Querying the Hidden Web

Andrea Calì

Computing Laboratory
University of Oxford

Alberto Mendenzon Workshop
Punta del Este, Uruguay, 24th October 2007

The Hidden Web

Data behind forms

- data accessible through Web forms
  - phone directories
  - auctions
  - stores
- assume that every form is queried with one click
- heterogeneous sources can be integrated in a Web information system

Modelling access limitations

Example: Yahoo! Real Estate

- No possibility of asking for all properties (filling in no fields)
- At least one field must be filled in
- The result is a table

Modelling

- ⋆ We model each source as a table
- ⋆ Filling in a field in the form corresponds to querying with a selection only

Modelling query answering

Modelling

- We consider a conjunctive queries in a relational setting
- We model each data source requiring a certain selection on attributes as a relation
- Query answering is done by a Turing machine that queries sources as oracles

Modelling query answering

Modelling

- We consider a conjunctive queries in a relational setting
- We model each data source requiring a certain selection on attributes as a relation
- Query answering is done by a Turing machine that queries sources as oracles

Observations

- Limitations restrict the answers we can retrieve
- we are interested in maximal answers (w.r.t. set inclusion)

Outline

Example

Superscripts denote input and output attributes

Schema

$R_1^{oio}(Title, Year, Artist)$
$R_2^{ioo}(Artist, Nationality, YOB)$

Example

Superscripts denote input and output attributes

Schema

$$R_1^{oio}(Title, Year, Artist)$$
$$R_2^{ioo}(Artist, Nationality, YOB)$$

Query

$$Q(A) \leftarrow R_2(A, uruguayan, 1950)$$

Example

Superscripts denote input and output attributes

Schema

$R_1^{oio}(Title, Year, Artist)$
$R_2^{ioo}(Artist, Nationality, YOB)$

Query

$Q(A) \leftarrow R_2(A, uruguayan, 1950)$

Best answering: Q cannot be executed directly!

- Starting from the constant 1950, we can access R_1
- then we can obtain tuples with new *Artist* constants
- with such values we can access R_2 and start over
- Need for considering abstract domains to distinguish e.g. years from artists' names

Providing maximal answers

- Basic technique in [Li & Chang 2000] for connection queries
- Answering is inherently recursive
- Need for a set of initial constants
- Notion of abstract domain associated to an attribute
- Encoding in positive Datalog

Naïve program for previous example

$$
\begin{array}{rcl}
Q(N) & \leftarrow & \hat{r}_2(A, uruguayan, 1950) \\
\hat{r}_1(T, Y, A) & \leftarrow & r_1(T, Y, A), dom_Y(Y) \\
\hat{r}_2(A, N, Y) & \leftarrow & r_2(A, N, Y), dom_A(A) \\
dom_A(A) & \leftarrow & \hat{r}_1(A, N, Y) \\
dom_N(N) & \leftarrow & \hat{r}_1(A, N, Y) \\
dom_Y(Y) & \leftarrow & \hat{r}_1(A, N, Y) \\
dom_T(T) & \leftarrow & \hat{r}_2(T, Y, A) \\
dom_Y(Y) & \leftarrow & \hat{r}_2(T, Y, A) \\
dom_A(A) & \leftarrow & \hat{r}_2(T, Y, A) \\
dom_N(uruguayan) & & \\
dom_Y(1950) & &
\end{array}
$$

Outline

Relevance

Definition: relevance

An relation r is relevant for a query Q if there are two instances D_1, D_2 that differ only on the tuples of R, and such that $\mathrm{ans}(Q, \mathcal{S}, D_1, I) \neq \mathrm{ans}(Q, \mathcal{S}, D_2, I)$.

$\mathrm{ans}(Q, \mathcal{S}, D, I)$: answers to Q over schema $\mathcal{S}$ (with limitations Λ), evaluated over database D using initial constants I (superset of those in Q)

Relevance

Definition: relevance

An relation r is relevant for a query Q if there are two instances D_1, D_2 that differ only on the tuples of R, and such that $\mathrm{ans}(Q, \mathcal{S}, D_1, I) \neq \mathrm{ans}(Q, \mathcal{S}, D_2, I)$.

$\mathrm{ans}(Q, \mathcal{S}, D, I)$: answers to Q over schema $\mathcal{S}$ (with limitations Λ), evaluated over database D using initial constants I (superset of those in Q)

Open problem:

Determining relevance for CQs was stated as open problem in [Li & Chang 2001]

Our approach

- Given a query and the schema, we represent dependencies among relations with a graph:
  - nodes are attributes
  - arcs tell which attributes provide values to feed attributes
- We prune non-relevant relations and accesses by deleting edges
- The deletion is based on a sort of stability of deletions

Complexity

Theorem: relevant sources

Relevant sources are exactly those appearing in the pruned graph

Complexity

Theorem: relevant sources

Relevant sources are exactly those appearing in the pruned graph

Tractability result

- The algorithm performs a visit of the graph, visiting all edges plus some "neighbours" for every node

⋆ polynomial time complexity in the size of the graph

Complexity

Theorem: relevant sources

Relevant sources are exactly those appearing in the pruned graph

Tractability result

- The algorithm performs a visit of the graph, visiting all edges plus some "neighbours" for every node
- ⋆ polynomial time complexity in the size of the graph

Extensions

- The same result holds for union of conjunctive queries with negation
- Determining relevance for Datalog queries is undecidable

Outline

Strong minimality of plans

$\forall$-minimality (strong)

A query plan Π is $\forall$-*minimal* iff, for every database D for $\mathcal{S}$, $Acc(D, \Pi) \subseteq Acc(D, \Pi')$ for every query plan Π' of Q.

$Acc(D, \Pi)$ are the accesses to sources done by a plan Π over a database D.

Strong minimality of plans

$\forall$-minimality (strong)

A query plan Π is $\forall$-*minimal* iff, for every database D for $\mathcal{S}$, $Acc(D, \Pi) \subseteq Acc(D, \Pi')$ for every query plan Π' of Q.

$Acc(D, \Pi)$ are the accesses to sources done by a plan Π over a database D.

Proposition

$\forall$-minimality does not always exist.

Weaker minimality of plans

Introduced in [Calì & Martinenghi 2008]

Preliminary criterion

$\Pi' \subseteq \Pi$ whenever, for every database D, $Acc(D, \Pi') \subseteq Acc(D, \Pi)$ and there is a database D' such that $Acc(D', \Pi') \subset Acc(D', \Pi)$.

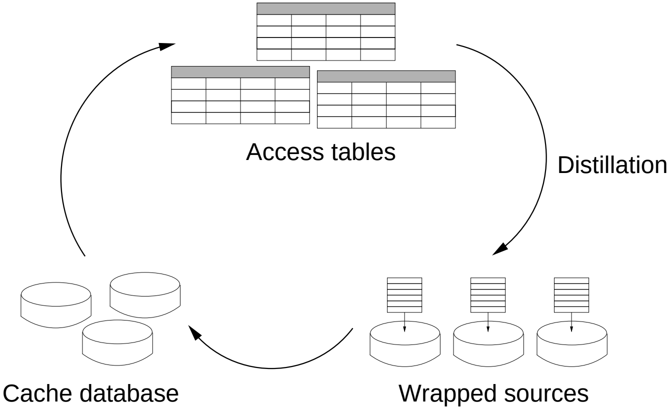
Minimality

Query plan Π is $\subseteq$-*minimal* iff for no query plan Π'' for Q it holds $\Pi'' \subseteq \Pi$.

Results on $\subseteq$-minimality

- A $\subseteq$-minimal plan always exists
- The system Toorjah computes $\subseteq$-minimal based on the optimised dependency graph
- Plans can be expressed in Datalog
  - the evaluation requires some ad-hoc strategies
- Toorjah adopts the fast-failing strategy

Answering queries in Toorjah



Experiments with Toorjah

Schema

$pub_1{}^{io}(Paper, Person)$
$pub_2{}^{oo}(Paper, Person)$
$conf^{ooo}(Paper, ConfName, Year)$
$rev^{ooi}(Person, ConfName, Year)$
$sub^{oi}(Paper, Person)$
$rev\_icde^{iio}(Person, Paper, Eval)$

Queries

- 3 sample queries
- 10,000 synthetic queries

Data

- for sample queries: 1 synthetic database, 10,000 tuples
- for synthetic queries: 100 instances, 10 to 10,000 tuples

Experiments with Toorjah (contd.)

Sample queries

1. $q_1(R) \leftarrow pub_1(P, R), conf(P, C, Y), rev(R, C, Y)$
   authors of publications in conferences where they were also reviewers.
2. $q_2(R) \leftarrow rev\_icde(R, P, rej), conf(P, C, Y), rev(R, C, Y)$
   papers rejected at ICDE by a reviewer and then accepted in a conference listing the same reviewer.
3. $q_3(R) \leftarrow rev\_icde(R, S, acc), sub(S, A), pub_1(P, R), pub_1(P, A), rev(R, icde, 2008), conf(P, icde, Y)$
   reviewers of ICDE 2008 who have accepted at ICDE a submission authored by an ICDE coauthor.

Experiments with Toorjah (contd.)

q_1

relation	accesses		returned rows	
	naive	opt.	naive	opt.
pub_1	4		996	
pub_2	399	364	991	884
conf	4	1	1000	1000
rev	20	20	999	999
sub	400		996	
rev_icde	159,600		997	

Experiments with Toorjah (contd.)

q_2

accesses		returned rows	
naive	opt.	naive	opt.
4		996	
399		991	
4	1	1000	1000
20	20	999	999
400		996	
159,600	133,588	997	818

Experiments with Toorjah (contd.)

q_3

accesses		returned rows	
naive	opt.	naive	opt.
4		996	
399	364	991	884
4	1	1000	1000
20	1	999	56
400	357	996	893
159,600	17,184	997	102

Experiments with Toorjah (contd.)

On synthetic queries:

	arcs	deleted arcs	strong arcs	saved accesses
min	10	4	0	9.10%
max	66	65	7	99.99%
avg	20.54	16.23	1.89	81.02%

Outline

The containment problem

Notation

- Conjunctive queries Q_1, Q_2
- Relational schema $\mathcal{S}$ with limitations Λ
- Initial constants $I \supseteq \text{const}(Q_1) \cup \text{const}(Q_2)$
- $\text{ans}(Q_1, \mathcal{S}, B, I)$: answers to Q evaluated on a schema $\mathcal{S}$ under limitations Λ using initial constants I

The containment problem

Notation

- Conjunctive queries Q_1, Q_2
- Relational schema $\mathcal{S}$ with limitations Λ
- Initial constants $I \supseteq \mathrm{const}(Q_1) \cup \mathrm{const}(Q_2)$
- $\mathrm{ans}(Q_1, \mathcal{S}, B, I)$: answers to Q evaluated on a schema $\mathcal{S}$ under limitations Λ using initial constants I

Containment

Containment $Q_1 \subseteq_{\Lambda,I} Q_2$ under limitations holds if for every database B for $\mathcal{S}$ we have

$$\mathrm{ans}(Q_1, \mathcal{S}, B, I) \subseteq \mathrm{ans}(Q_2, \mathcal{S}, B, I)$$

The containment problem (contd.)

- Checking containment amounts to check containment between two Datalog programs
- this because answering is inherently recursive
- however, programs have a special form
- Decidability?

The backward-chase

- Constructed starting from a query Q and a set of initial constants
- It is a set of databases, denoted bchase$(Q, \mathcal{S}, I)$
- every database represents one of the possible ways of "extracting" a tuple in the answer to the query
- it is possible that there is an infinite number of databases in a chase

Main property of the backward-chase

Theorem

$Q_1 \subseteq_{\Lambda,I} Q_2$ if and only if for every database $C \in \text{bchase}(Q_1, \mathcal{S}, I)$ there exists a homomorphism that sends:

1. body(Q_2) to facts of C, and
2. head(Q_2) to head(C) (head assoc. to all DBs in the chase)

Warning

No indication of a strategy for deciding containment!

Decidability

Theorem

IF If there exists a finite database $C \in \text{bchase}(Q_1, \mathcal{S}, I)$ such that $Q_1(C) \not\subseteq Q_2(C)$,

THEN there exists another finite database $C' \in \text{bchase}(Q_1, \mathcal{S}, I)$ such that

1. $Q_1(C') \not\subseteq Q_2(C')$, and
2. C' has maximum level $\delta = 2 \cdot |\mathcal{S}| + |Q_2| - 3$

Decidability

Theorem

IF If there exists a finite database $C \in \text{bchase}(Q_1, \mathcal{S}, I)$ such that $Q_1(C) \not\subseteq Q_2(C)$,

THEN there exists another finite database $C' \in \text{bchase}(Q_1, \mathcal{S}, I)$ such that

1. $Q_1(C') \not\subseteq Q_2(C')$, and
2. C' has maximum level $\delta = 2 \cdot |\mathcal{S}| + |Q_2| - 3$

Consequence

We can check all databases in the chase up to a certain number of levels

Complexity

Theorem

The complexity of checking containment of conjunctive queries under access limitations is in co-NEXPTIME.

Conclusions

- Answering queries over schemata with access limitations
  - determining relevance of sources
  - new minimisation criterion
  - optimised query plans
  - experiments
- Conjunctive query containment under access limitations
  - Notion of backward-chase
  - Decidability and complexity (upper bound)

Future work

- Including constraints in the schema
- Take into account different network delays
- Lower complexity bound for containment
- Optimisation of the containment check

Acknowledgments

This is a joint work with:

- Davide Martinenghi

Thanks to:

- ESPRC Data Exchange project
- Oxford-Man Institute of Quantitative Finance
- Keble College, Oxford
- Bertram Ludäscher
- Michael Benedikt