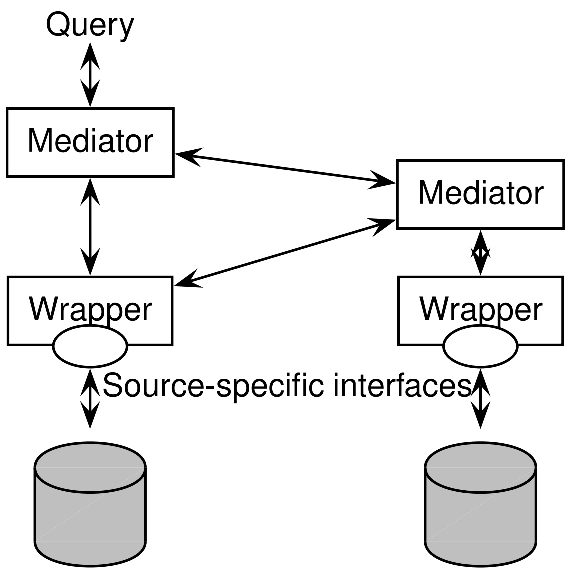
Chapter 3
Semistructured Data: Early Approaches

- Data integration
  - different, autonomous data sources
  - *different data models and schemata*
  - more advanced than the approach of SchemaSQL
- Knowledge representation, data exchange
  - schema- and meta-information inside the data
  - examples: KIF (Knowledge Interchange Format), F-Logic
  - up to ontology management ("Semantic Web")
- Management of data for presentation on the Web
- Extraction of data from the Web

SSD FOR DATA INTEGRATION/DATA EXCHANGE:

Wrapper/Mediator-Based Architectures

- *Mediator* (Vermittler): between users and data sources (Middleware),
- *Wrapper (Translator)*: provides homogeneous access to heterogeneus sources
  (especially for information extraction from the Web:
  programming of wrappers for Web pages and then collect the data)



WRAPPER/MEDIATOR-BASED ARCHITECTURES

- sources: databases, interfaces to databases via forms (e.g. library search), search engines, simple Web pages
- each relevant Web source is associated to a wrapper
- mediator contains knowledge about the accessible sources
- mediators can be composed hierarchically

Virtual Approach

The users state queries against the upper level mediator ("external view") which translates the queries against lower mediators and wrappers. Wrappers answer the queries from the sources. Mediators combine the answers and return them.

Materialized Approach

An integrated view of all data is completely materialized (and maintained). Users state their queries against the materialized database that directly answers them.

REQUIREMENTS FOR DATA INTEGRATION

- upper mediator level: a target data format
- interfaces between wrappers/mediators
  - a common data exchange format
  - a common query language/mechanism
- wrapper level: mapping from sources into the common format

Target Data Model and Languages

- flexible and extensible
  - "copy all properties of object X from data source A"
  - extensible to additional sources
  - different source data models and schemata
- handling metadata and content in combination
- self-describing data !?

3.1 TSIMMIS

(The Stanford-IBM Manager of Multiple Information Sources, 1995-2000)

Persons: J. Ullman, H. Garcia-Molina, J. Widom, Y. Papakonstantinou, etc.

Goal (several subprojects): construction of means for a consistent and efficient integrated access to information sources:

- Heterogeneous information sources
  - databases
  - Web pages

  ⇒ often no explicit schema known/present

  ⇒ mapping to a *common data model*:
  Object Exchange Model (OEM)

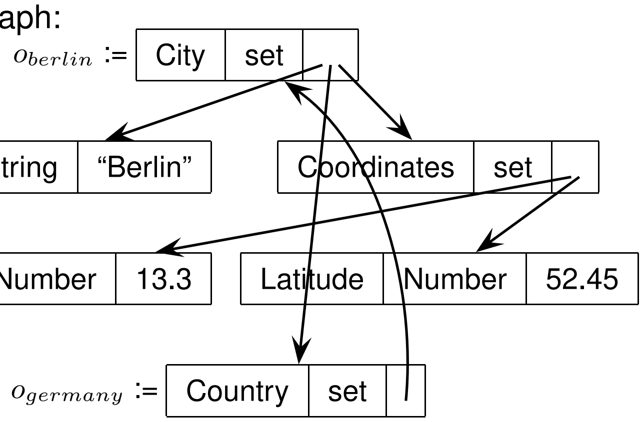
TSIMMIS: Concept

"Virtual" approach:

- users state queries against a mediator
- mediator forwards the subqueries to lower mediators or wrappers
- wrappers are programs that (logically) transform the objects of the data source into OEM and then answer the basic queries
- results of the wrappers are returned in OEM format to the mediator
- mediator integrates the results of the sources
- mediators can be composed hierarchically

3.1.1 OEM: Object Exchange Model

- very simple, "*self-describing*" object model
- knows only object identity and nesting as concepts:
- each object has an object-ID, a label ($\sim$ class), a (data)type and a value,
- values of complex types are sets of references to sub-objects
- labels: "self-describing data"
- top-level objects with semantic object identifiers as entry points (cf. OQL)
- can be represented as a graph:



OEM: EXAMPLE

Source 1: CIA World Factbook

Wrapper cia exports OEM objects as follows:

<&cont1, continent, set, {&a1, &n1, &c1, &c2, &c3, ...}>
<&n1, name, string, 'Europe'>
<&a1, area, number, 9562488>
<&c1, country, set, {&cn1, &cc1, &ca1, &cp1, &cap1}>
<&c2, country, set, {&cn2, &cc2, &ca2, &cp2, &cap2}>

<&cn1, name, string, 'Germany'>
<&cc1, code, string, 'D'>
<&ca1, area, number, 356910>
<&cp1, population, number, 83536115>
<&cap1, name, string, 'Berlin'>

<&cn2, name, string, 'Sweden'>
<&cc2, code, string, 'S'>
<&ca2, area, number, 449964>
<&cp2, population, number, 8900954>
<&cap2, name, string, 'Stockholm'>

OEM: Example

Source 2: Global Statistics

Wrapper gs exports OEM objects as follows:

<&cont1, continent, set, {&a1, &n1, &c1, &c2, &c3, . . .}>
<&n1, name, string, ‘Europe’>
<&c1, country, set, {&cn1, &ct11, &ct12, &ct13, . . . , &prov11, &prov21,. . .}>
<&c2, country, set, {&cn2, &ct21, &ct22, &ct23, . . . , &prov12, &prov22,. . .}>

<&cn1, name, string, ‘Germany’>
<&ct11, city, set, {&ctn11, &ctp11, &prov11}>
<&prov11, province, set, {&pn11, &pa11, &pp11, &ct11}>

<&ctn11, name, string, ‘Stuttgart’>
<&ctp11, population, number, 588482>

<&pn11, name, string, ‘Baden-Württemberg’>
<&pa11, area, number, 35742>
<&pp11, population, number, 10272069>

OEM

- another version of OEM has been presented that additionally allows for labeled edges; e.g. for *capital*-edges from a country to a city.

Exercise 3.1
Visualize an excerpt of the Mondial database with some countries, cities, continents and organizations as an OEM graph. □

... a very simple data model.

- how to query it?
- generally, the network model language could be used for navigating ...
- ... but in the meantime, *declarative* languages had been invented:
  - clause-based: SQL-style
  - logic-based: Datalog-style

3.1.2 TSIMMIS: Languages

Mediators are programmed in MSL (Mediator Specification Language; a rule-based query language for OEM):

MSL rules (cf. Prolog, Datalog):

head(Vars) :- body(Vars,databases)

- head and body consist of expressions over *patterns* of the form

  <oid label type value>
  or
  <oid label value>
  or
  <label value>

- *value* can be set-valued; in this case, the set consists itself of expressions of the form `<...>` .
- objects of different sources are identified by <object>@source.
- *body*: pattern that must be satisfied by suitable variable bindings,
- *head* describes the structure of the OEM object that is generated.

MSL

The country whose code is "D":

```
?- <C country {<code "D">}>@cia
```

- the query generates a set-valued result object, whose sub-objects are the individual answers:

result: <answer {&c1}>

The names of all south-american countries in which there is a city with name "Santiago":

```
<countryname N>:-
      <continent {<name "South America"> <country
           {<name N> <city {<name "Santiago">}>}>}>@gs
```

<&obj42, answer, set, {<countryname "Chile">,
<countryname "Paraguay">,
<countryname "Argentina">}>

EXAMPLE

Mediator med accesses wrappers cia and gs.

Query: all cities that are stored in gs whose names are mentioned in cia as names of capitals.

Mediator rule:

<capital {<name Cap> <country CN> R }>@med :-
    <country {<name CN> <capital Cap>}>@cia
    AND <country {<name CN> <city {<name Cap> | R}>}>@gs

- R is bound to the remaining sub-objects of the resulting city-objects.

⇒ object creation in the rule head $obj@med$ (cf. Views)

exported object e.g. <&cap, capital, set, {&ctn12, &c1, &ctp12, &ctprov12}>@med

- additionally: external predicates (string concatenation, substring, comparisons etc).
- variables can be bound to labels and to values
- *syntactically* 2nd order

⇒ queries against the logical schema are possible.

3.1.3 TSIMMIS: User Queries Against OEM

Users can state queries in MSL or LOREL:

- MSL: see above – rule- and pattern-based language
- LOREL (Lightweight Object Repository Language):
  (LORE: DBMS for OEM data model, Stanford)
  clause- and path-based language (based on OQL)

Lorel

- Entry points are named constants (e.g. europe) or extents (e.g. countries)
- result of each query is a collection of OEM objects.

Semantics of 1:N-relationships

multi-valued instead of *set-valued* semantics:

- germany.city yields *multiple unary* answers instead of (as in ODMG) *one set-valued* answer.
- germany.city.name yields the names of all these cities.

Lorel

clause-based SQL/OQL-style language: SELECT - FROM - WHERE

```
% all europ.  capitals:
select europe.country.capital % note:  multivalued semantics

% the country with the code "S":
select c
from europe.country c
where c.code = "S"

% South-american countries such that ...
select c
from southamerica.country c
where c.city.name = "Santiago"
```

implicit existential semantics: ... if there is any city whose name is Santiago.

3.2 Frame-Based Models

- objects are represented by *object frames*,
- Frame contains *slots* for storing properties
  ("signature" of the slots *can* be given by a schema, but not necessarily ($\Rightarrow$ semistructured data))
- Slots can be literal-valued or object-valued (*internal* storage by references) for describing attributes and relationships.

A Self-Describing OO-Data Model: F-Logic

(M. Kifer, G. Lausen, 1989/1995; SIGMOD Test of Time Award 1999)

- full object-orientation (class hierarchy, inheritance)
- objects have properties
- metadata (class names, method names) as first-class-members of the data model
- "frame-based" model
- deductive language (Prolog-style)

F-Logic: Data Model and Syntax

- is-a relationship:
  <object> : <class>
- subclass-relationship:
  <class> :: <class>
- properties:
  <object>[<property>$\rightarrow$<object/value>] (scalar)
  <object>[<property>$\twoheadrightarrow${<set-of-objects/values>}] (set-valued/multi-valued)
  Analogously with parameters:
  <object>[<property>@(<list-of-objects/values>)$\rightarrow$<object/value>]
  <object>[<property>@(<list-of-objects/values>)$\twoheadrightarrow$<{set-of-objects/values}>]
- inheritable properties:
  <class>[<property>$\bullet\!\!\rightarrow$<object/value>] (scalar)
  <class>[<property>$\bullet\!\!\twoheadrightarrow${<set-of-objects/values>}] (set-valued)
  nonmonotonic inheritance semantics with overriding.

F-Logic: Example

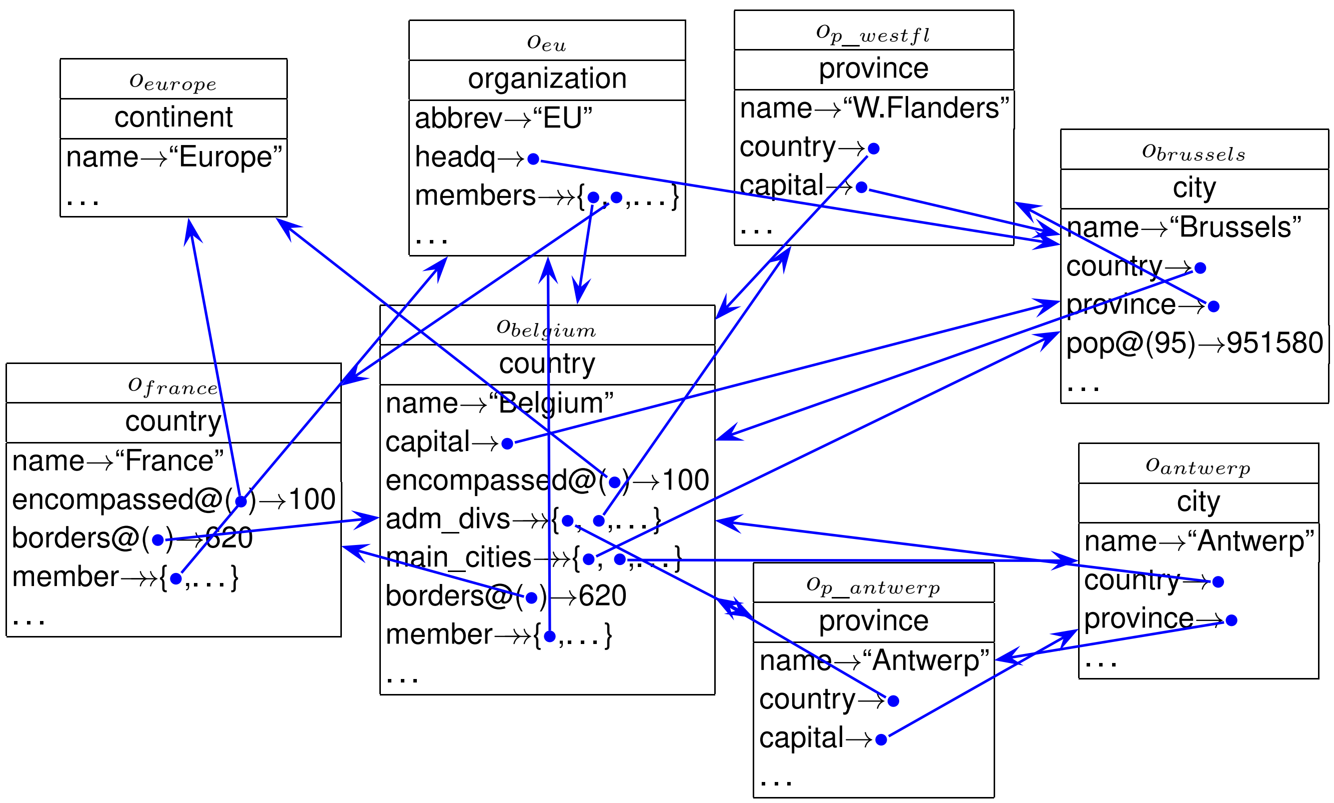
$o_{belgium}$: $country$[name$\rightarrow$"Belgium"; car_code$\rightarrow$"B";
capital$\rightarrow$$o_{brussels}$; independence$\rightarrow$"04 10 1830";
total_area$\rightarrow$30510; population$\rightarrow$10170241;
encompassed@(o_{europe})$\rightarrow$100; pop_growth$\rightarrow$0.33;
adm_divs$\twoheadrightarrow$$\{o_{p_antwerp}, o_{p_westfl}, \ldots\}$;
main_cities$\twoheadrightarrow$$\{o_{brussels}, o_{antwerp}, \ldots\}$;
borders@(o_{france})$\rightarrow$620; borders@($o_{germany}$)$\rightarrow$167;
borders@($o_{luxembourg}$)$\rightarrow$148; borders@($o_{netherlands}$)$\rightarrow$450].

$o_{brussels}$: city[name$\rightarrow$"Brussels"; country$\rightarrow$$o_{belgium}$;
province$\rightarrow$$o_{p_westfl}$; population@(95)$\rightarrow$951580].

o_{p_westfl} : prov[name$\rightarrow$"West Flanders"; country$\rightarrow$$o_{belgium}$;
capital$\rightarrow$$o_{brussels}$; area$\rightarrow$3358; population$\rightarrow$2253794].

o_{eu} : org[abbrev$\rightarrow$"EU"; name$\rightarrow$"European Union";
established$\rightarrow$"07 02 1992"; headq$\rightarrow$$o_{brussels}$;
members@("member")$\twoheadrightarrow$$\{o_{belgium}, o_{france}, \ldots\}$;
members@("applicant")$\twoheadrightarrow$$\{o_{hungary}, o_{slovakia}, \ldots\}$].

Representation as a Graph



Properties

- arbitrary properties of an objects can be stored – simply generate and fill a slot. *self-describing* data model
- null values need not to be stored explicitly, slots are simply not filled.
- *navigation* with *path expressions* in combination with *patterns*:
  - population of the province where the capital of Belgium is located:
    ?- C:country[name→”Belgium”].capital.province[population →P].
  - all names of cities in Belgium:
    ?- C:country[name→”Belgium”]..main_cities[name→N].
  - sum of population of all provinces of Belgium:
    ?- Z = sum{X | C:country[name→”Belgium”]..province[population→X]}.
  - can be nested in complex conditions:

?- C:country[encompassed@(_Cont[name→”Europe”])→_X; member→→_O[name→“EU”];
population→CP; area→CA]..city[population→CitP; name→N],
CP > 1000000, CA > 100000, CitP > 0.25 * CP.

F-LOGIC: SIGNATURE

The signature can also be formalized in F-Logic:

- properties:
  <type>[<property>⇒<type>] (scalar)
  <type>[<property>=>><type>] (set-valued)
  analogously with parameters:
  <type>[<property>@(<list-of-types>)⇒<type>]

- country[name⇒string; capital⇒city;
  total_area⇒number; population⇒number;
  encompassed@(continent)⇒number;
  adm_divs=>>province; main_cities=>>city;
  borders@(country)⇒number].

  city[name⇒string; country⇒country;
  province⇒province; population@(number)→number].

- queries against metadata: ?- X:country[M→(_V:C)] or ?- country[M⇒C].

F-LOGIC AS A PROGRAMMING LANGUAGE

- object-oriented logic
- *deductive* database language (i.e. Prolog-style) with fixpoint semantics

```
<head>:- <body>.
?- <query>.
```

- e.g., transitive closure can be expressed:

  R[transitive_flows↠S] :- R:river[to@(sea)→S].
  R1[transitive_flows↠S] :- R1:river[to@(river)→R2], R2:river[transitive_flows↠S].

Implementations

- The FLORID system (F-Logic Reasoning in Databases; C++):
  http://www.informatik.uni-freiburg.de/~dbis/florid
- FLORA/FLORA II; XSB-Prolog: http://flora.sourceforge.net/

... current use: reasoning in the Semantic Web.

DATA INTEGRATION FROM THE WEB WITH FLORID

- warehouse approach
- direct mapping from HTML trees to F-Logic
- queries against Web pages possible
- wrapper + mediator functionality programmed by F-Logic rules

1998: Generation of the MONDIAL database from the Web

- F-Logic wrappers for several Web pages
  - CIA World Factbook Country Pages
  - CIA World Factbook Organizations Pages
  - "Global Statistics"
  - some smaller sources + the original Karlsruhe TERRA database
  - ⇒ materialized F-Logic representation of each source
- F-Logic integration program that *stepwise* materializes an integrated database
- advantages of the warehouse approach for complex integration tasks (basic rules + exceptional and refining rules)

3.3 Summary: Database Aspects (1995)

- Integration of data from different, heterogeneous sources:
  - relational distributed/federated databases: metadata queries and -integration (SchemaSQL)
  - arbitrary sources (Tsimmis): metadata queries, general, flexible data model
  - sources can also be Web pages (HTML) (Tsimmis, Florid)
- semistructured nature of data
  - no fixed schema
  - implicit null values
  - easily extensible (adding new properties)
  - object as a collection of properties
  - self-describing data + metadata
- languages for semistructured data
  - metadata queries
  - generation of structure

⇒ flexible

QUERY LANGUAGES

- declarative
- clause-based; iterator variables and SELECT-FROM-WHERE-clause
  (SQL, OQL, Lorel)
- logical/deductive; binding of variables by patterns; constructive semantics of rule heads
  - patterns as terms
    (WSL/MSL)
  - patterns as extended path expressions
    (F-Logic)
  - as programming languages: fixpoint semantics
- multivalued vs. set-valued semantics
- rule-based (more or less explicit):
  - binding of variables in the "body" (SQL/OQL: FROM-WHERE clause)
  - result generation in the head

F-LOGIC [1989] AS A PREDECESSOR OF XML, RDF, OWL

- semistructured (⇒ XML, RDF)
- self-describing (⇒ XML, RDF)
- data model
  - database model: complex objects, properties, relationships (⇒ XML, RDF)
  - but also knowledge representation model with *built-in reasoning* (⇒ OWL)
  - optional schema information (⇒ XSD, RDFS, OWL)
- query language
  - navigation, path expressions with predicates and multivalued semantics (⇒ XPath)
- derivation rules (⇒ OWL + Rules [SWRL?])

RDF: Resource Description Format, 1997, see Lecture "Semantic Web"
OWL: Web Ontology Language, 2002 [OIL: 2000], see Lecture "Semantic Web"

3.4 Situation 1996

- Experiences with SQL (and ODMG/OQL) as database languages
  - standardization vs. products
- document management with SGML (Structured Generic Markup Language), CSS (Cascading Stylesheets) and DSSSL
- data exchange/access via internet/Web:
  - homogeneous solution necessary
  - availability of documents and data in HTML:
    * very simple variant of SGML
    * "native" HTML data (handwritten)
    * mapping of SGML (document management) to HTML (publication) by CSS
    * HTML-Web-Servers over relational databases

$\Rightarrow$ "Global" approach coordinated by the W3C (World Wide Web Consortium): development of a data model (+ language), that can handle (legacy-)databases, documents and Web (=HTML)

The W3C (World Wide Web Consortium)

- `http://www.w3.org`.
- founded in 1994 for developing common protocols and languages for the World Wide Web and to ensure interoperability of applications in the Web.
  (Tim Berners-Lee, MIT, CERN)
- following the principles of OMG/ODMG who developed the CORBA and ODL/OIF/OQL standards
- members: companies and research institutes
- definition of working groups
- notes $\rightarrow$ working drafts $\rightarrow$ recommendations
- not only XML, but also many other Web-related issues

3.5 Documents: SGML and HTML

- Structuring (und presentation) are called *(logical and optical)* *“markup”*.
  (document = content + markup)
- SGML (Standard Generalized Markup Language),
  development (IBM) since 1979, standard 1986.
  structuring and markup of documents, widely used in publishing.
- for publishing in the Web:
  HTML (Hypertext Markup Language), development since 1989 (CERN), standard 1991.

⇒ HTML is *an* SGML application with a fixed syntax
(tags, attributes, later: DTD).
goal: optical markup, as a side effect also some structuring of the documents (cf. <P>-Tag).

- SGML much more flexible than HTML → more complex → not suitable for browsers
  (HTML allows for efficient and fault-tolerant parsing)
- SGML sources can be transformed to HTML by stylesheets (CSS: Cascading Style Sheets).