Testing Conjunctive Query Containment under Relational Dependencies

Andrea Calì

Faculty of Computer Science
Free University of Bolzano
Italy

Seminar
Faculty of Mathematics, Computer Science and Mechanics
University of Warsaw
Warsaw, 26th April 2007
What is query containment?

**Definition**

Given two queries $Q_1$ and $Q_2$ we say $Q_1$ is contained in $Q_2$, denoted $Q_1 \subseteq Q_2$, if for every database $D$ we have

$$Q_1(D) \subseteq Q_2(D)$$

$Q(D)$: result of evaluation of $Q$ over $D$
Query containment

- Fundamental issue in query optimisation
- Important problem in:
  - consistent query answering
  - data integration and exchange
  - semantic web
We consider **conjunctive queries** over schemata with constraints (a.k.a. dependencies).

The presence of constraints makes query containment checking difficult.

Need for reasoning on **constraints** imposed by the database schema.

$Q_1$ contained in $Q_2$ **under** $\Sigma$, denoted $Q_1 \subseteq_\Sigma Q_2$, if for every database $D$ that satisfies $\Sigma$ we have $Q_1(D) \subseteq Q_2(D)$. 
Query homomorphism

Definition

A query homomorphism $\mu$ is a function from the symbols (appearing as arguments of predicates) of a query $Q$ to those of another query $Q'$ such that:

- every constant is mapped to itself, i.e., for every constant $c$: $\mu(c) = c$
- for every conjunct $R(v_1, \ldots, v_n)$ in $Q$, the conjunct $R(\mu(v_1), \ldots, \mu(v_n))$ is in $Q'$.
Conjunctive query containment: algorithm

1. freeze body$(Q_1)$ and head$(Q_1)$ by turning each variable into a distinct (fresh) constant
2. evaluate $Q_2$ over the frozen body of $Q_1$
3. $Q_1 \subseteq Q_2$ iff the evaluation returns the frozen head of $Q_1$

Testing containment amounts to checking the existence of a query homomorphism from $Q_2$ to $Q_1$ [Chandra & Merlin 1977].
Example

From [Ullman 1997]

\[ Q_1 : \ p(X, Z) \leftarrow a(X, Y), a(Y, Z) \]
\[ Q_2 : \ p(X, Z) \leftarrow a(X, U), a(V, Z) \]
Example

From [Ullman 1997]

\[ Q_1 : \ p(X, Z) \leftarrow a(X, Y), a(Y, Z) \]
\[ Q_2 : \ p(X, Z) \leftarrow a(X, U), a(V, Z) \]

Frozen body \((Q_1)\):

\[ a(0, 1) \leftarrow \]
\[ a(1, 2) \leftarrow \]
Example

From [Ullman 1997]

\[ Q_1 : \quad p(X, Z) \leftarrow a(X, Y), a(Y, Z) \]
\[ Q_2 : \quad p(X, Z) \leftarrow a(X, U), a(V, Z) \]

Frozen body (\( Q_1 \)):

\[ a(0,1) \leftarrow \]
\[ a(1,2) \leftarrow \]

Frozen head (\( Q_1 \)): \( p(0,2) \leftarrow \)
Example (contd.)

Applying $Q_2$ to the frozen $\text{body}(Q_1)$, we find a substitution:

$$X \rightarrow 0, \ U \rightarrow 1, \ V \rightarrow 1, \ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of $Q_1$. 
Example (contd.)

Applying $Q_2$ to the frozen $\text{body}(Q_1)$, we find a substitution:

$$X \rightarrow 0, \ U \rightarrow 1, \ V \rightarrow 1, \ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of $Q_1$. Therefore $Q_1 \subseteq Q_2$. 
Example (contd.)

Applying $Q_2$ to the frozen body($Q_1$), we find a substitution:

$$X \rightarrow 0, \ U \rightarrow 1, \ V \rightarrow 1, \ Z \rightarrow 2$$

that yields $p(0, 2)$ which is the frozen head of $Q_1$. Therefore $Q_1 \subseteq Q_2$.

Note

The frozen body of $Q_1$ is a representative of (a piece of) all databases that provide an answer to $Q_1$. 
Testing Conjunctive Query Containment under Relational Dependencies

Outline

1. Introduction
2. Query containment under database constraints
3. QC with the chase
4. Containment of queries over conceptual schemata
5. Conclusions
Query containment under constraints

Definition

Given a set $\Sigma$ of database dependencies, we say $Q_1$ is contained in $Q_2$ under $\Sigma$, denoted $Q_1 \subseteq_{\Sigma} Q_2$, if for every database $D$ such that $D \models \Sigma$ we have

$$Q_1(D) \subseteq Q_2(D)$$
Our setting

Queries

- conjunctive queries (CQs)

Dependencies

1. key dependencies (KD$s$)
   
   \[ \text{key}(R) = \{A_1, \ldots, A_k\} \]

2. inclusion dependencies (IDs) (generalisation of foreign key dependencies)
   
   \[ R_1[A_1, \ldots, A_m] \subseteq R_2[B_1, \ldots, B_m] \]
Testing Conjunctive Query Containment under Relational Dependencies

Query containment under database constraints

QC under constraints: example

Schema

employee(\textit{Emp\_id}, \textit{Salary}, \textit{Dept})
department(\textit{Dept}, \textit{Location})

with constraint employee[3] \subseteq department[1].
QC under constraints: example

Schema

employee(\(Emp\_id, Salary, Dept\))
department(\(Dept, Location\))

with constraint employee[3] \(\subseteq\) department[1].

Queries

\[Q_1: \; p(X) \leftarrow employee(X, Y, Z), department(Z, W)\]

\[Q_2: \; p(X) \leftarrow employee(X, Y, Z)\]
QC under constraints: example

Schema

employee($Emp\_id$, $Salary$, $Dept$)
department($Dept$, $Location$)


Queries

$Q_1 : \quad p(X) \leftarrow employee(X, Y, Z), department(Z, W)$
$Q_2 : \quad p(X) \leftarrow employee(X, Y, Z)$

$Q_1 \subseteq Q_2$ and $Q_2 \not\subseteq Q_1$, but notice that $Q_2 \subseteq_{\Sigma} Q_1$ (queries are equivalent under $\Sigma$).
Checking QC under database dependencies

Intuition

- Once we freeze $Q_1$ we are constructing a generic database that provides an answer to $Q_1$
- When we freeze, we must construct a database that satisfies $\Sigma$
- We do that by constructing the chase of the frozen query
Outline

1. Introduction
2. Query containment under database constraints
3. QC with the chase
4. Containment of queries over conceptual schemata
5. Conclusions
Chase and repairing

The chase “repairs” the frozen body of $Q_1$ in two ways:

1. “collapsing” pairs of facts that violate a KD (*KD chase rule*)
2. adding facts when an ID is violated (*ID chase rule*); fresh constants may be needed

Note

While collapsing two facts:

- we cannot make two distinct constants equal
- we can equate two variables
Testing Conjunctive Query Containment under Relational Dependencies

QC with the chase

**Containment test under dependencies**

**Theorem** [Johnson & Klug 1982]

\[ Q_2 \subseteq_\Sigma Q_1 \text{ iff there is a homomorphism that maps } body(Q_2) \text{ on the chase of the frozen } body(Q_1), \text{ and the head of } Q_2 \text{ to the frozen head of } Q_1 \]

**Note**

The chase is a representative for all databases that satisfy \( \Sigma \) and provide an answer for \( Q_1 \).
Infinite chase: example

Relations $R/2$, $S/2$

Dependencies

$\sigma_1 : R[1] \subseteq S[1]$
$\sigma_2 : S[2] \subseteq R[1]$
$\sigma_3 : S[2] \subseteq S[1]$
$\gamma_1 : \text{key}(R)=\{1\}$
Infinite chase: example

Relations $R/2$, $S/2$

Dependencies

\[
\sigma_1 : \quad R[1] \subseteq S[1] \\
\sigma_2 : \quad S[2] \subseteq R[1] \\
\sigma_3 : \quad S[2] \subseteq S[1] \\
\gamma_1 : \quad \text{key}(R) = \{1\}
\]

Initial database (frozen body)

\[
R(a, \alpha_0) \leftarrow \\
R(a, b) \leftarrow
\]

greek letters denote fresh constants
Infinite chase: example (contd.)

KD chase rule

We collapse the first two facts (due to $\gamma_1$) by forcing $\alpha_0 = b$.
Infinite chase: example (contd.)

**KD chase rule**

We collapse the first two facts (due to $\gamma_1$) by forcing $\alpha_0 = b$.

**ID chase rule**

Added facts due to IDs:

\[
S(a, \alpha_1) \leftarrow \\
R(\alpha_1, \alpha_2) \leftarrow \\
S(\alpha_1, \alpha_3) \leftarrow \\
S(\alpha_1, \alpha_4) \leftarrow \\
R(\alpha_3, \alpha_5) \leftarrow \\
S(\alpha_3, \alpha_6) \leftarrow \\
\ldots
\]

(... ad infinitum!)
Chase graph

- Facts in the frozen body of $Q_1$ have level 0. A fact derived from the ID chase rule from another fact that is at level $k$ has level $k + 1$

- If fact $f_2$ is derived from fact $f_1$ by an ID $\sigma$, there is an arc $(f_1, f_2)$ labelled with $\sigma$
Example: chase graph

\[
\begin{align*}
R(a, b) &
\xrightarrow{\sigma_1} S(a, \alpha_1) \\
R(\alpha_1, \alpha_2) &
\xrightarrow{\sigma_1} S(\alpha_1, \alpha_4) \\
S(\alpha_1, \alpha_3) &
\xrightarrow{\sigma_2} R(\alpha_3, \alpha_5) \\
S(\alpha_1, \alpha_3) &
\xrightarrow{\sigma_3} S(\alpha_3, \alpha_6)
\end{align*}
\]
Undecidability of QC under KDs and IDs

**Known result**

QC under general functional dep. and IDs is undecidable [Chandra & Vardi 1985]
Undecidability of QC under KDs and IDs

**Known result**

QC under general functional dep. and IDs is undecidable [Chandra & Vardi 1985]

**Theorem**

QC under general KDs and IDs is undecidable

**Proof sketch:** Reduction from implication of KDs and IDs. Consider \( R/n, S/m \), a set of dep. \( \Sigma \) and a constraint \( \sigma : R[1, \ldots, k] \subseteq S[1, \ldots, k] \).

\[
Q_1 : Q() \leftarrow R(X_1, \ldots, X_k, \ldots, X_n) \\
Q_2 : Q() \leftarrow R(X_1, \ldots, X_k, \ldots, X_n), S(X_1, \ldots, X_k, Y_1, \ldots, Y_{m-k})
\]

it is easy to see that \( Q_1 \subseteq_{\Sigma} Q_2 \) iff \( \Sigma \models \sigma \).
Testing Conjunctive Query Containment under Relational Dependencies

QC with the chase

QC under IDs alone

Theorem [Johnson & Klug 1984]

Containment is decidable in PSPACE.

Proof sketch:

- only a finite portion of the chase is necessary
- notion of equivalent conjuncts (agree on non-fresh constants)
- given a fact, an equivalent conjunct is found within
  \[ \delta = |\Sigma| \cdot (W + 1)^W, \]
  where \( W \) is the maximum “width” of IDs in \( \Sigma \)
- Taking into account joins in \( Q_2 \): the necessary depth is \( |Q_2| \cdot \delta \)
- A nontrivial guess shows memberships in PSPACE
- PSPACE-hardness is also proved (like undecidability)
QC under KDs and IDs: decidable cases

- unary IDs
- key-based IDs [Johnson & Klug 1984]; limited class, but is more general than foreign keys
- non-key-conflicting IDs [Calì & al. 2003]; more general class than key-based IDs
Non-key-conflicting IDs

Definition

Non-key-conflicting IDs (NKCIDs) are of the form

\[ R_1[A_1] \subseteq R_2[A_2] \]

where either:

1. no KD is defined over \( R_2 \)
2. \( A_2 \) is not a strict superset of key(\( R_2 \))
Testing Conjunctive Query Containment under Relational Dependencies

QC with the chase

Separation Theorem

Theorem

Given $Q_1$, $Q_2$, $\Sigma$ (NKCIDs), if the chase w.r.t. $\Sigma$ does not fail in the first applications of the FD chase rule:

$$Q_1 \subseteq Q_2 \text{ iff } Q_1 \subseteq_\Sigma Q_2$$

Proof

Based on the fact that if KDs are not violated in the first step of the chase, they are never violated
## Complexity

### Complexity (tight bounds) of containment

<table>
<thead>
<tr>
<th>KDs</th>
<th>IDs</th>
<th>complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>GEN</td>
<td>PSPACE</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>NP</td>
</tr>
<tr>
<td>yes</td>
<td>FK</td>
<td>PSPACE</td>
</tr>
<tr>
<td>yes</td>
<td>NKC</td>
<td>PSPACE</td>
</tr>
<tr>
<td>yes</td>
<td>GEN</td>
<td>undecidable</td>
</tr>
</tbody>
</table>
in data integration and exchange, we have an inconsistent database $B$ (materialised or not) in a global (a.k.a. target) schema

- we can repair $B$ w.r.t. the constraints in several ways
- **certain answers** to a query $Q$: those that are true for all possible repairs
- certain answers are found by evaluating $Q$ over the **chase** of $B$
Outline

1. Introduction
2. Query containment under database constraints
3. QC with the chase
4. Containment of queries over conceptual schemata
5. Conclusions
We consider a conceptual model (Extended ER, EER) derived by enriching Chen’s ER model.

- Conjunctive queries formulated on predicates referring to the constructs of the conceptual schema.

- Need for checking containment under constraints derived from the conceptual schema.

- Constraints are represented with inclusion dependencies (IDs) and key dependencies (KDs).

- Decidability of query containment approaches.
Extended ER schemata

We consider ER schemata enriched with:

- IS-A among entities and relationships
- mandatory (at least once) participation constraints
- functional (at most once) participation constraints
Extended ER schemata

We consider ER schemata enriched with:

- IS-A among entities and relationships
- mandatory (at least once) participation constraints
- functional (at most once) participation constraints

Relational representation of EER schemata:

- entities → unary relations
- relationships → n-ary relations
- attributes → n-ary relations (binary for entities)
- mandatory participation constraints → IDs
- functional participation constraints → KDs
Representing and querying ER schemata: example
Representing and querying ER schemata: example

Constraints: they are always key and inclusion dependencies

\[
\begin{align*}
\text{employee}[1] & \subseteq \text{works\_in}[1] \\
\text{key(works\_in)} & = \{1\} \\
\text{manager}[1] & \subseteq \text{employee}[1]
\end{align*}
\]
Representing and querying ER schemata: example

Employee \(\xrightarrow{(1, N)}\) Works_in \(\xrightarrow{1}\) Dept

Manager

Constraints: they are always key and inclusion dependencies

employee[1] \(\subseteq\) works_in[1]

key(works_in) = \{1\}

manager[1] \(\subseteq\) employee[1]

\ldots

\[ Q(X) \leftarrow \text{manager}(X), \text{works_in}(X, Y), \text{since}(X, Y, 1999) \]
The chase as a tool for containment checking: recall

The **chase** of a query is obtained by:

1. **“freezing”** the query:
   - turn each atom into a fact
   - leave constants unaltered
   - turn variables into “fresh” constants
2. adding facts to satisfy IDs;
3. collapsing (if possible) facts to satisfy KDs
The **chase** of a query is obtained by:

1. “freezing” the query:
   - turn each atom into a fact
   - leave constants unaltered
   - turn variables into “fresh” constants
2. adding facts to satisfy IDs;
3. collapsing (if possible) facts to satisfy KDs

To check $Q_1 \subseteq_{\Sigma} Q_2$:

1. we evaluate $Q_2$ over the chase of $Q_1$
2. if the evaluation returns the frozen head of $Q_1$, then $Q_1 \subseteq_{\Sigma} Q_2$
Chase for EER dependencies: example

Initial facts (frozen query):

\[
\begin{align*}
\text{manager}(m) & \leftarrow \\
\text{manages}(m, d) & \leftarrow
\end{align*}
\]
Chase for EER dependencies: example

Initial facts (frozen query):

`manager(m) ←`  
`manages(m, d) ←`

Facts added in the chase:

`employee(m) ←`  
`works_in(m, α) ←`  
`works_in(m, d) ←`  
`dept(α) ←`  
`dept(d) ←`
Chase for EER dependencies: example

**Initial facts (frozen query):**

- `manager(m) ←`
- `manages(m, d) ←`

**Facts added in the chase:**

- `employee(m) ←`
- `works_in(m, α) ←`
- `works_in(m, d) ←`
- `dept(α) ←`
- `dept(d) ←`

We must deduce $α = d$ (α is a fresh constant) and replace this value in all the segment of the chase constructed so far.
The need for unbounded models: example
The need for unbounded models: example

Initial fact (frozen query):

\[ \text{person}(p) \leftarrow \]
Graph representation of the chase

- person($p$)
- city_of_birth($p$, $\alpha_1$)
- city($\alpha_1$)
- mayor($\alpha_2$, $\alpha_1$)

Levels:
- level 0: person($p$)
- level 1: city_of_birth($p$, $\alpha_1$)
- level 2: city($\alpha_1$)
- level 3: mayor($\alpha_2$, $\alpha_1$)
Chase for EER dependencies: properties

- Violations of KDs are possible only in a very initial segment.
- We prove that a **finite segment** of the chase, until depth $|\Sigma| \cdot W! \cdot |Q_2|$, is sufficient to test containment.
  - $W$ is the maximum number of attributes involved by an ID in $\Sigma$.
- In principle we do not know how far we should go with the chase until we stop, since **collapses may propagate back** from some “deep” (late) part.
Query answering under EER dependencies: decidability and complexity

Containment check $Q_1 \subseteq_{\Sigma} Q_2$:

**Main result**

Propagation of collapses between constants does not go back more than a fixed “distance” in the chase; such distance is $|\Sigma| \cdot W!$
Query answering under EER dependencies: decidability and complexity

Containment check $Q_1 \subseteq_\Sigma Q_2$:

Main result

Propagation of collapses between constants does not go back more than a fixed “distance” in the chase; such distance is $|\Sigma| \cdot W!$

Technique

We proceed until level $|\Sigma| \cdot W! \cdot |Q_2|$ (initial segment) plus another $|\Sigma| \cdot W!$ levels

1. after that no collapse will affect the initial segment;
2. the segment we have is enough for checking query containment
Construction of the relevant segment of the chase

--------- level 0
Construction of the relevant segment of the chase (cont.)

\[ \text{level } 0 \]

\[ \text{level } |\Sigma| \cdot |W| \cdot |Q_2| \]
Construction of the relevant segment of the chase (cont.)

\[ \text{level 0} \]

\[ \text{level } |\Sigma| \cdot W! \cdot |Q_2| \]

\[ \text{level } |\Sigma| \cdot W! \cdot (|Q_2| + 1) \]
Construction of the relevant segment of the chase (cont.)

level 0

level $|\Sigma| \cdot W! \cdot |Q_2|$
Complexity of containment

**Complexity of checking** $Q_1 \subseteq_{\Sigma} Q_2$ (upper bound):

- exponential in $|Q_2|$
- polynomial in $|Q_1|$
- exponential in $|\Sigma|$
- double exponential in $W$
Outline

1. Introduction
2. Query containment under database constraints
3. QC with the chase
4. Containment of queries over conceptual schemata
5. Conclusions
Testing Conjunctive Query Containment under Relational Dependencies

Outline

1. Introduction

2. Query containment under database constraints

3. QC with the chase

4. Containment of queries over conceptual schemata

5. Conclusions
Conclusions

- Techniques for checking conjunctive query containment under constraints
- Use of the chase for containment checking
- Decidable cases and tractability
- Decidability of query containment under EER constraints
  - * proof technique based on chase
- Results extend to querying incomplete data
- Characterisation of complexity of query containment in different cases
Related work

Containment under KDs and IDs

- Chase technique: [Beeri & Vardi JACM 1984]
- Case of IDs alone: treated in [Johnson & Klug 1984]: containment in PSPACE
- Extension to in [Calì et al. 2003] to KDs and non-key-conflicting IDs (NKCDs), again in PSPACE
Containment on EER schemata

- Decidability of the QC problem on EER schemata derived from [Calvanese et al. 1998] by encoding into CPDL;
- We achieve a lower complexity by providing a direct insight into the structure of the chase
- Other relevant approaches:
  - DL-Lite [Calvanese et al. KR 2005] (captures conceptual schemata without IS-A among n-ary roles)
  - [Ortiz et al. 2005] (higher expressiveness and complexity)
Credits

Thanks to (alphabetical order):

- Leo Bertossi
- Diego Calvanese
- Giuseppe De Giacomo
- Michael Kifer
- Domenico Lembo
- Maurizio Lenzerini
- Thomas Lukasiewicz
- Riccardo Rosati
Thank you

Author’s contact

- http://www.andreacali.com
- ac@andreacali.com