Data integration
(heterogeneous data sources)

Differently from previous lectures, where we dealt with data sources sharing the same data model, in the presence of heterogeneous data sources, we need to change some steps of the integration approach:

1) identification of the sources data models
2) reverse engineering (conceptual models)
3) identification and resolution of conflicts
4) conceptual models integration
5) Choice of the target data model (for global conceptual schema translation)
6) Source schemata translation to the target data model (by means of adapters)
7) Conceptual model translation
8) Definition of data views (mappings)

Exercise:

3 different data sources:

**UNIVERSITY-DB (DS1)**
DEPARTMENT(dept-code, dept-name, budget)
RESEARCH_STAFF(name, email, dept-code, s-code)
SCHOOL_MEMBER(name, school, year, email)
SESSION(s-code, session-name, length, room-code)
ROOM(room-code, seats-number, notes)

**TAX-POSITION (DS2)**
<!ELEMENT listOfStudents (student*)>
<!ELEMENT student (name, s-code, school-name, email, tax-fee)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT s-code (#PCDATA)>
<!ELEMENT school-name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT tax-fee (#PCDATA)>

**COMPUTER-SCIENCE (DS3)**
CS_PERSON (first-name, last-name)
PROFESSOR:CS_PERSON (belongs_to:DIVISION, rank)
STUDENT:CS_PERSON (year, takes:set<COURSE>, rank, email)
DIVISION (code, description, address:LOCATION)
LOCATION (city, street, number, country)
COURSE (course-name, taught-by:PROFESSOR)

assume the content of s-code as key.
Step 1: Identification of data-sources' data model

DS1: relational
DS2: XML
DS3: Object Oriented

Step 2: Reverse engineering

DS1 conceptual schema

DS2 conceptual schema

DS3 conceptual schema
GS conceptual schema:

PERSON(p-code, first-name, last-name, email, role, year, rank, school-name, tax-fee, dept-code)
COURSE(course-code, course-name, p-code, room-code)
ENROLLED(p-code, course-code)
ROOM(room-code, seats, notes)
DEPARTMENT(dept-code, dept-name, description, budget)
LOCATION(street, number, city, country, dept-code)

Sources schemata translation
DS1: no translation needed
DS2: DS2.STUDENT(s-code, name, school-name, tax-fee)
DS3: DS3.PERSON(first-name, last-name, role, year, email, division-code, rank)
   DS3.COURSE(course-name, professor-first-name, professor-last-name)
   DS3.TAKES(student-first-name, student-last-name, course-name)
   DS3.DIVISION(code, description)
   DS3.LOCATION(street, number, city, country, division-code)

Conflicts analysis:

cardinality conflict between professor and course.
synonyms: department_division, course_session, research_staff-professor,
   school_member-student.
location <> room.
key-conflicts: PERSON.p-code - concat(first-name, last-name) to generate the key for DS3
Suppose that the sources’ schemata are very stable and there exists only the given sources in the system. What kind of integration strategy do you propose?

Mappings GAV

create view GS:PERSON (p-code, first-name, last-name, email, role, year, rank, school-name, tax-fee, dept-code) as

select email, f1(name), f2(name), email, "student", year, NULL, school, NULL, NULL
from DS1.SCHOOL_MEMBER

UNION

select email, f1(name), f2(name), email, "research-staff", year, NULL, school, NULL, dept-code
from DS1.RESEARCH_STAFF, DS1.SCHOOL_MEMBER
WHERE DS1.RESEARCH_STAFF.email=DS1.SCHOOL_MEMBER.email

UNION

select s-code, f1(name), f2(name), email, "student", NULL, NULL, school-name, tax-fee, NULL
from DS2.STUDENT

UNION

select concat(first-name, last-name), first-name, last-name, email, role, year, rank, NULL, NULL, division-code
from DS3.PERSON

)

f1(.) --> returns the first part of the name (that we assume to be the only first name)
f2(.) --> returns the second part of the name
concat(.,.) --> joins the first and the last name to build a valid key.

We need to perform the same steps for each entity in the global schema.

Adapters:

One of the possible solutions for the XML data-source is to have an adapter able to offer a relational view over the XML database (e.g., by mixing the OO language and XQuery).

The following piece of code will generate the java-inspired code for adding to a result set the result of the GAV mappings between GS.PERSON and DS2.PERSON

for $x$ in doc("tax-positions.xml")/listOfStudents/student
return resultSet.add({$x/s-code}, myfunct.f1({$x/name}), myfunct.f2({$x/name}), email, "student", null, null, {$x/school-name}, {$x/tax-fee}, null);

For DS3 the task is easier since we already have an object oriented data model:

for each person {
   if (person instanceof professor) {
      resultSet.add(professor.getFirstName()+professor.getLastName(),
                      professor.getFirstName(), professor.getLastName(), null, "professor", null,
                      professor.getRank(), null, null, professor.getDivision().getDivisionCode());
   } else {
      resultSet.add(student.getFirstName()+student.getLastName(),
                      student.getFirstName(), student.getLastName(), student.getEmail(), "student",
                      student.getYear(), student.getRank(),null, null, null);
   }
}