Integrity Constraint Translation Problem in an Federative Relational Database Environment

Yangjun Chen and Wolfgang Benn

Department of Computer Science, Technical University of Chemnitz-Zwickau,
09107 Chemnitz, Germany

Abstract In this paper, we address the problem of integrity constraint translations in federative relational databases and present a method to do the relevant task. This method is developed mainly based on the concepts of relation term structures and extended substitutions and can be used to translate integrity constraints automatically even if the local databases have schema conflicts.

1 Introduction

With the advent of applications involving increased cooperation among systems, the development of methods for integrating the pre-existing possibly heterogeneous databases becomes important [BOT86, CW93, HLM94, LA86, LHSC95, KC93, CB96a, CB96b, CB96c]. A common approach to integrating local databases is to design a global schema for the existing ones. In the implementation of such a global schema, some attention should be payed to the integrity maintenance problem. Several investigations have been reported in literature [AQFG95, CHJ96, RPG95]. In [AQFG95, CHJS96], a classification of integrity constraints has been suggested so that the global constraints can be defined manually based on it. In [RPG95], a method is developed tom derive new global constraints in terms of local ones. However, how to validate a global constraints in the component local databases in the case of schema conflicts has not been addressed. In this paper, we discuss this problem an propose a strategy to translate a global constraint into a set of local integrity constraints automatically.

2 A sample example

As an example, consider the following three heterogeneous schemas representing the information about a power network:

DB_1: wire(wire-type, from-to, voltage),
tube(from-to, t-type),
tube-type(t-type, protection).

DB_2: wire′(wire-type, from-to_1, from-to_2, ..., from-to_n),
tube(from-to, t-type),
tube-type(t-type, protection).

DB_3: from-to_1′(wire-type, voltage),
... ...
from-to_m′(wire-type, voltage),
tube(from-to, t-type),
tube-type(t-type, protection).

In DB_1, relation DB_1, “wire” contains a set of tuples with each for a wire type and a connection (from-to) storing the voltage value carried by the wire for the connection, while relations “tube” and “tube-type” are used to maintain the information about tubes utilized for protection.

In DB_2 id of a different structure, with one tuple per “wire-type”, and one attribute per connection, named by from-to_i (i = 1, 2, ..., n) and storing its voltage value. Finally, DB_3 has one relation per connection, named by from-to_j′ (j = 1, 2, ..., m); and each relation has one tuple per wire storing the voltage.

If we want to integrate these three databases together and the global schema is chosen to be the same as DB_1, then a global integrity constraints like:

$$\text{wire}(x_1, y_1, z_1) \wedge \text{tube}(y_1, w_1) \wedge \text{tube-type}(w_1, u_1) \wedge (z_1 \geq 5\text{k}) \Rightarrow (u_1 = \text{“protected”})$$

The work reported here is supported by DFG (Deutsche Forschungsgemeinschaft) under grant No. Be1786/1-1.

has to be translated so that it can be enforced in different local databases. For example, in order to validate this constraint in DB_3, it should be translated into a set of local constraints using the following statement:

for each *from-to* $\in$ {from-to$_1'$, ..., from-to$_m'$} do
{generate:
$from\text{-}to(x, z) \wedge \text{tube}(from\text{-}to, w) \wedge \text{tube-type}(w, u) \wedge (z \geq 5\text{k}) \Rightarrow (u = \text{“protected”})$}

Obviously, the constraint set represented by this statement is semantically equivalent to the global constraint above, but can be executed against DB_3. In the following, we propose a method to do such a translation automatically.

3 Relation term structures and derivation rules

First, we introduce a new concept of *relation structure terms* (RST) to capture high-order information concerning local databases. Then, for such RSTs, we define a set of derivation rules on them to establish the semantic relationships among the global schema and the local ones. Next, we develop another concept: *extended substitution* (ES) so that the translation of integrity constraints can be made in one forward reasoning step.

In our system, an RST is defined as follows:

$[re_{\{R1, ..., Rm\}} | \, a_1: x_1, a_2: x_2, ..., a_l: x_l, y: z_{\{A1, ..., An\}}]$,

where re is a variable ranging over the relation name set $\{R_1, ..., R_m\}$, y is a variable ranging over the attribute name set $\{A_1, ..., A_n\}$, x_1, ..., x_l and z are variables ranging over respective attribute values, and a_1, ..., a_l are attribute names. In the above term, each pair of the form: a_i: x_i (i = 1, ..., l) or y: z is called an attribute descriptor. Intuitively, such an RST can be used to represent either a collection of relations possessing the same structure or part structure of a relation. For example, $[re_{\{\text{from-to1}', ..., \text{from-to}m'\}} |$ wire-type: x, voltage: y] represents any relation in DB_3, while an RST of the form: $[re_{\{\text{wire}'\}} |$ wire-type: x, y: $z_{\{\text{rom-to1}, ..., \text{from-to}n\}}]$ (or simply [wire$'$| time: x, y: $z_{\{\text{rom-to1}, ..., \text{from-to}n\}}$]) represents a part structure of wire$'$ with the form: wire$'$(wire-type,..., from-to$_i$, ...) in DB_2. Since such a structure allows variables for relation names and attribute names, it can be regarded as a higher order predicate quantifying both data and metadata. When the variables (of an RST) appearing in the relation name position and attribute name positions are all instantiated to constants, it is degenerated to a first-order predicate. For example, [wire | wire-type: x_1, from-to: x_2, voltage: x_3] is a first-order predicate quantifying tuples of “wire”.

The purpose of RSTs is to formalize metadata information. Therefore, it can be used to declare schematic discrepancies. In fact, by combining a set of RSTs into a deductive rule, we can specify some semantic correspondences of heterogeneous local databases exactly. For convenience, an RST can be simply written as [re | a_1: x_1, a_2: x_2, ..., a_l: x_l, y: z] if the possible confusion can be avoided by the context.

For the RSTs, we can define derivation rules in a standard way, as implicitly universally quantified statements of the form: $\gamma_1 \,\&\, \gamma_2 \,...\, \&\, \gamma_l \Leftarrow \tau_1 \,\&\, \tau_2 \,...\, \&\, \tau_k,$ where both γ_i's and τ_k's are (partly) instantiated RSTs or normal predicates of the first-order logic. For example, using the following two rules

$r_{DB1\text{-}DB3}$: [y | wire-type: x, voltage: z] $\Leftarrow$ [wire | wire-type: x, from-to: y, voltage: z], $y \in$ {from-to$_1'$, ..., from-to$_m'$},
$r_{DB3\text{-}DB1}$: [wire| wire-type: x, from-to: y, voltage: z] $\Leftarrow$ [y | wire-type: x, voltage: z], $y \in$ {from-to$_1''$, ..., from-to$_l''$},

the semantic correspondence between DB_1 and DB_3 can be specified. (Note that in $r_{DB3\text{-}DB1}$, from-to$_1''$, ..., and from-to$_l''$ are the attribute values of “from-to” in “wire”.)

Similarly, using the following rules, we can establish the semantic relationship between DB_1 and DB_2:

$r_{DB1\text{-}DB2}$: [wire$'$ | wire-type: x, y: z] $\Leftarrow$ [wire | wire-type: x, from-to: y, voltage: z], $y \in$ {from-to$_1$, ..., from-to$_n$},
$r_{DB2\text{-}DB1}$: [wire | wire-type: x, from-to: y, voltage: z] $\Leftarrow$ [wire$'$ | wire-type: x, y: z], $y \in$ {from-to$_1''$, ..., from-to$_l''$}.

Finally, in a similar way, the semantic correspondence between DB_2 and DB_3 can be constructed as follows:

$r_{DB3\text{-}DB2}$: [wire$'$ | wire-type: x, y: z] $\Leftarrow$ [y | wire-type: x, voltage: z], $y \in$ {from-to$_1$, ..., from-to$_n$},
$r_{DB2\text{-}DB3}$: [y | wire-type: x, voltage: z] $\Leftarrow$ [wire$'$ | wire-type: x, y: z], $y \in$ {from-to$_1'$, ..., from-to$_m'$}.

In addition, for a complete specification of semantic correspondences, for each pair of databases, the rules concerning "tube" and "tube-type" have to be defined. For example, for $DB_1 \rightarrow DB_3$, the following rules should be specified:

$r_{DB1\text{-}DB3}'$: [tube | from-to: u, t-type: v] $\Leftarrow$ [tube | from-to: u, t-type: v],
$r_{DB1\text{-}DB3}''$: [tube-type | t-type: s, protection: t] $\Leftarrow$ [tube-type | t-type: s, protection: t].

In the remainder of the paper, a conjunction consisting of RSTs and normal first-order predicates is called a c-expression (standing for "complex expression"). For a derivation rule of the form: $A \Leftarrow B$, A and B are called the antecedent part and the consequent part of the rule, respectively.

4 Integrity constraint translation

In order to translate an integrity constraint by a reasoning step based on the derivation rules, we introduce another concept of extended substitutions to represent the result of matching the constraint with the antecedent part of a rule. Note that a variable involved in a constraint may appear in its left-hand side or in its right-hand side or in both of them. To characterize this feature, we associate each variable with a label $label \in \{l, r, b\}$, where l, r, and b stands for "left", "right" and "both", respectively. Then, an extended substitution can be defined as a finite set of the form: $\{x_1/v_1, ..., x_l/v_l\}$, where x_i ($i = 1, ..., l$) is a variable and v_i ($i = 1, ..., l$) is a set of pairs of the form: ($label$, v), where $label \in \{l, r, b, \_\}$ (here, "_" means "do not care") and v is a value (or a set of values) that x_i takes or "_". In contrast to the traditional substitution concept, the variables $x_1, ..., x_l$ may not be distinct, Each element x_i/v_i is called a binding for x_i and a variable may have several bindings.

Then, by matching our running constraint against the antecedent part of $r_{DB1\text{-}DB3}$, we will have the following ES_1:

$$\{x/\{l, x_1\}, y/\{l, y_1\}, y_1/\{\text{from-to}_1', ..., \text{from-to}_m'\}, z/\{l, \geq 5\text{k}\}.$$

Similarly, by matching the constraint against the antecedent parts of $r_{DB1\text{-}DB3}'$ and $r_{DB1\text{-}DB3}''$, respectively, the following two ESs will be produced:

ES_2: $\{x/\{l, y_1\}, v/\{l, w_1\}\}$,
ES_3: $\{s/\{l, w_1\}, t/\{l, u_1\}, u_1/\{r, = \text{"protected"}\}\}$.

First, in terms of $y_1/\{\text{from-to}_1', ..., \text{from-to}_m'\}$ in ES_1, we generate a statement of the form:

for each $y_1 \in$ from-to$_1'$, ..., from-to$_m'\}$ **do**.

Next, in terms of the remaining part of ES_1, ES_2 and ES_3 as well as the consequent part of $r_{DB1\text{-}DB3}$, $r_{DB1\text{-}DB3}'$ and $r_{DB1\text{-}DB3}''$, we can construct the following new constraint:

$$y_1(x_1, z) \wedge \text{tube}(y_1, w_1) \wedge \text{tube-type}(w_1, u_1) \wedge (z \geq 5\text{k}) \Rightarrow (u = \text{"protected"}).$$

Finally, we will have:

for each $y_1 \in$ from-to$_1'$, ..., from-to$_m'\}$ **do**
$\{y_1(x_1, z) \wedge \text{tube}(y_1, w_1) \wedge \text{tube-type}(w_1, u_1) \wedge (z \geq 5\text{k}) \Rightarrow (u = \text{"protected"})\}$,

which can be executed against DB_3.

5 Conclusion

In this paper, a method for translating integrity constraints automatically has been proposed. This method is mainly based on the concepts of relation term structures and extended substitutions, which make the automatic translation possible even if the schema heterogeneities exist.

References

AQFG95 R.M. Alzahrani, M.A. Qutaishat, N.J.Fiddian and W.A. Gray, "Integrity merging in an object-oriented federated database environment," in *Proc. 13th British National Conference on Databases*, Manchester, UK, July 1995, pp. 226 - 248.

BCG W. Benn, Y. Chen and I. Gringer, “A Rule-based Strategy for Transforming Relational Schemas into OO Schemas,” in *Proc. of the first Workshop on “Föderierte Datenbanken”*, Magdeburg, Germany, April 1996, pp. 19 - 27.

BOT86 Y. Breitbart, P. Olson, and G. Thompsom, “Database integration in a distributed heterogeneous database system,” in *Proc. 2nd IEEE Conf. Data Eng.*, 1986, pp. 301 - 310.

CB96a Y. Chen and W. Benn, “On the Query Translation in Federative Relational Databases,” in *Proc. of 7th Int. DEXA Workshop*, Zurich, Switzerland, Sept. 1996, pp. 491 - 498.

CB96b Y. Chen and W. Benn, “On the Query Optimization in Multidatabase,” accepted by *the first Int. Symposium on Cooperative Database Systems for Advanced Application*, Kyoto, Japan, Dec. 1996.

CB96c Y. Chen and W. Benn, “A Rule-based Strategy for Transforming Relational Schemas into OO Schemas,” accepted by *the first Int. Symposium on Cooperative Database Systems for Advanced Application*, Kyoto, Japan, Dec. 1996.

CW93 S. Ceri and J. Widom, “Managing Semantic Heterogeneity with Production Rules and Persistent Queues”, in *Proc. 19th Int. VLDB Conference*, Dublin, Ireland, 1993, pp. 108 -119.

HLM94 G. Harhalakis, C.P. Lin, L. Mark and P.R. Muro-Medrano, “Implementation of Rule-based Information Systems for Integrated Manufacturing”, *IEEE Trans. on Knowledge and Data Engineering*, vol. 6, No. 6, 892 - 908, Dec. 1994.

KC93 J-L. Koh and Arbee L.P. Chen, “Integration of Heterogeneous Object schemas”, in *Proc. 12th Int. Conf. on the Entity-Relationship Approach*, Arlington, Texas, USA, Dec. 1993, pp. 297 - 314.

LA86 W. Litwin and A. Abdellatif, “Multidatabase interoperability,” *IEEE Comput. mag.*, vol. 19, No. 12, pp. 10 - 18, 1986.

LBT93 A. Lefebvre, P. Bernus and R.W. Topor, “Querying heterogeneous databases: a case study”, in *Proc. 3th Australian Database Conf.*, Brisbane, February 1993, pp. 186-198.

LHSC95 E. Lim, S. Hwang, J. Srivastava, D. Clements and M. Ganesh, “Myriad: design and implementation of a federated database prototype”, *Software-Practice and Experience*, Vol. 25(5), 533 - 562, May 1995.

RPRG94 M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta, “A methodology for integration of heterogeneous databases,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 6, No. 6, 920 - 933, Dec. 1994.