Data Integration & Exchange

Guillaume Raschia — Polytech Nantes ; Nantes Université

Last update: December 4, 2023

Contents

[Source : L. Libkin - Univ. of Edinburgh, 2015]

[Source : A. Doan, A. Halevy and Z. Ives - “Principles of DI” Series of slides, 2012]

References

- **Text Book:** A. Doan, A. Halevy and Z. Ives - *Principles of Data Integration*, 2012
- **Text Book:** M. Arenas, P. Barceló, L. Libkin and F. Murlak - *Foundations of Data Exchange*, 2014
- **Chapter In Text Book:** S. Abiteboul et al. - *Web Data Management*, 2011

Introduction

Traditional Approach to Databases

As It Used To Be[1]

- A single large repository of data
  - may be distributed across several servers and sites
  - but remains under one single authority
- DBMS takes care of lots of things for you
  - query processing and optimisation
  - concurrency control
  - enforcing database integrity

[1] Title of a short film from Clément Gonzalez (2013) - link to the video

Traditional Approach to Databases cont'd

What do we expect from such a system?

1. Data is relatively clean: incompleteness is marginal
2. Data is consistent: enforced by the DBMS
3. Data is available: either on disk or on the local network
4. Queries have a well-defined semantics: you know what you pay for
5. Access to data is controlled

Traditional Approach to Databases cont'd

This model works well within a single organisation that either

- does not interact much with the outside world, or
- the interaction is heavily controlled by the DB administrator

Homogeneity still dominates but the rules are slightly changing...

What Happens These Days

- Many huge repositories are publicly available
  - In fact many are well-organised databases, e.g., `imdb.com`, the CIA World Factbook, many genome databases, open govs data platforms, the DBLP server of CS publications, etc etc etc
- Many queries *cannot* be answered using a single source
- Often data from various sources needs to be combined, e.g.
  - company mergers
  - restructuring databases within a single organisation
  - combining data from several private and public sources

Query Answering from Multiple Sources

- Data resides in several autonomous databases
- They may have different structures, different access policies etc
- Our view of the world may be very different from the view of the databases we need to use
- Only portions of the data from some database could be available
- That is, the sources do not conform to the schema of the database into which the data will be loaded

Heterogeneity is the rule!

What's Off-the-Shelf?

ETL tools: Extract-Transform-Load

1. Extract data from multiple sources
2. Transform it so it is clean and compatible with the target schema
3. Load it into a database

The Players

Big ones

IBM, Microsoft, Oracle, SAP – all have their ETL products;

- Microsoft and Oracle offer them with their database products

(Not so) Outsiders

- A few independent vendors, e.g. Informatica PowerCenter
- (Sort of) local player: Talend (aka. Qlik from 2023), R&D Center in Nantes
- Several open source products exist, e.g. CloverETL

The Gartner Magic Quadrant for Data Integration 2020

Figure 1. Magic Quadrant for Data Integration Tools

Source: Gartner (August 2020)

ETL Tools

Focus

- Data profiling
- Data cleaning
- Simple transformations
- Bulk loading
- Latency requirements

ETL Tools cont'd

What they don't do yet

- nontrivial transformations
- query answering

- But techniques now exist for data integration and for query answering
- They soon will be reflected in products

Data Profiling and Data Cleaning

Data profiling gives the user a view of data

- Samples over large tables
- Statistics (how many different values etc)
- Graphical tools for exploring the database

Cleaning

- Same properties may have different names
  - e.g. `Last_Name`, `LName`, `LastName`
- Same data may have different representations
  - e.g. `(0131)555-1111` vs `01315551111`,
  - `George Str.` vs `George Street`
- Some data may be just wrong

Data Transformation

- Most transformation rules tend to be simple:
  - Copy attribute `LName` to `Last_Name`
  - Set `age` to be `current year − DOB`
- Heavy emphasis on industry specific formats (MS Word, Excel, PDF, UN/EDIFACT, etc)
- Little to do with the general tasks of data integration

Integration and Exchange

Integration

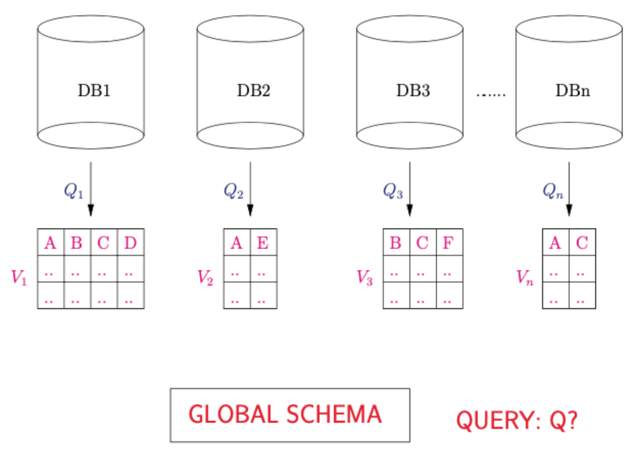
answer queries using multiple sources

- virtual approach, or
- materialization

Exchange

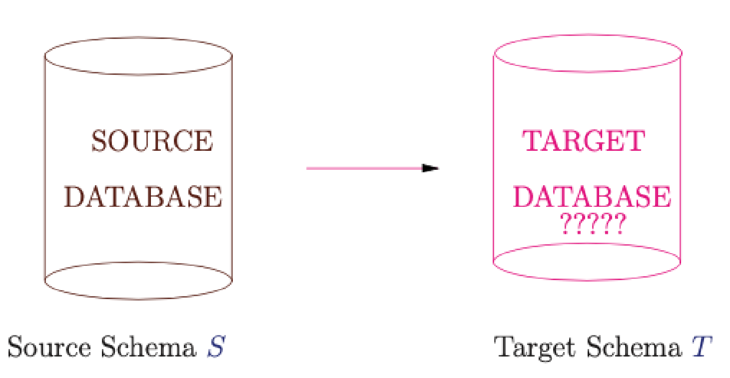
transfer data between two legacy database schemas

Data Integration: Scenario



Answer to Q is obtained by querying the views $V_1, V_2, \ldots, V_n$

Flip Coin: Data Exchange



Query over the target schema: Q

How to answer Q so that the answer is consistent with the source database?

What Changes?

- no clear notion of an answer to a query
- data is not clean: incomplete, inconsistent
- data may not even exist (virtual integration)

Our goal

Study the main concepts and techniques for creating and querying integrated/exchanged data

Logic and Database Queries

Questions

How does a data integration system decides which sources are relevant to a query? Which are redundant? How to combine multiple sources to answer a query?

Answer

By reasoning about the content of data sources

- Data sources are described by queries, by views

This section describes the fundamental tools for manipulating query expressions and reasoning about them.

Outline

- ▹ Review of basic database concepts
- Query unfolding
- Query containment

Relational Terminology

Relational schema

table/relation, attribute/column/field

Relation instance

set (or multi-set) of tuples/rows/records

Integrity constraints

key, foreign key, IND and FD, TGD, EGD

Relational Languages

SQL, RA, RC, Datalog (Rule-based)

Embedded Dependencies (ED)

A fragment of the First-Order logic (FO)

$$\forall \vec{x}, \vec{y} : \phi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} : \psi(\vec{x}, \vec{z}),$$

ϕ is a possibly empty and ψ is an non-empty conjunction of relational and equality atoms

ED

Two flavors

Tuple Generating Dependency (TGD)

Only relational atoms in ψ

Equality Generating Dependency (EGD)

All atoms in ψ are equalities

✏ Express FD, Key, IND, and Foreign Key with ED

Conjunctive Queries: CQ

- As logical statements: RC formula with $\exists$, $\wedge$ only
    - may have interpreted (comparison) predicates

$$\{d, r \mid \exists t, y, s : \text{Movie}(t, d, y) \wedge \text{Rating}(t, r, s) \wedge s > 4\}$$

- As an algebraic expression: Select-Project-Join query (σ, π, and $\bowtie$ only)

$$\pi_{\text{director, reviewer}}(\sigma_{\text{stars}>4}(\text{Movie} \bowtie \text{Rating}))$$

Conjunctive Queries cont'd

- Rule-based language

$$Q(d, r) \text{ :– } \mathsf{Movie}(t, d, y), \mathsf{Rating}(t, r, s), s > 4$$

  - $Q(d, r)$ is the head of the rule
  - $\mathsf{Movie}(t, d, y), \mathsf{Rating}(t, r, s), s > 4$ is the body
  - an atom like $\mathsf{Movie}(t, d, y)$ is called a subgoal
  - conjunctions ($\wedge$) are replaced by commas
  - head variables (d,r) are distinguished
  - variables that occur in the body but not in the head (t, y, and s) are assumed to be existentially quantified (implicit $\exists$)
  - essentially logic programming notation, without functions

Conjunctive Queries with Negation: $\mathsf{CQ}^{\neg}$

SPJ + Set Difference (\) queries

$$Q(d, r) :\!-\ \text{Movie}(t, d, y), \text{Rating}(t, r, s), s > 4, \neg\text{Rating}(t', r, s'), s' < 2$$

Safe rule

- Every distinguished variable (d, r) must appear in a positive subgoal

Union of CQ

SPJD + Set Union (∪) queries ≡ RA ≡ RC

- Multiple rules with the same head

$$Q(d,r) \coloneq \text{Movie}(t,d,y), \text{Rating}(t,r,s), s > 4, \neg\text{Rating}(t',r,s'), s' < 2$$
$$Q(d,r) \coloneq \text{Movie}(t,d,y), \text{Rating}(t,r,s), \text{Reviewer}(r,a), a = \text{`Nantes'}$$

Outline

- ✓ Review of basic database concepts
- ▹ Query unfolding
- · Query containment

Query Unfolding

- Query composition is an important mechanism for writing complex queries
  - Build query from views in a bottom up fashion
- Query unfolding “unwinds” query composition
- Important for:
  - Comparing between queries expressed with views
  - Query optimization (to examine all possible join orders)
  - Unfolding may even discover that the composition of two satisfiable queries is unsatisfiable! (✏ find such an example)

Query Unfolding: Example

Database schema: Flight(source, destination) and Hub(city)

$$Q_1(x, y) \coloneq \text{Flight}(x, z), \text{Hub}(z), \text{Flight}(z, y)$$
$$Q_2(x, y) \coloneq \text{Hub}(z), \text{Flight}(z, x), \text{Flight}(x, y)$$
$$Q_3(x, z) \coloneq Q_1(x, y), Q_2(y, z)$$

The unfolding of Q_3 is:

$$Q_3'(x, z) \coloneq \text{Flight}(x, u), \text{Hub}(u), \text{Flight}(u, y), \text{Hub}(w), \text{Flight}(w, y), \text{Flight}(y, z)$$

- mapping $f = \{x/x, y/y, z/u\}$[2] in Q_1 and $g = \{x/y, y/z, z/w\}$ in Q_2

[2] z/u denotes $f(z) = u$ and it roughly means “replace z by u”.

Query Unfolding Algorithm

1. Find a subgoal $P(x_1, \ldots, x_n)$ such that P is defined by a rule r
2. Unify $P(x_1, \ldots, x_n)$ with the head of r
3. Replace $P(x_1, \ldots, x_n)$ with the result of applying the unifier to the subgoals of r (use fresh variables for the existential variables of r)
4. Iterate until no unification can be found
5. If P is defined by a union of rules r_1, ..., r_m, create m rules of the main query, one for each of the r's

Query Unfolding: Summary

- Unfolding does not necessarily create a more efficient query!
  - Just let's the optimizer explore more evaluation strategies
  - Unfolding is the opposite of rewriting queries using views (see later on)
- The size of the resulting query can grow exponentially (✏ show how)

Outline

- ✓ Review of basic database concepts
- ✓ Query unfolding
- ▹ Query containment

Query Containment: Motivation

Intuitively, the unfolding of Q_3 is **equivalent** to Q_3'':

$$Q_3'(x, z) \coloneq \mathsf{Flight}(x, u), \mathsf{Hub}(u), \mathsf{Flight}(u, y), \mathsf{Hub}(w), \mathsf{Flight}(w, y), \mathsf{Flight}(y, z)$$

$$Q_3''(x, z) \coloneq \mathsf{Flight}(x, u), \mathsf{Hub}(u), \mathsf{Flight}(u, y), \mathsf{Flight}(y, z)$$

since, mainly, Hub(w) and Flight(w, y) can be unified with the same tuples than Hub(u) and Flight(u, y)

- How can we justify this intuition formally?

Query Containment: Motivation cont'd

Furthermore, the query Q_4 that requires going through *two* hubs is **contained** in Q_3'' (and Q_3' as well)

$$Q_3''(x,z) :- \mathsf{Flight}(x,u), \mathsf{Hub}(u), \mathsf{Flight}(u,y), \mathsf{Flight}(y,z)$$
$$Q_4(x,z) :- \mathsf{Flight}(x,u), \mathsf{Hub}(u), \mathsf{Flight}(u,y), \mathsf{Hub}(y), \mathsf{Flight}(y,z)$$

- We need algorithms to detect these relationships

Query Containment and Equivalence: Definitions

Definition (Query Containment)

Query Q_1 is contained in query Q_2 if for every database D, $Q_1(D) \subseteq Q_2(D)$

Definition (Query Equivalence)

Query Q_1 is equivalent to query Q_2 if for every database D:

$$Q_1(D) \subseteq Q_2(D) \quad \text{and} \quad Q_2(D) \subseteq Q_1(D)$$

Note

Containment and equivalence are properties of the queries, not the database!

Fact Checking #1

$$Q_1(x, z) :- P(x, y, z)$$
$$Q_2(x, z) :- P(x, x, z)$$

$$Q_2 \sqsubseteq Q_1$$

Fact Checking #2

$$Q_1(x, y) :- P(x, z), P(z, y)$$

$$Q_2(x, y) :- P(x, z), P(z, y), P(x, w)$$

$Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$

Why Do We Need It?

- When sources are described as views, we use containment to compare among them
- If we can remove subgoals—joins—from a query, its evaluation is more efficient
- Actually, containment arises everywhere...

Reconsidering the Example

Database schema: Flight(src, dest) and Hub(city)

Views:

$$Q_1(x, y) \text{ :– } \text{Flight}(x, z), \text{Hub}(z), \text{Flight}(z, y)$$
$$Q_2(x, y) \text{ :– } \text{Hub}(z), \text{Flight}(z, x), \text{Flight}(x, y)$$

And query: $Q_3(x, z) \text{ :– } Q_1(x, y), Q_2(y, z)$

The unfolding of Q_3 is:

$$Q_3'(x, z) \text{ :– } \text{Flight}(x, u), \text{Hub}(u), \text{Flight}(u, y), \text{Hub}(w), \text{Flight}(w, y), \text{Flight}(y, z)$$

Remove Redundant Subgoals

$$Q_3'(x, z) \coloneq \text{Flight}(x, u), \text{Hub}(u), \text{Flight}(u, y), \text{Hub}(w), \text{Flight}(w, y), \text{Flight}(y, z)$$

$$Q_3''(x, z) \coloneq \text{Flight}(x, u), \text{Hub}(u), \text{Flight}(u, y), \text{Flight}(y, z)$$

Is Q_3'' truly equivalent to Q_3'?

$\Leftrightarrow$ Do both $Q_3'' \sqsubseteq Q_3'$ and $Q_3' \sqsubseteq Q_3''$ hold?

Containment: Conjunctive Queries

$$Q(\vec{x}) \coloneq G_1(\vec{x_1}), \ldots, G_n(\vec{x_n})$$

- Assume there is no interpreted predicates ($\leq$, $\neq$) nor negative subgoal ($\neg$) …at least for now

Recall semantics

If f maps the body subgoals to tuples in D, then $f(\vec{x})$ is an anwser

Containment Mapping

$$Q_1(\vec{x}) \coloneq R_1(\vec{x_1}), \ldots, R_n(\vec{x_n})$$
$$Q_2(\vec{y}) \coloneq S_1(\vec{y_1}), \ldots, S_m(\vec{y_m})$$

Definition (Containment Mapping, aka. Homomorphism)

$$f : \text{Variables}(Q_1) \longrightarrow \text{Variables}(Q_2)$$

is a containment mapping from Q_1 to Q_2 if:

$$f(R_i(\vec{x_i})) \in \text{Body}(Q_2), \quad \forall i \in [\![1, n]\!] \tag{1}$$
$$\text{and} \qquad f(\vec{x}) = \vec{y} \tag{2}$$

Containment Mapping: Example

$$Q_3'(x, z) \coloneq \mathsf{Flight}(x, u), \mathsf{Hub}(u), \mathsf{Flight}(u, y), \mathsf{Hub}(w), \mathsf{Flight}(w, y), \mathsf{Flight}(y, z)$$

$$Q_3''(x, z) \coloneq \mathsf{Flight}(x, u), \mathsf{Hub}(u), \mathsf{Flight}(u, y), \mathsf{Flight}(y, z)$$

$f = \{x/x, y/y, u/u, w/u, z/z\}$ is a containment mapping from Q_3' to Q_3''

1. subgoals: $\mathsf{Hub}(w) \overset{f}{\mapsto} \mathsf{Hub}(u)$, and $\mathsf{Flight}(w, y) \overset{f}{\mapsto} \mathsf{Flight}(u, y)$, both in Body($Q_3''$)
2. head variables: $f(x) = x$ and $f(z) = z$

Containment Mapping: Main Result

Theorem [Chandra and Merlin, 1977]

Q_1 contains Q_2 (denoted $Q_2 \sqsubseteq Q_1$) if and only if there is a containment mapping f from Q_1 to Q_2

Deciding whether Q_1 contains Q_2 is NP-complete

Sketch of the Proof

$$Q_1(\vec{x}) \text{ :- } R_1(\vec{x_1}), \ldots, R_n(\vec{x_n})$$
$$Q_2(\vec{y}) \text{ :- } S_1(\vec{y_1}), \ldots, S_m(\vec{y_m})$$

The ‘if’ direction

Assume f exists: $f(R_i(\vec{x_i})) \in \text{Body}(Q_2)$ and $f(\vec{x}) = \vec{y}$

Let t be an answer to Q_2 over database D; then there is a unification g from the variables of Q_2 to D

Hence, g∘f is a unification from Variables(Q_1) to D and t is also an answer to Q_1 #

Sketch of the Proof cont'd

The 'only-if' direction

Assume the containment $Q_2 \sqsubseteq Q_1$ holds;

Consider the **frozen database** D' of Q_2:

- Variables of Q_2 are constants in D'

The unification from Q_1 to D' gives the containment mapping!

Frozen Database: Example

$$Q(x,z) \coloneq \text{Flight}(x,u), \text{Hub}(u), \text{Flight}(u,y), \text{Flight}(y,z)$$

Frozen DB for $Q(x,z)$

Flight = $\{(x,u),(u,y),(y,z)\}$

Hub = $\{(u)\}$

Two Views of This Result

1. **Variable mapping**:
   - a condition on variable mappings that guarantees containment
2. **Representative (canonical) databases**:
   - a (single) database that would offer a counter-example if any

Containment results typically fall into one of these two classes

Union of Conjunctive Queries

$$r_{1.1}: \quad Q_1(x, y) :\!- \mathsf{Flight}(x, z), \mathsf{Flight}(z, y)$$
$$r_{1.2}: \quad Q_1(x, y) :\!- \mathsf{Flight}(x, z), \mathsf{Flight}(y, z), \mathsf{Hub}(z)$$

and

$$Q_2(x, y) :\!- \mathsf{Flight}(x, z), \mathsf{Flight}(z, y), \mathsf{Hub}(z)$$

One can observe $Q_2 \sqsubseteq r_{1.1}$

Theorem

A CQ is contained in a union of CQ's iff it is contained in one of the conjunctive rules

Corollary

Containment is still NP-complete!

CQ's with Interpreted Predicates

A tweak on containment mappings provides a sufficient condition

$$Q_1(\vec{x}) \coloneq R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}), C_1$$
$$Q_2(\vec{y}) \coloneq S_1(\vec{y_1}), \ldots, S_m(\vec{y_m}), C_2$$

Containment mapping revisited

$$f : \text{Variables}(Q_1) \rightarrow \text{Variables}(Q_2), \text{ with}$$

$$f(R_i(\vec{x_i})) \in \text{Body}(Q_2), \quad \forall i \in [\![1, n]\!] \tag{3}$$
$$f(\vec{x}) = \vec{y} \tag{4}$$
$$\text{and} \qquad C_2 \models f(C_1) \tag{5}$$

Containment Mapping of CQ with IP: Example

$$Q_1(x, y) \coloneq \text{Flight}(x, z), \text{Flight}(z, y), \text{Population}(z, p), p \leq 500,000$$

$$Q_2(u, v) \coloneq \text{Flight}(u, w), \text{Flight}(w, v), \text{Hub}(w), \text{Population}(w, s), s \leq 100,000$$

Building the mapping f from Q_1 to Q_2 :

- $x \overset{f}{\mapsto} u$, $y \overset{f}{\mapsto} v$, $z \overset{f}{\mapsto} w$ and $p \overset{f}{\mapsto} s$, that one can denote $f = \{x/u, y/v, z/w, p/s\}$
- $s \leq 100,000 \models s \leq 500,000$

Containment Mapping: Not a Necessary Condition

$$Q_1(x, y) \coloneq R(x, y), S(u, v), u \leq v$$
$$Q_2(x, y) \coloneq R(x, y), S(u, v), S(v, u)$$

No containment mapping from Q_1 to Q_2:

- x/x and y/y for the head variables
- u/u and v/v to map the S subgoal
- but $\perp \not\models u \leq v$

Anyway, we have $Q_2 \sqsubseteq Q_1$!

Query Refinement

$$Q_1(x, y) \coloneq R(x, y), S(u, v), u \leq v$$
$$Q_2(x, y) \coloneq R(x, y), S(u, v), S(v, u)$$

We consider an equivalent rewriting of Q_2:

$$Q_2'(x, y) \coloneq R(x, y), S(u, v), S(v, u), u \leq v$$
$$Q_2'(x, y) \coloneq R(x, y), S(u, v), S(v, u), u > v$$

Q_2' rules are the **refinements** of Q_2

Query Refinement cont'd

$$Q_1(x,y) \coloneq R(x,y), S(u,v), u \leq v$$

$$r_{2.1} : Q'_2(x,y) \coloneq R(x,y), S(u,v), S(v,u), u \leq v$$

$$r_{2.2} : Q'_2(x,y) \coloneq R(x,y), S(u,v), S(v,u), u > v$$

Two containment mappings can then be defined:

1. from Q_1 to $r_{2.1}$: $\{x/x, y/y, u/u, v/v\}$ with $u \leq v \models u \leq v$, and
2. from Q_1 to $r_{2.2}$: $\{x/x, y/y, u/v, v/u\}$ with $u > v \models v \leq u$

Constructing Query Refinements

1. Consider all **complete orderings** of the variables and constants in the query
2. For each complete ordering, create a conjunctive query
3. The result is the union of conjunctive queries

Complete Ordering

- Given
  - a conjunction C of interpreted atoms over
  - a set of variables $x_1, \dots, x_n$, and
  - a set of constants $a_1, \dots, a_m$
- C_T is a **complete ordering** if:
  - $C_T \models C$, and
  - $\forall \vartheta_1, \vartheta_2 \in \{x_1, \dots, x_n, a_1, \dots, a_m\}$

$$C_T \models \vartheta_1 < \vartheta_2 \quad \text{or} \quad C_T \models \vartheta_1 > \vartheta_2 \quad \text{or} \quad C_T \models \vartheta_1 = \vartheta_2$$

Back to Query Refinement

$$Q_1(\vec{x}) \coloneq R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}), C_1$$

Let C_T be a complete ordering of C_1; then

$$Q'_1(\vec{x}) \coloneq R_1(\vec{x_1}), \ldots, R_n(\vec{x_n}), C_T$$

is a refinement of Q_1

Query Containement of CQ with IP

Theorem [Klug, 88; van der Meyden, 92]

Q_1 contains Q_2 if and only if there is a containement mapping from Q_1 to **every refinement** of Q_2

Deciding whether Q_1 contains Q_2 is Σ_2^p-complete

What about Negation?

$$Q_1(\vec{x}) \coloneq R_1(\vec{x}_1), \ldots, R_n(\vec{x}_n), \neg S_1(\vec{y}_1), \ldots, \neg S_m(\vec{y}_m)$$
$$Q_2(\vec{w}) \coloneq T_1(\vec{w}_1), \ldots, T_k(\vec{w}_k), \neg U_1(\vec{z}_1), \ldots, \neg U_\ell(\vec{z}_\ell)$$

Queries are assumed to be safe:

- every head variable appears in a positive subgoal in the body

Containment mapping revisited

Map negative subgoals in Q_1 to negative subgoals in Q_2

This is a sufficient condition, but not a necessary one

Bag Semantics

Flight =

Source	Destination	Departure Time
Nantes	Paris	8am
Nantes	Paris	10am
Paris	Lyon	1pm

$$Q(x, y) \coloneq \text{Flight}(x, z, w), \text{Flight}(z, y, t)$$

- Answers under set semantics: {(Nantes, Lyon)}
- Answers under bag semantics: {(Nantes, Lyon), (Nantes, Lyon)}

Equivalence Theorem

Q_1 is equivalent to Q_2 if and only if there is a 1-1 containment mapping

Trivial example of non-equivalence

$$Q_1(x) \coloneq P(x)$$
$$Q_2(x) \coloneq P(x), P(x)$$

What about query containment?

Answering Query Using Views

Motivating Example

```
Movie(mID, title, year, genre)
Director(mID, dName)
Actor(mID, aName)
```

$$Q(t, y, d) \coloneq \text{Movie}(i, t, y, g), y \geq 1950, g = comedy, \text{Director}(i, d), \text{Actor}(i, d)$$
$$V_1(t, y, d) \coloneq \text{Movie}(i, t, y, g), y \geq 1940, g = comedy, \text{Director}(i, d), \text{Actor}(i, d)$$

- Obviously $Q \sqsubseteq V_1$, hence $Q_0(t, y, d) \coloneq V_1(t, y, d), y \geq 1950$
- Containment is enough to show that V_1 can be used to answer Q

Motivating Example cont'd

$$Q(t, y, d) :\!- \text{Movie}(i, t, y, g), y \geq 1950, g = comedy, \text{Director}(i, d), Actor(i, d)$$
$$V_2(i, t, y) :\!- \text{Movie}(i, t, y, g), y \geq 1950, g = comedy$$
$$V_3(i, d) :\!- \text{Director}(i, d), \text{Actor}(i, d)$$

- Containment does not hold,
- but intuitively, V_2 and V_3 are useful for answering Q

How do we express that intuition? Answering queries using views!

Outline

- ▹ Review of basic database concepts
- Query unfolding
- Query containment

Schema Mapping Languages

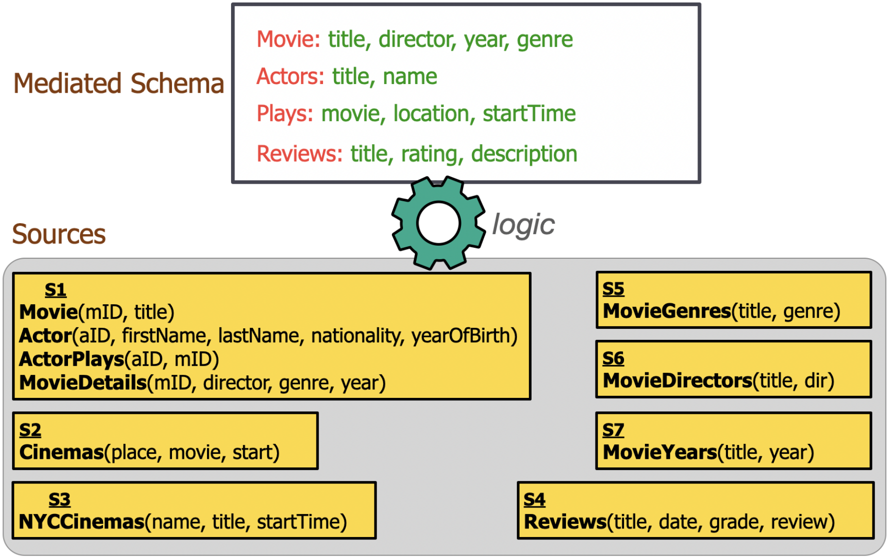
Motivation

How to map Mediated (aka. Global) Schema to Source Schemas?

Semantic Heterogeneity

- Difference in:
    - Naming of schema elements
    - Organization of tables
    - Coverage and detail of schema
    - Data-level representation: John Doe vs. J. Doe
- Why?
    - shema probably designed for different apps, by different people

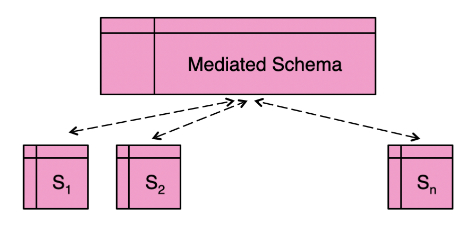
Schema Heterogeneity by Example



Principles of Schema Mapping

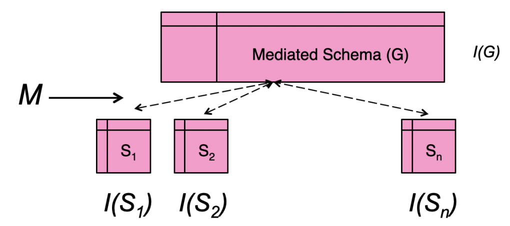
Schema mapping describes the relation between:

or between:



Semantics of Schema Mapping

Formally, schema mapping states: which instances of the mediated schema are consistent with the current instances of the data sources



$\mathbf{I}(G) \times (\mathbf{I}(S_i))_{1\leq i\leq n}$: the set of possible instances of the schema $(G, (S_i)_{1\leq i\leq n})$

$$\mathrm{M} \subseteq \mathbf{I}(G) \times \mathbf{I}(S_1) \times \ldots \times \mathbf{I}(S_n)$$

Possible Instances of the Mediated Schema: Simple Example

- Source 1: `(Director, Title, Year)` with tuples
  - {(Allen, Manhattan, 1979),
  - (Coppola, GodFather, 1972)}
- Mediated schema: `(Title, Year)`
  - Simple projection of Source 1
  - Only one possible instance: {(Manhattan, 1979), (GodFather, 1972)}

Possible Instances of the Mediated Schema: Second Example

- Source 1: `(Title, Year)` with tuples
  - {(Manhattan, 1979), (GodFather, 1972)}
- Mediated schema: `(Director, Title, Year)`
  - Possible instance 1: {(Allen, Manhattan, 1979), (Coppola, GodFather, 1972)}
  - Possible instance 2: {(Alice, Manhattan, 1979), (Bob, GodFather, 1972)}
- Why it is so important to know?
  - This matters at query time (see next slide)

Answering Queries over Possible Instances of Mediated Schema

Mediated schema: `(Director, Title, Year)`

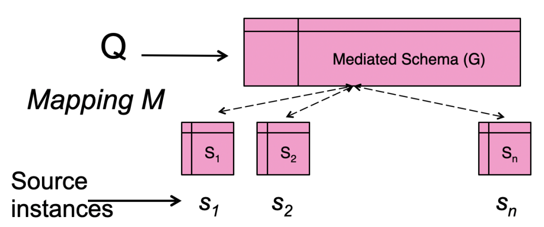
- Possible instance 1: {(Allen, Manhattan, 1979), (Coppola, GodFather, 1972)}
- Possible instance 2: {(Alice, Manhattan, 1979), (Bob, GodFather, 1972)}

- Query Q_1: return all years of movies
  - Answer: (1979, 1972) are certain answers
- Query Q_2: return all directors
  - No certain answer because no directors appear in all possible instances of the mediated schema

Certain Answers Make it Formal

An answer is **certain** if it is **true in every instance** of the mediated schema that is consistent with:

1. the instances of the sources, and
2. the mapping M

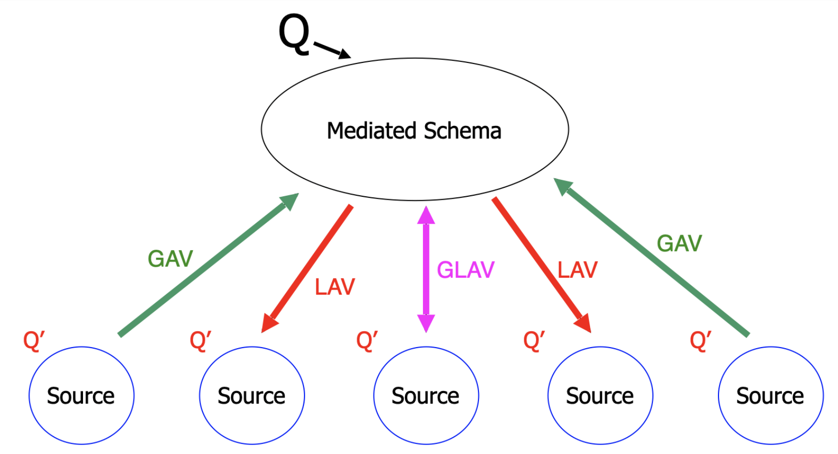


$$\text{certain}(Q, s_1, \ldots, s_n) = \bigcap Q(g), \quad \forall g \text{ such that } (g, s_1, \ldots, s_n) \in \mathrm{M}$$

Desiderata from Source Description Languages

- Flexibility:
  - Should be able to express relationships between real schemata
- Efficient reformulation:
  - Computational complexity of reformulation and finding answers
- Easy update:
  - Should be easy to add and delete sources

Languages for Schema Mapping



Global-As-View

Mediated schema is defined as a set of views over the data sources

- G : `Movie(title, director, year, genre)`
- S_1 :

  ```
  Movie(mID, title)
  MovieDetails(mDI, director, genre, year)
  ```

a sound GAV setting :

Movie(title, director, year, genre) $\supseteq$

S_1.Movie(mID, title), S_1.MovieDetails(mID, director, genre, year)

GAV: Formal Definition

A set of expressions of the form:

$$G_i(\vec{x}) \supseteq Q_i(\vec{S}) \qquad \text{or} \qquad G_i(\vec{x}) = Q_i(\vec{S})$$

resp. for sound (OWA) or exact (CWA) setting

- G_i is a relation in the mediated schema
- $Q_i(\vec{S})$ is a query over source relations

GAV Example

- Movie(title, director, year, genre) $\supseteq$
  S_1.Movie(mID, title), S_1.MovideDetails(mID, director, genre, year)
- Movie(title, director, year, genre) $\supseteq$
  S_5.MovieGenres(title, genre), S_6.MovieDirectors(title, director),
  S_7.MovieYear(title, year)

GAV Example cont'd

Remind global schema `Plays(movie, location, starTime)`

- Plays(m, ℓ, s) $\supseteq$ S_2.Cinemas(ℓ, m, s)
- Plays(m, ℓ, s) $\supseteq$ S_3.NYCCinemas(ℓ, m, s)

Reformulation in GAV

- Given a query Q on the mediated schema G;
  - Return the best query possible Q' on the data sources S_1,..., S_n
  - Global views unfolding is the swiss knife
- Query:

$$Q(t, \ell, s) :\!-\ \text{Movie}(t, d, y, \text{`comedy'}), \text{Plays}(t, \ell, s), s \geq \text{8pm}$$

- Two first GAV rules to unfold the Movie and Plays subgoals of Q:
  - Movie(title, director, year, genre) $\supseteq$
    S_1.Movie(mID, title), S_1.MovideDetails(mID, director, genre, year)
  - Plays(m, ℓ, s) $\supseteq$ S_2.Cinemas(ℓ, m, s)

First Reformulation

$$Q(t, \ell, s) \coloneq \text{Movie}(t, d, y, \text{‘comedy’}), \text{Plays}(t, \ell, s), s \geq 8\text{pm}$$

becomes

$$\begin{aligned} Q'(t, \ell, s) \coloneq\ & S_1.\text{Movie}(i, t), S_1.\text{MovieDetails}(i, d, \text{‘comedy’}, y), \\ & S_2.\text{Cinemas}(\ell, t, s), s \geq 8\text{pm} \end{aligned} \tag{6}$$

Another Reformulation

$$Q(t, \ell, s) \text{ :– } \text{Movie}(t, d, y, \text{'comedy'}), \text{Plays}(t, \ell, s), s \geq \text{8pm}$$

becomes

$$Q'(t, \ell, s) \text{ :– } S_1.\text{Movie}(i, t), S_1.\text{MovieDetails}(i, d, \text{'comedy'}, y), S_3.\text{NYCCinemas}(\ell, t, s), s \geq \text{8pm} \tag{7}$$

GAV Semantics

- Recall:
  - $(g, s_1, \ldots, s_n) \in$ M if $g \in \mathbf{I}(G)$ is a global database that is consistant with all the source extensions $(s_1, \ldots, s_n) \in \mathbf{I}(S_1) \times \ldots \times \mathbf{I}(S_n)$
- Then,
  - $G_i(\vec{x}) \supseteq Q_i(\vec{S})$ states the extension of G_i in g is a superset of evaluating Q_i on the sources, or
  - $G_i(\vec{x}) = Q_i(\vec{S})$ states the extension of G_i in g is equal to evaluating Q_i on the sources

Tricky GAV Example

S_8: stores pairs of (actor, director)

- Then the Actors and Movie global schema can be defined as views like
  - Actors($\perp$, actor) $\supseteq S_8$(actor, director)
  - Movie($\perp$, director, $\perp$, $\perp$) $\supseteq S_8$(actor, director)

Tricky GAV Example cont'd

- GAV setting:
  - Actors($\perp$, actor) $\supseteq S_8$(actor, director)
  - Movie($\perp$, director, $\perp$, $\perp$) $\supseteq S_8$(actor, director)
- Given the S_8 extension: {(Keaton, Allen), (Pacino, Coppola)}
- We'd get tuples for the mediated schema:
  - Actors $\supseteq$ {($\perp$, Keaton), ($\perp$, Pacino)}
  - Movie $\supseteq$ {($\perp$, Allen, $\perp$, $\perp$), ($\perp$, Coppola, $\perp$, $\perp$) }

Tricky GAV Example cont'd

- g extension:
  - Actors $\supseteq$ {($\perp$, Keaton), ($\perp$, Pacino)}
  - Movie $\supseteq$ {($\perp$, Allen, $\perp$, $\perp$), ($\perp$, Coppola, $\perp$, $\perp$) }
- Can't answer the query :

$$Q(a, d) :\!-\ \text{Actors}(t, a), \text{Movie}(t, d, g, y)$$

Actually, LAV (Local-As-View) setting will solve this problem

GAV Summary

- Mediated schema is defined as views over the sources
- Reformulation/unfolding is conceptually easy
    - Polynomial-time reformulation and query answering
- GAV forces everything into the mediated schema's perspective
    - Cannot capture a variety of tabular organizations

Local-As-View (LAV)

Data sources defined as views over the mediated schema

S5
MovieGenres(title, genre)

S6
MovieDirectors(title, dir)

S7
MovieYears(title, year)

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

- S_5.MovieGenres(t, g) $\subseteq$ Movie(t, d, y, g)
- S_6.MovieDirector(t, d) $\subseteq$ Movie(t, d, y, g)

LAV cont'd

S8
ActorDirectors(actor, dir)

Movie: title, director, year, genre
Actors: title, name
Plays: movie, location, startTime
Reviews: title, rating, description

- S_8.ActorDirectors(a, d) $\subseteq$ Movie(t, d, y, g), Actor(t,a), y$\geq$ 1970

LAV: Formal Definition

A set of expressions of the form:

$$S_i(\vec{x}) \subseteq Q_i(G) \qquad \text{or} \qquad S_i(\vec{x}) = Q_i(G)$$

resp. for sound (OWA) or exact (CWA) setting

- S_i is a source relation
- $Q_i(G)$ is a query over mediated schema

LAV Semantics

- Recall:
  - $(g, s_1, \ldots, s_n) \in$ M if $g \in \mathbf{I}(G)$ is a global database that is consistant with all the source extensions $(s_1, \ldots, s_n) \in \mathbf{I}(S_1) \times \ldots \times \mathbf{I}(S_n)$
- Then,
  - $S_i(\vec{x}) \subseteq Q_i(G)$ states that the result of Q_i over g is a superset of s_i, or
  - $S_i(\vec{x}) = Q_i(G)$ states that the result of Q_i over g is equal to s_i

Possible Databases

Unlike GAV, LAV definitions imply a set of possible databases for the mediated schema

- S_8.ActorDirectors$(a, d) \subseteq$ Movie(t, d, y, g), Actor(t, a)
- S_8 extension is {(Keaton, Allen)}
- Two possible databases for the mediated schema are:
  - Movie={(Manhattan, Allen, 1979, comedy)} and Actor={(Manhattan, Keaton)}
  - Movie={(Foobar, Allen, 1979, comedy)} and Actor={(Foobar, Keaton)}

Possible Databases cont'd

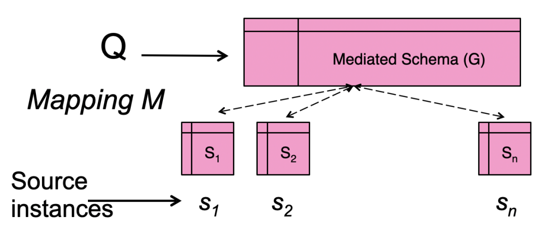
In a sound setting only, since the source may be incomplete, other tuples may be in the instance of the mediated schema:

- Movie={(Manhattan, Allen, 1979, comedy), (Leatherheads, Clooney, 2008, comedy)}
- Actor={(Manhattan, Keaton), (The Godfather, Keaton)}

Certain Answers: A Gentle Reminder

An answer is **certain** if it is **true in every instance** of the mediated schema that is consistent with:

1. the instances of the sources, and
2. the mapping M



$$\text{certain}(Q, s_1, \ldots, s_n) = \bigcap Q(g), \quad \forall g \text{ such that } (g, s_1, \ldots, s_n) \in \mathrm{M}$$

Certain Answers: Example #1

- S_8.ActorDirectors$(a, d) \subseteq$ Movie(t, d, y, g), Actor(t, a)
- S_8 extension is {(Keaton, Allen)}

$$Q(a, d) \coloneq \text{Movie}(t, d, y, g), \text{Actor}(t, a)$$

- Only one certain answer: (Keaton, Allen)

Certain Answers: Example #2

- S_9(dir) $\subseteq$ Director(t, dir), with s_9 = {Allen}
- S_{10}(actor) $\subseteq$ Actor(t, actor), with s_{10} = {Keaton}

$$Q(a, d) \coloneq \text{Director}(i, d), \text{Actor}(i, a)$$

- Under CWA: every possible DB follows the pattern
  - Director={(x, Allen)}, and Actor={(x, Keaton)}
- Then the only exact certain answer is (Keaton, Allen)
- Under OWA: no sound certain answer!

Reformulation in LAV

- We're given tuples for the sources (expressed as views)
- We're given a mediated schema (but no tuple)
- We have a query against that mediated schema

This is exactly the problem of Answering Queries Using Views!

LAV: Summary

- Reformulation = **answering queries using views**
- Algorithms work well in practice:
    - Reformulation is not the bottleneck
    - Under some conditions, guaranteed to find **all certain answers**
        - In practice, they typically do
    - LAV expresses **incomplete information**
        - GAV does not: only a single instance of the mediated schema is consistent with sources

LAV Limitations

S1
Movie(**mID**, title)
Actor(aID, firstName, lastName, nationality, yearOfBirth)
ActorPlays(aID, **mID**)
MovieDetails(**mID**, director, genre, year)

Movie: title, director, year, genre

- If a key is internal to as data source, LAV cannot use it
- So...

GLAV

Global-and-Local-As-View: The best of all Worlds

- A set of expressions of the form:

$$Q^S(\vec{x}) \subseteq Q^G(\vec{x}) \qquad \text{or} \qquad Q^S(\vec{x}) = Q^G(\vec{x})$$

  resp. for sound (OWA) or exact (CWA) setting

- Q^S is a query over the data sources
- Q^G is a query over the mediated schema

GLAV: Example

S_1.Movie(i, t), S_1.MovieDetails(i, d, g, y) $\subseteq$ Movie(t, d, "comedy", y), y $\geq$ 1970

Reformulation in GLAV

- Given a query Q
- Remind the GLAV setting pattern: $Q^S(\vec{x}) \subseteq Q^G(\vec{x})$, then
    1. Find a rewriting Q' using the views $Q_1^G, \ldots, Q_n^G$
    2. Create Q'' by replacing each Q_i^G by the corresponding Q_i^S, then
    3. Unfold $Q_1^S, \ldots, Q_n^S$ to get Q'''

Alternative Notation for GLAV

Tuple Generating Dependencies (TGD) can be used to specify GLAV expressions

$$(\forall \vec{x}) S_1(\vec{x_1}) \wedge \ldots \wedge S_n(\vec{x_n}) \rightarrow (\exists \vec{y}) G_1(\vec{y_1}) \wedge \ldots \wedge G_n(\vec{y_n})$$

is equivalent to the GLAV expression $Q^S(\vec{x}) \subseteq Q^G(\vec{x})$ where

- $Q^S(\vec{x}) \text{ :- } S_1(\vec{x_1}), \ldots, S_n(\vec{x_n})$
- $Q^G(\vec{x}) \text{ :- } G_1(\vec{y_1}), \ldots, G_n(\vec{y_n})$

GLAV to TGD Example

From

S_1.Movie(i, t), S_1.MovieDetails(i, d, g, y) $\subseteq$ Movie(t, d, "comedy", y), y $\geq$ 1970

To

S_1.Movie(i, t) $\wedge$ S_1.MovieDetails(i, d, g, y) $\longrightarrow$ Movie(t, d, "comedy", y) $\wedge$ y $\geq$ 1970

Reformulation with TGD's can be done relatively straightforwardly with the Inverse-Rules algorithm