GAV data integration under integrity constraints

Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”

Corso di Seminari di Ingegneria del Software, a.a. 2005/06

Lecture overview, part one

Query answering in GAV data integration systems:

- retrieved global database
- unfolding
- query answering
- complexity of query answering

Lecture overview, part two

Query answering in GAV under integrity constraints:

- 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-as-view (GAV)

(Reminder)

GAV mapping assertions $g \rightsquigarrow \phi_{\mathcal{S}}$ have the logical form:

$$\forall \mathbf{x} \;\; \phi_{\mathcal{S}}(\mathbf{x}) \rightarrow g(\mathbf{x})$$

where $\phi_{\mathcal{S}}$ is a conjunctive query, and g is an element of $\mathcal{G}$.

Semantics for GAV systems

(Reminder)

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 it satisfies the mapping with respect to $\mathcal{C}$

model for $(\mathcal{I}, \mathcal{C})$ = legal database for $(\mathcal{I}, \mathcal{C})$

assumption of **sound mapping** (open-world assumption)

Semantics: Certain Answers

(Reminder)

- 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 $cert(Q, \mathcal{I}, \mathcal{C})$

Retrieved global database

Given a source database $\mathcal{C}$, we call **retrieved global database**, denoted $ret(\mathcal{I}, \mathcal{C})$, the global database obtained by "applying" the queries in the mapping, and "transferring" to the elements of $\mathcal{G}$ the corresponding retrieved tuples.

GAV: example

Consider $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with

Global schema $\mathcal{G}$:

$$\text{student}(code, name, city)$$
$$\text{university}(code, name)$$
$$\text{enrolled}(Scode, Ucode)$$

Source schema $\mathcal{S}$: relations $\mathsf{s}_1(X, Y, W, Z)$, $\mathsf{s}_2(X, Y)$, $\mathsf{s}_3(X, Y)$

Mapping $\mathcal{M}$:

$$\text{student}(X, Y, Z) \rightsquigarrow \{ (X, Y, Z) \mid \mathsf{s}_1(X, Y, Z, W) \}$$
$$\text{university}(X, Y) \rightsquigarrow \{ (X, Y) \mid \mathsf{s}_2(X, Y) \}$$
$$\text{enrolled}(X, W) \rightsquigarrow \{ (X, W) \mid \mathsf{s}_3(X, W) \}$$

GAV: example

university

code	name
AF	*bocconi*
BN	*ucla*

student

code	name	city
15	*bill*	*oslo*
12	*anne*	*florence*

enrolled

Scode	Ucode
12	*AF*
16	*BN*

$s_1^{\mathcal{C}}$

12	anne	florence	21
15	bill	oslo	24

$s_2^{\mathcal{C}}$

AF	bocconi
BN	ucla

$s_3^{\mathcal{C}}$

12	AF
16	BN

Example of source database $\mathcal{C}$ and retrieved global database $ret(\mathcal{I}, \mathcal{C})$

GAV: minimal model

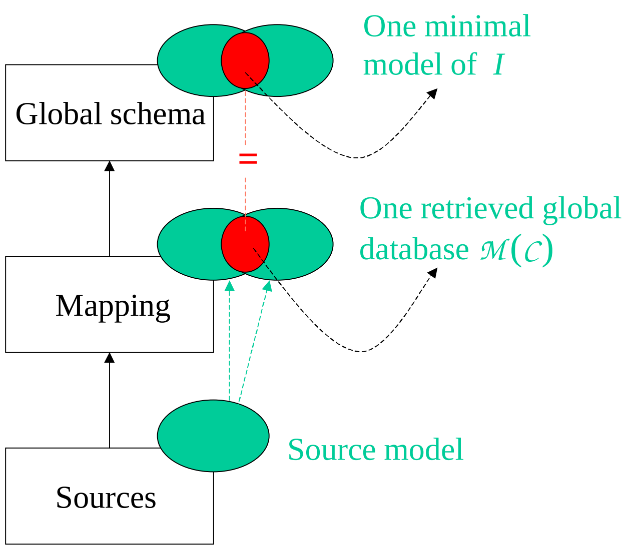
GAV mapping assertions $g \rightsquigarrow \phi_{\mathcal{S}}$ have the logical form:

$$\forall \mathbf{x} \;\; \phi_{\mathcal{S}}(\mathbf{x}) \rightarrow g(\mathbf{x})$$

where $\phi_{\mathcal{S}}$ is a conjunctive query, and g is an element of $\mathcal{G}$.

In general, given a source database $\mathcal{C}$ there are several databases that are legal with respect to $(\mathcal{I}, \mathcal{C})$

However, it is easy to see that $ret(\mathcal{I}, \mathcal{C})$ is the intersection of all such databases, and therefore, is the **only** "minimal" model of $\mathcal{I}$.



GAV: query answering

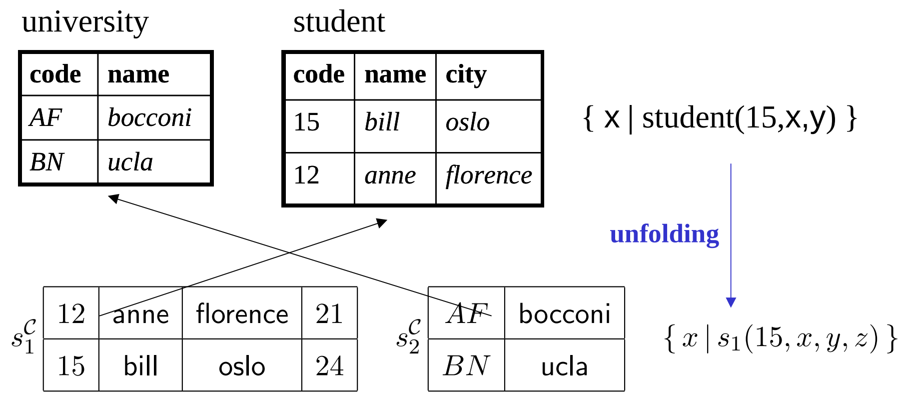
- If q is a conjunctive query, then $\mathbf{t} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\mathbf{t} \in q^{ret(\mathcal{I}, \mathcal{C})}$
- If q is query over $\mathcal{G}$, then the **unfolding** of q wrt $\mathcal{M}$, *unf*$_{\mathcal{M}}(q)$, is the query over $\mathcal{S}$ obtained from q by substituting every symbol g in q with the query $\phi_{\mathcal{S}}$ that $\mathcal{M}$ associates to g
- It is easy to see that evaluating a query q over $ret(\mathcal{I}, \mathcal{C})$ is equivalent to evaluating *unf*$_{\mathcal{M}}(q)$ over $\mathcal{C}$. It follows that, if q is a conjunctive query, then $\mathbf{t} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\mathbf{t} \in$*unf*$_{\mathcal{M}}(q)^{\mathcal{C}}$

  Unfolding is therefore sufficient

GAV: complexity of query answering

- **Data complexity** of query answering is **polynomial** (actually **LOGSPACE**): the query $unf_{\mathcal{M}}(q)$ is first-order (in fact conjunctive)

GAV: example



GAV: more expressive queries?

- More expressive queries in the mapping?
  - Same results hold if we use any computable query in the mapping
- More expressive user queries?
  - Same results hold if we use Datalog queries as user queries
  - Same results hold if we use union of conjunctive queries with inequalities as user queries

GAV: another view

Let B_1 and B_2 be two global databases with values in $\Gamma \cup$ Var.

- A homomorphism $h : B_1 \rightarrow B_2$ is a mapping from ($\Gamma \cup$ Var(B_1)) to ($\Gamma \cup$ Var(B_2)) such that
  1. $h(c) = c$, for every $c \in \Gamma$
  2. for every fact $R_i(t)$ of B_1, we have that $R_i(h(t))$ is a fact in B_2 (where, if $t = (a_1, \ldots, a_n)$, then $h(t) = (h(a_1), \ldots, h(a_n))$)
- B_1 is homomorphically equivalent to B_2 if there is a homomorphism $h : B_1 \rightarrow B_2$ and a homomorphism $h' : B_2 \rightarrow B_1$

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system. If $\mathcal{C}$ is a source database, then a universal solution for $\mathcal{I}$ relative to $\mathcal{C}$ is a model J of $\mathcal{I}$ relative to $\mathcal{C}$ such that for every model J' of $\mathcal{I}$ relative to $\mathcal{C}$, there exists a homomorphism $h : J \rightarrow J'$ (see [Fagin&al. ICDT'03]).

GAV: another view

- Homomorphism preserves satisfaction of conjunctive queries: if there exists a homomorphism $h : J \to J'$, and q is a conjunctive query, then $\mathbf{t} \in q^J$ implies $\mathbf{t} \in q^{J'}$
- Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GAV data integration system without constraints in the global schema. If $\mathcal{C}$ is a source database, then $ret(\mathcal{I}, \mathcal{C})$ is the minimal universal solution for $\mathcal{I}$ relative to $\mathcal{C}$
- We derive again the following results
  - if q is a conjunctive query, then $\mathbf{t} \in cert(q, \mathcal{I}, \mathcal{C})$ if and only if $\mathbf{t} \in q^{ret(\mathcal{I},\mathcal{C})}$
  - complexity of query answering is polynomial

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

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

- 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 with respect to $\mathcal{C}$ (i.e., $\mathcal{B}$ is constituted by a superset of the retrieved global database $ret(\mathcal{I}, \mathcal{C})$)

Example

Global schema: player$(Pname, YOB, Pteam)$
team$(Tname, Tcity, Tleader)$

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

Mapping:

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

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

Example (cont'd)

Source database $\mathcal{C}$

s_1:

Totti	1976	Roma

s_2:

Juve	Torino	Del Piero

s_3:

Vieri	1974	Inter

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

player:

Totti	1976	Roma
Vieri	1974	Inter

team:

Juve	Torino	Del Piero

Example (cont'd)

player :

Totti	1976	Roma
Vieri	1974	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	1976	Roma
Vieri	1974	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	1976	Roma
Vieri	1974	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	1976	Roma
Vieri	1974	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 $\mathsf{q}(X, Z) \leftarrow \mathsf{player}(X, Y, Z)$:

Example (cont'd)

player :

Totti	1976	Roma
Vieri	1974	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 $\mathsf{q}(X, Z) \leftarrow \mathsf{player}(X, Y, Z)$:

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

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)

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 $q(X) \ :- \ r(X,Y), s(Y,Y)$:

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

$\Rightarrow$ unsound method!

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

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!

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

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

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

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

Query rewriting for IDs

Intuition: Use the IDs as basic rewriting rules

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

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

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

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

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

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** $Q' := Q' \cup \{\tau(\textit{reduce}(q, g_1, g_2))\}$;
  (b) **for each** $g \in body(q)$ **do**
    for each $I \in \Sigma_I$ **do**
      if I is applicable to g **then** $Q' := Q' \cup \{ q[g/\textit{gr}(g, I)] \}$
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:** $unf_{\mathcal{M}}(\text{ID-rewrite}(q))$ 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)

The inconsistency issue

- ID are “repaired” by the sound semantics
- KD violations are NOT repaired
- need for a more “tolerant” semantics
- issue studied by research in consistent query answering

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)