

# **Multiplex: Integrating Autonomous, Heterogeneous and Inconsistent Information Sources**

Amihai Motro

Proceedings of NGITS 99  
Fourth International Workshop on Next Generation Information  
Technologies and Systems  
Zichron Yaacov, Israel, July 1999

Lecture Notes in Computer Science No. 1649  
Springer-Verlag, pp. 138–158

## Outline

1. Introduction: multidatabase systems.
2. Multiplex: a formal model for multidatabases.
3. Approximative answers.
4. Inconsistency resolution.
5. Conclusion: summary and remaining problems.

# 1. Multidatabase Systems

**The problem:** How to provide unified access to a collection of *autonomous, heterogeneous* and possibly *inconsistent* databases.

Problem has been the subject of research for almost 20 years.

## Some Recent Attacks on the Problem

1. Pegasus (Hewlett-Packard)
2. UniSQL (UniSQL)
3. TSIMMIS (Stanford)
4. SIMS (ISI)
5. Information Manifold (AT&T Research)
6. Virtual DataBase (Junglee)
7. VirtualDB

## 2. The Multiplex Model

### Important Features

1. **Extension** of the relational model.
2. Formal assumptions of **integrability**.
3. **Abstraction** of many ad-hoc solutions.
4. Full support for **heterogeneity**.
5. Flexibility via three **degrees of freedom**.
6. Support for **approximative answers**.
7. Quick adaptation to **evolving environments**.
8. **Practical platform** for implementation.

## Derivative Databases

A database  $(D', d')$  is a *derivative* of a database  $(D, d)$ , if

1. Scheme  $D'$  is a set of relation schemes that are *views* of the relation schemes of  $D$ .
2. Instance  $d'$  is the *extension* of the relation schemes in  $D'$  in the database instance  $d$ .

## View Equivalence

- Let  $(D_1, d_1)$  and  $(D_2, d_2)$  be derivatives of  $(D, d)$ .
- View  $V_1$  of  $D_1$  and view  $V_2$  of  $D_2$  are *equivalent*, if for every instance  $d$  of  $D$ , the extension of  $V_1$  in  $d_1$  and the extension of  $V_2$  in  $d_2$  are identical.
- Intuitively, view equivalence allows us to substitute the answer to one query for an answer to another query, although these are different queries on different schemes.

## Scheme Mapping

The commonality of two database schemes (which are both derivatives of the same database scheme) is expressed by scheme mapping.

Assume two database schemes  $D_1$  and  $D_2$ , both derivatives of  $D$ .

A *scheme mapping*  $(D_1, D_2)$  is a collection of view pairs  $(V_{i,1}, V_{i,2})$  ( $i = 1, \dots, m$ ), where

1. Each  $V_{i,1}$  is a view of  $D_1$  ,
2. Each  $V_{i,2}$  is a view of  $D_2$  ,
3.  $V_{i,1}$  is equivalent to  $V_{i,2}$  .

## Scheme Mapping (cont.)

### Example:

1. In one database:  $Emp = (Name, Title, Salary, Supervisor)$ .
2. In another database:  $Manager = (Ename, Level, Sal, Sup)$ .

The retrieval of the salaries of managers is performed differently in each database:

$$\left( \text{project}_{Name, Salary} \text{select}_{Title=manager} Emp, \right. \\ \left. \text{project}_{Ename, Sal} Manager \right)$$

### Limitation

We consider only views and queries that are *conjunctive*. Queries may also include *aggregate functions*.

## Multidatabase

1. A scheme  $D$ .
2. A collection  $(D_1, d_1), \dots, (D_n, d_n)$  of databases.
3. A collection  $(D, D_1), \dots, (D, D_n)$  of scheme mappings.

**Note:** The scheme mappings are pairs of views.

Views in the first position are *available views*.

Views in the second position are their *materialization*.

## Multidatabase (cont.)

To define the scheme mappings we need to assume that all  $n + 1$  database schemes are derivatives of a single database scheme.

1. We assume a (hypothetical) database that represents the *real world*. Its scheme and instance are *perfect*.
2. There exist  $n$  databases schemes and instances that are approximations of this real world database.
3. We construct yet another database scheme that approximates the real world scheme.
4. We relate the new scheme to the existing schemes via scheme mappings, and construct a multidatabase.

## Types of Inconsistency Among Multiple Databases

Unless the existing databases are mutually inconsistent, their integration is easy.

We distinguish two types of inconsistency:

1. **Intensional inconsistencies:** differences in schemes.
2. **Extensional inconsistencies:** differences in contents.

## Two Assumptions of Integrability

### 1. The Scheme Consistency Assumption

- All database schemes *derivatives* of the real world scheme.
- In each database scheme, every relation scheme is derived from the real world scheme.
- The different ways in which reality is modeled are all correct.
- There are no *modeling errors*, only *modeling differences*.
- All intensional inconsistencies among independent database schemes are *reconcilable*.

## Two Assumptions of Integrability (cont.)

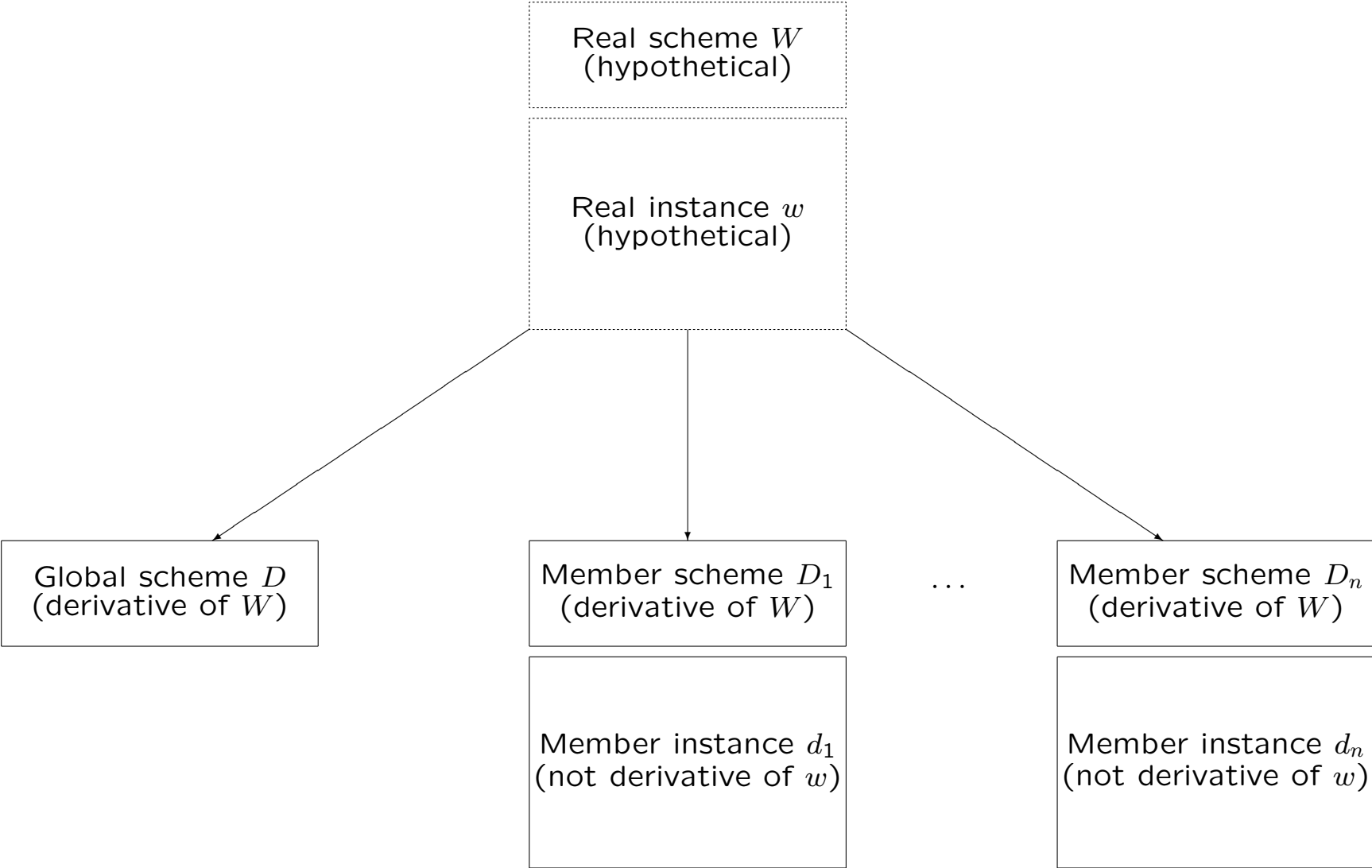
### 2. The Instance Consistency Assumption

- All database instances are *derivatives* of the real world instance.
- In each database instance, every relation instance is derived from the real world instance.
- The information stored in databases is always correct.
- There are no factual *errors*, only different *representations* of the facts.
- All extensional inconsistencies among independent database instances are *reconcilable*.

## Two Assumptions of Integrability (cont.)

- These assumptions have not been articulated before (in this context), but *tacit* assumptions have often been made.
- Most previous work on scheme integration has tacitly subscribed to the SCA.
- Most previous work on database integration has tacitly subscribed to the ICA.
- The Multiplex model assumes that the SCA *holds*, and that the ICA *does not hold*.
- The assumption that the SCA holds was used in the definition of multidatabases (the feasibility of scheme mappings).

# Consistency Assumptions in Multiplex



## Degrees of Freedom

Our model reflects realities of multidatabase environments:

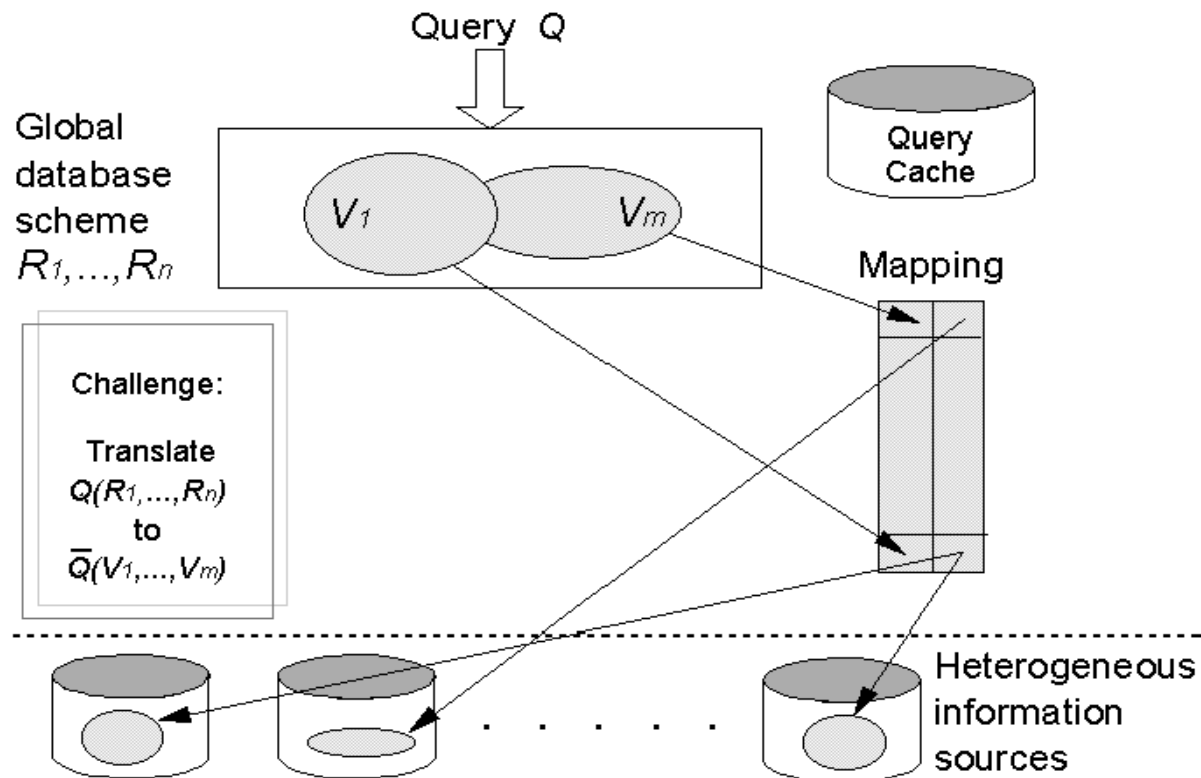
1. Scheme mapping not *total*: databases become temporarily unavailable.
2. Scheme mapping not *onto*: ad-hoc partial integration.
3. Scheme mapping not *single-valued*: framework for inconsistencies.

## Heterogeneity

Independent databases need not be relational!

Only requirement is that they compute *tabular answers*.

# A Diagrammatic View of the Multiplex System



## Multidatabase Queries

- **Syntactically:** multidatabase query is query on global scheme.
- **Semantically:** we must define how it is evaluated.
- **Intuitively:** we must transform a query on the global scheme to a query on the “available views”.
- **Formally:** Given
  1. Database scheme  $D = \{R_1, \dots, R_n\}$ .
  2. Set  $M = \{V_1, \dots, V_m\}$  of views of scheme  $D$ .

Translate a given query  $Q_D$  of the database scheme to an equivalent query  $Q_M$  of the view schemes.

- **Unfortunately:** a solution may not exist, or there could be multiple solutions. [Larson+Yang, Levy et al, Brodsky+Motro]

## Multidatabase Queries (cont.)

Assume multidatabase with scheme  $D$  and mapped views  $M$ , and consider query  $Q$  on this multidatabase.

The **answer** to  $Q$  is the *set of answers* produced by a sound and complete translation algorithm.

## Multidatabase Queries (cont.)

Two possible cases:

1. When the algorithm produces at least one solution, these solutions may evaluate to different answers. Each such answer is a *candidate answer*. The answer to  $Q$  is the **set of all candidate answers**.
2. When a solution to the translation problem does not exist, the answer to  $Q$  is the **empty set of answers**. This empty set of answers should be interpreted as *answer unavailable*.

**Note:** If the ICA holds, and the algorithm produces at least one solution, these solutions are guaranteed to evaluate to the same answer. The answer to  $Q$  is this unique answer.

### 3. Approximative Answers

Users expect every query to evaluate to a single answer.

- When multiple answers are available (case 1): **consolidate**.
- When no answers are available (case 2): **approximate**.

Treat two cases:

1. The ICA holds.
2. The ICA does not hold.

## Sound and Complete Answers

- Denote:
  - $q$  = database answer to  $Q$ .
  - $q_0$  = real answer to  $Q$ .
- $q$  is an *estimate* of  $q_0$  from the present database.
- $q$  is a *sound* answer if  $q \subseteq q_0$ .
- $q$  is a *complete* answer if  $q \supseteq q_0$ .
- The perfect answer  $q_0$  is *between* any sound answer  $q_s$  and any complete answer  $q_c$ .

## Sound and Complete Answers (cont.)

- When no additional information is available on the individual answers, all answers are assumed to *claim* soundness and completeness, and their claims are treated identically.
- A sound answer is a *yes* vote for its members, and a *maybe* vote for its non-members.
- A complete answer is a *maybe* vote for its members and a *no* vote for its non-members.

## Approximation and Consolidation Assuming Consistency

No answers possible (multiple answers not possible).

- When the perfect answer cannot be computed, sound/complete answers serve as “below” / “above” *approximations*.
- We want tightest approximation: *largest* sound answer and *smallest* complete answer.
- **Maximal Sound Approximation**: union of all subviews of  $Q$ .
- **Minimal Complete Approximation**: intersection of all superviews of  $Q$ .
- **Approximative answer**:

$$\langle \bigcup \{W_M \mid W_M \sqsubseteq Q\} , \bigcap \{W_M \mid W_M \sqsupseteq Q\} \rangle$$

- If  $Q$  is answerable:  
estimate converges to a single answer.

## Approximation and Consolidation Not Assuming Consistency

No answers and multiple answers are both possible.

- $q_i$  ( $i = 1, \dots, k$ ) denote the available answers to  $Q$ .  
 $s_i$  ( $i = 1, \dots, m$ ) denote the available subviews of  $Q$ .  
 $c_i$  ( $i = 1, \dots, m$ ) denote the available superviews of  $Q$ .
- $q_i$  claim to be sound and complete.  
 $s_i$  claim to be sound.  
 $c_i$  claim to be complete.
- **Consolidated approximation:** pair of *estimates* of sound and complete approximations:  
 $\langle \{t \mid t \text{ has only yes votes} \}, \{t \mid t \text{ does not have all no votes} \} \rangle$
- If the ICA holds and  $Q$  is answerable:  
estimate converges to a single answer.

## 4. Inconsistency (Conflict) Resolution

- Every tuple (possibly extended with nulls) is voted upon.
- No attempt to *coalesce* tuples (data fusion).
- **Example:** Two employee-salary tuples contributed by two sources: (Jones, 33,000), (Jones, 35,000).  
Each would receive a single “yes” vote, and both will only be included in the upper bound.
- Assume all relations have identifying attributes (*keys*).  
Assume all mapped views include key attributes.
- Under these assumptions we could resolve this inconsistency by (for example) the *average* of the salaries.

## Conflict Resolution [Anokhin+Motro 99]

- All contributed views are intersected to discover the subviews that are provided by more than one source.
- For each such subview, the multidatabase designer is prompted to provide a **conflict resolution strategy**.
- A special **resolve** statement allows the designer to specify how each attribute would be calculated.
- Types of resolution:
  1. External: Timestamp, quality, cost, etc.
  2. Internal: Average, mode, etc.
- The **resolve** statements become part of a new multidatabase definition:  
multidatabase scheme, contribution scheme, resolution scheme.

## **Conflict Resolution** (cont.)

The overall process:

1. Use previous voting technique on the key attributes to determine the lower and upper bounds.
2. Use conflict resolution strategies, to coalesce (“fuse”) the non-key attributes.

# The Multiplex System

## Highlights

- **Client-server architecture:** The Multiplex server is written in pure Java; the client applications (global queries) are CGI-scripts, usually invoked through the Multiplex user interface. All communications among server, clients, and member databases are carried on the Internet.
- **Heterogeneity:** presently integrates information from (1) relational (Oracle); (2) object-oriented (Ode); (3) simple files (Unix shell scripts); (4) Wide Area Information Services (WAIS), (5) spreadsheets (Excell), (6) menu-based (Xlibris).
- **Answer format:** answers are presented (using colors) as two relations: (1) the sound estimate; (2) the tuples that augment it to a complete estimate.
- **Query language:** conjunctive queries with aggregates and ordering (both SQL and query “wizard” interfaces available).

## 5. Conclusion

### Contributions

- Multiplex is a *formal* model for multidatabases:
  - Simple extension of the relational model.
  - Formal assumptions of integrability.
  - Abstraction of many ad-hoc approaches.
- Multiplex is *powerful* and *practical*:
  - Support for heterogeneity.
  - Support for ad-hoc integration and quick reconfiguration.
  - Support for sources unavailability and inconsistency:  
approximations by lower/upper bounds, conflict resolution.

## Some of the Issues under Investigation

1. New Implementation: Version 1.1 available; Version 2.0 under development.
2. Weighted voting: When not all sources are equal.
3. Database constraints: Use constraints to
  1. Invalidate candidate answers,
  2. “Clean up” global answers that do not satisfy the constraints.
4. Optimization of multidatabase queries: Can a source provide anything *less* than promised?
5. Mapping design: Does a present set of mapped views cover the global scheme?