PhD course on

View-based query processing

Data integration – lecture 2

Riccardo Rosati

Dipartimento di Informatica e Sistemistica

Università di Roma “La Sapienza”

{rosati}@dis.uniroma1.it

Corso di Dottorato in Ingegneria Informatica, Università di Roma “La Sapienza”

Course overview

1. Introduction to view-based query processing [Lenzerini]
2. Conjunctive query evaluation [Gottlob]
3. Data exchange [Gottlob]
4. **Data integration** [De Giacomo, Rosati]
5. Data integration through ontologies [De Giacomo]
6. View-based query processing over semistructured data [Calvanese]
7. Reasoning about views [Lenzerini]

Lecture overview

- the role of global integrity constraints
- inclusion dependencies
- query reformulation under inclusion dependencies
  - chase
  - canonical model
  - query rewriting algorithm
- key dependencies
- decidability and separation

Global integrity constraints

- integrity constraints (ICs) posed over the global schema
- specify intensional knowledge about the domain of interest
- add semantics to the information
- but: data in the sources can conflict with global integrity constraints
- the presence of global integrity constraints rises semantic and computational problems
- open research problems

Integrity constraints for relational schemas

Most important ICs for the relational model:

- key dependencies (KDs)
- functional dependencies (FDs)
- inclusion dependencies (IDs)
- foreign keys (FKs)
- exclusion dependencies (EDs)

Inclusion dependencies (IDs)

- an ID states that the presence of a tuple in a relation implies the presence of a tuple in another relation where t' contains a projection of the values contained in t
- syntax: $r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$
- e.g., the ID $r[1] \subseteq s[2]$

  corresponds to the FOL sentence

$$\forall x, y, z \,.\, r(x, y, z) \rightarrow \exists x', z' \,.\, s(x', x, z')$$

- IDs are a special form of **tuple-generating dependencies**

Semantics for GAV systems under integrity constraints

We refer only to databases over a fixed infinite domain Γ.

Given a source database $\mathcal{C}$ for a system $\mathcal{I}$, a global database $\mathcal{B}$ is **legal** for $(\mathcal{I}, \mathcal{C})$ if:

1. it satisfies the ICs on the global schema
2. it satisfies the mapping, i.e. $\mathcal{B}$ is constituted by a **superset** of the **retrieved global database** $ret(\mathcal{I}, \mathcal{C})$

- $ret(\mathcal{I}, \mathcal{C})$ is obtained by evaluating, for each relation in $\mathcal{G}$, the mapping queries over the source database
- assumption of **sound mapping** (open-world assumption)

Semantics: Certain Answers

- we are interested in **certain answers**
- a tuple t is a **certain answer** for a query Q if t is in the answer to Q for **all** (possibly infinite) legal databases for $(\mathcal{I}, \mathcal{C})$
- the certain answers to Q are denoted by $cert(Q, \mathcal{I}, \mathcal{C})$

Example

Global schema: $\mathsf{player}(Pname, YOB, Pteam)$

$\mathsf{team}(Tname, Tcity, Tleader)$

Constraints: $\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$

Mapping:

$$\mathsf{player} \rightsquigarrow \begin{cases} \mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_1(X, Y, Z) \\ \mathsf{player}(X, Y, Z) \leftarrow \mathsf{s}_3(X, Y, Z) \end{cases}$$

$$\mathsf{team} \rightsquigarrow \mathsf{team}(X, Y, Z) \leftarrow \mathsf{s}_2(X, Y, Z)$$

Example (cont'd)

Source database $\mathcal{C}$

s$_1$:

Totti	1971	Roma

s$_2$:

Juve	Torino	Del Piero

s$_3$:

Vieri	1970	Inter

Retrieved global database $ret(\mathcal{I}, \mathcal{C})$

player:

Totti	1971	Roma
Vieri	1970	Inter

team:

Juve	Torino	Del Piero

Example (cont'd)

player :

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero **is a player of** Juve

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain Γ.

Example (cont'd)

player :

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero **is a player of** Juve

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain Γ.

Warning 1 there may be an **infinite number** of legal databases for $\mathcal{I}$

Example (cont'd)

player :

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero **is a player of** Juve

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain Γ.

Warning 1 there may be an **infinite number** of legal databases for $\mathcal{I}$

Warning 2 in case of cyclic IDs, legal databases for $\mathcal{I}$ may be of **infinite size**

Example (cont'd)

player :

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero **is a player of** Juve

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain Γ.

Consider the query $\;$ q$(X, Z) \leftarrow$ player(X, Y, Z) :

Example (cont'd)

player :

Totti	1971	Roma
Vieri	1970	Inter
Del Piero	α	Juve

team :

Juve	Torino	Del Piero

The ID on the global schema tells us that Del Piero **is a player of** Juve

All legal global databases for $\mathcal{I}$ have **at least** the tuples shown above, where α is some value of the domain Γ.

Consider the query $\mathsf{q}(X, Z) \leftarrow \mathsf{player}(X, Y, Z)$:

$$cert(\mathsf{q}, \mathcal{I}, \mathcal{C}) = \{\langle \mathsf{Totti}, \mathsf{Roma} \rangle, \langle \mathsf{Vieri}, \mathsf{Inter} \rangle, \langle \mathsf{Del\ Piero}, \mathsf{Juve} \rangle\}$$

Query processing under inclusion dependencies

- intuitive strategy: add new facts until IDs are satisfied
- problem: infinite construction in the presence of **cyclic IDs**
- example 1: $r[2] \subseteq r[1]$

  suppose $ret(\mathcal{I}, \mathcal{C}) = \{r(a, b)\}$

  1) add $r(b, c_1)$

  2) add $r(c_1, c_2)$

  3) add $r(c_2, c_3)$

  ....

  (infinite construction)

Query processing under inclusion dependencies

- example 2: $r[1] \subseteq s[1], \quad s[2] \subseteq r[1]$

  suppose $ret(\mathcal{I}, \mathcal{C}) = \{r(a, b)\}$

  1) add $s(a, c_1)$

  2) add $r(c_1, c_2)$

  3) add $s(c_1, c_3)$

  4) add $r(c_3, c_4)$

  5) add $s(c_3, c_5)$

  ....

  (infinite construction)

The chase

- chase of a database: exhaustive application of a set of **rules** that transform the database, in order to make the database consistent with a set of integrity constraints

- the chase for IDs has only one rule, the ID-chase rule

The ID-chase rule

- **if** the schema contains the ID $r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$
  and there is a fact in $\mathcal{DB}$ of the form $r(a_1, \ldots, a_n)$
  and there are no facts in $\mathcal{DB}$ of the form $s(b_1, \ldots, b_m)$
  such that $a_{i_\ell} = b_{j_\ell}$ for each $\ell \in \{1, \ldots, k\}$,
  then add to $\mathcal{DB}$ the fact $s(c_1, \ldots, c_m)$,
  where for each h such that $1 \leq h \leq m$,
  if $h = j_\ell$ for some ℓ then $c_h = a_{i_\ell}$
  otherwise c_h is a new constant symbol
  (not occurring already in $\mathcal{DB}$)
- notice: new existential symbols are introduced (skolem terms)

Properties of the chase

- bad news: the chase is in general infinite
- good news: the chase identifies a canonical model
- canonical model = a database that “represents” of all the models of the system
- we can use the chase to prove soundness and completeness of a query processing method
- but: only for positive queries!

Query processing under inclusion dependencies

why don't we use a finite number of existential constants in the chase?

example: $r[1] \subseteq s[1], \quad s[2] \subseteq r[1]$

suppose $ret(\mathcal{I}, \mathcal{C}) = \{r(a, b)\}$

compute *chase*$(ret(\mathcal{I}, \mathcal{C}))$ with only one new constant c_1:

0) $r(a, b)$; 1) add $s(a, c_1)$; 2) add $r(c_1, c_1)$; 3) add $s(c_1, c_1)$

this database is **not** a canonical model for $(\mathcal{I}, \mathcal{C})$

e.g., for the query $q(X) \text{ :– } r(X, Y), s(Y, Y)$:

$a \in q^{\textit{chase}(ret(\mathcal{I},\mathcal{C}))}$ while $a \notin cert(q, \mathcal{I}, \mathcal{C})$

$\Rightarrow$ unsound method!

(and is unsound for **any** finite number of new constants)

An algorithm for rewriting CQs under IDs

- basic idea: let's chase the query, not the data!
- query chase: dual notion of database chase
- IDs are applied from right to left
- advantage: much easier termination conditions! which imply:
  - decidability properties
  - efficiency

Query rewriting under inclusion dependencies

Given a user query Q over $\mathcal{G}$

- we look for a rewriting R of Q expressed over $\mathcal{S}$
- a rewriting R is **perfect** if $R^{\mathcal{C}} = cert(Q, \mathcal{I}, \mathcal{C})$ for every source database $\mathcal{C}$.

With a perfect rewriting, we can do **query answering by rewriting**

Note that we avoid the construction of the retrieved global database $ret(\mathcal{I}, \mathcal{C})$

Query rewriting for IDs

Intuition: Use the IDs as basic rewriting rules

$$\mathsf{q}(X, Z) \leftarrow \mathsf{player}(X, Y, Z)$$

$$\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$$

as a logic rule: $\mathsf{player}(W_3, W_4, W_1) \leftarrow \mathsf{team}(W_1, W_2, W_3)$

Query rewriting for IDs

Intuition: Use the IDs as basic rewriting rules

$$\mathsf{q}(X, Z) \leftarrow \mathsf{player}(X, Y, Z)$$

$$\mathsf{team}[Tleader, Tname] \subseteq \mathsf{player}[Pname, Pteam]$$

as a logic rule: $\mathsf{player}(W_3, W_4, W_1) \leftarrow \mathsf{team}(W_1, W_2, W_3)$

Basic rewriting step:

when the atom unifies with the **head** of the rule

substitute the atom with the **body** of the rule

We add to the rewriting the query

$$\mathsf{q}(X, Z) \leftarrow \mathsf{team}(Z, Y, X)$$

Query Rewriting for IDs: algorithm ID-rewrite

Iterative execution of:

1. **reduction:** atoms that unify with other atoms are eliminated and the unification is applied
2. **basic rewriting step**

The algorithm ID-rewrite

Input: relational schema Ψ, set of IDs Σ_I, UCQ Q

Output: perfect rewriting of Q

$Q' := Q$;

repeat

$Q_{aux} := Q'$;

for each $q \in Q_{aux}$ **do**

(a) **for each** $g_1, g_2 \in body(q)$ **do**

if g_1 and g_2 unify **then** $Q' := Q' \cup \{\tau(\mathit{reduce}(q, g_1, g_2))\}$;

(b) **for each** $g \in body(q)$ **do**

for each $I \in \Sigma_I$ **do**

if I is applicable to g **then** $Q' := Q' \cup \{\ q[g/\mathit{gr}(g, I)]\ \}$

until $Q_{aux} = Q'$;

return Q'

Properties of ID-rewrite

- ID-rewrite terminates
- ID-rewrite produces a perfect rewriting of the input query
- more precisely:
  - $unf_{\mathcal{M}}(q)$ = unfolding of the query q w.r.t. the GAV mapping $\mathcal{M}$
- **Theorem:** $unf_{\mathcal{M}}($ID-rewrite$(q))$ is a perfect rewriting of the query q
- **Theorem:** query answering in GAV systems under IDs is in PTIME in data complexity (actually in LOGSPACE)

Key dependencies (KDs)

- a KD states that a set of attributes functionally determines all the relation attributes
- syntax: $key(r) = \{i_1, \ldots, i_k\}$
- e.g., the KD $key(r) = \{1\}$ corresponds to the FOL sentence

$$\forall x, y, y', z, z'. r(x, y, z) \wedge r(x, y', z') \rightarrow y = y' \wedge z = z'$$

- KDs are a special form of **equality-generating dependencies**
- we assume that **only one key** is specified on every relation

Query answering under IDs and KDs

- possibility of inconsistencies (recall the **sound** mapping)
- when $ret(\mathcal{I}, \mathcal{C})$ violates the KDs, no legal database exists and query answering becomes trivial!

Theorem: Query answering under IDs and KDs is undecidable.

Proof: by reduction from implication of IDs and KDs.

Separation for IDs and KDs

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

$$r_1[\mathbf{A}_1] \ \subseteq \ r_2[\mathbf{A}_2]$$

where $\mathbf{A}_2$ is **not** a strict superset of $key(r_2)$

Theorem (IDs-KDs separation): Under KDs and NKCIDs:

if $ret(\mathcal{I}, \mathcal{C})$ satisfies the KDs

then the KDs can be ignored wrt certain answers of a user query Q

Separation for IDs and KDs

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

$$r_1[\mathbf{A}_1] \ \subseteq \ r_2[\mathbf{A}_2]$$

where $\mathbf{A}_2$ is **not** a strict superset of $key(r_2)$

> **Theorem (IDs-KDs separation):** Under KDs and NKCIDs:
>
> if $ret(\mathcal{I}, \mathcal{C})$ satisfies the KDs
>
> then the KDs can be ignored wrt certain answers of a user query Q

the problem is undecidable as soon as we extend the language of the IDs

Separation for IDs and KDs

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

$$r_1[\mathbf{A}_1] \;\subseteq\; r_2[\mathbf{A}_2]$$

where $\mathbf{A}_2$ is **not** a strict superset of $key(r_2)$

Theorem (IDs-KDs separation): Under KDs and NKCIDs:

if $ret(\mathcal{I}, \mathcal{C})$ satisfies the KDs

then the KDs can be ignored wrt certain answers of a user query Q

the problem is undecidable as soon as we extend the language of the IDs

foreign keys (FKs) are a special case of NKCIDs

Query processing under separable KDs and IDs

- global algorithm:
  1. verify consistency of $ret(\mathcal{I}, \mathcal{C})$ with respect to KDs
  2. compute ID-rewrite of the input query
  3. unfold the query computed at previous step
  4. evaluate the query over the sources
- the KD consistency check can be done by suitable CQs with inequality
- (exercise: choose a key dependency and write a query that checks consistency with respect to such a key)
- computation of $ret(\mathcal{I}, \mathcal{C})$ can be avoided (by unfolding the queries for the KD consistency check)

Example: checking KD consistency

relation: player$[Pname, Pteam]$

key dependency: $key(\text{player}) = \{Pname\}$

KD (in)consistency query:

$q()\ \text{:-}\ \text{player}(X, Y), \text{player}(X, Z), Y \neq Z$

q true iff the instance of player violates the key dependency

Example: unfolding a KD consistency query

mapping:
$$\begin{aligned} \mathsf{player}(X,Y) &\leftarrow \mathsf{s}_1(X,Y) \\ \mathsf{player}(X,Y) &\leftarrow \mathsf{s}_2(X,Y) \end{aligned}$$

q' = unfolding of q:

$$\begin{aligned} q'() \quad = \quad & \mathsf{s}_1(X,Y), \mathsf{s}_1(X,Z), Y \neq Z \vee \\ & \mathsf{s}_1(X,Y), \mathsf{s}_2(X,Z), Y \neq Z \vee \\ & \mathsf{s}_2(X,Y), \mathsf{s}_1(X,Z), Y \neq Z \vee \\ & \mathsf{s}_2(X,Y), \mathsf{s}_2(X,Z), Y \neq Z \end{aligned}$$

Query answering under separable KDs and IDs

Computational characterization:

- **Theorem:** query answering in GAV systems under KDs and NKCIDs is in PTIME in data complexity (actually in LOGSPACE)

Information integration under integrity constraints

- the above algorithms are applicable in information integration systems with GAV mappings and (separable) KDs and IDs
- what happens in the presence of LAV mappings?
- what happens in the presence of other integrity constraints (exclusion dependencies)?
- see next lecture

The inconsistency issue

- ID are “repaired” by the sound semantics
- KD violations are NOT repaired
- need for a more “tolerant” semantics
- see next lecture

More expressive queries

- under KDs and FKs, can we go beyond CQs?
- union of CQs (UCQs): YES

  ID-rewrite$(q_1 \vee \ldots \vee q_n) =$ ID-rewrite$(q_1) \cup \ldots \cup$ ID-rewrite(q_n)
- recursive queries: NO
- answering recursive queries under KDs and FKs is undecidable

  [Calvanese & Rosati, 2003]
- (same undecidability result holds in the presence of IDs only)