Database Heterogeneity

Lecture 13

Outline

• Database Integration
• Wrappers
• Mediators
• Integration Conflicts
Motivation

• If we all use the same database, things are “quite simple”
• However, we often use
  – Heterogeneous data sources
  – Heterogeneous DBMS
  – Different data formats/data types etc.
• Key word: company merger

1. Database Integration

• Goal: providing a uniform access to multiple heterogeneous information sources
• More than data exchange (e.g., ASCII, EDI, XML)
• Old problem, difficult, well-know (partial) solutions
Data Integration

• We practised it in the project
• Typically requires (some) manual interaction
• What is your experience with it?

…still A Big Problem in Practice

Top IT Spending Priorities

1CIO Magazine Survey, February 2002

33% of firms surveyed have EAI projects (Forrester, March 2002 Business Technographics benchmark)
Old-School Approach (1)
Manual, Global Integration

- Manually merge multiple databases into a **new global database**
- Time consuming and error prone
- Local autonomy lost
- Static solution
- Does not scale with number of databases

\[
\begin{align*}
\text{Book} & \left( \text{ISBN, Title, Price, Author} \right) \\
\text{Author} & \left( \text{Name, ISBN} \right) \\
\text{Livre} & \left( \text{ISBN, Prix, Titre} \right) \\
\text{Auteur} & \left( \text{Prenom, Nom, ISBN} \right)
\end{align*}
\]

Old-School Approach (2)
Multidatabase Language Approach

- No attempt at integrating schemas
- Language (e.g., MSQL) used to integrate information sources at run-time
- Simple example:

\[
\begin{align*}
\text{Use} & \ S1, \ S2 \\
\text{Select} & \ \text{Titre} \\
\text{From} & \ S1.\text{Book}, \ S2.\text{Livre} \\
\text{Where} & \ S1.\text{Book.ISBN} = S2.\text{Livre.ISBN}
\end{align*}
\]

- Not transparent (you need to know all databases!)
- Heavy burden on (expert) users
- Global queries subject to local changes
How to Deal with Distribution?

- **Problems**
  - data access over the network
  - inconsistent replicated data
- **Solutions**
  - solved by using Web access (over HTTP)
  - Web Services, Java RMI, …
  - publishing using JSP
  - JDBC to access remote databases
  - etc.
How to Deal with Autonomy?

- **Problems**
  - changing structure of Web page
  - different coverage of Web sites
  - availability of services

- **Solutions**
  - manually adapt to changes
  - replication, materialization (availability)
  - contacts, agreements, … standards

How to Deal with Heterogeneity?

- **Problems**
  - Data models
  - Schemas
  - Data

- **Solutions**
  - Mappings, schema integration
  - Standards
Solution Variants

• General issues
  – Bottom-up vs. top-down engineering
  – Virtual vs. materialized integration
  – Read-only vs. read-write access
  – Transparency: language, schema, location

• What did you do?

A Generic System Architecture

• One solution: the Wrapper-Mediator architecture

mediators integrate the data from the DBs
wrappers convert to a common representation
2. Wrappers

Wrapper Tasks

- Translate among different data models
- Data Model consists of
  - Data types
  - Integrity constraints
  - Operations (e.g. query language)
- Overcome other "syntactic" heterogeneity
A Closer Look at Data Models

- **Data model used by sources**
- **Data model used by integrated DB**
  - canonical data model (e.g. relational, XML)
- **Query models**
  - Structured queries, retrieval queries, data mining (statistics)

Example: Wrapping Relational Data in XML/HTML

- **Data types**
  - trivial
- **Integrity Constraints (e.g. primary keys)**
  - requires XML Schema
- **Operations**
  - none in HTML
Example: Wrapping XML/HTML into Relational

- Data Types
  - which difficulties?
- Integrity Constraints
  - none in HTML
- Operations
  - requires generally XQuery
  - form fields can be considered as hard-coded queries

3. The Mediator

- Integrate data with same "real-world meaning", but different representations
  - Semantics are important
  - integration mapping \implies schema integration
  - can be implemented, e.g., as database views
- Decompose queries against the integrated schema to queries against source DBs
  - only for virtual integration
An Example: LAV

- **Local As View** approach
- Each local database is defined as a view on the integrated schema

A simple Example:

**Source A**: R1(prof, course, university)

**Source B**: R2(title, prof, course)

Definition of the global, integrated schema:
Global(prof,course,title,university)

Source A defined as:
Create view R1 as
SELECT prof, course, university FROM Global

Source B defined:
SELECT title, prof, course FROM Global

---

Schema Integration

- **Standard Methodology**

```
Schema translation (wrapper)
Correspondence investigation
Conflict resolution and schema integration
```
Identifying Schema Correspondences

Sources of information
- source schema
- source database
- source application
- database administrator, developer, user

• Semantic correspondences
  - e.g. related names
  - One of the important current research topics
    • No obvious solutions yet

• Structural correspondences
  - reachability by paths

• Data analysis
  - distribution of values
A Closer Look at Schemas

- Tight vs. loose integration
  - Is there a global schema?
- Support for semantic integration
  - collection, fusion, abstraction

A Widely Used Architecture: Federated DBMS

- accepted model for integrated database systems with integrated schema
- 5-level architecture
- data independence
Export Schema

- provided by *data source*
- source DB can change w/o changing export schema

Import Schema

- provided by *wrapper*
- export schema can change w/o changing import schema
Integrated Schema

- provided by **mediator**
- import schemas can change w/o changing integrated schema

Application View

- provided by **application**
- integrated DB can change w/o changing application (code)
4. Handling Integration Conflicts

- What types of problems did you encounter integrating corresponding data?
- different structural representation (e.g. attribute vs. table)
- different naming schemes

Types of Conflicts

- Schema level
  - Naming conflicts
  - Structural conflicts
  - Classification conflicts
  - Constraint and behavioral conflicts
- Data level
  - Identification conflicts
  - Representational conflicts
  - Data errors
Conflict Resolution

- Depends on type of conflict
- Requires construction of mappings
- Mappings might be complex, e.g. not expressible as SQL views

Naming Conflicts

- Homonyms
  - same name used for different concepts
  - Resolution: introduce prefixes to distinguish the names
- Synonyms
  - different names for the same concepts
  - Resolution: introduce a mapping to a common name
Structural Conflicts

- Different, non-corresponding attributes
  - Resolution: create a relation with the union of the attributes
- Different datatypes
  - Resolution: build a mapping function
- Different data model constructs
  - e.g. attribute vs. relation
  - Resolution: requires higher order mappings

Classification Conflicts

- Relations can have different coverage (inclusion, non-empty intersection)
  - Resolution: build generalization hierarchies
- Additional problem
  - Identification of corresponding data instances
  - "real world" correspondence is application dependent
Data Correspondences

• Corresponding data instances
  – similar to naming conflicts at schema level
  – Resolution: mapping tables and functions
  – Similarity functions
• Corresponding data values, data conflicts
  – of corresponding data instances
  – Resolution: mapping tables and functions
  – Prefer data from more trusted data source

Constraint and Behavioral Conflicts

• Cardinality conflicts
  – different types of cardinality constraints on relationships
  – Resolution: use the more general constraint
• Behavioral conflicts for relation update
  – E.g. cascading delete vs. non-cascading
  – Resolution: add missing behavior at global level
More?

- Security
  - protecting data
- Data Quality
  - actively managing data quality
- Integration as Agreement Process
  - "emergent semantics"

Questions to Lecture