View Integration

José Luis Ambite
USC/Information Sciences Institute

Outline

- Database Theory Background
  - Datalog
  - Query Containment
- Dimensions of Data Integration
  - Architecture
  - Content Descriptions
    - Global-as-View
    - Local-as-View:
      - Bucket Algorithm
      - Inverse-Rules Algorithm
    - Source Capabilities: Recursive Rewritings

Datalog

- Datalog Program = set of datalog rules
- Datalog rule = conjunctive query

\[ \text{big-LA-buyers(Buyer,Seller, Price)} :\]
\[ \begin{array}{l}
\text{head} \\
\text{body}
\end{array} \]
\[
\forall \text{ Buyer, Seller, Price}
\]
\[
\exists \text{ Product} \left( \text{person(Buyer, "Los Angeles")} \land \text{purchase(Buyer, Seller, Product, Price)} \land \text{Price > 10000} \right)
\]
\[
\rightarrow \text{big-LA-buyers(Buyer,Seller, Price)}
\]

Recursion in Datalog

\[ \text{path}(X, Y) :\]
\[ \begin{array}{l}
\text{arc}(X, Y) \\
\text{path}(X, Z), \text{path}(Z, Y)
\end{array} \]

Semantics: evaluate the rules bottom-up until a fixpoint:
- Iteration #0: \( \{ \text{arc}(a,b), \text{arc}(a,c), \text{arc}(b,d), \text{arc}(c,d), \text{arc}(d,e) \} \)
- Iteration #1: \( \{ \text{path}(a,b), \text{path}(a,c), \text{path}(b,d), \text{path}(c,d), \text{path}(d,e) \} \)
- Iteration #2: path gets the new tuples: \( \{ \text{arc}(a,e) \} \)
- Iteration #3: Nothing changes => stop.

Conjunctive Queries and Views

CREATE VIEW Big-LA-buyers AS
SELECT buyer, seller, price
FROM Person, Purchase
WHERE Person.city = "Los Angeles" AND
Person.name = Purchase.buyer AND
Purchase.price > 10000.

Datalog rule ~ view definition
Rule body ~ select-from-where construct of SQL

Query Containment

- Query Containment: \( q' \subseteq q \)
  - \( \forall D \ q'(D) \sqsubseteq q(D) \)
  - \( q' \models q \)
- Query Equivalence: \( q' = q \leftrightarrow q' \sqsubseteq q \land q \sqsubseteq q' \)
- Complexity of Query Containment
  - Conjunctive Queries (CQ), Union of CQs: NP-complete
  - CQ with comparisons (=, <, \( \neq \)): \( \Pi^2 \)-complete
  - FOL, recursive queries: Undecidable
Query Containment for Conjunctive Queries and Datalog

Method of Canonical Databases
1. Create a canonical database D that is the “frozen” body of q1
2. Compute q2(D)
3. If q2(D) contains the “frozen” head of q1, then q1 ⊆ q2, otherwise not.

Query Containment Example

q1 is the CQ: path(X,Y) :- arc(X,Z) & arc(Z,W) & arc(W,Y)
q2 is the value of path in the following recursive Datalog program:
path(X,Y) :- arc(X,Y)
path(X,Y) :- path(X,Z) & path(Z,Y)
Intuitively, q1 = paths of length 3; q2 = paths of length 1 or more, q1 ⊆ q2

1. Freeze q1, say with 0, 1, 2, 3 as constants for X, Z, W, Y, respectively.
D = {arc(0, 1), arc(1, 2), arc(2, 3)}
Frozen head of q1 is path(0, 3).
2. Compute q2(D) Ext(path) = { (0,1), (1,2), (2,3), (0,2), (1,3), (0,3) }
3. Since frozen head of q1, path(0, 3), is in q2(D) then q1 ⊆ q2

Principal Dimensions of Data Integration

• Virtual vs. materialized architecture
• Access: query only or query & update?
• Mediated schema and query reformulation
  – Content Descriptions
    • Global-as-view
    • Local-as-view
  – Language for descriptions and queries: conjunctive queries (CQs), union of CQs, Datalog (recursion), first-order logic (\(\land,\lor,\neg\)), description logics...
• Types of Sources
  – Structured (DB’s) vs. semi-structured (Web)
  – Source capabilities: positive and negative

Materialized Architecture: Data Warehouse

User
Application
Data Extraction
Data Warehouse
Wrapper
Wrapper
Wrapper
Data Source
Data Source
Data Source

Virtual Architecture: Mediator

User
Mediator
Wrapper
Wrapper
Wrapper
Wrapper
Data Source
Data Source
Data Source

Describing Information Sources

Mediated Schema Relations
Content Descriptions
Information Source Relations

User queries refer to the mediated schema.
Sources store data in their local schemas.
Content descriptions provide the mappings between the mediated and local schemas.
Mediator Architecture

- User queries in global (mediator) schema
- Mediator translates and decomposes user query into multiple source queries

Source Descriptions

Elements of source descriptions:
- Contents: source contains movies, directors, cast.
- Constraints: only movies produced after 1965.
- Completeness: contains all American movies.
- Capabilities:
  - Negative: source requires movie title or director as input
  - Positive: source can perform selections, joins, ...

Desiderata for Source Descriptions

- Distinguish between sources with closely related data: so we can prune access to irrelevant sources.
- Enable easy addition of new information sources: because sources are dynamically being added and removed.
- Be able to find sources relevant to a query: reformulate queries such that we obtain guarantees on which sources we access.

Approaches to Specification of Source Descriptions

- Global-as-View (GAV):
  Mediator relation defined as a view over source relations
  Ex: TSIMMIS (Stanford), HERMES (Maryland).
- Local-as-View (LAV):
  Source relation defined as view over mediator relations
  Ex: Information Manifold (AT&T), Tukwila(UW), InfoMaster (Stanford).

Query Reformulation

Problem: rewrite the user query expressed in the mediated schema into a query expressed in the source schemas.

Given a query \( Q \) in terms of the mediated-schema relations, and descriptions of the information sources,
Find a query \( Q' \) that uses only the source relations, such that
- \( Q' \models Q \) (i.e., answers are correct; i.e., \( Q' \subseteq Q \)) and
- \( Q' \) provides all possible answers to \( Q \) given the sources

Answering queries using views

Given query \( q \) and view definitions \( V \equiv \{ V_1 \ldots V_n \} \)
- \( q' \) is an Equivalent Rewriting of \( q \) using \( V \) if:
  - \( q' \) refers only to views in \( V \), and
  - \( q' = q \)
- \( q' \) is a Maximally-Contained Rewriting of \( q \) using \( V \) if:
  - \( q' \) refers only to views in \( V \), and
  - \( q' \subseteq q \), and
  - there is no rewriting \( q' \), such that \( q' \subseteq q' \subseteq q \) and \( q' \neq q \)
Global-as-View (GAV)

Each mediator relation is defined as a view over source relations.
MovieActor(title,actor) ← DB1(id,title,actor,year)
MovieActor(title,actor) ← DB2(title,director,actor,year)
MovieReview(title,review) ← DB1(id,title,actor,year) ^ DB3(id,review)

Query Reformulation in GAV

Query reformulation = rule unfolding + simplification
Query: Find reviews for 'DeNiro' movies
q(title,review) :- MovieActor(title,'DeNiro'),
                MovieReview(title,review)
1. q'(title,review) :- DB1(id,title,'DeNiro',year),
                  DB2(id,review)
2. q'(title,review) :- DB2(title,director,'DeNiro',year),
                  DB1(id,title,actor,year), DB3(id,review)

Local-as-View (LAV)

Each source relation is defined as a view over mediator relations
V1(title, year, director) ⊆
    Movie(title,year,director,genre) ^
    American(director) ^ year ≥1960 ^
    genre = 'Comedy'
V2(title,review) ⊆ Movie(title,year,director,genre) ^
    year ≥1990 ^ MovieReview(title,review)

Query Reformulation in LAV

Query: Reviews for comedies produced after 1950
q(title,review) :- Movie(title,year,director,'Comedy'),
                  year ≥1950, MovieReview(title,review)
Reformulated query:
q'(title,review) :- V1(title,year,director),
                  V2(title,review)
V1(title,year,director) → Movie(title,year,director,genre) ^
    American(director) ^ year ≥1960 ^ genre = 'Comedy'
V2(title,review) → Movie(title,year,director,genre) ^
    year ≥1990 ^ MovieReview(title,review)

LAV vs. GAV

See [Ullman, ICDT-1997] for a detailed comparison.
- Local-as-View:
  - Easier to add sources: specify the query expression.
  - Easier to specify constraints on contents of the sources: they are part of the query expression describing them.
- Global-as-View:
  - Easier query reformulation
GLAV combines both (Friedman & Millstein 1999)

Query Reformulation in LAV

The Bucket Algorithm

[Levy 1996]

Given: user query q, source descriptions {Vi}
1. Find relevant sources (fill buckets)
   For each relation g in query q
   - Find Vj that contains relation g
   - Check that constraints in Vj are compatible with q
2. Combine source relations {Vj} from each bucket into a conjunctive query q' and check for containment (q' ⊆ q)
The Bucket Algorithm: Example

\[ V(S, number, year) \rightarrow Enrolled(S, student, course, year), Course(course, number) \]
\[ \geq 500, year \geq 1992 \]
\[ V(S, dept, course) \rightarrow Registered(S, student, course, year), Enrolled(S, student, dept) \]
\[ V(S, course) \rightarrow Registered(S, student, course, year), Course(course, number) \]
\[ \geq 100 \]

User Query (using mediator relations):
\[ q(S, D) \rightarrow Enrolled(S, D), Registered(S, C, Y), Course(C, N) \]
\[ \rightarrow V4(student, course, number) \]
\[ \rightarrow V3(student, course) \]
\[ \rightarrow V2(student, dept, course) \]
\[ \rightarrow V1(student, number, year) \]
\[ \Rightarrow q \]

\[ \Rightarrow q \]

\[ \Rightarrow q \]

\[ \Rightarrow q \]

\[ \Rightarrow q \]

1. Filling the Buckets

\[ V1(student, number, year) \rightarrow Enrolled(student, course, year), Course(course, number) \]
\[ \leq 500, year \geq 1992 \]
\[ V2(student, dept, course) \rightarrow Registered(student, course, year), Enrolled(student, dept) \]
\[ V3(student, course) \rightarrow Registered(student, course, year), year \leq 1990 \]
\[ V4(student, course, number) \rightarrow Registered(student, course, year), Course(course, number) \]
\[ \leq 100 \]

\[ q(S, D) \rightarrow Enrolled(S, D), Registered(S, C, Y), Course(C, N), N \geq 300, Y \geq 1995 \]

\[ V2(S, D, C') \rightarrow V1(S, D', C), V1(S, N, Y) \]

\[ V4(S, C', N') \rightarrow V2(S, D', C), V1(S', N, Y') \]

2. Checking Containment

\[ q(S, D) \rightarrow Enrolled(S, D), Registered(S, C, Y), Course(C, N) \]
\[ \geq 300, Y \geq 1995 \]
\[ q(S, D) \rightarrow Enrolled(S, D), V1(S, D', C), V1(S, N, Y) \]
\[ q(S, D) \rightarrow Enrolled(S, D), Registered(S, C, Y), Course(C, N) \]
\[ \geq 300, Y \geq 1995 \]
\[ \Rightarrow q' \subseteq q \] (and q' is a maximally-contained rewriting of q)

\[ \Rightarrow q' \subseteq q \] (and q' is a maximally-contained rewriting of q)

\[ \Rightarrow q' \subseteq q \] (and q' is a maximally-contained rewriting of q)

\[ \Rightarrow q' \subseteq q \] (and q' is a maximally-contained rewriting of q)

Inverse-Rules Algorithm

Idea: Construct an equivalent logic program which evaluation yields the answer to the query

– The antecedent of the query and views is in term of mediator predicates
– Would like to have source predicates in antecedent so that program can be evaluated

\[ \Rightarrow \] Invert the rules

(simply by using standard logical manipulations)

The Inverse-Rules Algorithm: Example

\[ V(D, course) \rightarrow Enrolled(D, student, dept) \]

\[ \rightarrow Enrolled(D, student, course) \]

\[ \forall D, C \ [ v5(D,C) \rightarrow \exists S \ [ e(S, D) \land r(S, C)] ] \]

\[ \forall v5(D,C) \land [ e(f(D,C), D) \land r(f(D,C), C)] \]

\[ \Rightarrow [ v5(D,C) \land e(f(D,C), D) \land r(f(D,C), C) ] \]

\[ \Rightarrow v5(D,C) \land e(f(D,C), D) \land r(f(D,C), C) \]

\[ e(f(D,C), D) \leftarrow v5(D,C) \]

\[ r(f(D,C), C) \leftarrow v5(D,C) \]

The Inverse-Rules Algorithm: Example

\[ q(D) \leftarrow Enrolled(S, D) \land Registered(S, "DB") \]

\[ v5(D,C) \rightarrow Enrolled(D, student, dept) \]

\[ \rightarrow Enrolled(D, student, course) \]

\[ \rightarrow Enrolled(D, student, course) \]

\[ Ex(E) \rightarrow Enrolled(E, DB) \]

\[ Ex(E) \rightarrow Enrolled(E, DB) \]

\[ Ex(Registered) \rightarrow Enrolled(E, DB) \]

\[ Ex(Registered) \rightarrow Enrolled(E, DB) \]

\[ Ex(e) \rightarrow Enrolled(E, DB) \]

\[ Ex(e) \rightarrow Enrolled(E, DB) \]
Modeling Source Capabilities

Negative capabilities:
- A web-site may require certain inputs (in an HTML form) to answer a query
- Need to consider only valid query execution plans.

Positive capabilities:
- A source may be database (understands SQL)
- Need to decide the placement of operations according to capabilities.

Problem: how to describe and exploit source capabilities

Recursive Rewritings

q(X) :- AwardPaper(X)

Problem: Unbounded union of conjunctive queries
q1(X) :- AAAIdb(X), AwardDB(X)
q1(X) :- AAAIdb(X1), CitationDB(X1,X), AwardDB(X)
...
q1(X) :- AAAIdb(X1), CitationDB(X1,X2), ..., CitationDB(Xn,X), AwardDB(X)

Solution: Recursive Rewriting
papers(X) :- AAAIdb(X)
papers(X) :- papers(Y), CitationDB(Y,X)
q'(X) :- papers(X), AwardDB(X)

Inverse-Rules Algorithm

Inverse + Domain Rules (1)

Inverted Rules:
AAAIPapers(X) ← AAAIdb(X)
Cites(X,Y) ← dom(X) ^ CitationDB(X,Y)
AwardPaper(X) ← dom(X) ^ AwardDB(X)

Domain Rules:
dom(Y) ← dom(X) ^ CitationDB(X,Y)
dom(X) ← AAAIdb(X)

Query:
q(X) :- AwardPaper(X)

Negative Capabilities: Binding Patterns

Sources:
AAAIdb^f(X) → AAAIPapers(X)
CitationDB^b(X,Y) → Cites(X,Y)
AwardDB^p(X) → AwardPaper(X)

Query: find all the award winning papers:
q(X) :- AwardPaper(X)

Inverse-Rules Algorithm

Binding Patterns

Sources:
AAAIdb^f(X) → AAAIPapers(X)
CitationDB^b(X,Y) → Cites(X,Y)
AwardDB^p(X) → AwardPaper(X)

Query: find all the award winning papers:
q(X) :- AwardPaper(X)

Inverse-Rules Algorithm

Inverse + Domain Rules (2)

Simplyfing the program:
q(X) :- paper(X) ^ AwardDB(X)
paper(Y) ← paper(X) ^ CitationDB(X,Y)
paper(X) ← AAAIdb(X)
<table>
<thead>
<tr>
<th>Source Descriptions</th>
<th>User Query</th>
<th>conjunctive</th>
<th>conjunctive + inequality</th>
<th>positive</th>
<th>datalog</th>
<th>first order</th>
</tr>
</thead>
<tbody>
<tr>
<td>conjunctive</td>
<td>PTIME</td>
<td>PTIME</td>
<td>PTIME</td>
<td>PTIME</td>
<td>PTIME</td>
<td>undec.</td>
</tr>
<tr>
<td>conjunctive + inequality</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
</tr>
<tr>
<td>positive</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>co-NP</td>
<td>undec.</td>
</tr>
<tr>
<td>datalog</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
</tr>
<tr>
<td>first order</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
<td>undec.</td>
</tr>
</tbody>
</table>

Data complexity of the problem to decide whether a tuple is in the maximal answer:
- (Open world Assumption (source descriptions with containment))