Lecture #9

Data Integration
May 30th, 2002

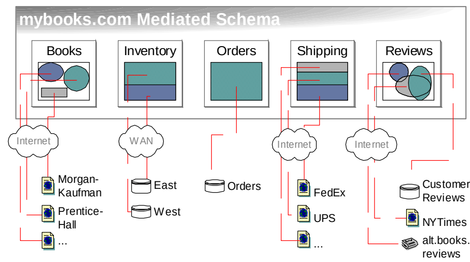
Agenda/Administration

- Project demo scheduling.
- Reading pointers for exam.

What is Data Integration

- Providing
  - Uniform (same query interface to all sources)
  - Access to (queries; eventually updates too)
  - Multiple (we want many, but 2 is hard too)
  - Autonomous (DBA doesn’t report to you)
  - Heterogeneous (data models are different)
  - Structured (or at least semi-structured)
  - Data Sources (not only databases).

The Problem: Data Integration



Uniform query capability across *autonomous*, *heterogeneous* data sources on LAN, WAN, or Internet

Motivation(s)

- Enterprise data integration; web-site construction.
- WWW:
  - Comparison shopping
  - Portals integrating data from multiple sources
  - B2B, electronic marketplaces
- Science and culture:
  - Medical genetics: integrating genomic data
  - Astrophysics: monitoring events in the sky.
  - Environment: Puget Sound Regional Synthesis Model
  - Culture: uniform access to all cultural databases produced by countries in Europe.

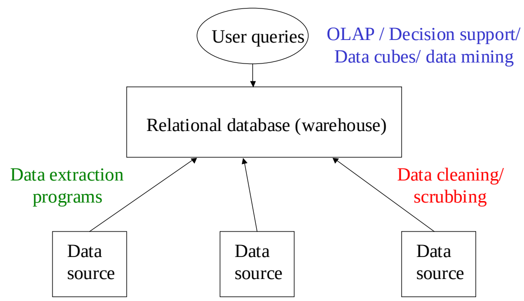
Discussion

- Why is it hard?
- How will we solve it?

Current Solutions

- Mostly ad-hoc programming: create a special solution for every case; pay consultants a lot of money.
- Data warehousing: load all the data periodically into a warehouse.
  - 6-18 months lead time
  - Separates *operational* DBMS from *decision support* DBMS. (not only a solution to data integration).
  - Performance is good; data may not be fresh.
  - Need to clean, scrub you data.

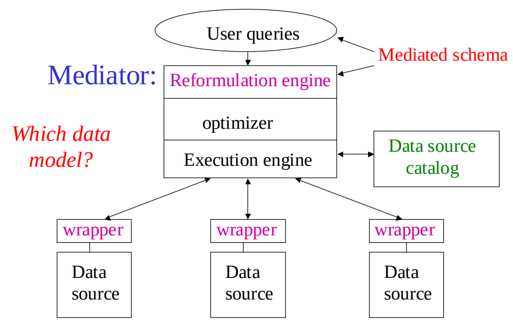
Data Warehouse Architecture



The Virtual Integration Architecture

- Leave the data in the sources.
- When a query comes in:
  - Determine the relevant sources to the query
  - Break down the query into sub-queries for the sources.
  - Get the answers from the sources, and combine them appropriately.
- Data is fresh.
- Challenge: performance.

Virtual Integration Architecture



Sources can be: relational, hierarchical (IMS), structure files, web sites.

Research Projects

- Garlic (IBM),
- Information Manifold (AT&T)
- Tsimmis, InfoMaster (Stanford)
- The Internet Softbot/Razor/Tukwila (UW)
- Hermes (Maryland)
- DISCO, Agora (INRIA, France)
- SIMS/Ariadne (USC/ISI)

Industry

- Nimble Technology
- Enosys Markets
- IBM starting to announce stuff
- BEA marketing announcing stuff too.

Dimensions to Consider

- How many sources are we accessing?
- How autonomous are they?
- Meta-data about sources?
- Is the data structured?
- Queries or also updates?
- Requirements: accuracy, completeness, performance, handling inconsistencies.
- Closed world assumption vs. open world?

Outline

- Wrappers
- Semantic integration and source descriptions:
  - Modeling source completeness
  - Modeling source capabilities
- Query optimization
- Query execution
- Peer-data management systems
- Creating schema mappings

Wrapper Programs

- Task: to communicate with the data sources and do format translations.
- They are built w.r.t. a specific source.
- They can sit either at the source or at the mediator.
- Often hard to build (very little science).
- Can be "intelligent": perform source-specific optimizations.

Example

Transform:

 Introduction to DB
<i> Phil Bernstein </i>
<i> Eric Newcomer </i>
Addison Wesley, 1999

into:

<book>
<title> Introduction to DB </title>
<author> Phil Bernstein </author>
<author> Eric Newcomer </author>
<publisher> Addison Wesley </publisher>
<year> 1999 </year>
</book>

Data Source Catalog

- Contains all meta-information about the sources:
  - Logical source contents (books, new cars).
  - Source capabilities (can answer SQL queries)
  - Source completeness (has *all* books).
  - Physical properties of source and network.
  - Statistics about the data (like in an RDBMS)
  - Source reliability
  - Mirror sources
  - Update frequency.

Content Descriptions

- User queries refer to the *mediated schema*.
- Data is stored in the sources in a *local schema*.
- Content descriptions provide the semantic mappings between the different schemas.
- Data integration system uses the descriptions to translate user queries into queries on the sources.

Desiderata from Source Descriptions

- Expressive power: distinguish between sources with closely related data. Hence, be able to prune access to irrelevant sources.
- Easy addition: make it easy to add new data sources.
- Reformulation: be able to reformulate a user query into a query on the sources efficiently and effectively.

Reformulation Problem

- Given:
  - A query Q posed over the mediated schema
  - Descriptions of the data sources
- Find:
  - A query Q' over the data source relations, such that:
    - Q' provides only *correct answers* to Q, and
    - Q' provides *all* possible answers from to Q given the sources.

Approaches to Specifying Source Descriptions

- Global-as-view: express the mediated schema relations as a set of views over the data source relations
- Local-as-view: express the source relations as views over the mediated schema.
- Can be combined with no additional cost.

Global-as-View

Mediated schema:
 Movie(title, dir, year, genre),
 Schedule(cinema, title, time).

```
Create View Movie AS
  select *  from S1     [S1(title,dir,year,genre)]
  union
  select  * from S2     [S2(title, dir,year,genre)]
  union                 [S3(title,dir), S4(title,year,genre)]
  select S3.title, S3.dir, S4.year, S4.genre
  from  S3, S4
  where S3.title=S4.title
```

Global-as-View: Example 2

Mediated schema:
 Movie(title, dir, year, genre),
 Schedule(cinema, title, time).

```
Create View Movie AS [S1(title,dir,year)]
  select title, dir, year, NULL
  from S1
  union                 [S2(title, dir,genre)]
  select  title, dir, NULL, genre
  from S2
```

Global-as-View: Example 3

Mediated schema:
 Movie(title, dir, year, genre),
 Schedule(cinema, title, time).

Source S4: S4(cinema, genre)

```
Create View Movie AS
  select NULL, NULL, NULL, genre
  from S4
 Create View Schedule AS
   select cinema, NULL, NULL
   from S4.
```

But what if we want to find which cinemas are playing comedies?

Global-as-View Summary

- Query reformulation boils down to view unfolding.
- Very easy conceptually.
- Can build hierarchies of mediated schemas.
- You sometimes loose information. Not always natural.
- Adding sources is hard. Need to consider all other sources that are available.

Local-as-View: example 1

Mediated schema:
  Movie(title, dir, year, genre),
  Schedule(cinema, title, time).
Create Source S1 AS
  select * from Movie
Create Source S3 AS   [S3(title, dir)]
  select title, dir from Movie
Create Source S5 AS
  select title, dir, year
  from Movie
  where year > 1960 AND genre="Comedy"

Local-as-View: Example 2

Mediated schema:
  Movie(title, dir, year, genre),
  Schedule(cinema, title, time).
Source S4: S4(cinema, genre)
Create Source S4
  select cinema, genre
  from Movie m, Schedule s
  where m.title=s.title
.
Now if we want to find which cinemas are playing comedies, there is hope!

Local-as-View Summary

- Very flexible. You have the power of the entire query language to define the contents of the source.
- Hence, can easily distinguish between contents of closely related sources.
- Adding sources is easy: they're independent of each other.
- Query reformulation: *answering queries using views!*

The General Problem

- Given a set of views V1,…,Vn, and a query Q, can we answer Q using only the answers to V1,…,Vn?
- Many, many papers on this problem.
- The best performing algorithm: The MiniCon Algorithm, (Pottinger & Levy, 2000).
- Great survey on the topic: (Halevy, 2001).

Local Completeness Information

- If sources are incomplete, we need to look at each one of them.
- Often, sources are *locally complete*.
- Movie(title, director, year) complete for years after 1960, or for American directors.
- Question: given a set of local completeness statements, is a query Q' a complete answer to Q?

Example

- Movie(title, director, year) (complete after 1960).
- Show(title, theater, city, hour)
- Query: find movies (and directors) playing in Seattle:

  Select m.title, m.director

  From Movie m, Show s

  Where m.title=s.title AND city="Seattle"
- Complete or not?

Example #2

- Movie(title, director, year), Oscar(title, year)
- Query: find directors whose movies won Oscars after 1965:

  select m.director

  from Movie m, Oscar o

  where m.title=o.title AND m.year=o.year AND o.year > 1965.
- Complete or not?

Query Optimization

- Very related to query reformulation!
- Goal of the optimizer: find a physical plan with minimal cost.
- Key components in optimization:
  - Search space of plans
  - Search strategy
  - Cost model

Optimization in Distributed DBMS

- A distributed database (2-minute tutorial):
  - Data is distributed over multiple nodes, but is uniform.
  - Query execution can be distributed to sites.
  - Communication costs are significant.
- Consequences for optimization:
  - Optimizer needs to decide locality
  - Need to exploit independent parallelism.
  - Need operators that reduce communication costs (semi-joins).

DDBMS vs. Data Integration

- In a DDBMS, data is distributed over a set of *uniform* sites with *precise* rules.
- In a data integration context:
  - Data sources may provide only limited access patterns to the data.
  - Data sources may have additional query capabilities.
  - Cost of answering queries at sources unknown.
  - Statistics about data unknown.
  - Transfer rates unpredictable.

Modeling Source Capabilities

- Negative capabilities:
  - A web site may require certain inputs (in an HTML form).
  - Need to consider only valid query execution plans.
- Positive capabilities:
  - A source may be an ODBC compliant system.
  - Need to decide placement of operations according to capabilities.
- Problem: *how to describe and exploit source capabilities.*

Example #1: Access Patterns

Mediated schema relation: Cites(paper1, paper2)

Create Source S1 as
  select *
  from Cites
  given paper1
Create Source S2 as
  select paper1
  from Cites

Query: select paper1 from Cites where paper2="Hal00"

Example #1: Continued

Create Source S1 as
  select *
  from Cites
  given paper1
Create Source S2 as
  select paper1
  from Cites
Select p1
From S1, S2
Where S2.paper1=S1.paper1 AND S1.paper2="Hal00"

Example #2: Access Patterns

Create Source S1 as
  select *
  from Cites
  given paper1
Create Source S2 as
  select paperID
  from UW-Papers
Create Source S3 as
  select paperID
  from AwardPapers
  given paperID
Query: select * from AwardPapers

Example #2: Solutions

- Can't go directly to S3 because it requires a binding.
- Can go to S1, get UW papers, and check if they're in S3.
- Can go to S1, get UW papers, feed them into S2, and feed the results into S3.
- Can go to S1, feed results into S2, feed results into S2 again, and then feed results into S3.
- Strictly speaking, we can't a priori decide when to stop.
- Need *recursive query processing*.

Handling Positive Capabilities

- Characterizing positive capabilities:
  - Schema independent (e.g., can always perform joins, selections).
  - Schema dependent: can join R and S, but not T.
  - Given a query, tells you whether it can be handled.
- Key issue: how do you search for plans?
- Garlic approach (IBM): Given a query, STAR rules determine which subqueries are executable by the sources. Then proceed bottom-up as in System-R.

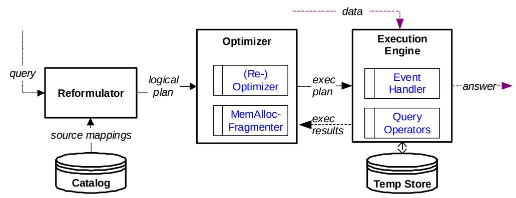
Matching Objects Across Sources

- How do I know that A. Halevy in source 1 is the same as Alon Halevy in source 2?
- If there are uniform keys across sources, no problem.
- If not:
  - Domain specific solutions (e.g., maybe look at the address, ssn).
  - Use Information retrieval techniques (Cohen, 98). Judge similarity as you would between documents.
  - Use concordance tables. These are time-consuming to build, but you can then sell them for lots of money.

Optimization and Execution

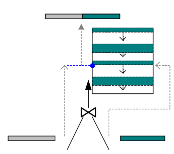
- Problem:
  - Few and unreliable statistics about the data.
  - Unexpected (possibly bursty) network transfer rates.
  - Generally, unpredictable environment.
- General solution: (research area)
  - Adaptive query processing.
  - Interleave optimization and execution. As you get to know more about your data, you can improve your plan.

Tukwila Data Integration System

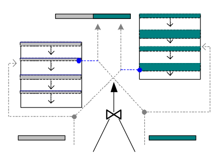


Novel components:
- Event handler
- Optimization-execution loop

Double Pipelined Join (Tukwila)



Hash Join
- 8 Partially pipelined: no output until inner read
- 8 Asymmetric (inner vs. outer) — optimization requires source behavior knowledge



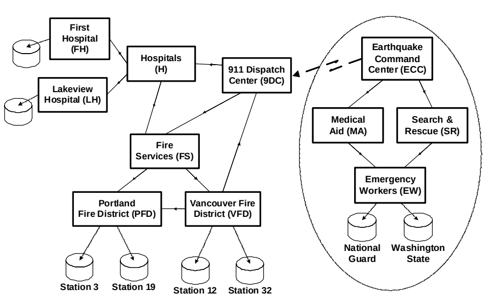
Double Pipelined Hash Join
- 4 Outputs data immediately
- 4 Symmetric — requires less source knowledge to optimize

Piazza: A Peer-Data Management System

***Goal**: To enable users to share data across local or wide area networks in an ad-hoc, highly dynamic distributed architecture.*

- § Peers share data, mediated views.
- § Peers act as both clients and servers
- § Rich semantic relationships between peers.
- § Ad-hoc collaborations (peers join and leave at will).

Extending the Vision to Data Sharing

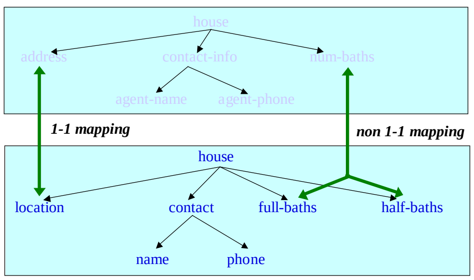


The *Structure* Mapping Problem

- Types of structures:
  - Database schemas, XML DTDs, ontologies, …,
- Input:
  - Two (or more) structures, S_1 and S_2
  - (perhaps) Data instances for S_1 and S_2
  - *Background* knowledge
- Output:
  - A mapping between S_1 and S_2
    - Should enable translating between data instances.

Semantic Mappings between Schemas

- Source schemas = XML DTDs



Why Matching is Difficult

- Structures represent same entity differently
  - different names => same entity:
    - area & address => location
  - same names => different entities:
    - area => location or square-feet
- Intended semantics is typically subjective!
  - IBM Almaden Lab = IBM?
- ***Schema, data and rules never fully capture semantics!***
  - not adequately documented, certainly not for machine consumption.
- Often hard for humans (committees are formed!)

Desiderata from Proposed Solutions

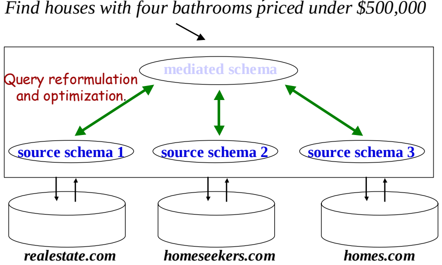
- Accuracy, efficiency, ease of use.
- Realistic expectations:
  - Unlikely to be fully automated. Need user in the loop.
- Some notion of semantics for mappings.
- Extensibility:
  - Solution should exploit additional background knowledge.
- "Memory", knowledge reuse:
  - System should exploit previous manual or automatically generated matchings.
  - Key idea behind LSD.

Learning for Mapping

- Context: generating semantic mappings between a *mediated schema* and a large set of *data source schemas*.
- Key idea: generate the first mappings manually, and learn from them to generate the rest.
- Technique: multi-strategy learning (extensible!)
- L(earning) S(ource) D(escriptions) [SIGMOD 2001].

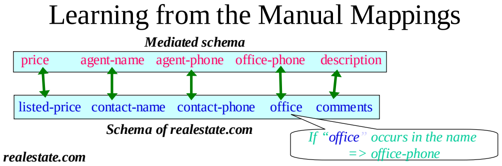
Data Integration (a simple PDMS)

Find houses with four bathrooms priced under $500,000



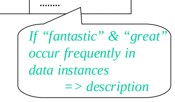
Applications: WWW, enterprises, science projects
Techniques: virtual data integration, warehousing, custom code.

Learning from the Manual Mappings



realestate.com

listed-price	contact-name	contact-phone	office	comments
$250K	James Smith	(305) 729 0831	(305) 616 1822	Fantastic house
$320K	Mike Doan	(617) 253 1429	(617) 112 2315	Great location
....	....	....	....	....



homes.com

sold-at	contact-agent	extra-info
$350K	(206) 634 9435	Beautiful yard
$230K	(617) 335 4243	Close to Seattle
$190K	(512) 342 1263	Great lot

Multi-Strategy Learning

- Use a set of *base* learners:
  - Name learner, Naïve Bayes, Whirl, XML learner
- And a set of *recognizers:*
  - County name, zip code, phone numbers.
- Each base learner produces a prediction weighted by confidence score.
- Combine base learners with a *meta-learner*, using stacking.

The Semantic Web

- How does it relate to data integration?
- How are we going to do it?
- Why should we do it? Do we need a killer app or is the semantic web a killer app?