Integrating Heterogeneous OO Schemas

Yangjun Chen
IPSI Institute, GMD GmbH
64293 Darmstadt, Germany
yangjun@darmstadt.gmd.de

Wolfgang Benn
Department of Computer Science
TU Chemnitz, 09107 Chemnitz, Germany
benn@informatik.tu-chemnitz.de

To eliminate semantic conflicts among the heterogeneous databases involved in a cooperation, a set of correspondence assertions for declaring their semantic relationships have to be constructed by DBAs or by users. Normally, four set relationships between object classes: equivalence, inclusion, intersection, and exclusion will be defined to provide knowledge about correspondences that exist among the local schemas [1].

In this paper, we would like to introduce a new assertion, the so-called *derivation* assertion, to accommodate more heterogeneities, which can not be treated by the existing methodologies. As an example, consider two local object-oriented schemas: S_1 and S_2 and assume that S_1 contains two classes: *parent* and *brother*, and S_2 contains a class: *uncle*. A derivation assertion of the form: $S_1(parent, brother) \rightarrow S_2(uncle)$ can specify their corresponding semantic relationship clearly, which can not be established otherwise. We claim that this kind of assertions is necessary for the following reason. Imagine a query concerning *uncle*, submitted to the integrated schema from S_1 and S_2. If the above assertion is not specified, the query evaluation will not take schema S_1 into account and thus the answers to the query can not be correctly computed in the sense of cooperations. Some more complicated examples will be given in a full paper to show that derivation assertions can always be used to handle intricate semantic relationships.

In the following definition, $\boldsymbol{C}$ represents a set of classes constituting a schema. The type of a class C in $\boldsymbol{C}$ is denoted by $type(C)$.

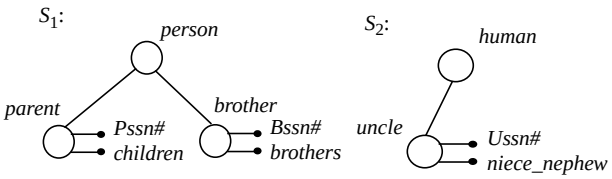
Definition A *path* w.r.t. a class C is a sequence of the form: $C \bullet a_i \bullet a_{ij} \bullet a_{ijk} \ldots \bullet b$, where a_i is an attribute name of C, a_{ij} is an attribute name of $type(a_i)$ (if $type(a_i) \in type(\boldsymbol{C})$, i.e., a_i itself is a class), ..., $a_{ij...hl}$ is an attribute name of $type(a_{ij...h})$ (if $type(a_{ij...h}) \in type(\boldsymbol{C})$), ... and b is of the form $a_{ij...hl...s}$ or "$a_{ij...hl...s}$". If the path is of the form $C \bullet a_i \bullet a_{ij} \bullet a_{ijk} \ldots \bullet a_{ij...hl...s}$, then the attribute values (or the aggregation function's range) of $C \bullet a_i \bullet a_{ij} \bullet a_{ijk} \ldots \bullet a_{ij...hl...s}$ are represented. Otherwise, the path is of the form: $C \bullet a_i \bullet a_{ij} \bullet a_{ijk} \ldots \bullet$"$a_{ij...hl...s}$", used to refer to the attribute name (or the aggregation function name) $a_{ij...hl...s}$ itself.

In terms of the above concept, a correspondence assertion can be generally described as follows.

$S_1(A_1, A_2, ..., A_n)\ \theta\ S_2 \bullet B$
value correspondence of attributes in S_1:
… …
$path_{ik}\ \delta\ path_{jl}$
… …
value correspondence of attributes in S_2:
… …
attribute correspodence:
… …
$S_1 \bullet path_{ms}\ \gamma\ S_2 \bullet path_{Bt}$ with $P_1, ..., P_g$
……
agg_function correspondence:
… …
$S_1 \bullet path_{uv}\ \lambda\ S_2 \bullet path_{Bw}$
… …

where θ is a correspondence assertion, $\delta ::= = | \neq | \in | \supseteq | \varnothing | \cap$,
$\gamma ::= \alpha | \beta | \equiv | \supseteq | \varnothing | \cap$, $\lambda ::= \aleph | \equiv | \supseteq | \varnothing | \cap$,
P_j (j = 1, ..., g) are the predicates.

Below are two schemas for the genealogical applications.



The correspondence between them can be specified as follows:

$S_1(parent, brother) \rightarrow S_2 \bullet uncle$
value correspondence of attributes in S_1:
$parent \bullet Pssn\# \in brother \bullet brothers$
value correspondence of attributes in S_2: no constraints
attribute correspodence:
$S_1 \bullet brother \bullet Bssn\# \equiv S_2 \bullet uncle \bullet Ussn\#$
$S_1 \bullet parent \bullet children \supseteq S_2 \bullet uncle \bullet niece\_nephew$

References

[1] P. Scheuermann and E.I. Chong, "Role-based query processing in multidatabase systems", in: *Proc. of 4th Int. Conf. on Extending Database Technology*, Cambridge, United Kingdom, March 1994, pp. 95 - 108.