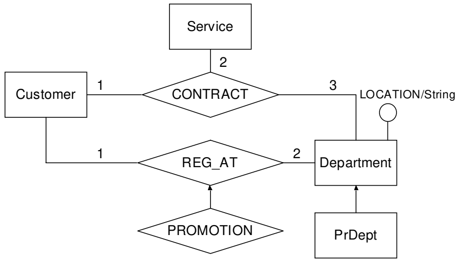
The modeling problem: the DWQ architecture

Slide 28



$\mathcal{DLR}$ for conceptual modeling of global view

Slide 30



$$
\begin{array}{rcl}
\mathsf{CONTRACT} & \sqsubseteq & (\$1:\mathsf{Customer}) \sqcap (\$2:\mathsf{Service}) \sqcap (\$3:\mathsf{Department}) \\
\mathsf{REG\text{-}AT} & \sqsubseteq & (\$1:\mathsf{Customer}) \sqcap (\$2:\mathsf{Department}) \\
\mathsf{PROMOTION} & \sqsubseteq & \mathsf{REG\text{-}AT} \\
\mathsf{LOCATION} & \sqsubseteq & (\$1:\mathsf{Dept}) \sqcap (\$2:\mathsf{String}) \\
\mathsf{Department} & \sqsubseteq & \exists^{\leqslant 1}[\$1]\mathsf{LOCATION} \sqcap \neg\mathsf{Service} \\
\mathsf{Customer} & \sqsubseteq & \neg\mathsf{Service} \sqcap \neg\mathsf{Department} \sqcap \exists^{\geqslant 1}[\$1]\mathsf{CONTRACT} \\
\mathsf{PrDept} & \sqsubseteq & \mathsf{Department} \sqcap \exists^{\geqslant 1}[\$2]\mathsf{PROMOTION}
\end{array}
$$

$\mathcal{L}_{\mathcal{G}}$ in DWQ: The description logic $\mathcal{DLR}$

Slide 29

- **relationships:** interpreted as **sets of tuples** of a given arity

$$R \ ::= \ \top_n \ | \ P \ | \ \neg R \ | \ R_1 \sqcap R_2 \ | \ (\$i/n:C)$$

- **concepts:** interpreted as **sets of objects**

$$C \ ::= \ \top_1 \ | \ A \ | \ \neg C \ | \ C_1 \sqcap C_2 \ | \ \exists^{\leqslant k}[\$i]R$$

- $\mathcal{DLR}$ **conceptual model** is a set of **assertions** on relationships and concepts

$$
\begin{array}{rclcrcl}
R & \sqsubseteq & R' & \qquad & R & \not\sqsubseteq & R' \\
C & \sqsubseteq & C' & \qquad & C & \not\sqsubseteq & C'
\end{array}
$$

$\Longrightarrow$ $\mathcal{DLR}$ **is a fragment of FOL** (formal first order semantics)

Encoding conceptual data models in $\mathcal{DLR}$

Slide 31

$\mathcal{DLR}$ **can encode**

- object-oriented data models
- semantic data models
- frame-based ontology languages

Correspondences between models of $\mathcal{DLR}$ and legal database states of the schemas can be proved formally

Slide 32

Expressing constraints in $\mathcal{DLR}$

- arbitrary **boolean constructs**
- **unary inclusion** dependencies (e.g., referential integrity)
  $\exists[\$1]\mathsf{REG\text{-}AT} \sqsubseteq \exists[\$3]\mathsf{CONTRACT}$
- special forms of **typed inclusion** dependencies
  $\mathsf{REG\text{-}AT} \sqcap (\$1 : \mathsf{BigCustomer}) \sqsubseteq \mathsf{PROMOTION}$
- **existence** and **exclusion** dependencies (by means of $\exists$ and $\neg$)
- **unary functional** dependencies
  $\mathsf{Customer} \sqsubseteq \exists^{\leqslant 1}[\$1]\mathsf{REG\text{-}AT}$
- **view definitions**
  $P \sqsubseteq R$, $R \sqsubseteq P$ defines the view P in terms of the complex expression R

Slide 33

Extensions of $\mathcal{DLR}$

- $\mathcal{DLR}_{reg}$: regular expressions (recursive views) – see [Calvanese, De Giacomo, Lenzerini '98]
- $\mathcal{DLR}_{\mu}$: fixpoints (XML DTDs, inductive structures) – see [Calvanese, De Giacomo, Lenzerini '99]
- $\mathcal{TDLR}$: temporal constructs to model temporal databases – see [Artale, Franconi '99]
- $\mathcal{DLR}_{key}$: key constraints – see [Calvanese, De Giacomo, Lenzerini '00]

Slide 34

Modeling the global view

At the conceptual level, the global view contains both general concepts, and local concepts. Relationships involving these concepts (both between local and general, and between local) can be specified by means of **interschema assertions**:

- **extensional** interschema assertions

  $R_i \sqsubseteq_{ext} R'_j \quad R_i \not\sqsubseteq_{ext} R'_j \quad C_i \sqsubseteq_{ext} C'_j \quad C_i \not\sqsubseteq_{ext} C'_j$

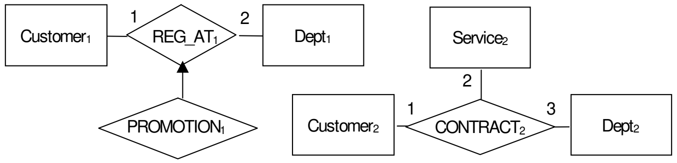
  express relationships between extensions of relationships and concepts
- **intensional** interschema assertions

  $R_i \sqsubseteq_{int} R'_j \quad R_i \not\sqsubseteq_{int} R'_j \quad C_i \sqsubseteq_{int} C'_j \quad C_i \not\sqsubseteq_{int} C'_j$

  express conceptual relationships concerning the meaning of the expressions

Slide 35

Interschema assertions in $\mathcal{DLR}$ – Example



$$
\begin{array}{rcl}
\mathsf{Dept}_1 & \equiv_{ext} & \mathsf{PrDept}_0 \\
\mathsf{REG\text{-}AT}_1 & \sqsubseteq_{ext} & \mathsf{REG\text{-}AT}_0 \\
\mathsf{Customer}_1 & \equiv_{ext} & \mathsf{Customer}_0 \sqcap \exists[\$1](\mathsf{REG\text{-}AT}_0 \sqcap (\$2 : \mathsf{PrDept}_0)) \\
\mathsf{Customer}_0 \sqcap \exists[\$1]\mathsf{CONTRACT}_0 & \sqsubseteq_{ext} & \exists[\$1]\mathsf{PROMOTION}_1 \\
\mathsf{Customer}_2 & \sqsubseteq_{ext} & \mathsf{Customer}_0 \sqcap \exists[\$1]\mathsf{CONTRACT}_0 \\
\mathsf{Dept}_2 & \sqsubseteq_{ext} & \mathsf{Dept}_0 \\
\mathsf{Service}_2 & \equiv_{ext} & \mathsf{Service}_0 \\
\mathsf{Customer}_1 & \equiv_{int} & \mathsf{Customer}_2 \\
\mathsf{Dept}_1 & \equiv_{int} & \mathsf{Dept}_2
\end{array}
$$

LAV and GAV conceptually reconciled

Slide 36

- Concepts and relations at the sources are first-order citizens
- A concept (or a relation) in the global view may be equivalent to a combination of concepts (relations) in the sources
- A concept (or a relation) in the global view may be even a subset of a corresponding concept (relation) in one source
- The interschema assertions constitute a form of sophisticated ontology (or, network of inter-connected ontologies) and meta-data repository
- **Reasoning** on interschema assertions provide support to the designer

Reasoning on the global view

Slide 37

Reasoning technique based on the fact that **logical implication in $\mathcal{DLR}$ is EXPTIME-complete**

$\Longrightarrow$ We can check

- Consistency of entities and relationships
- Subsumption of entities and relationships
- Completeness of interschema assertion
- Correctness and completeness of data in the sources
- Redundancy, e.g., equivalence of entities and relationships

Taking into account the logical level

Slide 38

Data at the sources (and, in the case of materialization, in the global view) are stored in logical structures (e.g. relational tables)

To take into account such logical level, we introduce the notion of $\mathcal{DLR}$ query

A $\mathcal{DLR}$ query is simply a query whose atomes are concepts and relations in $\mathcal{DLR}$

Queries in $\mathcal{DLR}$ – Examples

Slide 39

- Set of tuples (c, d, l) s.t. c is a customer that has a contract for a service with department d located at l:

  $q_1(c, d, l) \ \leftarrow \ \mathsf{Customer}_0(c) \wedge \mathsf{CONTRACT}_0(c, s, d) \wedge \mathsf{LOCATION}_1(d, l)$

- Set of tuples (c, d, l) s.t. c is a customer registered at a public relations department and that has a contract for a service with department d located at l:

  $$q_2(c, d, l) \ \leftarrow \ \mathsf{Customer}_0(c) \wedge \mathsf{CONTRACT}_0(c, s, d) \wedge \mathsf{LOCATION}_1(d, l) \wedge \mathsf{REG\text{-}AT}_0(c, p) \wedge \mathsf{PrDept}_0(p)$$

GAV: Specification of the global view

Slide 40

We specify the content of each global view structure q by means of a view over $\mathcal{A}^{\mathcal{S}}$ of the form:

$$q(\vec{\mathbf{x}}) \leftarrow conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

where

- q is a global view structure whose arity is the arity of $\vec{\mathbf{x}}$
- each $conj_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of **atoms** (with variables $\vec{\mathbf{x}}$, $\vec{\mathbf{y}}_i$)
- each **atom** refers to a source structure

Query answering typically easy (unfolding), but

Problem 1: How do we take the constraints of $\mathcal{G}$ into account?

Problem 2: How do we take source access limitations into account?

LAV: Specification of the sources

Slide 41

We specify the content of each source structure q by means of a view over $\mathcal{A}^{\mathcal{G}}$ of the form:

$$q(\vec{\mathbf{x}}) \leftarrow conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

where

- q is a source structure whose arity is the arity of $\vec{\mathbf{x}}$
- each $conj_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of **atoms** (with variables $\vec{\mathbf{x}}$, $\vec{\mathbf{y}}_i$)
- each **atom** refers to a global view structure

Query answering typically very complex, and

Problem 1: How do we take the constraints of $\mathcal{G}$ into account?

Problem 2: How do we take source access limitations into account?

Limitations in accessing the sources

Slide 42

We use a simple notation for denoting a source structure (relations) with access limitations

To model the situation where column X_i must be bound when the accessing relation R, we write: $R(X_1, \ldots, \underline{X_i}, \ldots)$

$$q_2(c, \underline{d}, l) \leftarrow \mathsf{Customer}(c) \wedge \mathsf{CONTRACT}(c, s, d) \wedge \mathsf{LOCATION}(d, l) \wedge \mathsf{REG\text{-}AT}(c, p) \wedge \mathsf{PrDept}(p)$$

GAV and access limitations: Query processing

Slide 43

Sources: $\mathsf{r}_1(\underline{Tit}, Cd)$, $\mathsf{r}_2(Tit, \underline{Cd})$, $\mathsf{r}_3(\underline{Cd}, Aut, P)$, $\mathsf{r}_4(Cd, \underline{A}, P)$

Global view:

$$\mathsf{In}(Tit, Cd) \leftarrow \mathsf{r}_1(Tit, Cd) \vee \mathsf{r}_2(Tit, Cd)$$

$$\mathsf{Cd}(Cd, Aut, P) \leftarrow \mathsf{r}_3(Cd, Aut, P) \vee \mathsf{r}_4(Cd, Aut, P)$$

Query: $q(P) \leftarrow \mathsf{In}(t_1, Cd) \wedge \mathsf{Cd}(Cd, Aut, P)$

Unfolding:

$$\begin{aligned} q(P) \leftarrow\ & (\mathsf{r}_1(t_1, Cd) \wedge \mathsf{r}_3(Cd, A, P)) \vee \\ & (\mathsf{r}_1(t_1, Cd) \wedge \mathsf{r}_4(Cd, A, P)) \vee \\ & (\mathsf{r}_2(t_1, Cd) \wedge \mathsf{r}_3(Cd, A, P)) \vee \\ & (\mathsf{r}_2(t_1, Cd) \wedge \mathsf{r}_4(Cd, A, P)) \end{aligned}$$

Better Solution:

$$q_1(A) \leftarrow (\mathsf{r}_1(t_1, Cd) \wedge \mathsf{r}_3(Cd, A, P))$$

$$q(P) \leftarrow (\mathsf{r}_1(t_1, Cd) \wedge \mathsf{r}_3(Cd, A, P)) \vee (q_1(A) \wedge \mathsf{r}_4(Cd, A, P))$$

GAV and access limitations: Query processing

Slide 44

The example shows that unfolding is not sufficient in the presence of access limitations

Why ?

Because one access may in principle allow to access a new constant, that may in turn be used in a bound column for another access

We can conclude that we may need recursion to obtain the maximal set of answer

Computing all obtainable answers

Slide 45

```
query(P)           :- global_in(t,C), global_cap(C,A,P)
global_in(T,C)     :- r1(T,C)
global_in(T,C)     :- r2(T,C)
global_cap(C,A,P) :- r3(C,A,P)
global_cap(C,A,P) :- r4(C,A,P)
r1(S,C)            :- song(S), source_r1(S,C)
r2(S,C)            :- cd(C), source_r2(S,C)
r3(C,A,P)          :- cd(C), source_r3(C,A,P)
r4(C,A,P)          :- author(A), source_r4(C,A,P)
cd(C)              :- song(S), source_r1(S,C)
cd(C)              :- author(A), source_r4(C,A,P)
song(S)            :- cd(C), source_r2(S,C)
author(A)          :- cd(C), source_r3(C,A,P)
price(P)           :- author(A), source_r4(C,A,P)
price(P)           :- cd(C), source_r3(C,A,P)
song(t)            :-
```


Reasoning about the integration system specification

Slide 46

Both in LAV and in GAV, there is the need of comparing queries

Query containment is the fundamental reasoning task: Given a global view $\mathcal{G}$ and two queries

$$q_1(\vec{\mathbf{x}}) \leftarrow body_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1)$$
$$q_2(\vec{\mathbf{x}}) \leftarrow body_2(\vec{\mathbf{x}}, \vec{\mathbf{y}}_2)$$

check whether $\mathcal{G} \models q_1 \subseteq q_2$, i.e., $ans(q_1, \mathcal{DB}) \subseteq ans(q_2, \mathcal{DB})$ for every $\mathcal{DB}$ that satisfies $\mathcal{G}$

Query satisfiability is a special case: verify whether q is interpreted as the empty set in every $\mathcal{DB}$ that satisfies $\mathcal{G}$ (e.g., $u(\vec{\mathbf{x}}) \leftarrow P(\vec{\mathbf{x}}) \wedge \neg P(\vec{\mathbf{x}})$ is unsatisfiable)

Query equivalence is another special case: q_1 and q_2 are equivalent if $\mathcal{G} \models q_1 \subseteq q_2$ and $\mathcal{G} \models q_2 \subseteq q_1$

Query containment at work

Slide 47

In GAV:

- Two equivalent structures in the global view can be merged
- Unsatisfiable queries can be detected
- Minimize redundancies and incompleteness in the global view
- Check whether the queries associated to the concepts in the gloabl views are coherent with the constraints in the global view

In LAV:

- Subsumption between source structures can be detected
- Equivalence between source structures can be detected
- Redundant source accesses can be minimized
- **Crucial in query processing**

The problem of query containment

Slide 48

Without limitations on the query language, query containment is undecidable, since it subsumes the problem of checking whether

$$\models \alpha \supset \beta$$

where α, β are first-order formulas

The basic decidability result concerns conjunctive queries:

- algorithm based on the notion of containment mapping
- NP-complete problem

The algorithm for conjunctive query containment

Slide 49

In checking $Q_1 \subseteq Q_2$, a mapping σ from the variables of Q_2 to the variables of Q_1 is called a **containment mapping** if for each $R(x_1, \ldots, x_n)$ in Q_2, $R(\sigma(x_1), \ldots, \sigma(x_n))$ is in Q_1

Examples:

- yes: $R(X,Y), Q(Y,Z), Q(X,W) \quad \subseteq \quad Q(T,S), R(U,T)$
- no: $R(X,Y), Q(Y,Z), Q(X,W) \quad \subseteq \quad Q(T,S), R(U,T)$
- yes: $R(X,Y), Q(Y,Z), Q(X,W) \quad \subseteq \quad Q(T,S), R(U,T)$

NP algorithm: Guess a mapping (polynomial size), and check (polynomial time) whether it is a containment mapping

Another algorithm for conjunctive query containment

Slide 50

Build a canonical database by freezing the variables in Q_1, and evaluate Q_2 on this database

NP algorithm: Guess a binding from the variables of Q_2 to the objects in the database (polynomial size), and check (polynomial time) whether the head of Q_2 is derived

If the evaluation fails, the canonical database is a counterexample database, that shows that containment does not hold

Complexity of conjunctive query containment

Slide 51

Clique is NP-complete: given a graph (V, E) and a positive integer $J \leq |V|$, is there a clique of size J or more in the graph?

$$edge(Y_1, Y_2), \ldots, \quad \subseteq \quad edge(X_1, X_2), edge(X_2, X_3), edge(X_1, X_3), \ldots$$

if and only if the graph has a clique. Indeed, if the containment does not hold, no matter how we chose to assign the Y's (the nodes of the graph) to the X's, they do not form a clique

$\Longrightarrow$ Conjunctive query containment is **NP-hard**, and therefore NP-complete

Beyond conjunctive queries: union

Slide 52

$$Q_1 \ \vee\ Q_2 \ \vee\ \ldots \quad \subseteq \quad R_1 \ \vee\ R_2 \ \vee\ \ldots$$

iff for all i, there exists a j such that

$$Q_i \ \subseteq\ R_j$$

$\Longrightarrow$ Containment of unions of conjunctive queries is in NP, and therefore **NP-complete**

Beyond conjunctive queries: comparison predicates

Slide 53

Containment mappings aren't enough

$$\begin{aligned} Q_1(X,Y) &\leftarrow e(X,Y) \wedge r(U,V) \wedge r(V,U) \\ Q_2(X,Y) &\leftarrow e(X,Y) \wedge r(U,V) \wedge U \leq V \end{aligned}$$

$Q_1 \ \subseteq\ Q_2$, but there is no containment mapping

- we need to consider a set of canonical databases
- in each canonical database, we try an ordering on the objects in the database

The containment holds if a containment mapping exists for all canonical databases. It follows that containment of unions of conjunctive queries is Π_2^p-complete

Beyond conjunctive queries: negation

Slide 54

Containment mappings aren't enough

$$\begin{aligned} Q_1(X,Z) &\leftarrow e(X,Y) \wedge e(Y,Z) \wedge \neg\, e(X,W) \\ Q_2(X,Y) &\leftarrow e(X,Y) \wedge e(Y,Z) \wedge \neg\, e(X,Z) \end{aligned}$$

$Q_1 \ \not\subseteq\ Q_2$, but the only counterexample database is the one where Y and Z denote the same object

- we need to consider all possible partitions of the variables of Q_1
- for each partition, we identify the variables in the same equivalence class, and we evaluate Q_2

The containment holds if for each partition Q_2 is not empty. It follows that containment of unions of conjunctive queries is Π_2^p-complete

Query containment under constraints

Slide 55

Traditionally, to check query containment, only the structure of the queries needs to be taken into account

$$\{(x,z) \mid \mathsf{red}(x,y) \wedge \mathsf{green}(y,z)\} \ \subseteq\ \{(x,z) \mid \mathsf{red}(x,y) \wedge \mathsf{green}(y',z)\}$$

In $\mathcal{DLR}$, query containment needs to be verified in all database states **that satisfy the constraints in the conceptual model**

$$\begin{aligned} \mathcal{G} \models \{(x,z) \mid \mathsf{red}(x,y) \wedge \mathsf{blue}(y,z) \wedge \mathsf{yellow}(y,z)\} \ &\subseteq \\ \{(x,z) \mid \mathsf{red}(x,y) \wedge \mathsf{green}(y',z)\} & \end{aligned}$$

where $\quad \mathcal{G} \ \supseteq\ \{\ \mathsf{blue} \sqcap \mathsf{yellow} \ \sqsubseteq\ \mathsf{green}\ \}$

Example of query containment in $\mathcal{DLR}$

Slide 56

$$q_1(c,d,l) \leftarrow \mathsf{Customer}_0(c) \wedge \mathsf{CONTRACT}_0(c,s,d) \wedge \mathsf{LOCATION}_1(d,l)$$

$$q_2(c,d,l) \leftarrow \mathsf{Customer}_0(c) \wedge \mathsf{CONTRACT}_0(c,s,d) \wedge \mathsf{LOCATION}_1(d,l) \wedge \mathsf{REG\text{-}AT}_0(c,p) \wedge \mathsf{PrDept}_0(p)$$

$\mathcal{G} \models q_1 \subseteq q_2$ because $\mathcal{G}$ implies

$$\mathsf{Customer}_0 \sqcap \exists[\$1]\mathsf{CONTRACT}_0 \sqsubseteq \exists[\$1]\mathsf{PROMOTION}_1$$

$$\mathsf{PROMOTION}_1 \sqsubseteq \mathsf{REG\text{-}AT}_1 \sqcap (\$2 : \mathsf{Dept}_1) \sqsubseteq \mathsf{REG\text{-}AT}_0 \sqcap (\$2 : \mathsf{PrDept}_0)$$

Reasoning on queries under $\mathcal{DLR}$ constraints

Slide 57

Deciding whether $\mathcal{G} \models q_1 \subseteq q_2$ can be done in time $O(2^{p(|\mathcal{G}|\cdot \ell_1^{\ell_2})})$

where

- $|\mathcal{G}|$ is the size of $\mathcal{G}$
- ℓ_1 is the sum of the number of variables in q_1 and the number of constants in q_1 and q_2
- ℓ_2 is the number of existentially quantified variables in q_2 that appear in a semi-cycle of the tuple-graph for q_2

The reconciliation problem

Slide 58

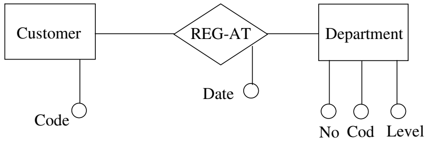
Sources are **autonomous** and **heterogeneous**:

- different data models $\Longrightarrow$ **wrappers**
- schematic differences
- differences in representing values
- differences in representing objects
- inconsistencies in the data

When answering queries posed on the global view, we must consider these problems
$\Longrightarrow$ specification of **reconciling mediators**

The reconciliation problem: Example

Slide 59



R_1	C_1	D_1	T_1
	c_1	d_1	t_1
	c_2	d_2	t_2

R_2	D_2	L_2
	d'_2	l_3
	d_4	l_4
	d_5	l_5

$$r_1(x,t,z) \leftarrow \mathtt{Code}(v,x) \wedge \mathsf{No}(y,t) \wedge \mathsf{REG\text{-}AT}(v,y,z)$$

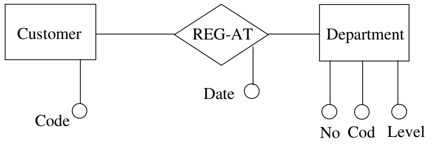
$$r_2(y,w) \leftarrow \mathtt{Cod}(x,y) \wedge \mathsf{Level}(x,w)$$

$$q(x,w) \leftarrow \mathsf{REG\text{-}AT}(v,y,z) \wedge \mathtt{Code}(v,x) \wedge \mathsf{Level}(y,w)$$

Query: $r_1(x,t,z) \wedge r_2(y,w) \wedge$ (t denotes the same Department as y)

The reconciliation problem: Example

Slide 60



R_1	C_1	D_1	T_1
	c_1	d_1	t_1
	c_2	d_2	t_2

Match(d_2, d'_2)

R_2	D_2	L_2
	d'_2	l_3
	d_4	l_4
	d_5	l_5

$r_1(x,t,z) \;\leftarrow\; \texttt{Code}(v,x) \wedge \textsf{No}(y,t) \wedge \textsf{REG-AT}(v,y,z) \mid \mathit{identify}([t],y)$

$r_2(y,w) \;\leftarrow\; \texttt{Cod}(x,y) \wedge \textsf{Level}(x,w) \mid \mathit{identify}([y],x)$

$q(x,w) \;\leftarrow\; \textsf{REG-AT}(v,y,z) \wedge \texttt{Code}(v,x) \wedge \textsf{Level}(y,w)$

Query: $\; r_1(x,t,z) \;\wedge\; r_2(y,w) \;\wedge\; \mathit{match}(t,y)$

Refining source description in LAV

Slide 61

Wrappers present each source as a set of relations

Each relation is described as an **adorned query**

$$T(\vec{\mathbf{x}}) \;\leftarrow\; \mathit{body}(\vec{\mathbf{x}},\vec{\mathbf{y}}) \mid \langle \mathit{adornment} \rangle$$

- $T(\vec{\mathbf{x}})$ defines the schema of the relation
- $\mathit{body}(\vec{\mathbf{x}},\vec{\mathbf{y}})$ describes the content of the relation in terms of a query over the global conceptual model
- The *adornment* declares
  - the domains of the columns of the relation
  - which relation attributes identify the entities at the conceptual level

Example of source schema: Source 1

Slide 62

One relation with information on customers (ssn) registered at (either in promotion or not) a certain department (name)

$\textsf{TABLE}_1(S,M,P) \;\leftarrow$

$\quad \textsf{REG-AT}(X,D) \wedge \neg\textsf{PROMOTION}(X,D) \wedge P = \textsf{false} \wedge$
$\quad \texttt{SSN}(X,S) \wedge \textsf{Name}(D,M)$
$\quad \vee$
$\quad \textsf{PROMOTION}(X,D) \wedge P = \textsf{true} \wedge \texttt{SSN}(X,S) \wedge \textsf{Name}(D,M)$

$\quad |$

$\quad \mathit{identify}([S],X),\; S :: \texttt{SSNString},$
$\quad \mathit{identify}([M],D),\; M :: \texttt{DeptNameString},$
$\quad P :: \textsf{Boolean}$

Example of source schema: Source 2

Slide 63

One relation with information on contracts between customers (name, date of birth) and departments (name) for services (number)

$\textsf{TABLE}_2(N,B,I,M) \;\leftarrow$

$\quad \textsf{CONTRACT}(X,S,D) \wedge \textsf{Name}(X,N) \wedge \texttt{DOB}(X,B) \wedge$
$\quad \texttt{ServNo}(S,I) \wedge \textsf{Name}(D,M)$

$\quad |$

$\quad \mathit{identify}([N,B],X),\; N :: \texttt{PersNameString},\; B :: \texttt{Date}$
$\quad \mathit{identify}([I],S),\; I :: \texttt{ServNoInteger},$
$\quad \mathit{identify}([M],D),\; M :: \texttt{DeptNameString}$

Dealing with schematic differences

Assume a new relationship $\mathsf{ServiceCost}(d, s, p)$

$$
\begin{array}{rcl}
\mathtt{v1\_1}(D,P) & \leftarrow & \mathsf{ServiceCost}(D,S,P) \wedge \mathtt{ServNo}(S,'\mathtt{v1}') \mid \\
& & P :: \mathtt{ItalianLira},\ D :: \mathtt{Date},\ \mathit{identify}(['\mathtt{v1}'], S) \\
\mathtt{v2\_1}(D,P) & \leftarrow & \mathsf{ServiceCost}(D,'\mathtt{v2}',P) \mid \\
& & P :: \mathtt{ItalianLira},\ D :: \mathtt{Date} \\
\mathtt{Serv\_2}(X,Y,D) & \leftarrow & \mathsf{ServiceCost}(D,S_3,X) \wedge \mathsf{ServiceCost}(D,S_4,Y) \wedge \\
& & \mathtt{ServNo}(S_3,'\mathtt{v3}') \wedge \mathtt{ServNo}(S_4,'\mathtt{v4}') \mid \\
& & \mathit{identify}(['\mathtt{v3}'], S_3),\ \mathit{identify}(['\mathtt{v4}'], S_4), \\
& & X :: \mathtt{Euro},\ Y :: \mathtt{Euro},\ D :: \mathtt{Date} \\
\mathtt{Serv\_3}(S1,P) & \leftarrow & \mathsf{ServiceCost}('\mathtt{d}',S,P) \wedge \mathtt{ServNo}(S,S1) \mid \\
& & P :: \mathtt{Euro},\ \mathit{identify}([S1], S),\ S1 :: \mathsf{String}
\end{array}
$$

Slide 64

Interschema Correspondences

Specify how to **match**/**convert**/**reconcile** data in different sources.
Once for all, and **not** on a query-by-query basis

Conversion correspondences

$\mathit{convert}([\vec{\mathbf{x}}],[\vec{\mathbf{y}}]) \leftarrow \mathit{body}(\vec{\mathbf{x}},\vec{\mathbf{y}},\vec{\mathbf{z}})$ through $\mathit{program}(\vec{\mathbf{x}},\vec{\mathbf{y}},\vec{\mathbf{z}})$

Matching Correspondences

$\mathit{match}([\vec{\mathbf{x}}_1],\ldots,[\vec{\mathbf{x}}_k]) \leftarrow \mathit{body}(\vec{\mathbf{x}}_1,\ldots,\vec{\mathbf{x}}_k,\vec{\mathbf{z}})$ through $\mathit{program}(\vec{\mathbf{x}},\ldots,\vec{\mathbf{x}}_k,\vec{\mathbf{z}})$

Reconciliation Correspondences

$\mathit{reconcile}([\vec{\mathbf{x}}_1],\ldots,[\vec{\mathbf{x}}_k],[\vec{\mathbf{z}}]) \leftarrow \mathit{body}(\vec{\mathbf{x}},\vec{\mathbf{z}},\vec{\mathbf{w}})$ through $\mathit{program}(\vec{\mathbf{x}},\vec{\mathbf{z}},\vec{\mathbf{w}})$

Slide 65

Interschema Correspondences – Examples

$$
\begin{array}{rcl}
\mathit{convert}_1([N,B],[S]) & \leftarrow & \mathtt{PersNameString}(N) \wedge \mathtt{Date}(B) \wedge \mathtt{DOB}(N,B) \wedge \\
& & \mathtt{SSNString}(S) \ \text{ through } \mathtt{name\_to\_SSN}(N,B,S) \\
\mathit{convert}_2([M],[C]) & \leftarrow & \mathtt{DeptNameString}(M) \wedge \mathtt{DeptCodeInteger}(C) \\
& & \text{through } \mathtt{dept\_name\_to\_code}(M,C)
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{match}_1([N,B],[S]) & \leftarrow & \mathit{convert}_1([N,B],[S]) \ \text{ through } \mathit{none} \\
\mathit{match}_2([M],[C]) & \leftarrow & \mathit{convert}_2([M],[C]) \ \text{ through } \mathit{none} \\
\mathit{reconcile}_{1,1}([N,B],[S_1],[S_2]) & \leftarrow & \mathit{match}_1([N,B],[S_1]) \wedge \mathit{convert}_1([N,B],[S_2]) \\
& & \text{through } \mathit{none}
\end{array}
$$

Slide 66

Designing the mediator for a query in DWQ

Based on a technique for query rewriting under constraints and interschema correspondences:

- The designer expresses the requested relation as an adorned query
- Compute the rewriting in terms of source relations (quality factors can be used for choosing among different rewritings)

  The computed rewriting is a disjunction of conjunctive queries, where each atom is

  - a relation in a source
  - a matching/conversion/reconciliation predicate defined in the interschema correspondences
- The designer analyses the rewriting and adds **merging clauses**

Slide 67

Query posed to the global view – Example

Slide 68

$\mathtt{TDW}(S, I, C, P) \leftarrow$

$\mathsf{CONTRACT}_0(X, R, D) \wedge \mathsf{PROMOTION}_1(X, D) \wedge$

$\mathtt{SSN}(X, S) \wedge \mathtt{ServNo}(R, I) \wedge \mathtt{Code}(D, C) \wedge P = \mathsf{true}$

$\vee$

$\mathsf{CONTRACT}_0(X, R, D) \wedge \mathsf{REG\text{-}AT}_1(X, D) \wedge \neg\mathsf{PROMOTION}_1(X, D) \wedge$

$\mathtt{SSN}(X, S) \wedge \mathtt{ServNo}(R, I) \wedge \mathtt{Code}(D, C) \wedge P = \mathsf{false}$

$\vee$

$\mathsf{PROMOTION}_1(X, D) \wedge \mathtt{SSN}(X, S) \wedge \mathtt{Code}(D, C) \wedge P = \mathsf{true} \wedge I = \mathsf{NULL} \mid$

$\mathit{identify}([S], X),\ S :: \mathtt{SSNString},$

$\mathit{identify}([I], R),\ I :: \mathtt{ServNoInteger},$

$\mathit{identify}([C], D),\ C :: \mathtt{DeptCodeInteger},\ P :: \mathsf{Boolean}$

Query reformulated – Example

Slide 69

$\mathtt{TDW}(S_0, I_0, C, P_0) \leftarrow$

$\mathsf{TABLE}_1(S_1, M_1, P_1) \wedge \mathsf{TABLE}_2(N_2, B, I_2, M_2) \wedge$

$\mathit{reconcile}_{1,1}([N_2, B], [S_1], [S_0]) \wedge \mathit{reconcile}_{3,2}([M_1], [M_2], [C]) \wedge$

$\mathit{convert}_5([P_1], [P_0]) \wedge \mathit{convert}_6([I_2], [I_0])$

$\vee$

$\mathsf{TABLE}_1(S_1, M_1, P_1) \wedge I_0 = \mathsf{NULL} \wedge P_1 = \mathsf{true} \wedge$

$\mathit{convert}_2([M_1], [C]) \wedge \mathit{convert}_4([S_1], [S_0]) \wedge \mathit{convert}_5([P_1], [P_0])$

merging $\mathtt{TDW}(S_a, I_a, C_a, P_a)$ and $\mathtt{TDW}(S_a, I_b, C_a, P_a)$

such that $P_a = \mathsf{true} \wedge I_b = \mathsf{NULL}$

into $\mathtt{TDW}(S_a, I_a, C_a, P_a)$