Overview of Data Integration

Helena Galhardas
IST/DEI

(based on the slides of the course: CIS 550 – Database & Information Systems, Univ. Pennsylvania, Zachary Ives)

References

- Chapter 1 of the book: “Principles of Data Integration” by AnHai Doan, Alon Halevy, Zachary Ives
  (http://research.cs.wisc.edu/dibook/)
- Slides of the course: CIS 550 – Database & Information Systems, Univ. Pennsylvania, Zachary Ives)
Outline

- Overview of Data Integration
- Data Integration Architectures

The Problem

- Often people build databases in isolation, then want to share their data
  - Different systems within an enterprise
  - Different information brokers on the Web
  - Scientific collaborators
  - Researchers who want to publish their data for others to use
- Even with normalization and the same needs, different people will arrive at different schemas
Application Area 1: Business

Enterprise Databases

Legacy Databases

Services and Applications

EII Apps:

CRM

ERP

Portals

...

50% of all IT $$$ spent here!

Application Area 2: Science

Phenotype | Gene | Sequenceable Entity | Structured Vocabulary | Experiment
---|---|---|---|---
Gene-
Clinics | HUGO | LOCUS-
Link | Swiss-
Prot | GO

OMIM

Entrez

GEO

Hundreds of biomedical data sources available; growing rapidly!
Application Area 3: The Web

Solution: Data Integration

Data integration: set of techniques that enable a uniform access to a set of autonomous and heterogeneous data sources, controlled by different people, through a common schema.
Motivating example (1)

- **FullServe**: company that provides internet access to homes and sells products to support the home computing infrastructure (ex: modems, wireless routers, etc)

- FullServe is a predominantly American company and decided to acquire **EuroCard**
  - an European company that is mainly a credit card provider, but has recently started leveraging its customer base to enter the internet market

---

<table>
<thead>
<tr>
<th><strong>FullServe databases</strong></th>
<th>HR</th>
<th>Training &amp; Development</th>
<th>Sales</th>
<th>Customer-care</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Employee Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FullTimeEmp(ssn, empId, firstName, middleName, lastName)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hire(empId, hireDate, recruiter)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TempEmployees(ssn, hireStart, hireEnd, name, hourlyRate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Training Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Courses(courseID, name, instructor)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enrollments(courseID, empId, date)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Services Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Services(packName, textDescription)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customers(name, id, zipCode, streetAdr, phone)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contracts(custID, packName, startDate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sales Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Products(prodName, prodId)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales(prodName, customerName, address)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Resume Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interview(interviewDate, name, recruiter, hireDecision, hireDate)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CV(name, resume)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HelpLine Database</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calls(date, agent, custId, text, action)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Observations

Why data resides in multiple DBs in a company, rather than in a single well-organized DB?

1. When companies go through internal restructuring, they do not always align their DBs
   - Ex: FullServe services and products databases
2. Most DBs are created by a group within the company with a specific need
   - Not all the information needs in the future are anticipated
   - Ex: FullServe training database

EuroCard databases

**Employee Database**
Emp(ID, firstNameMiddleInitial, lastName)  
Hire(ID, hireDate, recruiter)

**CreditCard Database**
Customer(CustID, cardNum, expiration, currentBalance)  
CustDetail(CustID, name, address)

**Resume Database**
Interview(ID, date, location, recruiter)  
CV(name, resume)

**HelpLine Database**
Calls(date, agent, custId, description, followup)
Interesting queries in FullServe

- The HR Department wants to query for all of its employees whether in the US or in Europe
  - Requires access to 1 database in the American side (Employee) and 1 in the European side (Employee)

- FullServe has a single customer support hot-line, where customers can call about any service or product they obtain from the company. When a representative is on the phone with a customer, it’s important to see the entire set of services the customer is getting from FullServe (internet service, credit card or products purchased). Furthermore, it is useful to know whether the customer is a big spender on its credit card.
  - Requires access to 2 databases in the US side (Sales and Services) and 1 in the European side (CreditCard)

Other Reasons to Integrate Data

- Create a (useful) web site for tracking services
- Collaborate with third parties
  - E.g., create branded services
- Comply with government regulations
  - Find “risky” employees
- Business intelligence
  - What’s really wrong with our products?
Another example: searching for a new job (1)

Problem

- Each form (site) asks for a slightly different set of attributes (ex: keywords describing job, location and job category or employer and job type)
- It’s tiring to go daily to some of these sites
- Ideally, would like to have a single web site to pose our queries and have that site integrating data from all relevant sites in the Web
Create a single site to search for jobs/rentals/…

**Goal of data integration**

To offer **uniform access** to a set of **autonomous** and **heterogeneous** data sources:

- The focus is on **querying** the data sources, but updating may also be interesting
- **Large** number of sources
- **Heterogeneous** data sources (different systems, diff. schemata, some structured other unstructured)
- **Autonomous** data sources: we may not have full access to the data or source may not be available all the time.
Why is it hard?

- **Systems reasons**
  - SQL supported across multiple systems not exactly the same
  - Managing different platforms
  - Distributed query processing

- **Logical reasons**
  - Schema and data heterogeneity
    - Different schemas (e.g., FullServe and EuroCard temporary employees), diff attributes (e.g., ID), diff attribute names for the same knowledge (e.g., text and action), diff. representations of data (e.g. First-name, last name) also known as semantic heterogeneity
  - Social and administrative reasons
    - Locating and capturing relevant data in the enterprise.
    - Convincing people to share (data fiefdoms)

---

Practical goals of data integration

- **Ideally**: data integration system access a set of data sources and **automatically configures itself** to correctly and efficiently answer queries over multiple sources

- **Actually**:
  - Build tools to reduce the **user effort** required to integrate a set of data sources
  - Improve the ability of the system to answer queries in **uncertain** environments

- **Trade-off**:
  - User effort vs Accuracy
Outline

- Overview of Data Integration
- Virtual Data Integration Architecture
- Data Warehouse Architecture

Building a Data Integration System

Create a middleware mediator or data integration system over the sources

- Can be warehoused (a data warehouse) or virtual
- Presents a uniform query interface and schema
- Abstracts away multitude of sources; consults them for relevant data
  - Unifies different source data formats (and possibly schemas)
  - Sources are generally autonomous, not designed to be integrated
- Sources may be local DBs or remote web sources/services
- Sources may require certain input to return output (e.g., web forms)
  - binding patterns describe these
Generic Data Integration Architecture

Logical components of a virtual integration system

Is built for the data integration application and contains only the aspects of the domain relevant to the application. Most probably will contain a subset of the attributes seen in sources.

Specify the properties of the sources the system needs to know to use their data. Main component are semantic mappings that specify how attributes in the sources correspond to attributes in the mediated schema, how to resolve differences in how data values are specified in different sources. Other info is whether sources are complete.

Programs whose role is to send queries to a data source, receive answers and possibly apply some basic transformation to the answer.
Example: a data integration scenario

Source descriptions

- The relationship between the sources and the mediated schema (semantic mappings), for example:
  - mapping of source S1 states it contains movies; the attribute name in Movies maps to attribute title in the mediator Movie relation; the Actors relation in the mediated schema is a projection of the Movies source on the attributes name and actors
- Whether the sources are complete, for example:
  - Source S2 may not contain all the movie showing times in the entire country
- Can specify limited access patterns to the sources, e.g.,:
  - All the playing times sources require a movie title as input
Query Processing in a Virtual Data Integration System

Query reformulation

- Rewrite the user query that was posed in terms of the relations in the mediated schema, into queries referring to the schemas of data sources
- Result is called a logical query plan:
  - Set of queries that refer to the schemata of the data sources and whose combination will yield the answer to the original query
Example

**Ex:** SELECT title, startTime
    FROM Movie, Plays
    WHERE Movie.title = Plays.movie
    AND location = 'New York'
    AND director = 'Woody Allen'

- Tuples for `Movie` can be obtained from source S1 but attribute `title` needs to be reformulated to `name`.
- Tuples for `Plays` can be obtained from S2 or S3. Since S3 is complete for showings in NY, we choose it.
- Since source S3 requires the `title` of a movie as input and the title is not specified in the query, the query plan must first access S1 and then feed the movie titles returned from S1 as inputs to S3.

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies: name, actors, director, genre</td>
<td>Cinemas: place, movie, start</td>
<td>Cinemas in NYC: cinema, title, startTime</td>
<td>Cinemas in SF: location, movie, startingTime</td>
<td>Reviews: title, date, grade, review</td>
</tr>
</tbody>
</table>

Query optimization

- Accepts a logical query plan as input and produces a physical query plan.
  - Specifies the exact order in which sources are accessed, when results are combined, which algorithms are used for performing operations on the data, and the amount of resources allocated to each operation.
- **Ex:**
  - The optimizer will decide which join algorithm to combine results from S1 and S3. It may stream tuples from S1 and input them into S3, or it may batch them up before sending them to S3.

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movies: name, actors, director, genre</td>
<td>Cinemas: place, movie, start</td>
<td>Cinemas in NYC: cinema, title, startTime</td>
<td>Cinemas in SF: location, movie, startingTime</td>
<td>Reviews: title, date, grade, review</td>
</tr>
</tbody>
</table>
Query Processing in a Virtual Data Integration System

Query (formulated on mediated schema)

Query reformulation

Logical query plan

Query optimizer

Physical query plan

Execution engine

wrapper
source

wrapper
source

wrapper
source

wrapper
source

wrapper
source

Query execution

- Responsible for the execution of the physical query plan
  - Dispatches the queries to the individual sources through the wrappers and combines the results as specified by the query plan.
  - Also may ask the optimizer to reconsider its plan based on its monitoring of the plan’s progress (e.g., if source S3 is slow)

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>Movies: name, actors, director, genre</th>
<th>Cinemas: place, movie, start</th>
<th>Cinemas in NYC: cinema, title, startTime</th>
<th>Cinemas in SF: location, movie, startingTime</th>
<th>Reviews: title, date grade, review</th>
</tr>
</thead>
</table>
Outline

- Overview of Data Integration
- Virtual Data Integration Architecture
  - Data Warehouse Architecture

---

Data Warehouses – Offline Replication

- Define a **database** with this schema
- Define procedural **mappings** using an **ETL (Extract, Transform, Load)** tool to import the data and clean it.
- Periodically copy all of the data from the data sources
  - Note that the sources and the warehouse are basically independent at this point

![Data Warehouse Diagram](image)
Pros and Cons of Data Warehouses

- Need to spend time to design the physical database layout, as well as logical
  - This actually takes a lot of effort!
- Data is generally not up-to-date (lazy or offline refresh)

- Queries over the warehouse don’t disrupt the data sources
- Can run very heavy-duty computations, including data aggregation, data mining and data cleaning

Next Lecture

- Manipulation of Query Expressions