Ontological Query Answering under Extended Entity-Relationship Schemata

Andrea Calì

(joint work with Georg Gottlob and Andreas Pieris)

Dept. of Information Systems and Computing, Brunel University
Oxford-Man Institute of Quantitative Finance, University of Oxford

Politecnico di Bari
6th September 2010
The (extended) Entity-Relationship model

Introduced by Peter Chen (1975) for conceptual modelling and data design

More on the formalism later
Why to adopt the ER model?

Good properties of the ER model

- well understood
- general
- expressive
- equipped with formal semantics
  - allows for logical inference
Why to adopt the ER model?

Peter Chen and I, Auckland 2007
The ER model: basic constructs (recall)
The ER model: basic constructs (recall)

Possible instance of relationship Teaches
The ER model: IS-A and disjointness (recall)

IS-A and disjointness

```
Diagram:
- memb_name
  - Member
    - Professor
      - BSc_student
```
The ER model: IS-A and disjointness (recall)

Entity hierarchy

- **Member**
  - **memb_name**
  - \{disjoint, covering\}
  - **PhD_student**
  - **Professor**
The role of a conceptual schema

- Conceptual Schema
- Logical Schema
- Query
- Results
- Data Store
The role of a conceptual schema (cont’d)
The role of a conceptual schema (cont’d)
Outline

- Reasoning on conceptual schemata (brief hint)
Outline

- Reasoning on conceptual schemata (brief hint)
- Answering queries on conceptual schemata with incomplete data
Outline

- Reasoning on conceptual schemata (brief hint)
- Answering queries on conceptual schemata with incomplete data
- Identification of a tractable class of schemata
Outline

- Reasoning on conceptual schemata (brief hint)
- Answering queries on conceptual schemata with incomplete data
- Identification of a tractable class of schemata
- Efficient answering by rewriting in the tractable cases
Outline

- Reasoning on conceptual schemata (brief hint)
- Answering queries on conceptual schemata with incomplete data
- Identification of a tractable class of schemata
- Efficient answering by rewriting in the tractable cases
- Complexity results
Reasoning (adapted from Enrico’s example)
Ontological Query Answering under Extended Entity-Relationship Schemata

Reasoning (cont’d)

Person

{disjoint}

Artist

{disjoint, covering}

Lazy

LatinLover

Scientist

Gentleman

Bore
Extended ER schemata (EER)

We consider ER schemata extended with:

- IS-A among entities and relationships
- mandatory participation constraints
- functional participation constraints

Only cardinality constr. allowed: (0,1), (1,1), (1,N)
Representing and querying EER schemata: example
Representing and querying EER schemata: example

\[
q(B) \leftarrow \text{phd\_student}(A), \text{memb\_name}(A, B), \text{works\_in}(A, C), \\
\text{since}(A, C, 2006), \text{memb\_name}(C, db)
\]
Representing and querying EER schemata: example

Some constraints (in first-order logic form)

- $\text{leads}(X, Y) \rightarrow \text{works\_in}(X, Y)$
- $\text{professor}(X) \rightarrow \text{member}(X)$
- $\text{professor}(X) \rightarrow \exists Y \text{ leads}(X, Y)$
- $\text{key}(\text{works\_in}) = \{1\}$
- $\text{member}(X) \rightarrow \exists Y \text{ works\_in}(X, Y)$
- $\text{key}(\text{leads}) = \{2\}$
The relational constraints: CDs

- From EER schemata we get:
  - Tuple-Generating Dependencies (TGDs) (actually Inclusion Dependencies)
  - Key Dependencies (KDs)
- We call such constraints Conceptual Dependencies (CDs)
Data incompleteness is a long-standing problem in data warehousing, data integration, Semantic Web, ontological reasoning.

Inconsistency/incompleteness w.r.t. constraints in the schema representing the data:
- The data “behave” independently of the global conceptual schema.
Data incompleteness is a long-standing problem in data warehousing, data integration, Semantic Web, ontological reasoning.

Inconsistency/incompleteness w.r.t. constraints in the schema representing the data:
- The data “behave” independently of the global conceptual schema.

We want correct answers w.r.t. the constraints in the case of incomplete data.
The role of a conceptual schema
Deduction under EER dependencies

Initial data:

- professor\( (p) \)
- works_in\( (p, g) \)
- memb_name\( (p, gottlob) \)
- gr_name\( (g, database) \)
Deduction under EER dependencies

Initial data:

- professor\( (p) \)
- works_in\( (p, g) \)
- memb_name\( (p, gottlob) \)
- gr_name\( (g, database) \)

Added facts (the \( z_i \) are nulls):

- professor\( (p) \)
- works_in\( (p, g) \)
- memb_name\( (p, gottlob) \)
- gr_name\( (g, database) \)
Deduction under EER dependencies

**Initial data:**
- professor(p)
- works_in(p, g)
- memb_name(p, gottlob)
- gr_name(g, database)

**Added facts (the z_i are nulls):**
- member(p)
Deduction under EER dependencies

Initial data:
- professor\((p)\)
- works\_in\((p, g)\)
- memb\_name\((p, gottlob)\)
- gr\_name\((g, database)\)

Added facts (the \(z_i\) are nulls):
- member\((p)\)
- leads\((p, z_1)\)
Deduction under EER dependencies

Initial data:
- professor(p)
- works_in(p, g)
- memb_name(p, gottlob)
- gr_name(g, database)

Added facts (the z₁ are nulls):
- member(p)
- leads(p, z₁)
- works_in(p, z₁)
Deduction under EER dependencies

**Initial data:**
- professor($p$)
- works_in($p$, $g$)
- memb_name($p$, gottlob)
- gr_name($g$, database)

**Added facts (the $z_i$ are nulls):**
- member($p$)
- leads($p$, $z_1$)
- works_in($p$, $z_1$)
- $z_1 = g$
Deduction under EER dependencies

Initial data:
professor(p)
works_in(p, g)
memb_name(p, gottlob)
gr_name(g, database)

Added facts (the z_i are nulls):
member(p)
leads(p, g)
works_in(p, g)
Deduction under EER dependencies

Initial data:
- professor\( (p) \)
- works_in\( (p, g) \)
- memb_name\( (p, gottlob) \)
- gr_name\( (g, database) \)

Added facts (the \( z_i \) are nulls):
- member\( (p) \)
- group\( (g) \)
- leads\( (p, g) \)
- works_in\( (p, g) \)
Deduction under EER dependencies

Initial data:
- professor($p$)
- works_in($p, g$)
- memb_name($p, gottlob$)
- gr_name($g, database$)

Added facts (the $z_i$ are nulls):
- member($p$)
- group($g$)
- leads($p, g$)
- works_in($p, g$)
Ontological Query Answering under Extended Entity-Relationship Schemata

Deduction under EER dependencies

Initial data:

- professor(p)
- works_in(p, g)
- memb_name(p, gottlob)
- gr_name(g, database)

Query

\[ q(X) \leftarrow leads(p, X) \]
Deduction under EER dependencies

Initial data:

- professor(p)
- works_in(p, g)
- memb_name(p, gottlob)
- gr_name(g, database)

Query:

\[ q(X) \leftarrow leads(p, X) \]

Answer: \{g\}
Deduction under EER dependencies (cont’d)

- **Hard** violations of KDs: equating two distinct constants
- Can be checked with **negative constraints**
  - Same complexity as query answering
  - Preliminary check on initial data
  - No added complexity
Negative constraints

Example

\[ phd\_student(X), professor(X) \rightarrow \bot \]
\[ phd\_student(X), enrolled(X, Y) \rightarrow \bot \]
Negative constraints

Example

\[ \text{phd\_student}(X), \text{professor}(X) \rightarrow \bot \]
\[ \text{phd\_student}(X), \text{enrolled}(X, Y) \rightarrow \bot \]

- Negative constraints can be checked preliminarily
- Same complexity as query answering
- No added complexity
- Can be used to check hard violations [C. et al. KR 2008]
# Separability

**Definition (intuitive)**

Conceptual dependencies: **TGDs plus KDs**. The KDs are **separable** from the TGDs iff either:

1. there is a hard violation, or
2. we can compute the answers ignoring the KDs
Separability

Definition (intuitive)

Conceptual dependencies: **TGDs plus KDs.**
The KDs are **separable** from the TGDs iff either:

1. there is a hard violation, or
2. we can compute the answers ignoring the KDs
   ✴ taking **TGDs only** into account
   ✴ this is more efficient!
Separability: example

The fact $\text{leads}(p, g)$ would not exist without KDs

$\Rightarrow$ KDs and TGDs are not separable here
Result: characterisation of separability

**Syntactic condition**

*Non-conflicting KDs and TGDs*: can be checked on a graph representation of the EER schema.

**Theorem**

An EER schema is separable if and only if is non-conflicting.
Result: characterisation of separability

Consequence
CQ answering on a non-conflicting EER schema can be done by evaluating a suitable select-project-join-union query (implementable in SQL).

Theorem
CQ answering on a non-conflicting EER schema is in \( AC_0 \) in data complexity.
EER languages in this talk

Our results hold for two variants of the EER model
EER languages in this talk

Our results hold for two variants of the EER model

**Variant One**

- *n*-ary relationships
- **No** permutations in IS-A
- All TGDs of the form:
  \[ r_1(X_1, \ldots, X_n) \rightarrow r_2(X_1, \ldots, X_n) \]

EER languages in this talk

Our results hold for two variants of the EER model:

**Variant One**
- *n*-ary relationships
- No permutations in IS-A
- All TGDs of the form:
  \[ r_1(X_1, \ldots, X_n) \rightarrow r_2(X_1, \ldots, X_n) \]

**Variant Two**
- *binary* relationships
- Permutations in IS-A
- For example:
  \[ r_1(X_1, X_2) \rightarrow r_2(X_2, X_1) \]
Characterising separability: the CD-graph

Types of nodes:

- entity nodes
- relationship nodes
- key nodes (shaded in the figure)
- (attributes do not take part in this)
Non-conflicting CDs

Definition

A set of CDs is non-conflicting if:

- whenever there is a “bad” path $v_1 \sim v_2 \sim \ldots \sim v_m$ ($m \geq 3$) in the CD-graph with:
  - $v_1$ entity node
  - $v_2, \ldots, v_n$ relationship nodes
  - $v_n$ key node
- then there is a “good” path of only rel. nodes from $v_m$ to $v_2$. 
Example: conflicting CDs – EER schema
Example: conflicting CDs – bad path
Example: non-conflicting CDs – EER schema
Example: non-conflicting CDs – good path
Algorithm: query answering by rewriting

- Technique analogous to the one of [C. et al. IJCAI 2003]
- A query \( Q \) is rewritten into another query \( Q_R \) taking the constraints (TGDs) \( \Sigma \) into account
- \( Q_R \) is evaluated on the initial data \( D \)
- The correct answers are obtained: \( Q_R(D) = answers(Q, \Sigma, D) \)
The role of a conceptual schema
Query rewriting: example

Given the query $Q$

$$q(M) \leftarrow \text{member}(M)$$
Query rewriting: example

Given the query $Q$

$q(M) \leftarrow \text{member}(M)$

We take the CDs as rewriting rules using resolution
Query rewriting: example

Given the query $Q$

$q(M) \leftarrow member(M)$

We take the CDs as rewriting rules using resolution

$professor(X) \rightarrow member(X)$
Query rewriting: example

Given the query $Q$

$q(M) \leftarrow member(M)$

We take the CDs as rewriting rules using resolution

$professor(X) \rightarrow member(X)$
Query rewriting: example

Given the query $Q$

$$q(M) \leftarrow member(M)$$

We take the CDs as rewriting rules using resolution

$$professor(X) \rightarrow member(X)$$

New query $Q_1$ (in logical union with $Q$)

$$q(M) \leftarrow professor(M)$$
Theorem

The rewriting returns exactly the correct answers if evaluated on the (incomplete) data.
Results on query rewriting

Theorem

The rewriting returns exactly the correct answers if evaluated on the (incomplete) data.

- Query answering is efficient
- Can be done with the evaluation of an SQL query
- in $AC_0$ in data complexity
More complexity results

Note:
- all variants below are **non-conflicting**
- the first variant below enjoys a different syntactic condition to be non-conflicting

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>yes</td>
<td>PSPACE-complete</td>
<td>$\text{AC}_0$</td>
</tr>
<tr>
<td>any</td>
<td>no</td>
<td>NP-complete</td>
<td>$\text{AC}_0$</td>
</tr>
<tr>
<td>$= 2$</td>
<td>yes</td>
<td>NP-complete</td>
<td>$\text{AC}_0$</td>
</tr>
</tbody>
</table>

- Data complexity is **wrt the data only** (all the rest fixed)
- All results (no EGDs so far) extend to **finite instances** due to **finite controllability** (results by Barany, Gottlob and Otto)
- Applications, e.g., to Data Exchange
Wrap-up and open problems

- EER schemata are general and well-understood
- Tractable query answering algorithms ($AC_0$ in data compl.)
  - we can use SQL!
- Applications in data integration, Semantic Web, ontological reasoning
Wrap-up and open problems

- EER schemata are general and well-understood
- Tractable query answering algorithms ($AC_0$ in data compl.)
  - we can use SQL!
- Applications in data integration, Semantic Web, ontological reasoning

Ontology modelling

- Our results generalise most tractable ontology languages
  - in particular, we properly generalise DL-lite
- If DL-lite is relevant for ontological reasoning, a fortiori our extended ER is
The End

THANK YOU