Data become more Important than Applications

The Data Evolution Timeline

- Age of Programs
  - "Data is less important than code"
  - 1945 - 1970

- Age of Proprietary Data (Office)
  - "Data is as important as code"
  - 1970 - 1994

- Age of Open Data (HTML)
  - "Data is more important than code"
  - 1994 - 2000

- Age of Open Metadata (XML)
  - 2000 - 2003

- Age of Semantic Models (OWL)
  - 2003 -
What is Data Integration?

- **Data integration** is the problem of providing *unified* and *transparent access* to a collection of data stored in *multiple*, *autonomous*, and *heterogeneous* data sources.

- At least two contexts:
  - **Intra-organization** data integration (e.g., EIS)
  - **Inter-organization** data integration (e.g., integration on the Web)

- One of the major challenges for the future of IT:

  - Estimated Annual Integration costs + data quality costs worldwide: $1 Trillion/year\(^1\,^2\)
  - ...and this is just a beginning
  - "The Grand Challenge is now becoming mission critical"

---

**Top 10 Business and Technology Priorities in 2010**

<table>
<thead>
<tr>
<th>Top 10 Business Priorities</th>
<th>Ranking</th>
<th>Top 10 Technology Priorities</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business process improvement</td>
<td>1</td>
<td>Virtualization</td>
<td>1</td>
</tr>
<tr>
<td>Reducing enterprise costs</td>
<td>2</td>
<td>Cloud computing</td>
<td>2</td>
</tr>
<tr>
<td>Increasing the use of information/analysis</td>
<td>3</td>
<td>Web 2.0</td>
<td>3</td>
</tr>
<tr>
<td>Improving enterprise workforce effectiveness</td>
<td>4</td>
<td>Networking, voice and data communications</td>
<td>4</td>
</tr>
<tr>
<td>Attracting and retaining new customers</td>
<td>5</td>
<td>Business Intelligence</td>
<td>5</td>
</tr>
<tr>
<td>Managing change initiatives</td>
<td>6</td>
<td>Mobile technologies</td>
<td>6</td>
</tr>
<tr>
<td>Creating new products or services (innovation)</td>
<td>7</td>
<td>Data/document management and storage</td>
<td>7</td>
</tr>
<tr>
<td>Targeting customers and markets more effectively</td>
<td>8</td>
<td>Service-oriented applications and architecture</td>
<td>8</td>
</tr>
<tr>
<td>Consolidating business operations</td>
<td>9</td>
<td>Security technologies</td>
<td>9</td>
</tr>
<tr>
<td>Expanding current customer relationships</td>
<td>10</td>
<td>IT management</td>
<td>10</td>
</tr>
</tbody>
</table>

*Source: Gartner EXP (January 2010)*
Why Data Integration is a Difficult Problem?

- **Number of sources / size of the problem**
  - the Web is large! New optimization techniques are needed (Pipelining, Distributed processing)

- **Location of the sources / source discovery**
  - does a source that supposedly fulfills my info needs exist?
  - where is it located? (The Semantic Web helps—"google" for SW agents)

- **Heterogeneity of the sources**
  - system (Web Services, WSDL/SOAP, etc.)
  - syntactic (HTML, XML, RDF, RDBS, ORDBS, etc.)
  - structural (DB schemas, XML DTDs, RDF/OWL Ontologies)

Why Data Integration is a Difficult Task?

- **Autonomy of the sources**
  - sources may change their data, schemas especially on the Internet and they will not notify the integration system about these changes

- **Volatility of the sources**
  - sources may come and go, the system should be flexible to recover from these changes with minimal effort (use flexible ways to map sources to the system, use of adaptive query answering techniques)

- **Different source capabilities**
  - some source are more intelligent than others, e.g. they allow to perform e.g. join like queries on them)
  - some sources are more restrictive than others, e.g. they require certain input (filling out some forms) before they provide results
Integrated Access to Distributed Data: Existing Architectures

- Distributed databases
  - data sources are *homogeneous* databases under the control of the distributed database management system

- Multidatabase or federated databases
  - data sources are *autonomous, heterogeneous* databases; procedural specification

- (Mediator-based) data integration
  - access through a global schema mapped to autonomous and heterogeneous data sources; declarative specification

- Peer-to-peer data integration
  - *network* of autonomous systems mapped one to each other, without a global schema; declarative specification

---

History of Heterogeneous Database Integration

```
Multidatabase System
  Nonfederated DBS  Federated DBS
    |     |
    v     v
Loosely Coupled  Tightly Coupled
  |     |
  v  v
Single Federation  Multiple Federation
```
Data Interoperability

- LOCATION TRANSPARENCY
  - global schema
    - global access only: DISTRIBUTED DBS
    - global + local access: FEDERATED DBS

- LOCATION VISIBILITY
  - no global schema
    - multiDB views, multi DB access language:
      - MULTIDATABASE SYSTEMS

- unstructured or semi-structured data
  - (files, repositories, knowledge bases, spreadsheets, …)
  - information exchange protocols / languages:
    - INTEROPERABLE SYSTEMS

The Materialized Integration Approach

- Data warehousing: Materialized views
  - global (materialized) schema
  - all data periodically loaded into a central repository,
    - need to clean, scrub you data
    - should be automated (ETL tools)
  - “easy” querying;
    - performance is good;
  - after the data is “in house” the system is independent from the sources
  - difficult to keep up-to-date; data may not be fresh
  - separates operational DBMS from decision support DBMS
    - not only a solution to data integration
    - good for intranet applications (e.g. enterprise data integration)
The Virtual Integration Approach

- Mediators: Virtual views
  - global (mediated) schema; leave the data in the sources
  - data retrieved on demand (no central repository)
  - 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, filter them if needed and combine them appropriately
  - data always up-to-date (fresh)
  - good for most WWW applications e.g. shopbots, different internet portals

Materialized Vs Virtual Data Integration

Materialized views

Virtual views
Translating Between Different Data Models

Wrappers access the sources (provide a view in a uniform data model of the data stored in the sources)

Peer-to-peer Data Integration

Operations: – Answer(Q,Pi) – Materialize(Pi)
Traditional

- Each relation is exported by a single database or at least in a controlled way by several distributed databases.
- All sources are assumed to be fully relational.
- Completeness is expected; optimize cost.
- Cost dominated by transfer time.
- Deterministic execution; availability of statistics.

Data Integration

- Multiple sources export partial and overlapping portions of a relation.
  - Need to minimize plans to remove redundancy.
- Sources may allow only limited types of queries.
  - Need to model query capabilities.
- Optimize cost/coverage together.
  - First few tuples?
- Access costs need to be considered.
- Unreliable sources; networks.
  - Adaptive execution.
  - Non-blocking plans.
  - Symmetric hash joins etc.

Research Prototypes

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>RESEARCH GROUP</th>
<th>Multiple Sources</th>
<th>DM</th>
<th>QL</th>
<th>Subscriptions</th>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataServer</td>
<td>DARPA (JTF-ATD)</td>
<td>Yes</td>
<td>O2</td>
<td>QL</td>
<td>No</td>
<td>No</td>
<td>CBR, Optimization, Integration, CORBA-based infrastructure, ALP-like</td>
</tr>
<tr>
<td>DISCO</td>
<td>INRIA, Paris</td>
<td>Yes</td>
<td>Logic-based</td>
<td>SQL-like</td>
<td>No</td>
<td>No</td>
<td>CBR, Cost-model</td>
</tr>
<tr>
<td>Garlic</td>
<td>L. Haas, Almaden IBM Research Center</td>
<td>Yes</td>
<td>O2</td>
<td>OQL7</td>
<td>No</td>
<td>No</td>
<td>CBR, Cost-model, Query Optimizer</td>
</tr>
<tr>
<td>Hermes</td>
<td>National Taiwan University of Science and Technology, National Taiwan University of Science and Technology</td>
<td>Yes</td>
<td>Logic-based</td>
<td>User SQL, GUI, Internal Rule-based (PROLOG like)</td>
<td>No</td>
<td>No</td>
<td>Toolkit for generating mediators (merging, Semantic reconciliation),</td>
</tr>
<tr>
<td>Information Manifold</td>
<td>A. Levy, AT&amp;T Research Lab</td>
<td>Yes</td>
<td>Desc, Logic, SQL7</td>
<td>No</td>
<td>No</td>
<td>CBR, Guarantees about completeness of answer</td>
<td>Implemented, discontinued</td>
</tr>
<tr>
<td>InfoSleuth</td>
<td>M. Ruenkiewicz, MCC, Austin</td>
<td>Yes</td>
<td>KQML</td>
<td>No</td>
<td>Yes</td>
<td>Agent-based architecture, resource discovery + broker, ontologies, CBR, Sem. reconciliation</td>
<td>Implemented, first version, ongoing</td>
</tr>
<tr>
<td>MARIA</td>
<td>GRC International, Inc.</td>
<td>Yes</td>
<td>O2</td>
<td>Web-based: SQL, C++, Java</td>
<td>Yes</td>
<td>Yes</td>
<td>Agent-based, information search, and retrieval, data translation and correction</td>
</tr>
<tr>
<td>KOMET</td>
<td>University of Karlsruhe, Germany</td>
<td>Yes</td>
<td>?</td>
<td>Logic-based (KAMEL)</td>
<td>No</td>
<td>No</td>
<td>Query planning and CBR, Sem. Reconciliation planned</td>
</tr>
<tr>
<td>TSIMMIS</td>
<td>H. Garcia-Molina, Honkai, Stanford University</td>
<td>Yes</td>
<td>OEM</td>
<td>LOREL, MSL,</td>
<td>In combination with C3</td>
<td>Auto, Med, &amp; Wrapper generator, Capabilities-based mediation, Semistructured data, Sem. Reconciliation through external service access</td>
<td>Implemented, ongoing, used in I**3</td>
</tr>
</tbody>
</table>
Research Prototypes

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>RESEARCH GROUP</th>
<th>Multiple Sources</th>
<th>DM</th>
<th>QL</th>
<th>Subscriptions</th>
<th>Push</th>
<th>Pull</th>
<th>Features</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>C3</td>
<td>J. Widom, InfoLab, Stanford University</td>
<td>N/A</td>
<td>OEM, DOM</td>
<td>LOREL</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Change Detection, Subscriptions</td>
<td>Partially implemented, on-going, used in I**3</td>
</tr>
<tr>
<td>CQ</td>
<td>C. Pu, Oregon Graduate Institute</td>
<td>Yes</td>
<td>Relational</td>
<td>SQL+</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Change Detection, Subscriptions</td>
<td>Partially implemented, on-going</td>
</tr>
<tr>
<td>Sentinel Active OODBMS</td>
<td>S. Chakravarthy, University of Florida</td>
<td>N/A</td>
<td>DOM</td>
<td>SQL</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>ECA-rules, Event spec. lang., Event detector, Rule editor</td>
<td>Partially implemented, on-going</td>
</tr>
<tr>
<td>COIN</td>
<td>S. Madden, M. Steigl, MIT</td>
<td>Yes</td>
<td>Relational</td>
<td>SQL+</td>
<td>No</td>
<td>No</td>
<td></td>
<td>Meta-attributes to specify context, Semantic reconciliation</td>
<td>Partially implemented, on-going</td>
</tr>
<tr>
<td>InfoHarness</td>
<td>A. Sheth, University of Georgia</td>
<td>Yes</td>
<td>DOM</td>
<td>User: Web-based Internal ??</td>
<td>No</td>
<td>No</td>
<td>Maintains metadata repository to support real-time access to net courses, Sem. reconciliation</td>
<td>Partially implemented, on-going at Bellcore</td>
<td></td>
</tr>
<tr>
<td>CoBase</td>
<td>Wes Chu, UCLA</td>
<td>Yes</td>
<td>DOM, SQL+</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
<td>Query relaxation, Integration, GIS integration</td>
<td>Implemented, on-going at Bellcore</td>
</tr>
<tr>
<td>SIMS</td>
<td>Y. Arens, ISU/ISC</td>
<td>Yes</td>
<td>LOOM, LOOM, SQL</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
<td>Dynamic Query Planning</td>
<td>Implemented, on-going, used in P**3</td>
</tr>
</tbody>
</table>

DM – Data Model
QL – Query Language
CBR – Capabilities-Based Rewriting

Some DB Theory

- **Schema**
- **Conjunctive queries**

For example:
*Give me students together with their advisors who took courses given by their advisors after the 1st semester 2010*
DB Theory

- **SQL**

```
SELECT Advises.prof, Advises.student 
FROM Registered, Teaches, Advises 
WHERE Registered.c-number = Teaches.c-number and 
    Registered.semester=Teaches.semester and 
    Advises.prof=Teaches.prof and 
    Advises.student=Registered.student and 
    Registered.semester > "2010\1"
```

- **DATALOG**

```
Q(prof, student) :-
    Registered(student,course, semester), Teaches(prof,course, semester), Advises(prof, student), semester > "2010\1"
```

DB Theory: Query Containment

- A query $Q'$ is **contained** in $Q$ if for all databases $D$, the set of tuples computed for $Q'$ is a subset of those computed for $Q$
  
  i.e., $Q' \subseteq Q$ iff $\forall D \ Q'(D) \subseteq Q(D)$

- A query $Q'$ is **equivalent** to $Q$ iff $Q' \subseteq Q$ and $Q \subseteq Q'$

- **Example:**

  $Q'(\text{prof}, \text{student}) :-$
  
  Registered(student,course, semester), Teaches(prof,course, semester), Advises(prof, student), semester > "2010\2"
  
  is contained in

  $Q(\text{prof}, \text{student}) :-$
  
  Registered(student,course, semester), Teaches(prof,course, semester), Advises(prof, student), semester > "2010\1"
Simple Query Containment Check

- Create a canonical database $D$ that contains a tuple for each subgoal in $Q'$ ("frozen" body)
- Execute $Q$ over it: $Q(D)$
- If $Q(D)$ returns a tuple that matches the head of $Q'$ (i.e. contains the "frozen" head of $Q'$) then $Q' \subseteq Q$ otherwise not
- This is an NP-complete algorithm in the size of the query
  - Testing for full first-order logic queries is undecidable!!!)

Example:
$Q'$: $p3(x,y) \leftarrow \text{arc}(x,z), \text{arc}(z,w), \text{arc}(w,y)$
$Q$: $\text{path}(x,y) \leftarrow \text{arc}(x,y)$
  $\text{path}(x,y) \leftarrow \text{path}(x,z), \text{path}(z,y)$

1. Freeze $Q'$ with say 0,1,2,3 for $x,z,w,y$ respectively:
   $Q'(0,3) \leftarrow \text{arc}(0,1), \text{arc}(1,2), \text{arc}(2,3)$
   $D=\{\text{arc}(0,1), \text{arc}(1,2), \text{arc}(2,3)\}$
2. Compute $Q(D)$:
   $Q(D)=\{(0,1), (1,2), (2,3), (0,2), (1,3), (0,3)\}$
3. The frozen head of $Q' = (0,3)$ is in the $Q(D)$ ⇒ $Q' \subseteq Q$

Virtual Data Integration: Approaches & Languages
The Modeling Problem

- How to model the global schema?
  - data model the user poses queries (entities and attributes of interest)
  - possible constraints (key/uniqueness, keys/foreign keys)
- How to model the sources?
  - data model (tables/objects)
  - access limitations (processing capabilities like join, selects, …)
  - data values (common vs. different domains)
- How to model the mapping between global schemas and sources?
  - overlapping and contradictory information
  - semantic mismatches (different naming conventions)

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

---

Virtual Integration Architecture

**Design-Time**
- Mapping Tool
- Mediation Language
- Web Services
- XML

**Run-Time**
- Query Reformulation
- Optimization & Execution

---

End User

Mediator

Global Schema

Query

Result

Wrapper

Data Source

Local Schema

Wrapper

Data Source

Local Schema
Virtual Integration Architecture

**Design-Time**
- Mapping Tool
- Mediation Language
- Web Services
- XML

**Run-Time**
- Query Reformulation
- Optimization & Execution

**Diagram**
- Mediator
- Global Schema
- Wrapper
- Local Schema
- Query
- End User
- Result

**Overview**
- Data Source
- Data Source
- Wrapper
- Wrapper
- End User
- Mapping Tool
- XML
- Web Services
- Query Reformulation
- Optimization & Execution
Virtual Integration Architecture

**Design-Time**
1. XML
2. Web Services
3. Mediation Language
4. Mapping Tool

**Run-Time**
1. Query Reformulation
2. Optimization & Execution

**Mediation Process**
- End User initiates a query.
- Mediator maps the query to the global schema.
- Wrappers transform data sources to local schemas.
- Results are transformed and returned to the end user.

** Tools and Technologies**
- XML
- Web Services
- Mediation Language
- Mapping Tool

**Optimization and Execution**
- Query Reformulation
- Optimization & Execution
Virtual Integration Architecture

**Design-Time**

1. XML
2. Web Services
3. Mediation Language
4. Mapping Tool

**Run-Time**

5. Query Reformulation
6. Optimization & Execution

![Diagram](diagram.png)

Mediation Languages

- Requirements:
  - **accuracy**, **completeness**, **performance**, handling of **inconsistencies**
  - Closed world assumption vs. open world?

![Diagram](diagram2.png)
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 global schema
  - A set of 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
Languages for Schema Mapping

Global-as-View (GAV)

- Each global relation is defined as a view wrt the source views while queries are expressed over these views
- Thus, it seems that we can simply evaluate the query over the data satisfying the global relations (as if we had a single database at hand)
  - Query answering means unfolding the mappings/views
GAV: First Example

Global Schema:
Movie(title, dir, year, genre)
Schedule(cinema, title, time)

Integrating View:
Create View Movie AS
SELECT * FROM S1
union
SELECT * FROM S2
union
SELECT S3.title, S3.dir, S4.year, S4.genre
FROM S3, S4
WHERE S3.title = S4.title

GAV: Second Example

Global Schema:
Movie(title, dir, year, genre)
Schedule(cinema, title, time)

Integrating View:
Create View Movie AS
SELECT title, dir, year, NULL
FROM S1
union
SELECT title, dir, NULL, genre
FROM S2
GAV: Third Example

**Global Schema:**
- Movie(title, dir, year, genre)
- Schedule(cinema, title, time)

**Integrating Views:**

Create View Movie AS
```
SELECT NULL, NULL, NULL, genre
FROM S4 [S4(cinema, genre)]
```

Create View Schedule AS
```
SELECT cinema, NULL, NULL
FROM S4 [S4(cinema, genre)]
```

*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 global schemas
- You sometimes loose information
  - Not always natural
- Adding sources is hard
  - Need to consider all other sources that are available
Local-as-View (LAV)

- Each data source is described by one or more views with respect to the mediated schema.
- The views might not be complete, i.e., they may only contain a subset of tuples satisfying their definition.
- Query answering is formulated as a query rewriting problem:
  - reformulate a user query (against the mediated schema) into a query that refers directly to the available mappings/views.

LAV: First Example

Global Schema:
- Movie(title, dir, year, genre)
- Schedule(cinema, title, time)

Source Views:
- Create View S1 AS [S1(title, dir, year, genre)]
  - SELECT * FROM Movie
- Create View S3 AS [S3(title, dir)]
  - SELECT title, dir FROM Movie
- Create View S5 AS [S5(title, dir, year)]
  - SELECT title, dir, year FROM Movie
  - WHERE year > 1960 AND genre='Comedy'
LAV: Second Example

Global Schema:
- Movie(title, dir, year, genre)
- Schedule(cinema, title, time)

Source Views:
- Create View S4 [S4(cinema, genre)]
  ```sql
  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!

LAV Summary

- Very expressive, 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!
Global and Local as View (GLAV)

- In order to take advantage of both LAV and GAV, an other approach has been proposed: GLAV

- GLAV approach combines the expressive power of both LAV and GAV, allowing flexible schema definitions independent of the particular details of the sources

- In GLAV we can find both the following expressions:
  \[ \text{loci} (X) : - \text{gl1}(X1), \text{gl2}(X2), ..., \text{glk}(Xk) \text{ and } \text{loc1}(X1), \text{loc2}(X2), ..., \text{lock}(Xk) : - \text{glj}(X) \text{ or even } \text{loc1}(X1), \text{loc2}(X2), ..., \text{lock}(Xk) : - \text{gl1}(X1), \text{gl2}(X2), ..., \text{glm}(Xm) \]

Interpretation of Views

- A view describes some of the facts that are available at the source

- A view does not define exactly what is at the source
  - Example: View \( S(t,d) : - \text{Movie}(\text{title}, \text{director}, 1960) \) says that the source has some Movie-facts with third component 1960, not all of them
  - \( S \) could even be empty although \( \text{Movie}(\text{title}, \text{director}, 1960) \) is not

- In other words:
  - The \( : - \) separator between head and body of a view definition should not be interpreted as “if”
  - Rather, it is “only if”
Differences Between OWA and CWA

CWA

S1(t,d) :- Movie(title, director, year)
S2(t,d) :- Movie(title, director, year)

S1 = S2 = All movie tuples

S1 and S2 have all movie tuples
E.g.: S1 and S2 are computed from the same movie table in a database

OWA

S1 and S2 have some movie tuples
E.g.: S1 and S2 are from two different web sites

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?

Advertised description

True source contents

Guarantees
Example

- Movie(title, director, year)
  - complete after 1960
- Show(title, theater, city, hour)

- Query: find movies (and directors) playing in Seattle:
  ```sql
  SELECT M.title, M.director
  FROM Movie M, Show S
  WHERE M.title=S.title
  AND city='Seattle'
  ```

- Complete or not?

Second Example

- Movie(title, director, year), Oscar(title, year)

- Query: find directors whose movies won Oscars after 1965:
  ```sql
  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?
Usability Conditions for Views

Query: q(X,Z) :- r(X,Y), s(Y,Z), t(X,Z), Y > 5

What can go wrong?

V1(A,B) :- r(A,C), s(C1,B) (join predicate not applied)

V2(A,B) :- r(A,C), s(C,B), C > 1 (predicate too weak)

V3(A) :- r(A,B), s(B,C), t(A,C), B > 5: needed argument is projected out. Can be recovered if we have a functional dependency t: A → C

GAV and LAV – Comparison

- **GAV** (e.g., Carnot, SIMS, Tsimmis, IBIS, Momis, DisAtDis, . . .)
  - Quality depends on how well we have compiled the sources into the global schema through the mapping
  - Whenever a source changes or a new one is added, the global schema needs to be reconsidered
  - Adding constraints to sources requires modifications of the global schema description
  - Query processing can be based on some sort of unfolding; query answering looks easier (polynomial)– without constraints

- **LAV** (e.g., Information Manifold, DWQ, Picsel)
  - Quality depends on how well we have characterized the sources
  - High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected)
  - Adding constraints to sources requires some extensions to the source descriptions
  - Query processing needs reasoning; query answering is complex (NP-Complete)
Answering Queries Using Views

The General Problem

- Given a query \( Q \) and a set of view definitions \( V_1, \ldots, V_n \):
  - Is it possible to answer \( Q \) using only the \( V \)'s?
- Assumptions:
  - There is a set of predicates that define the global schema
    - These do not exist as stored relations
  - Each data source has its capabilities defined by views, which are conjunctive queries (CQs) whose subgoals involve the global predicates
  - A query is a CQ over the global predicates
  - A rewriting is an expression (union of CQ's) involving the views
    - Ideally, the rewriting is equivalent to the query
    - In practice, we have to be happy with a rewriting maximally contained in the query
Formal Definition of Query Rewriting

Given a query $Q$ and a set $V$ of view definitions $V_1, \ldots, V_n$

$Q'$ is a rewriting of the query using $V$'s if it refers only to the views (or to arithmetic predicates)

$Q'$ is an equivalent rewriting of $Q$ using the $V$'s if $Q'$ is equivalent to $Q$

$Q'$ is a maximally-contained rewriting of $Q$ w.r.t. $L$ (mapping language) using the $V$'s if there is no other $Q''$ such that $Q''$ strictly contains $Q'$, and $Q''$ is contained in $Q$

- $Q'$ refers only to views in $V$ and
- $Q' \subseteq Q$, and
- there is no rewriting $Q''$, such that $Q' \subseteq Q''$ and $Q'' \neq Q''$

Example

Source Views:
- $V_1(A,B) :- \text{cites}(A,B), \text{cites}(B,A)$
- $V_2(C,D) :- \text{sameTopic}(C,D), \text{cites}(C,C_1), \text{cites}(D,D_1)$

Query:
- $q(X,Y) :- \text{sameTopic}(X,Y), \text{cites}(X,Y), \text{cites}(Y,X)$

Query rewriting:
- $q'(X,Y) :- V_1(X,Y), V_2(X,Y)$

Unfolding of the rewriting:
- $q''(X,Y) :- \text{cites}(X,Y), \text{cites}(Y,X), \text{sameTopic}(X,Y), \text{cites}(X,Z), \text{cites}(Y,W)$

Note: Rewriting is not equivalent to the query, but we can’t do any better
More on Rewritings

- What makes a rewriting $R$ useful?
  - There must be no other rewriting containing $R$
  - When views in $R$ are unfolded into global predicates, $R$ is contained in the original query

- If $V(X,Y)$ is a subgoal in a rewriting, then substitute $V(X,Y)$ with $V$'s body by:
  1. Finding unique variables that appear only in the view’s body (local variables of the body)
  2. Substituting variables of the subgoal $V(X,Y)$ for variables of $V$'s head

Example

- Consider the subgoal $V2(X,Y)$ in the rewriting of our example:
  $q'(X,Y) :- V1(X,Y), V2(X,Y)$
- $V2$'s definition:
  $V2(C,D) :- \text{sameTopic}(C,D), \text{cites}(C,C1), \text{cites}(D,D1)$

- After step 1:
  $V2(C,D) :- \text{sameTopic}(C,D), \text{cites}(C,Z), \text{cites}(D,W)$

- After step 2 by substituting $C\rightarrow X$ and $D\rightarrow Y$:
  $V2(X,Y) :- \text{sameTopic}(X,Y), \text{cites}(X,Z), \text{cites}(Y,W)$

- Rewriting becomes:
  $q'(X,Y) :- V1(X,Y), \text{sameTopic}(X,Y), \text{cites}(X,Z), \text{cites}(Y,W)$

- Subgoal $V1(X,Y)$ is unfolded similarly
Important Points

- To test containment of a rewriting in a query, we unfold the views in the rewriting first, then test CQ containment of the unfolding in the query.
- The view definition describes what any tuples of the view look like, so CQ containment implies that the rewriting will provide only true answers.

**Query:**
\[ q(X,Y) :- \text{sameTopic}(X,Y), \text{cites}(X,Y), \text{cites}(Y,X) \]

**Query rewriting:**
\[ q'(X,Y) :- V1(X,Y), V2(X,Y) \]

**Unfolding of the rewriting:**
\[ q''(X,Y) :- \text{cites}(X,Y), \text{cites}(Y,X), \text{sameTopic}(X,Y), \text{cites}(X,Z), \text{cites}(Y,W) \]

Is there a containment mapping \( q \rightarrow q'' \)?

---

Important Points

- There is no guarantee a rewriting supplies any answers to the query.
- Comparing different rewritings by testing if one rewriting is contained in another must be done at the level of the folded views.
- Two sources might have similar views, defined by:
  \[ V2(C,D) :- \text{sameTopic}(C,D), \text{cites}(C,C1), \text{cites}(D,D1) \]
  \[ V3(E,F) :- \text{sameTopic}(E,F), \text{cites}(E,E1), \text{cites}(F,F1) \]
  - but the sources actually have different sets of tuples.
- Then, the two rewritings:
  \[ q'(X,Y) :- V1(X,Y), V2(X,Y) \]
  \[ q''(X,Y) :- V1(X,Y), V3(X,Y) \]
  - have the same unfolding, but there is no reason to believe one rewriting is contained in the other.
- One view could provide lots of tuples, the other, few or none.
**Important Points**

- On the other hand, when one rewriting, folded, is contained in another, we can be sure that if the first provides no answers the second does not.
- Here are two rewritings:
  
  \[
  q'(X,Y) \leftarrow V1(X,Y), V2(X,Y)
  \]
  
  \[
  q''(X,"WSDL") \leftarrow V1(X,"WSDL"), V2(X,"WSDL")
  \]
- There is a containment mapping \( q' \rightarrow q'' \)
  
  - Thus, \( q'' \subseteq q' \) at the level of views
- No matter what tuples \( V1 \) and \( V2 \) represent, \( q' \) provides all answers \( q'' \) provides
- Intuitively, \( q'' \) must contain the same or fewer answers vs. \( q' \):
  
  - It has all of the same conditions, except one extra conjunction (i.e., it’s more restricted)
  - There’s no union or any other way it can add more data

**Finding All Rewritings**

- For conjunctive queries with no arithmetic predicates, the following holds: *If \( Q \) has an equivalent rewriting using \( V \), then there exists one with no more conjuncts than \( Q \) [Levy, Mendelzon, Sagiv & Srivastava, PODS 95]*
  
  - The rewriting problem is NP-complete
- Maximally-contained rewriting: union of all conjunctive rewritings of the length of the query or less

**LMSS Test:**

- If a query has \( n \) subgoals, then we only need to consider rewritings with at most \( n \) subgoals
  
  - Any other rewriting must be contained in one with \( \leq n \) subgoals
Complexity of Finding Maximally-Contained Queries

- The ability to find a maximally-contained rewriting depends in subtle ways on other properties of the problem [Abiteboul S., Duschka O. PODS 98]
- Complexity doesn’t depend on the query
  - Can handle recursive queries just as easily
- Complexity does change if the sources are not "conjunctive queries"
  - Sources as unions of conjunctive queries (NP-hard)
    - Disjunctive descriptions
  - Sources as recursive queries (Undecidable)
    - Comparison predicates
- Complexity also changes based on Open vs. Closed world assumption

A Naive Algorithm

- Consider all rewritings containing up to as many views as Q has subgoals
- Test each unfolding for containment in Q
- Take the union of the contained ones
- Exponential and brute force
- Makes use of the LMSS test
- Can we do better?
Practical Query Rewriting Algorithms

The Bucket Algorithm

- **input** = set of views $V$
- **output** = maximally-contained rewriting composed from (some of the) views
- informally, a view can be useful for a query if the set of relations it mentions overlaps with that of the query, and it selects some of the attributes selected by the query. The constrains of the view must be equivalent or weaker than those from the query.
- **The main idea**: the number of rewritings that need to be considered can be reduced by considering each sub-goal of the query in isolation.

- **Step 1**:
  - Create a bucket for each sub-goal of $Q$ and fill it in with views/sources that are relevant for that particular sub-goal.
  - A view $V$ is put in the bucket of the relation/sub-goal $g$ if it contains this relation AND the constraints in $V$ are compatible (weaker than) those in $Q$.

- **Step 2**:
  - Combine source relations from each bucket into conjunctive queries (each consisting of one conjunct from every bucket).
  - Check containment wrt to $Q$, and add the rewriting to the answer.
  - The result is a union of conjunctive rewritings.
The Bucket Algorithm: Example

V1(Std,Crs,Sem,Title) :- reg(Std,Crs,Sem), course(Crs,Title), Crs ≥ 500, Sem ≥ Aut2010
V2(Std,Prof,Crs,Sem) :- reg(Std,Crs,Sem), teaches(Prof,Crs,Sem)
V3(Std,Crs) :- reg(Std,Crs,Sem), Sem ≤ Aut2005
V4(Prof,Crs,Title,Sem) :- reg(Std,Crs,Sem), course(Crs,Title), teaches(Prof,Crs,Sem), Sem ≤ Aut2008

q(S,C,P) :- teaches(P,C,Q), reg(S,C,Q), course(C,T), C ≥ 300, Q ≥ Aut2006

Step 1: For each query subgoal, put the relevant sources into a bucket

Note: Arithmetic predicates don’t pose a problem
ICS-FORTH & Univ. of Crete

The Bucket Algorithm: Example

V1(Std,Crs,Sem,Title) :- reg(Std,Crs,Sem), course(Crs,Title), Crs ≥ 500, Sem ≥ Aut2010
V2(Std,Prof,Crs,Sem) :- reg(Std,Crs,Sem), teaches(Prof,Crs,Sem)
V3(Std,Crs) :- reg(Std,Crs,Sem), Sem ≤ Aut2005
V4(Prof,Crs,Title,Sem) :- reg(Std,Crs,Sem), course(Crs,Title), teaches(Prof,Crs,Sem), Sem ≤ Aut2008
q(S,C,P) :- teaches(P,C,Q), reg(S,C,Q), course(C,T), C ≥ 300, Q ≥ Aut2006

Note: V3 doesn’t work: arithmetic predicates not consistent
V4 doesn’t work: S not in the output of V4

ICS-FORTH & Univ. of Crete

The Bucket Algorithm: Example

V1(Std,Crs,Sem,Title) :- reg(Std,Crs,Sem), course(Crs,Title), Crs ≥ 500, Sem ≥ Aut2010
V2(Std,Prof,Crs,Sem) :- reg(Std,Crs,Sem), teaches(Prof,Crs,Sem)
V3(Std,Crs) :- reg(Std,Crs,Sem), Sem ≤ Aut2005
V4(Prof,Crs,Title,Sem) :- reg(Std,Crs,Sem), course(Crs,Title), teaches(Prof,Crs,Sem), Sem ≤ Aut2008
q(S,C,P) :- teaches(P,C,Q), reg(S,C,Q), course(C,T), C ≥ 300, Q ≥ Aut2006

Note: V3 doesn’t work: arithmetic predicates not consistent
V4 doesn’t work: S not in the output of V4
The Bucket Algorithm: Example

Step 2:
- Try all combinations of views, one each from a bucket
- Test satisfaction of arithmetic predicates in each case
  - e.g., two views may not overlap, i.e., they may be inconsistent
- Desired rewriting = union of surviving ones

Query rewriting 1:

<table>
<thead>
<tr>
<th>teaches</th>
<th>reg</th>
<th>course</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2</td>
<td>V1</td>
<td>V1</td>
</tr>
</tbody>
</table>

q1(S,C,P) :- V2(S',P,C,Q), V1(S,C,Q,T'), V1(S'',C,Q',T)
- no problem from arithmetic predicates (none in V2)
- May or may not be minimal (why?)

Unfolding of rewriting 1:

Unfolded rewriting 1:

q1'(S,C,P) :- r(S',C,Q), t(P,C,Q), r(S,C,Q), c(C,T'), r(S'',C,Q'),
c(C,T), C ≥ 500, Q ≥ Aut2010, C ≥ 500, Q' ≥ Aut2010

- Black r's can be mapped to green r:
  S'⇒S, S''⇒S, Q'⇒Q

- Black c can be mapped to green c:
  just extend above mapping to T⇒T'

Minimized unfolding of rewriting 1:

Minimized rewriting 1:

q1m'(S,C,P) :- t(P,C,Q), r(S,C,Q), c(C,T'), C ≥ 500, Q ≥ Aut2010

Minimized rewriting 1:

q1m(S,C,P) :- V2(S',P,C,Q), V1(S,C,Q,T')
Query Rewriting 2:

\[ q_2(S,C,P) :\neg V_2(S',P,C,Q), V_1(S,C,Q,T'), V_4(P',C,T',Q') \]
\[ q_2'(S,C,P) : - r(S',C,Q), t(P,C,Q), r(S,C,Q), \]
\[ r(S,C,Q), c(C,T'), C \geq 500, Q \geq \text{Aut2010}, \]
\[ r(S'',C,Q'), c(C,T), t(P',C,Q'), Q' \leq \text{Aut2008} \]

- This combination is infeasible: consider the conjunction of arithmetic predicates in \( V_1 \) and \( V_4 \)

Query rewriting 3:

\[ q_3(S,C,P) : - V_2(S',P,C,Q), V_2(S,P',C,Q), V_4(P'',C,T,Q') \]

Unfolding of rewriting 3:

\[ q_3'(S,C,P) : - r(S',C,Q), t(P,C,Q), r(S,C,Q), t(P',C,Q), r(S'',C,Q'), \]
\[ c(C,T), t(P'',C,Q'), Q' \leq \text{Aut2008} \]

- The green subgoals can cover the black ones under the mapping:
  \( S' \rightarrow S, S'' \rightarrow S, P' \rightarrow P, P'' \rightarrow P, Q' \rightarrow Q \)

Minimized rewriting 3:

\[ q_{3m}(S,C,P) : - V_2(S,P,C,Q), V_4(P,C,T,Q) \]

Verify that there are only two rewritings that are not covered by others

Maximally Contained Rewriting:

\[ q' = q_{1m} \cup q_{3m} \]
The Bucket Algorithm: Second Example

- **Query:** \( q(X) :- \text{cites}(X,Y), \text{cites}(Y,X), \text{sameTopic}(X,Y) \)
- **Views:**
  - \( V4(A) :- \text{cites}(A,B), \text{cites}(B,A) \)
  - \( V5(C,D) :- \text{sameTopic}(C,D) \)
  - \( V6(F,H) :- \text{cites}(F,G), \text{cites}(G,H), \text{sameTopic}(F,G) \)

**Note:** Should we list \( V4(X) \) twice in the buckets?

- Consider all combinations & containment check of the unfolded rewriting
- \( V4(X) \) cannot be combined with anything (why?)
  - Try \( q1(X) :- V4(X), V4(X), V5(X,Y) \)
  - Try \( q2(X) :- V4(X), V6(X,Y), V5(X,Y) \)
- Does any of these work?
- When can we discard a view from consideration?

**Buckets**

<table>
<thead>
<tr>
<th></th>
<th>cites</th>
<th>cites</th>
<th>sameTopic</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V4 )</td>
<td>( V4 )</td>
<td>( V5 )</td>
<td></td>
</tr>
<tr>
<td>( V6 )</td>
<td>( V6 )</td>
<td>( V6 )</td>
<td></td>
</tr>
</tbody>
</table>

Remarks:
- \( V4 \) didn’t contribute to any rewrite, but the bucket algorithm doesn’t recognize it ahead
- On the other hand, consider
  - \( q2(X,Y) :- \text{cites}(X,Y), \text{cites}(Y,X) \)
- Then both cites predicates can be folded into \( V4 \)
  - Not recognized by the bucket algorithm

Here is a successful rewriting:
- \( q3(X) :- V6(X,Y), V6(X,Y), V6(X,Y) \)
- By itself is not contained in \( q \)
- But, with subgoal \( X=Y \) added, it is!
- By minimizing the rewriting, we get:
  - \( q3m(X) :- V6(X,X) \)

Remarks:
- \( V4 \) didn’t contribute to any rewrite, but the bucket algorithm doesn’t recognize it ahead
- On the other hand, consider
  - \( q2(X,Y) :- \text{cites}(X,Y), \text{cites}(Y,X) \)
- Then both cites predicates can be folded into \( V4 \)
  - Not recognized by the bucket algorithm
The State of Affairs

- **Bucket algorithm:**
  - deals well with predicates, Cartesian product can be large (containment check required for every candidate rewriting)

- **Inverse rules:**
  - modular (extensible to binding patterns, FD’s)
  - no treatment of predicates
  - resulting rewritings need significant further optimization

Neither scales up

- **The MiniCon algorithm:**
  - change perspective: look at query variables

The MiniCon Algorithm

- A “much smarter” bucket algorithm:
  - In many cases, we don’t need to perform the cross-product of all items in all buckets
  - Eliminates the need for the containment check

- Basically, a modification to the bucket approach concentrating on variables rather than subgoals (MiniCon Descriptions – MCD)
  - “head homomorphism” – defines what variables must be equated
  - variable-substituted version of the subgoals (mapping of variable names)
  - info about what’s covered
The MiniCon Algorithm

Property 1:
- If a variable occurs in the head of a query, then there must be a corresponding variable in the head of the MCD view
- If a variable participates in a join predicate in the query, then it must be in the head of the MCD view

For each subgoal of the query:
- For each subgoal of each view
  - Choose the least restrictive head homomorphism to match the subgoal of the query
  - If we can find a way of mapping the variables, then add MCD for each possible "maximal" extension of the mapping that satisfies Property 1

MiniCon Example

Query:
q(X) :- cites(X,Y), cites(Y,X), sameTopic(X,Y)

Views:
V4(A) :- cites(A,B), cites(B,A)
V5(C,D) :- sameTopic(C,D)
V6(F,H) :- cites(F,G), cites(G,H), sameTopic(F,G)

Form buckets (aka MCDs) more intelligently:
- Ask what is the minimal set of query subgoals that must be covered (via mappings) by each view
- First, look at join conditions in q
- Then, follow joins on existential variables in views
MiniCon Example

Consider V4(A) :- cites(A,B), cites(B,A)
- Can cover query subgoal cites(X,Y)
- To do this, we map: X[A, Y[B
- Y is an existential join variable
- So V4 needs to cover the query subgoals that contain Y, which are:
  - cites(Y,X) and sameTopic(X,Y)
- V4 can cover the cites subgoal (X[A, Y[B), but not the sameTopic
- Hence, no MCD is created for V4

MiniCon Example

Consider V5(C,D) :- sameTopic(C,D)
- Can cover sameTopic(X,Y) (and nothing else)
- To do this, we map: X[C, Y[D
- The MCD says how the query subgoals may be covered by V5

<table>
<thead>
<tr>
<th>MiniCon Descriptions (MCDs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>V5</td>
</tr>
</tbody>
</table>
MiniCon Example

Consider V6(F,H) :- cites(F,G), cites(G,H), sameTopic(F,G)

- Can cover query subgoal \( \text{sameTopic}(X,Y) \)
- To do this, we map: \( X \mapsto F, Y \mapsto G \)
- \( Y \) is an existential join variable
- So V6 needs to cover the query subgoals that contain \( Y \), which are: cites(\( X,Y \)) and cites(\( Y,X \))
- To do this, we map: \( X \mapsto F, Y \mapsto G \) and \( X \mapsto H \)

We get the following MCD for V6:

**MiniCon Descriptions (MCDs)**

<table>
<thead>
<tr>
<th>View</th>
<th>Mappings</th>
<th>Query Subgoals Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>V5</td>
<td>( X \mapsto C, Y \mapsto D )</td>
<td>3</td>
</tr>
<tr>
<td>V6</td>
<td>( X \mapsto F, Y \mapsto G, X \mapsto H )</td>
<td>1,2,3</td>
</tr>
</tbody>
</table>

**Problem:** Mappings do **not** define a function (homomorphism)

- \( X \) is mapped to both \( F \) and \( H \)
- Can we fix this?
  - Yes, if both \( F \) and \( H \) are **distinguished** in V6
  - No, otherwise
- We fix the problem by making \( F \) and \( H \) **equal** when producing the rewriting:
  - **Query rewriting:** \( q'(F) :- V6(F,F) \)
  - Since V6 covers all query subgoals, no other view is needed!!
MiniCon Algorithm

- When forming MCDs:
  - Join obligations in query via existential variables cannot be fulfilled by another view
  - Unless those variables are mapped to the view’s distinguished variables
  - Current view should fulfill them all

- When combining MCDs:
  - Combined MCDs must cover pairwise disjoint sets of query subgoals
    - Greatly reduces the number of candidates!
    - Also proven to be correct without the use of a containment check!
  - For every possible way of covering all query subgoals, “chain” the corresponding MCDs of views in the covering, and form a rewriting
    - All query subgoals must be covered
    - The union of all such rewritings is the rewriting of the query w.r.t. the given views

MiniCon Second Example

Query:
q(X) :- cites(X,Y), cites(Z,X), inSIGMOD(X)

Views:
V7(A) :- cites(A,B), inSIGMOD(A)
V8(C) :- cites(D,C), inSIGMOD(C)

Step 1:

<table>
<thead>
<tr>
<th>View</th>
<th>Mappings</th>
<th>Query Subgoals Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>V7</td>
<td>X⇒A, Y⇒B</td>
<td>1</td>
</tr>
<tr>
<td>V7</td>
<td>X⇒A</td>
<td>3</td>
</tr>
<tr>
<td>V8</td>
<td>Z⇒D, X⇒C</td>
<td>2</td>
</tr>
<tr>
<td>V8</td>
<td>X⇒C</td>
<td>3</td>
</tr>
</tbody>
</table>
MiniCon Second Example

MiniCon Descriptions (MCDs)

<table>
<thead>
<tr>
<th>View</th>
<th>Mappings</th>
<th>Query Subgoals Covered</th>
</tr>
</thead>
<tbody>
<tr>
<td>V7</td>
<td>X⇒A, Y⇒B</td>
<td>1</td>
</tr>
<tr>
<td>V7</td>
<td>X⇒A</td>
<td>3</td>
</tr>
<tr>
<td>V8</td>
<td>Z⇒D, X⇒C</td>
<td>2</td>
</tr>
<tr>
<td>V8</td>
<td>X⇒C</td>
<td>3</td>
</tr>
</tbody>
</table>

Step 2:
Query rewriting 1: \( q_1(X) : \neg V7(X), V8(X), V7(X) \)
Query rewriting 2: \( q_2(X) : \neg V7(X), V8(X), V8(X) \)

Final rewriting: \( q'(X) : \neg V7(X), V8(X) \)

MiniCon and LAV Summary

- **Variations** need to be made for:
  - Constants in general (I sneaked those in)
  - "Semi-interval" predicates (\( x \leq c \))
    - Note that full-blown inequality predicates are co-NP-hard in the size of the data, so they don’t work
  - State-of-the-art algorithm for answering queries using views (AQUV) in the relational world of data integration
    - It’s been extended to support “conjunctive XQuery” as well
  - Scales to large numbers of views, which we need in LAV data integration
    - Empirically shown to scale to 1000s of views, i.e., sources
  - A similar approach: Chase & Backchase by Tannen et al.
    - Slightly more general in some ways – but:
      - Produces equivalent rewritings, not maximally contained ones
      - Not always polynomial in the size of the data
MiniCon Performance, Many Rewritings

Chain queries with 5 subgoals and all variables distinguished

- **MiniCon**
- **Inverse**
- **Bucket**

Larger Query, Fewer Rewritings

Chain queries; 2 variables distinguished, query of length 12, views of lengths 2, 3, and 4

- **Minicon**
- **Inverse**
Summary

Paradigms of Data Integration

Integration

- global defined from local
  - CWA
    - global-schema-as-view
    - Database Schema Integration
- global “independent” of local
  - OWA
    - global-as-view-of-local
    - Data Warehousing
    - local-as-view-of-global
    - Mediation
More Issues to Tackle

- Performance issues
  - the size of the problem, remember?
  - network behavior (delay, bursting data)
- Source Discovery
- Schema Discovery
- Approximate answers
  - a less precise answer is often better than no (precise) answer at all
- Choosing the right sources wrt to user demands
  - not everybody has the same evaluation criterion for an answer especially when multimedia are involved, e.g. (e.g. most “vivid”, with highest resolution, fastest, most reliable, etc.)

Complexity of Relational Calculus

- We consider the complexity of the recognition problem, i.e., checking whether a tuple of constants is in the answer to a query:
  - measured wrt the size of the database; data complexity
  - measured wrt the size of the query and the database; combined complexity

- Complexity of relational calculus
  - data complexity: \textit{polynomial}, actually in \textit{LogSpace}
  - combined complexity: \textit{PSPACE-complete}

- Complexity of conjunctive queries
  - data complexity: in \textit{LogSpace}
  - combined complexity: \textit{NP-complete}
Complexity of Query Containment

- Conjunctive Queries (CQ) (NP-Complete)
  - Q1: p(X,Z) :- a(X,Y) & a(Y,Z)
  - Q2: p(X,Z) :- a(X,Y) & a(V,Z)

- CQ’s With Negation (\(\Pi^p\)-Complete)
  - Q1: p(X,Z) :- a(X,Y) & a(Y,Z) & NOT a(X,Z)

- CQ’s With Arithmetic Comparison (\(\Pi^p\)-Complete)
  - Q1: p(X,Z) :- a(X,Y) & a(Y,Z) & X < Y

- Datalog Programs
  - p(A,C) :- a(A,B) & b(B,C)

\[
\sum_k^p \cup \prod_k^p \subseteq \Delta_{k+1}^p \subseteq \sum_k^p \cap \prod_k^p
\]
Complexity of View-based Query Answering: Regular Path Queries

<table>
<thead>
<tr>
<th>domain</th>
<th>views</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all sound</td>
<td>$\text{coNP}$</td>
</tr>
<tr>
<td>closed</td>
<td>all exact</td>
<td>$\text{coNP}$</td>
</tr>
<tr>
<td></td>
<td>arbitrary</td>
<td>$\text{coNP}$</td>
</tr>
<tr>
<td>open</td>
<td>all sound</td>
<td>$\text{PSPACE}$</td>
</tr>
<tr>
<td></td>
<td>all exact</td>
<td>$\text{PSPACE}$</td>
</tr>
<tr>
<td></td>
<td>arbitrary</td>
<td>$\text{PSPACE}$</td>
</tr>
</tbody>
</table>

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Moshe Y. Vardi 2000

Readings

- Chapter 9 in “Web Data Management” Book
- For formal foundations and query rewriting algorithms read:
  - Alon Halevy
    - *Data Integration: a Status Report* Invited Talk German Database Conference (BTW), 2003
  - Maurizio Lenzerini
    - *Data Integration: A Theoretical Perspective* Invited Talk PODS 2002
Acknowledgements

- Diego Calvanese
  - **Query Processing in Data Integration** Lecture Slides
- Jeffrey D. Ullman
  - Lecture Slides [www-db.stanford.edu/~ullman/cs345-notes.html](http://www-db.stanford.edu/~ullman/cs345-notes.html)
- Laks VS Lakshmanan
- Zachary G. Ives
  - Lecture Slides [www.seas.upenn.edu/~zives/05s/cis650/](http://www.seas.upenn.edu/~zives/05s/cis650/)
- Michalis Petropoulos
  - Lecture Slides [http://www.cse.buffalo.edu/~mpetropo/CSE636-FA08/](http://www.cse.buffalo.edu/~mpetropo/CSE636-FA08/)