Information Integration in XML: The MONDIAL Case Study

Wolfgang May
may@informatik.uni-freiburg.de

PRELIMINARY REPORT

Institut für Informatik
Albert-Ludwigs-Universität
Georges-Koehler-Allee, Geb. 51
79110 Freiburg, Germany

March 2001

Keywords: Information Integration, XML, XPathLog

Abstract

This report describes how several information sources on the Web containing political and geographical data are integrated to a geographical database using the LoPiX system [LoP, May01].

Homepage of the MONDIAL Case Study:
`http://www.informatik.uni-freiburg.de/~may/Mondial/`.
Homepage of the LoPiX system:
`http://www.informatik.uni-freiburg.de/~may/lopix/`.
Homepage of the MONDIAL-LoPiX case study:
`http://www.informatik.uni-freiburg.de/~may/lopix/lopix-mondial.html`.

1 Introduction

The present case study continues the work on information integration. In an earlier case-study [May99], the individual sources of the MONDIAL database have been extracted from HTML pages on the Web. The current work now uses their XML representations, focusing in information integration in XML, using the LoPiX system.

2 The Main Data Sources

In an earlier case-study [May99], the individual sources of the MONDIAL database have been extracted from HTML pages on the Web. The current work now uses their XML representations.

2.1 The CIA World Factbook Countries

The CIA World Factbook (`http://www.odci.gov/cia/publications/pubs.html`) provides political, economic, and social and some geographical information about the countries.

```
<!-- XML DTD "cia.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->


<!ELEMENT cia (continent*, country*)>

<!ELEMENT continent EMPTY>
<!ATTLIST continent id ID #REQUIRED
                    name CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>

<!ELEMENT country (ethnicgroups*, religions*, languages*, borders*, coasts*)>

<!ATTLIST country
  id ID #REQUIRED
  name CDATA #REQUIRED
  datacode CDATA #IMPLIED
  continent CDATA #IMPLIED
  total_area CDATA #IMPLIED
  population CDATA #IMPLIED
  population_growth CDATA #IMPLIED
  infant_mortality CDATA #IMPLIED
  gdp_agri CDATA #IMPLIED
  gdp_ind CDATA #IMPLIED
  gdp_serv CDATA #IMPLIED
  gdp_total CDATA #IMPLIED
  inflation CDATA #IMPLIED
  gdp_total CDATA #IMPLIED
  indep_date CDATA #IMPLIED
```

```
  government CDATA #IMPLIED
  capital CDATA #IMPLIED>

<!ELEMENT ethnicgroups (#PCDATA)>
<!ATTLIST ethnicgroups name CDATA #REQUIRED>

<!ELEMENT religions (#PCDATA)>
<!ATTLIST religions name CDATA #REQUIRED>

<!ELEMENT languages (#PCDATA)>
<!ATTLIST languages name CDATA #REQUIRED>

<!ELEMENT borders (#PCDATA)>
<!ATTLIST borders country IDREF #REQUIRED>

<!ELEMENT coasts (#PCDATA)>
```

2.2 The CIA World Factbook: Organizations

A separate part of the CIA World Factbook provides information about political and economical organizations.

```
<!-- XML DTD "orgs.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->

<!ELEMENT orgs (organization*)>

<!ELEMENT organization (member_names*)>
<!ATTLIST organization id ID #REQUIRED
                       abbrev CDATA #REQUIRED
                       name CDATA #REQUIRED
                       established CDATA #IMPLIED
                       hqcity CDATA #IMPLIED
                       hqcountry CDATA #IMPLIED>

<!ELEMENT member_names (#PCDATA)>
<!ATTLIST member_names type CDATA #REQUIRED>
```

2.3 Global Statistics: Cities and Provinces

The *Global Statistics* Data (`http://www.stats.demon.nl`) collected by Johan van der Heijden provides information about administrative divisions (area and population) and main cities (population).

```
<!-- XML DTD "gs.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->

<!ELEMENT gs (continent*, country*, city*, province*)>
```

```
<!ELEMENT continent EMPTY>
<!ATTLIST continent id ID #REQUIRED
                    name CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>

<!ELEMENT country (name+)>
<!ATTLIST country id ID #REQUIRED
                  capital IDREF #IMPLIED
                  population CDATA #IMPLIED
                  continent IDREF #IMPLIED
                  main_cities IDREFS #IMPLIED
                  adm_divs IDREFS #IMPLIED>


<!ELEMENT province EMPTY>
<!ATTLIST province id ID #REQUIRED
                  name CDATA #REQUIRED
                  country IDREF #REQUIRED
                  capital IDREF #IMPLIED
                  population CDATA #IMPLIED
                  area CDATA #IMPLIED>

<!ELEMENT city (population*, name+)>
<!ATTLIST city id ID #REQUIRED
               country IDREF #REQUIRED
               province IDREF #IMPLIED>

<!ELEMENT population (#PCDATA)>
<!ATTLIST population year CDATA #REQUIRED>
```

2.4 Qiblih: Geographical Coordinates

The Qiblih pages (http://www.bcca.org/misc/qiblih/latlong.html) provide the geographical coordinates of many cities around the world.

```
<!-- XML DTD "qiblih.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->

<!ELEMENT qiblih (city*)>

<!ELEMENT city EMPTY>
<!ATTLIST city id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               longitude CDATA #REQUIRED
               latitude CDATA #REQUIRED
               province CDATA #IMPLIED>
```

2.5 The Original TERRA Database

The Karlsruhe TERRA database can be seen as the origin and the "kernel" of the idea of the MONDIAL database. There, geographical and political information about countries, administrative divisions, cities, organizations, waters, mountains, deserts etc. is stored in a relational schema used for a practical training in ORACLE at Karlsruhe University.

```
<!-- XML DTD "terra.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->

<!ELEMENT terra (country*, province*, city*,
   (mountain, desert, island, river, lake, sea)*)>

<!ELEMENT country (encompassed*)>
<!ATTLIST country id ID #REQUIRED
                  name CDATA #REQUIRED
                  code CDATA #REQUIRED
                  area CDATA #REQUIRED
                  population CDATA #REQUIRED
                  capital CDATA #REQUIRED>

<!ELEMENT encompassed (#PCDATA)>
<!ATTLIST encompassed continent CDATA #REQUIRED>

<!ELEMENT province (#PCDATA)>
<!ATTLIST province id ID #IMPLIED
                   name CDATA #IMPLIED
                   abbrev CDATA #IMPLIED
                   country CDATA #IMPLIED
                   pop CDATA #IMPLIED
                   capital CDATA #IMPLIED>

<!ELEMENT city (province*,located_at*)>
<!ATTLIST city id ID #REQUIRED
               name CDATA #REQUIRED
               country CDATA #REQUIRED
               population CDATA #IMPLIED
               longitude CDATA #IMPLIED
               latitude CDATA #IMPLIED>

<!ELEMENT located_at EMPTY>
<!ATTLIST located_at type (river|sea|lake) #REQUIRED
                     water IDREF #IMPLIED>

<!ELEMENT river (to*, located*)>
<!ATTLIST river id ID #REQUIRED
                name CDATA #REQUIRED
                length CDATA #IMPLIED>

<!ELEMENT to EMPTY>
```

```
<!ATTLIST to type (river|sea|lake) #REQUIRED
             water IDREF #REQUIRED>

<!ELEMENT located EMPTY>
<!ATTLIST located country_code CDATA #REQUIRED
                  province_id CDATA #REQUIRED>

<!ELEMENT lake (located*)>
<!ATTLIST lake id ID #REQUIRED
               name CDATA #REQUIRED
               area CDATA #IMPLIED>

<!ELEMENT sea (located*)>
<!ATTLIST sea id ID #REQUIRED
              name CDATA #REQUIRED
              depth CDATA #IMPLIED
              bordering IDREFS #REQUIRED>

<!ELEMENT desert (located*)>
<!ATTLIST desert id ID #REQUIRED
                 name CDATA #REQUIRED
                 area CDATA #IMPLIED>

<!ELEMENT island (located*)>
<!ATTLIST island id ID #REQUIRED
                 name CDATA #REQUIRED
                 area CDATA #IMPLIED
                 longitude CDATA #IMPLIED
                 latitude CDATA #IMPLIED>

<!ELEMENT mountain (located*)>
<!ATTLIST mountain id ID #REQUIRED
                   name CDATA #REQUIRED
                   height CDATA #REQUIRED
                   longitude CDATA #IMPLIED
                   latitude CDATA #IMPLIED>
```

2.6 Countries, Names, and Car Codes

The CIA World Factbook does not provide the commonly used and well-known car codes of the countries. These are needed for integration with the Terra Database. Additionally, the german country names proved to be very helpful for integrating the Terra Database.

```
<!-- XML DTD "codes.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, Oct 2000) -->

<!ELEMENT codes (country*)>
```

```
<!ELEMENT country (name,name,name)>
<!ATTLIST country id CDATA #REQUIRED
                  car_code CDATA #REQUIRED>

<!ELEMENT name (#PCDATA)>
<!ATTLIST name language CDATA #REQUIRED>
```

3 Integration

Target DTD

The data sources are the integrated into a common XML document according to the following DTD:

```
<!-- XML DTD "mondial-3.0.dtd":
  (Wolfgang May, may@informatik.uni-freiburg.de, March 2001)
  a hierarchical DTD for the MONDIAL database -->

<!ELEMENT mondial (country*,continent*,organization*,
                   mountain*,(sea*,river*,lake*,desert*,island*)*)>
<!ELEMENT country (name,encompassed+,
                   ethnicgroups*,religions*,languages*,border*,
                   province*,city*)>
<!ATTLIST country car_code ID #IMPLIED
                  capital IDREF #IMPLIED
                  memberships IDREFS #IMPLIED
                  id ID #REQUIRED
                  name CDATA #REQUIRED
                  datacode CDATA #IMPLIED
                  continent CDATA #IMPLIED
                  area CDATA #IMPLIED
                  total_area CDATA #IMPLIED
                  population CDATA #IMPLIED
                  population_growth CDATA #IMPLIED
                  infant_mortality CDATA #IMPLIED
                  gdp_agri CDATA #IMPLIED
                  gdp_ind CDATA #IMPLIED
                  gdp_serv CDATA #IMPLIED
                  gdp_total CDATA #IMPLIED
                  inflation CDATA #IMPLIED
                  gdp_total CDATA #IMPLIED
                  indep_date CDATA #IMPLIED
                  government CDATA #IMPLIED
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT population (#PCDATA)>
<!-- note that population is also a subelement of city -->
<!ATTLIST population year CDATA #IMPLIED>
<!ELEMENT encompassed EMPTY>
<!ATTLIST encompassed continent IDREF #REQUIRED
                      percentage CDATA #REQUIRED>
<!ELEMENT ethnicgroups (#PCDATA)>
<!ATTLIST ethnicgroups percentage CDATA #REQUIRED>

<!ELEMENT religions (#PCDATA)>
<!ATTLIST religions percentage CDATA #REQUIRED>

<!ELEMENT languages (#PCDATA)>
```

```
<!ATTLIST languages percentage CDATA #REQUIRED>

<!ELEMENT border EMPTY>
<!ATTLIST border country IDREF #REQUIRED
                 length CDATA #REQUIRED
                 justice IDREF 'org-UN'>

<!ELEMENT province (city*)>
<!ATTLIST province id ID #REQUIRED
                   name CDATA #REQUIRED
                   country IDREF #REQUIRED
                   capital IDREF #IMPLIED
                   area CDATA #IMPLIED
                   population CDATA #IMPLIED>

<!ELEMENT city (name, population*,located_at*)>
<!ATTLIST city id ID #REQUIRED
               is_country_cap CDATA #IMPLIED
               is_state_cap CDATA #IMPLIED
               longitude CDATA #IMPLIED
               latitude CDATA #IMPLIED
               country IDREF #REQUIRED
               province IDREF #IMPLIED>

<!ELEMENT located_at EMPTY>
<!ATTLIST located_at type (river|sea|lake) #REQUIRED
                     water IDREF #IMPLIED>

<!ELEMENT organization (name,abbrev,established?,members*)>
<!ATTLIST organization id ID #REQUIRED
                       name CDATA #IMPLIED
                       abbrev CDATA #IMPLIED
                       established CDATA #IMPLIED
                       headq IDREF #IMPLIED>
<!ELEMENT members EMPTY>
<!ATTLIST members type CDATA #REQUIRED
                 country IDREFS #REQUIRED>

<!ELEMENT continent EMPTY>
<!ATTLIST continent id ID #REQUIRED
                    area CDATA #IMPLIED
                    name CDATA #REQUIRED>

<!ELEMENT river (to*, located*)>
<!ATTLIST river id ID #REQUIRED
                name CDATA #REQUIRED
                length CDATA #IMPLIED>

<!ELEMENT to EMPTY>
<!ATTLIST to type (river|sea|lake) #REQUIRED
```

```
             water IDREF #REQUIRED>

<!ELEMENT located EMPTY>
<!ATTLIST located country IDREF #REQUIRED
                  province IDREF #REQUIRED>

<!ELEMENT lake (located*)>
<!ATTLIST lake id ID #REQUIRED
               name CDATA #REQUIRED
               area CDATA #IMPLIED>

<!ELEMENT sea (located*)>
<!ATTLIST sea id ID #REQUIRED
              name CDATA #REQUIRED
              depth CDATA #IMPLIED
              bordering IDREFS #REQUIRED>

<!ELEMENT desert (located*)>
<!ATTLIST desert id ID #REQUIRED
                 name CDATA #REQUIRED
                 area CDATA #IMPLIED>

<!ELEMENT island (located*)>
<!ATTLIST island id ID #REQUIRED
                 name CDATA #REQUIRED
                 area CDATA #IMPLIED
                 longitude CDATA #IMPLIED
                 latitude CDATA #IMPLIED>

<!ELEMENT mountain (located*)>
<!ATTLIST mountain id ID #REQUIRED
                   name CDATA #REQUIRED
                   height CDATA #REQUIRED
                   longitude CDATA #IMPLIED
                   latitude CDATA #IMPLIED>
```

Integration Process

The program mondial-integra integrates the databases into an XML result tree and exports it to the file `mondial-3.0.xml`:

```
?- sys.theOMAccess.debugOn.
?- sys.theOM.eqTraceOn. % trace derived equalities
?- sys.prn.style@("bound").     % set output mode
?- sys.prn.annotatedLiterals@("on").
```

Loading Data Sources

```
gs[@xml->"file:/home/may/Mondial/Mondial-Sources/gs.xml" isa url].
cia[@xml->"file:/home/may/Mondial/Mondial-Sources/cia.xml" isa url].
```

```
orgs[@xml->"file:/home/may/Mondial/Mondial-Sources/orgs.xml" isa url].
terra[@xml->"file:/home/may/Mondial/Mondial-Sources/terra.xml" isa url].
qiblih[@xml->"file:/home/may/Mondial/Mondial-Sources/qiblih.xml" isa url].
codes[@xml->"file:/home/may/Mondial/Mondial-Sources/codes.xml" isa url].
mondial.system = "mondial-3.0.dtd".
mondial[@dtd->"file:/home/may/Mondial/Mondial-Sources/mondial-3.0.dtd" isa url].
?- sys.strat.doIt.
U.parse@(xml,S) :- S[@xml->U].
U.parse@(dtd) :- mondial[@dtd->U].
result isa mondial.
S = X :-  S[@xml->U], U.parse@(xml,S) = Doc, Doc[S:S->X].
```

Now, gs, cia etc are root nodes of separate trees

```
?- sys.strat.doIt.
?- sys.garbageCollection.
```

some properties are projected directly from the sources to the result:

```
gs:population = population.
gs:country = country.
gs:province = province.
gs:capital = capital.
gs:area = area.
gs:population = population.
gs:year = year.
gs:name = name.
gs:text() = text().
cia:name = name.
cia:borders = border.
cia:ethnicgroups = ethnicgroups.
cia:religions = religions.
cia:languages = languages.
cia:area = area.
cia.population = population.
cia:datacode =  datacode.
cia:total_area = total_area.
cia:population_growth = population_growth.
cia:infant_mortality= infant_mortality.
cia:gdp_agri = gdp_agri.
cia:gdp_ind = gdp_ind.
cia:gdp_serv = gdp_serv.
cia:gdp_total = gdp_total.
cia:inflation = inflation.
cia:indep_date = indep_date.
cia:government = government.
codes:car_code = car_code.
qiblih:longitude = longitude.
qiblih:latitude = latitude.
orgs:abbrev = abbrev.
orgs:name = name.
```

```
orgs:established = established.
?- sys.strat.doIt.
```

in the terra source, we cannot project complete properties but have to filter wrt the types the apply to. All gepopolitical entities are taken from terra.

```
terra:located_at = located_at.
terra:mountain = mountain.
  terra:height = height.
  M[@longitude->L1 and @latitude->L2] :-
    terra/mountain->M[@terra:longitude->L1 and @terra:latitude->L2].
terra:desert = desert.
  D[@area->A] :- terra/desert->D[@terra:area->A].
terra:island = island.
  I[@area->A] :- terra/island->I[@terra:area->A].
terra:lake = lake.
  L[@area->A] :- terra/lake->L[@terra:area->A].
terra:river = river.
  terra:to = to.
    terra:type = type.
    terra:water = water.
  terra:length = length.
terra:sea = sea.
  terra:depth = depth.
D[@name->A] :- terra/desert->D[@terra:name->A].
D[@name->A] :- terra/mountain->D[@terra:name->A].
D[@name->A] :- terra/island->D[@terra:name->A].
D[@name->A] :- terra/lake->D[@terra:name->A].
D[@name->A] :- terra/river->D[@terra:name->A].
D[@name->A] :- terra/sea->D[@terra:name->A].
?- sys.strat.doIt.
```

continents are taken from gs:

```
result[continent->C] :- gs/gs:continent->C.
```

some cities are not present in the database, but are needed as headquarters for organizations or as capital:

```
C/city[name->"Abidjan"] :-
      gs/gs:country->C[gs:name/gs:text()->"Cote dIvoire"].
C/city[name->"Jeddah"] :-
      gs/country->C[gs:name/gs:text()->"Saudi Arabia"].
C/city[name->"El Aaiun"], C[@capital->elaaiun] :-
      gs/gs:country->C[gs:name/gs:text()->"Western Sahara"].
?- sys.strat.doIt.
```

all cities from gs are taken as result cities:

```
result[city->C[@name->N]] :- gs/gs:city->C[name/text()->N].
?- sys.strat.doIt.
?-sys.echo@("resolving multiple city names:").
```

Cities names like "Mexico", "Mexico City" and "Ciudad de Mexico" denote the same object – same for Panama:

```
C1 = C2 :-
   gs/gs:city->C1[@gs:country->C and @name->N1],
   strcat(N1," City", N2),
   gs/gs:city->C2[@gs:country->C and @name->N2].
C1 = C2 :-
   gs/gs:city->C1[@gs:country->C and @name->N1],
   strcat("Ciudad de ", N1, N2),
   gs/gs:city->C2[@gs:country->C and @name->N2].
?- sys.strat.doIt.
?-sys.echo@("CIA-relevant countries are those for which a capital is given:").
```

Only those countries are relevant, where a capital is given in cia:

```
C[@cia:rel_country->1] :- cia/cia:country->C[@cia:capital].
?- sys.strat.doIt.
?- sys.strat.doIt.
```

Fusing CIA and GlobalStatistics Countries. Fuse two countries if they have the same name:

```
result[country->C1],
C1 = C2 :-
   cia/cia:country->C1[@cia:rel_country and @cia:name->N],
   gs/gs:country->C2[gs:name/gs:text()->N].
C[@name->N] :- result/country->C/@cia:name->N.
result[country->C[@cia:rel_country->1 and @name->"Serbia and Montenegro"]] :-
   gs/gs:country->C[gs:name/gs:text()->"Serbia and Montenegro"].
?- sys.strat.doIt.
?- sys.strat.doIt.
```

Fusing CIA and GlobalStatistics Cities. In MONDIAL-CIA, the only cities mentioned are the capitals. Some of them are stored under different names than in GlobalStatistics.

```
city_synonym("Bucharest","Bucuresti").
city_synonym("Warsaw","Warszawa").
city_synonym("New Delhi","Delhi").
city_synonym("Tashkent","Toshkent").
city_synonym("Addis Ababa","Addis Abeba").
city_synonym("La Hague","s'-Gravenhage").
?- sys.strat.doIt.
C1 = C2 :-
   gs/gs:city->C1[@gs:country->Ctry and @name->N1],
   city_synonym(N1,N2),
   gs/gs:city->C2[@gs:country->Ctry and @name->N2].
?- sys.strat.doIt.
    not C/@gs:main_cities.
?- sys.strat.doIt.
```

add provinces and cities from gs to the result tree if they belong to relevant countries:

```
result[province->P] :- gs/gs:province->P/@country[@cia:rel_country].
result[city->Cty]   :- gs/gs:city->Cty/@country[@cia:rel_country].
?- sys.strat.doIt.
```

Fusing CIA and codes Countries. The mondial-codes data source contains the english, german, and local names of the countries and their car-codes. Countries are objects.
For most countries, the CIA and english name coincides:

```
C1 = C2 :-
   codes/codes:country->C1/codes:name[@codes:language = "english"]/codes:text()->N,
   result/country->C2[@name->N].
?- sys.strat.doIt.
```

For other countries, the CIA name coincides with the german or local name:

```
C1 = C2 :-
   codes/codes:country->C1/codes:name[@codes:language = "german"]/codes:text()->N,
   result/country->C2[@name->N].
C1 = C2 :-
   codes/codes:country->C1/codes:name[@codes:language = "local"]/codes:text()->N,
   result/country->C2[@name->N].
?- sys.strat.doIt.
```

As a side effect, for all CIA-relevant countries, also the attribute country/@car_code is defined (taken from codes).

Linking Qiblih-Cities to Countries. In mondial-qiblih, the countries are only represented by the string-valued method qcity[country@(qiblih)⇒countryname]. All Qiblih cities for which qcity.country@(qiblih) matches an N which is any of country.name@(cia) or country.name@(gs) are linked to the respective country.

```
?- sys.strat.doIt.
Cty[@country->C] :-
   qiblih/qiblih:city->Cty[@qiblih:country->N], result/country->C[@name->N].
```

There are some CIA capitals which are also contained in Qiblih, but not with the exact country name. They are matched by name, excluding the US and Canadian Cities (which can be detected since for them, province@(qiblih) is defined).

```
?- sys.strat.doIt.
Cty[@country->Country] :-
   qiblih/qiblih:city->Cty[@qiblih:name->N], not Cty/@country,
   not Cty/@qiblih:province,
   result/country->Country[@cia:rel_country]/@gs:capital[@name->N].
```

US and Canadian Qiblih cities are linked to the corresponding province object by the province name:

```
?- sys.strat.doIt.
Cty[@province->P] :-
   qiblih/qiblih:city->Cty[@country->S and @qiblih:province->N],
   gs//gs:province->P[@gs:country->S and @gs:name->N].
```

Merging Qiblih-Cities with GlobalStatistics main cities. If no province is known in Qiblih (all except US and CDN), cities are matched by name (comparing with both GlobalStatistics or CIA name – note that capitals can have both of them) and country:

```
?- sys.strat.doIt.
C1 = C2 :-
   qiblih/qiblih:city->C1[@qiblih:name->N and @country->Ctry
                           and not @qiblih:province],
   result/city->C2[@name->N and @country->Ctry].
?- sys.strat.doIt.
```

In the USA, Qiblih and GlobalStatistics main cities are matched by name/state/country:

```
C1 = C2 :-
   qiblih/qiblih:city->C1[@qiblih:name->N and not @name and
                           @country->Ctry and @province->P],
   result/city->C2[@name->N and
                   @province->P and @country->Ctry].
?- sys.strat.doIt.
```

For Canada, Qiblih provides provinces, but GlobalStatistics does not. Cities are matched by name and country:

```
C1 = C2 :-
   qiblih/qiblih:city->C1[@qiblih:name->N and not @name and
                           @country->CDN[@name="Canada"]],
   result/city->C2[@name->N and @country->CDN].
?- sys.strat.doIt.
```

Organizations and Memberships. Note that the CIA country data and the CIA organizations data is not fully consistent: there are 71 memberships which are not mentioned in the Country data, and there are 6 organizations which are mentioned in the country data which do not exist in the organization data.

Since the organizations data is more detailed (membership type), the membership information is taken from there. The memberships are linked to the countries. Note that there are members which are no legal countries, mostly international commissions or local interest groups. Note that the member lists contains some countries which are both members and associate members/observers (IFRCS). Here, we take only the full membership.

```
result[organization->O] :- orgs/orgs:organization->O.
?- sys.strat.doIt.
O/members[@type->"member" and @country->C] :-
   result/organization->O/orgs:member_names[@orgs:type="member"]/orgs:text()->CN,
   result/country->C[@name->CN].
```

```
?- sys.strat.doIt.
O/members[@type->T and @country->C] :-
   result/organization->O/orgs:member_names[@orgs:type->T]/orgs:text()->CN,
   not O/orgs:member_names[@orgs:type="member"]/orgs:text()->CN,
   result/country->C[@name->CN].
```

The headquarters of organizations are linked to the respective cities:

```
?- sys.strat.doIt.
O[@headq->Cty] :-
   result/organization->O[@orgs:hqcity->N and @orgs:hqcountry->CN],
   result/city->Cty[@name->N and @country/@name->CN].
?- sys.strat.doIt.
```

Terra: Fusing Countries. The Terra country objects are fused with the country objects. In most cases, the objects can be merged using terra_country[code@(terra)⇒car_code] and country[car_code⇒car_code]:

```
L = C :-
  terra/terra:country->L[@terra:code->Code],
  result/country->C[@car_code->Code].
?- sys.strat.doIt.
```

Others are fused by matching the Terra name with the german name given in the codes source, or with the name given in gs:

```
L = C :-
   terra/terra:country->L[@terra:name->N and not @car_code],
   result/country->C/codes:name[@codes:language ="german"]/codes:text()->N.
L = C :-
   terra/terra:country->L[@terra:name->N and not @car_code],
   result/country->C/@name->N.
```

If the name of a capital is the same in CIA/GlobalStatistics and Terra, the countries are also the same.

```
L = C :-
   terra/terra:country->L[@terra:capital->CN and not @car_code],
   result/country->C/@capital/@name->CN.
?- sys.strat.doIt.
```

The only remaining Terra countries are those of the SU, GDR and CS (countries which have been dissolved).

Fusing Administrative Divisions. Terra contains data about cities belongig to administrative divisions which is not contained in GlobalStatistics. This information has to be integrated. Some of the administrative divisions can be fused by country/name:

```
L = P :-
    terra/terra:province->L[@terra:country->CC and @terra:abbrev->AB
                          and @terra:name->N],
    not CC = AB,
    result/province->P[@gs:name -> N]/@country[@car_code->CC].
?- sys.strat.doIt.
```

For the remaining administrative divisions, the capital coincides with the capital of an administrative division in GlobalStatistics of the same country (which is already merged). Since Terra contains the country itself as a province if no other provinces are known, these have to be excluded.

```
L = P :-
    terra/terra:province->L[@terra:country->CC and @terra:capital->CN],
    not result/province->L,
    count{X [CC]; terra/terra:province->X[@terra:country->CC]} > 1,
    result/province->P[@capital[@name->CN] and
                       @country/@car_code->CC].
?- sys.strat.doIt.
```

The provinces of Finland and Norway are only contained in Terra. They are made administrative divisions of their countries, the capital Cap is made a city and the links from the capitals to the provinces and to the country are set:

```
p(finland,"SF"), finland = C :- result/country->C[@car_code->"SF"].
p(norway,"N"), norway = C :- result/country->C[@car_code->"N"].
?- sys.strat.doIt.
result[city->Cap[@name->CN and @country->Ctry and @province->L]],
result[province->L[@country->Ctry and @name->N and
                   @capital->Cap and @population->E]] :-
   p(Ctry,CC),
   terra/terra:province->L[@terra:country->CC and @terra:abbrev->AB and
           @terra:name->N and @terra:capital->CN and @terra:pop->E],
   terra/terra:city->Cap[@terra:country->CC and
           terra:province/terra:text()->AB and @terra:name->CN].
```

For the former USSR, all its administrative divisions (the SSRs) are now independent and are fused with the respective country objects. Here, the matching is done by the first three letters of the (german) name. Note that this leads to objects which are countries and terra:provinces.

```
L = C :-
   terra/terra:province->L[@terra:name->N1 and @terra:country->"SU"],
   result/country->C/codes:name[@codes:language="german"]->N2,
   not terra/terra:country->C,
   pmatch(N1,"/\A(...)/","$1",N1short),
   pmatch(N2,"/\A(...)/","$1",N2short), N1short = N2short.
?- sys.strat.doIt.
```

Continent Information. Terra contains information about the distribution of countries over continents in its encompassed property. Note that this is not contained in CIA or GlobalStatistics for countries which belong to two continents (Russia, Egypt, Turkey).

```
C/encompassed[@continent->CT and @percentage->Perc] :-
   result/country->C/terra:encompassed[@terra:continent->CN]/terra:text()->Perc,
   result/continent->CT[@gs:name->CN].
C/encompassed[@continent->Asia and @percentage->80],
C/encompassed[@continent->Europe and @percentage->20] :-
```

```
   result/country->C[@name->"Russia"],
   result/continent->Europe[@gs:name->"Europe"],
   result/continent->Asia[@gs:name->"Asia"].
?- sys.strat.doIt.
```

all other countries belong with 100% to the continent which is given in CIA/GlobalStatistics:

```
C/encompassed[@continent->Cont and @percentage->100] :-
   result/country->C[@gs:continent->Cont and not encompassed].
?- sys.strat.doIt.
```

Fusing Cities. TERRA contains geographical coordinated for all its cities, many of them are also contained in GlobalStatistics without coordinates and Qiblih also does not provide their coordinates.>

For countries where no administrative divisions are distinguished in TERRA (i.e. terra:city[@terra:country = @terra:province), cities are merged by country/name.

```
S = C :-
   terra/terra:city->S[@terra:name->N and @terra:country->CC and
                       terra:province/terra:text()->CC],
   result/country->Country[@terra:code->CC],
   result/city->C[@country->Country and @name->N].
```

Cities where GlobalStatistics knows the province (US cities and capitals of provinces) are fused using name/province/country:

```
S = C :-
   terra/terra:city->S[@terra:name->N and @terra:country->CC and
                       terra:province/terra:text()->AB],
   not CC = AB,
   result/country->Country[@terra:code->CC],
   terra/terra:province[@terra:name->PName and @terra:country->CC
            and @terra:abbrev->AB],
   result/city->C[@country->Country and @name->N],
   result/province[@gs:name->PName].
?- sys.strat.doIt.
```

There are cities where TERRA knows the province but GlobalStatistics does not know it although the province as an object is known. These can be fused and linked using the TERRA information.

```
S = C, C[@province -> P] :-
   terra/terra:city->S[@terra:name->N and @terra:country->CC and
                       terra:province/terra:text()->AB],
   not CC = AB,
   result/country->Country[@terra:code->CC],
   terra/terra:province->P[@terra:country->CC and @terra:abbrev->AB],
   result/city->C[@country->Country and @name->N and not @province].
?- sys.break.doIt.
```

Terra cities which are capitals are fused with the CIA/GlobalStatistics capital of the same country. Here, all countries (i.e., Germany) where the capital changed must be excluded!

```
S = C :-
   result/country[@terra:code->CC and @terra:capital->N and @capital->C],
   not CC="D",
   terra/terra:city->S[@terra:name->N and @terra:country->CC],
   not result/city->S.
?- sys.break.doIt.
```

the same for provinces

```
S = C :-
   terra/terra:city->S[@terra:name->N and @terra:country->CC and
                       terra:province/terra:text()->AB],
   not result/city->S,
   terra/terra:province[@terra:country->CC and @terra:abbrev->AB and
                        @terra:capital->N and @capital->C].
?- sys.break.doIt.
```

Again, the former USSR needs special treatment: capitals of SSRs are now the capitals of the new countries:

```
S = C :-
   terra/terra:city->S[@terra:name->N and @terra:country->"SU" and
                       terra:province/terra:text()->AB],
   terra/terra:province[@terra:country->"SU" and @terra:abbrev->AB and
                        @terra:capital->N and @capital->C].
?- sys.strat.doIt.
```

non-capital USSR cities can only be matched by name (note that the former terra province has been equated with the new country)

```
S = C :-
   terra/terra:city->S[@terra:name->N and @terra:country->"SU" and
                       terra:province/terra:text()->AB],
   result/city->C[@country->Country[@terra:country->"SU" and
                       @terra:abbrev->AB] and
                  @name->N].
?- sys.strat.doIt.
```

Additional Terra Information. Terra contains some more data which has not yet been integrated, e.g. cities which are not mentioned in any of the other databases, or additional information about cities which are known.

There are some countries where GlobalStatistics gives no main cities (except the capital), but Terra gives some additional cities. Then, these cities can be added to the database without any risk to have a city under two different names. This also holds when all GlobalStatistics cities are capitals of provinces and Terra also knows these provinces/capitals (Then the capitals are already fused).
First, all GlobalStatistics cities are determined which are no capitals:

```
C[@notcapital->true] :-
   result/city->C[@country->X], not X[@capital->C],
   not X/gs:adm_divs[@capital->C].
?- sys.strat.doIt.
```

For countries which have provinces in TERRA and GlobalStatistics knows no non-capital cities (effectively, this implies that GlobalStatistics knows no provinces, these are only N and SF), all TERRA cities can be added with their provice information.

```
result[city->S[@country->C and @province->P]] :-
   result/country->C[@terra:code->CC],
   not C/gs:main_cities[@notcapital],
   terra/terra:city->S[@terra:country->CC and terra:province/terra:text()->AB],
   not CC = AB, not result/city->S,
   result/province->P[@country->C and @terra:abbrev->AB].
?- sys.strat.doIt.
```

TERRA cities which are known with their province, and GlobalStatistics knows no non-capital cities in this province can be added with their provice information.

```
result[city->S[@country->C and @province->P]] :-
   result/province->P[@country->C and @terra:abbrev->AB],
   N = count{City [P]; result/city->City[@province->P]}, N = 1,
   result/country->C[@terra:code->CC],
   terra/terra:city->S[@terra:country->CC and terra:province/terra:text()->AB],
   not CC = AB, not result/city->S.
```

For countries (except the former USSR) where GlobalStatistics knows no main cities and TERRA knows no provinces, these cities are added.

```
result[city->S[@country->C]],
C[@gs:main_cities->S] :-          %% other countries
   result/country->C[@terra:code->CC], not CC = "SU",
   N = count{City [C]; C/@gs:main_cities->City}, N = 1,
   terra/terra:city->S[@terra:country->CC and terra:province/terra:text()->CC],
   not result/city->S.
?- sys.strat.doIt.
```

TERRA city longitude and latitude data is added if there is no other (Qiblih) information:

```
C[@longitude->Long and @latitude->Lat] :-
   result/city->C[@terra:longitude->Long and @terra:latitude->Lat],
   not C/@latitude, not C/@longitude.
?- sys.strat.doIt.
```

Take the city name from TERRA if there is no other name (cities which are taken from TERRA and not equated with any gs city)

```
C[@name->N and name->N] :-  result/city->C[@terra:name->N and not name].
?- sys.strat.doIt.
```

The TERRA city population data is added if there is no other (GlobalStatistics) information:

```
C/population[@year->87 and text()->E] :-
    result/city->C[@terra:population->E and not population].
?- sys.strat.doIt.
```

For countries where the car-code is not yet given, it is taken from TERRA (currently this is only Burma and French Guiana)

```
C[@car_code->CC] :- result/country->C[@terra:code->CC and not @car_code].
```

TERRA geo objects locations. For TERRA geo objects (rivers, lakes, seas, deserts, mountains, islands), the country/province data is translated.

```
mountain subcl geo_obj.
desert subcl geo_obj.
island subcl geo_obj.
water subcl geo_obj.
lake subcl water.
river subcl water.
sea subcl water.
?- sys.strat.doIt.
result[Class->Obj] :-  terra/Class->Obj, Class subcl geo_obj.
?- sys.strat.doIt.
```

First, the countries are determined, for the former USSR, (SU,prov) is changed into (prov):

```
Obj[@country->C] :-
   terra/Class->Obj/terra:located[@terra:country_code->CC],
   Class subcl geo_obj,
   not CC =  "SU",
   result/country->C[@terra:code->CC].
Obj[@country->C] :-
   terra/Class->Obj/terra:located[@terra:country_code="SU" and @terra:province_id->AB],
   result/country->C[@terra:abbrev->AB].
Obj[country->C] :-
   terra/Class->Obj/terra:located[@terra:country_code="CS"],
   result/country->C[@car_code="CZ"].
?- sys.strat.doIt.
```

As far as provinces are given, this information is also added:

```
L[@country->C and @province->P] :-
   terra/Class/terra:located->L[@terra:country_code->CC and @terra:province_id->AB],
   result/province->P[@terra:abbrev->AB and @country->C/@terra:code->CC].
?- sys.strat.doIt.
```

There are some cities and geo objects whose provinces are not given yet. They are contained in an additional file:

```
?- sys.load@("../Mondial-programs/mondial-addprovs.flp").
?-sys.strat.doIt.
format.name="$out[0]=$in[0];
             $out[0]=~tr/áâãäåÉéèíïÍÖóôöúüçñ/aaaaaEeeiiIOooouucn/".
?-sys.strat.doIt.
addprov2(N2,CC,PN2) :- addprov(N,_,CC,PN),
  perl(format.name, N, N2),  perl(format.name, PN, PN2).
?-sys.strat.doIt.
C[@province->P] :-
  addprov2(N,CC,PN),
  result/city->C[@name->N]/@country[@car_code->CC],
  result/province->P[@name->PN]/@country[@car_code->CC].
?-sys.strat.doIt.
Obj[@country->C] :-
  addgeocountry(Geo,Name,Code), Class subcl geo_obj,
  terra/Class->Obj[@name->Name], result/country->C[@car_code->Code].
?- sys.strat.doIt.
```

For province information about geo objects, for every country, a semicolon-list of provinces is given.

```
addgeoprov(Geo, Name, Code, ProvName) :-
  addgeo(Geo, Name, Code, Strg),
  pmatch(Strg, "/([ÄA-Za-z][^;]*)/g", "$1", ProvName).
?- sys.strat.doIt.
Obj/located[@country->C and @province->P] :-
  addgeoprov(Geo, Name, Code, ProvName),
  terra/Class->Obj[@name->Name and @country->C[@car_code->Code]],
  result/province->P[@country->C and @name->ProvName].
?- sys.strat.doIt.
```

The formatting of CIA ethnicgroups etc is changed:

```
E[text()->N and @percentage->P] :-
   result/country->C/ethnicgroups->E[@cia:name->N and cia:text()->P].
R[text()->N and @percentage->P] :-
   result/country->C/religions->R[@cia:name->N and cia:text()->P].
L[text()->N and @percentage->P] :-
   result/country->C/languages->L[@cia:name->N and cia:text()->P].
?- sys.strat.doIt.
B[@country-> C and @length->L] :-
    result/country/border->B[@cia:country-> C and cia:text()->L].
?- sys.strat.doIt.
```

cities and provinces are linked as subelements

```
C[province->P] :- result/province->P[@country->C].
C[city->City] :- result/city->City[@country->C and not @province].
P[city->City] :- result/city->City[@province->P].
?- sys.strat.doIt.
```

Recall that above, the MONDIAL DTD has been parsed, yielding an *export signature*. The result tree is now exported according to this signature:

```
?- sys.theOMAccess.export@("xml","mondial-3.0.xml","mondial","result").
```

Available Files. The sources and results of this case study are freely available for educational purposes:

- The LoPiX system:
  `http://www.informatik.uni-freiburg.de/~may/lopix`.
- The MONDIAL Case Study:
  `http://www.informatik.uni-freiburg.de/~may/Mondial`.
  - The XML representations of the Sources:
    `mondial-{cia|gs|orgs|qiblih|terra|codes}.xml`
    with corresponding
    DTDs `mondial-{cia|gs|orgs|qiblih|terra|codes}.dtd`.
    Note that the programs are designed for the 1996 versions of the data sources. Depending on changes of the sources, adaptations could be necessary.
  - The XML files: `mondial-3.0.xml` and `mondial-3.0.dtd` as a hierarchical model.

References

[LoP] The LoPiX System. `http://www.informatik.uni-freiburg.de/~may/lopix/`.

[May99] W. May. Information Extraction and Integration with FLORID: The MONDIAL Case Study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999. Available from `http://www.informatik.uni-freiburg.de/~may/Mondial/`.

[May01] W. May. XPath-Logic and XPathLog: A Logic-Based Approach for Declarative XML Data Manipulation (Preliminary Report). Technical report, Universität Freiburg, Institut für Informatik, 2001. Available from `http://www.informatik.uni-freiburg.de/~may/lopix/`.