AG Heterogene Informationssysteme
Prof. Dr.-Ing. Stefan Deßloch
Fachbereich Informatik
Technische Universität Kaiserslautern

Middleware for Heterogeneous and Distributed Information Systems – Exercise Sheet 11

Wednesday, January 21, 2009 – 10:00 to 11:30 – Room 48-379

Schema Integration

Schema integration (also referred to as schema merging) aims at combining given source schemas into one integrated schema. Schema merging takes as input a set of semantic correspondences between the source schema elements. These correspondences are identified earlier during schema matching, either automatically or manually.

1. In class, several desirable properties of an integrated schema (the result of schema merging) have been discussed. Name each of these properties and explain their importance!
2. Table 1 depicts two schema definitions of lecture catalogs. Try to think of an integrated schema; what problems and ambiguities do you encounter? What forms of heterogeneity cause these problems?
3. In class, a concrete schema merging algorithm[1] has been presented. Here, as a first step, data model heterogeneity between the source schemas is resolved by translating each of them into an (extended) entity-relationship representation. Transfer the sample schemas depicted in Table 1 into such an entity-relationship representation! (Model primitive data types as entities and use the relationship types *composition*, *association*, and *type-of*)
4. As a second step, create a mapping between the source schemas, i.e. perform manual schema matching. We speak of a *representational conflict* if the same real-world concept is modeled differently in different schemas. Do you see a representational conflict between the sample schemas? Think of options to resolve this conflict! Use the mapping to specify the way the conflict is to be resolved!
5. Perform schema merging following the algorithm presented in class. A *fundamental conflict* occurs when the merged model is not a valid entity-relationship model (i.e. it violates the meta-meta-model). Do you encounter any fundamental conflicts? As a hint, consider that an entity must not have more than one type. Resolve the fundamental conflict in the way presented in class!

[1] Rachel Pottinger, Philip A. Bernstein: Merging Models Based on Given Correspondences. VLDB 2003:826-873

6. Finally, convert the merged entity-relationship model into a relational schema. We speak of a *meta-model conflict* when any meta-model-specific constraints are violated during such a conversion. Do you see any meta-model conflicts? How could they be resolved?

a)

```
CREATE SCHEMA lectureCatalog;

CREATE TABLE lectureCatalog.prof (
    pid INTEGER PRIMARY KEY NOT NULL,
    firstName VARCHAR(255),
    lastName VARCHAR(255)
);

CREATE TABLE lectureCatalog.lecture (
    lid INTEGER PRIMARY KEY NOT NULL,
    title VARCHAR(255),
    lecturer INTEGER REFERENCES lectureCatalog.prof(pid),
    desc VARCHAR(255),
    room INTEGER
);
```

b)

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lectureCatalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="lecture">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="professor">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="name" type="xs:string" />
                    <xs:element name="department" type="xs:string" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="title" type="xs:string" />
              <xs:element name="description" type="xs:string" />
              <xs:element name="roomNumber" type="xs:string" />
              <xs:element name="language" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Table 1: Two sample lecture catalog schemas