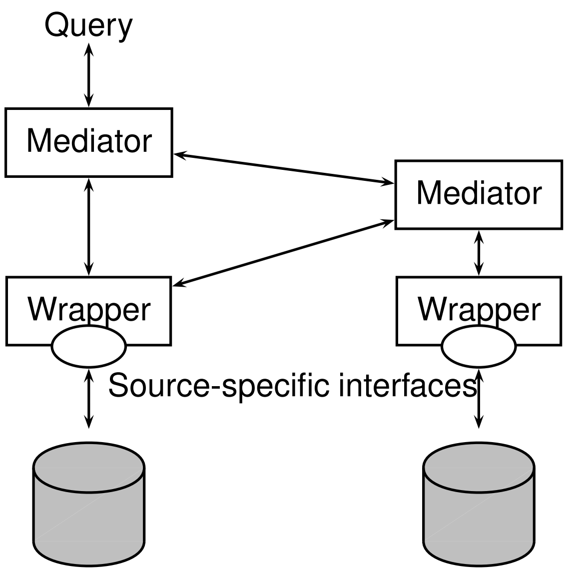
Kapitel 3
Semistrukturierte Daten: Erste Ansätze

- Datenintegration
  - verschiedene, autonome Datenquellen
  - verschiedene Datenmodelle und Schemata
  - weitergehend als der Ansatz von SchemaSQL
- Wissensrepräsentation, Datenaustausch
  - Schema- und Metainformationen innerhalb der Daten
  - Beispiele: KIF (Knowledge Interchange Format), F-Logic
  - bis hin zum Ontologie-Management ("Semantic Web")
- Verwaltung von Daten für Web-Bereitstellung
- Extraktion von Daten aus dem Web

SSD ZUR DATENINTEGRATION/DATENAUSTAUSCH:

Wrapper/Mediator-basierte Architekturen

- *Mediator* (Vermittler): Ebene zwischen Benutzern und Datenquellen (Middleware),
- *Wrapper (Translator)*: bieten dem Mediator homogenen Zugriff zu heterogenen Quellen
  (insb. Informationsgewinnung aus dem Web:
  Wrapper für Web-Seiten programmieren und Daten dann zusammenfassen)



Wrapper/Mediator-basierte Architekturen

- Quellen: Datenbanken, Oberflächen auf Datenbanken mit Forms/Suchmasken (z.B. Bibliotheks-Suchsysteme), Suchmaschinen, einfache Web-Seiten
- jede verwendete Datenquelle ist einem Wrapper zugeordnet
- Mediator enthält Wissen über die zur Verfügung stehenden Datenquellen
- Mediatoren können hierarchisch angeordnet sein.

⇒ Schnittstellen zwischen Wrappern/Mediatoren

⇒ Notwendigkeit eines Datenaustauschformats
  - für Metadaten
  - für Contents
  - beides zusammen !?
  - self-describing data !?

3.1 TSIMMIS

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

Personen u.a. J. Ullman, H. Garcia-Molina, J. Widom, Y. Papakonstantinou

Projektziel (mehrere Teilprojekte): Schaffung von Hilfsmitteln zum konsistenten integrierten Zugriff zu Informationsquellen:

- Heterogene Informationsquellen
  - Datenbanken
  - Web-Seiten

  ⇒ häufig kein explizites Schema vorhanden
  ⇒ Abbildung auf ein *gemeinsames Datenmodell*
  Object Exchange Model (OEM)

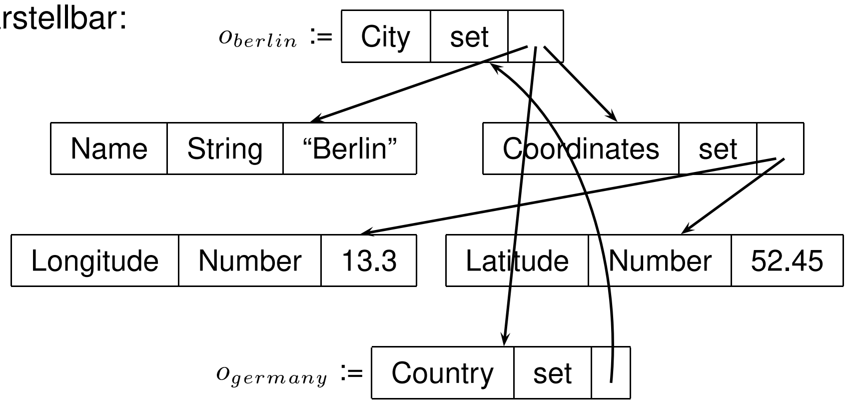
TSIMMIS: Konzept

Virtueller Ansatz:

- Benutzer stellt Anfrage an einen Mediator
- Mediator gibt Anfrage an die Wrapper (“Translatoren”) weiter
- Translatoren sind Programme, die die Datenobjekte der Quellen (logisch) in OEM transformieren und die Anfragen aus den Quellen beantworten können
- Resultate der Quellen werden in OEM an den Mediator zurückgegeben
- Mediator integriert die Antworten der Quellen.
- Mediatoren können hierarchisch angeordnet sein.

3.1.1 OEM: Object Exchange Model

- sehr einfaches, “*selbstbeschreibendes*” Objektmodell
- kennt nur Objektidentität und Schachtelung als Konzepte:
- Jedes Objekt hat eine Objekt-ID, ein Label ($\sim$ Klasse), einen Typ und einen Wert,
- Werte komplexer Typen sind Mengen von Referenzen auf Subobjekte
- Labels: “selbstbeschreibende Daten”
- top-level Objekte mit semantischen Objektbezeichnern als Einstiegspunkte (vgl. OQL)
- als Graph darstellbar:



OEM: BEISPIEL

Quelle 1: CIA World Factbook

Wrapper cia exportiert OEM-Objekte wie folgt:

<&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: Beispiel

Quelle 2: Global Statistics

Wrapper gs exportiert OEM-Objekte wie folgt:

<&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

- Es gibt eine Version von OEM, die zusätzlich Namen ("Labels") an den Kanten erlaubt; z.B. für *capital*-Kanten von einem Land zu einer Stadt.

Aufgabe 4
Stellen Sie einen Ausschnitt der Mondial-Datenbank mit einigen Ländern, Städten, Kontinenten und Organisationen als OEM-Graphen dar. □

3.1.2 TSIMMIS: Sprachen

Mediatoren werden in MSL (Mediator Specification Language; eine regelbasierte Anfragesprache an OEM) programmiert:

MSL-Regeln (Vgl. Prolog, Datalog):

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

- head und body bestehen aus Ausdrücken über *Patterns* der Form

  <*oid label type value*>
  oder
  <*oid label value*>
  oder
  <*label value*>

- *value* kann mengenwertig sein; in diesem Fall besteht die Menge wieder aus Ausdrücken der Form $<\ldots>$.
- Objekte der verschiedenen Quellen werden durch <objekt>@source identifiziert.
- $body$: Pattern, das mit geeigneten Variablenbindungen erfüllt werden muß,
- $head$ beschreibt die Struktur des OEM-Objekts, das generiert werden soll.

MSL

Das Land, das “D” als Code besitzt:

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

- Anfrage erzeugt ein mengenwertiges answer-Objekt, dessen Subobjekte die einzelnen Antworten sind.

Ergebnis: <answer {&c1}>

Die Namen aller südamerikanischen Länder, in denen es eine Stadt namens Santiago gibt:

- Annahme: *southamerica* als top-level-Objekt zugreifbar

```
<countryname N>:-
<southamerica {<country
      {<name N> <city{<name ‘‘Santiago’’>}>}>}>@gs
```

<&obj42, answer, set, {<countryname “Chile”>,
<countryname “Paraguay”>,
<countryname “Argentinien”>}>

BEISPIEL

Mediator med hat Zugriff zu Wrappern cia und gs.

Anfrage: alle Städte die in cia als capital-Name auftreten und in gs gespeichert sind:

Mediator-Regel:

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

- R wird an die restlichen Subobjekte des gefundenen city-Objektes gebunden.

⇒ Objekterzeugung im Regelkopf $obj@med$ (vgl. Views)

exportiertes Objekt z.B. <&cap, capital, set, {&ctn12, &c1, &ctp12, &ctprov12}>@med

- außerdem: externe Prädikate zugelassen (Stringverkettung, Substring, Vergleiche etc).
- Variablen können an Labels und an Werte gebunden werden
- *syntaktisch* 2.Ordnung.

⇒ Anfragen an das logische Schema möglich

3.1.3 TSIMMIS: Benutzeranfragen an OEM

Benutzer können Anfragen in MSL oder LOREL stellen.

- MSL: siehe oben – regel- und patternbasierte Sprache
- LOREL (Lightweight Object Repository Language):
  (LORE: DBMS für OEM-Datenmodell, Stanford)
  Klausel- und pfadbasierte Sprache (basiert auf OQL)

Lorel

- Einstiegspunkte sind benannte Konstanten (z.B. europe) oder Extents (z.B. countries)
- Ergebnis jeder Anfrage ist eine Kollektion von OEM-Objekten

Semantik von 1:N-Beziehungen

Mehrwertige statt *mengenwertige* Semantik:

- germany.city ergibt *mehrere einelementige* Antworten anstatt (wie in ODMG) *eine mehrelementige*.
- germany.city.name ergibt die Namen aller dieser Städte.

LOREL

```
% alle europ. Hauptstädte:
select europe.country.capital

% Das Land mit dem Code ``S'':
select c
from europe.country c
where c.code = ``S''

% Südamerikanische Länder mit ...
select c
from southamerica.country    % Top-Level-Objekt benennen
where c.city.name = ``Santiago''
```

Existenz-Semantik: ... wenn es eine Stadt gibt, die Santiago heißt.

3.2 Frame-basierte Modelle

- Objekte werden als *Objektframes* repräsentiert,
- Frame enthält *Slots* um Eigenschaften abzuspeichern
  (“Signatur” der Slots *kann* vorgegeben sein, muß aber nicht (⇒ Semistructured Data))
- Slots können literal- oder objektwertig sein (Speicherung *intern* durch Referenzen) und damit *Beziehungen* beschreiben.

Ein selbstbeschreibendes OO-Datenmodell: F-Logic

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

- volle Objektorientierung (Klassenhierarchie, Vererbung)
- Objekte haben Eigenschaften
- Metadaten (Klassennamen, Methodennamen) als first-class-members des Datenmodells
- “frame-basiertes” Modell
- deduktive Sprache (Prolog-artig)

F-Logic: Datenmodell und Syntax

- is-a Beziehung:
  <object> : <class>
- Subklassen-Beziehung:
  <class> :: <class>
- Eigenschaften:
  <object>[<property>→<object/value>] (skalar)
  <object>[<property>↠{<set-of-objects/values>}] (mengenwertig/mehrwertig)
  Analog mit Parametern:
  <object>[<property>@(<list-of-objects/values>)→<object/value>]
  <object>[<property>@(<list-of-objects/values>)↠<{set-of-objects/values}>]
- vererbbare Eigenschaften:
  <class>[<property>•→<object/value>] (skalar)
  <class>[<property>•↠{<set-of-objects/values>}] (mengenwertig)
  nichtmonotone Wertvererbungssemantik mit Overriding.

F-Logic: Beispiel

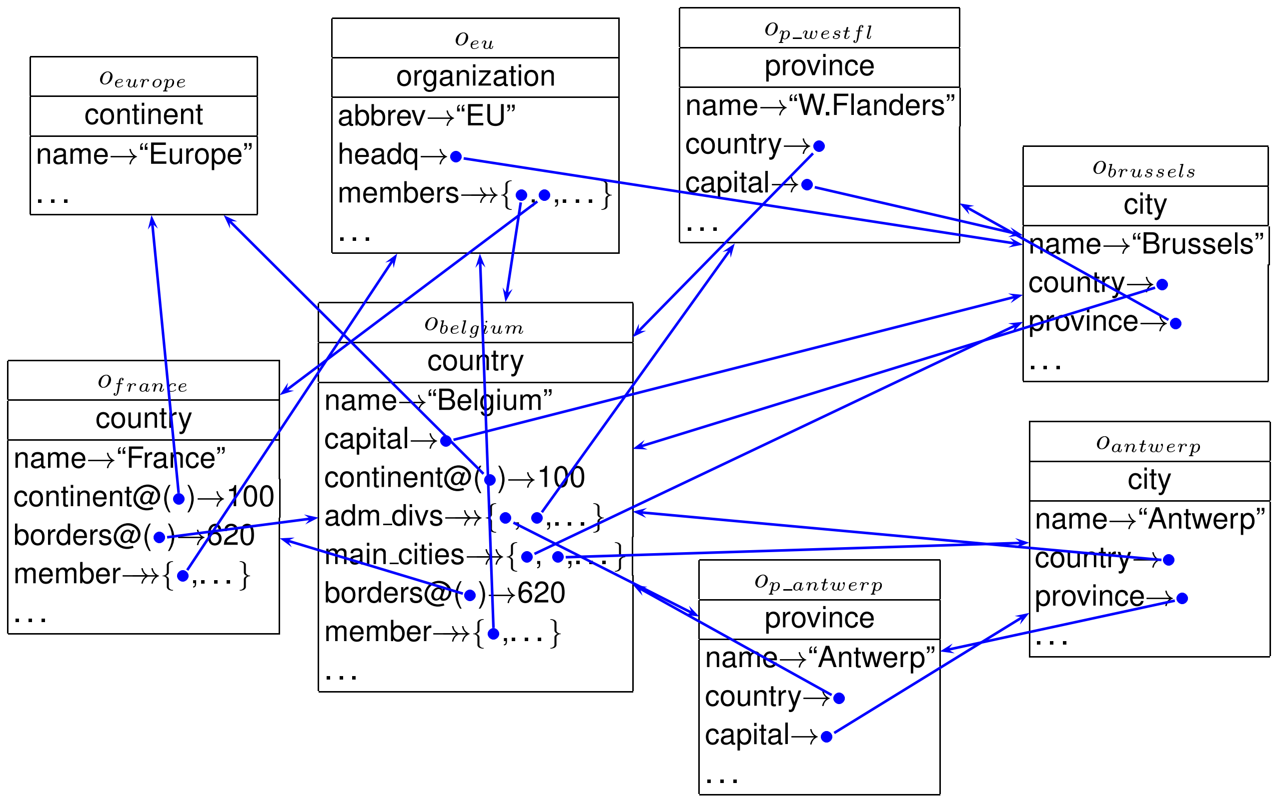
$o_{belgium}$: *country*[name→ "Belgium" ; car_code→ "B" ;
capital→$o_{brussels}$; independence→ "04 10 1830" ;
total_area→30510; population→10170241;
continent@(o_{europe})→100; pop_growth→0.33;
adm_divs→→{$o_{p\_antwerp}$, o_{p_westfl}, . . . };
main_cities→→{$o_{brussels}$, $o_{antwerp}$, . . . };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

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

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

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

Repräsentation als Graph



EIGENSCHAFTEN

- beliebige Eigenschaften eines Objektes speicherbar – einfach Slot anlegen. *selbstbeschreibendes* Datenmodell
- Nullwerte brauchen nicht explizit gespeichert werden, Slots werden einfach nicht belegt.
- *Navigation* mit *Pfadausdrücken* in Kombination mit *Patterns*:
  - Einwohnerzahl der Provinz, in welcher die belgische Hauptstadt liegt:
    $o_{belgium}$.capital.province.population = P
  - Alle Namen von belgischen Städten:
    $o_{belgium}$..main_cities[name→N]
  - Die Summe der Einwohnerzahlen aller belgischen Provinzen:
    ?- Z = sum{X | $o_{belgium}$..province[population→X]}
  - beliebig komplex geschachtelt in den Bedingungen:
    o_{europe}..country[memberships↠O[name→"EU"];
    population→CP; area→CA]..city[population→CitP; name→N],
    CP > 1000000, CA > 100000, CitP > 0.25 * CP.

F-LOGIC: SIGNATUR

Auch die Signatur kann in F-Logic formalisiert werden:

- Eigenschaften:
  <type>[<property>⇒<type>] (skalar)
  <type>[<property>⇛<type>] (mengenwertig)
  Analog mit Parametern:
  <type>[<property>@(<list-of-types>)→<type>]
- country[name⇒string; capital⇒city;
  total_area⇒number; population⇒number;
  continent@(continent)⇒number;
  adm_divs⇛province; main_cities⇛city;
  borders@(country)⇒number]

  city[name⇒string; country⇒country;
  province⇒province; population@(number)→number]
- Anfragen an Metadaten: ?- X:country[M→(_V:C)] oder ?- country[M⇒C]

F-Logic als Programmiersprache

- Objektorientierte Logik
- *Deduktive* Datenbanksprache (d.h. Prolog-Stil) mit Fixpunktsemantik

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

- Z.B. Transitive Hülle

  R[transitive_flows→→S] :- R:river[to@(sea)→S].

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

- FLORID-System:

http://www.informatik.uni-freiburg.de/~dbis/florid

- Erweiterung: Zugriff auf Web-Seiten zur Integration von HTML-Daten (Wrapper + Mediator-Funktionalität)
  So ist 1998 die MONDIAL-Datenbank entstanden ...

3.3 Zusammenfassung: Datenbankaspekte (ca. 1995)

- Integration von Daten aus unterschiedlichen Quellen:
  - Relationale Föderationen: Metadatenanfragen und -einbindung (Schema SQL)
  - Beliebige Quellen (Tsimmis): Metadatenanfragen, allgemeines flexibles Datenmodell
  - Quellen können auch z.B. Web-Seiten (HTML) sein (Tsimmis, Florid)
- Semistrukturiertheit
  - kein festes Schema
  - Nullwerte implizit
  - leicht erweiterbar
  - Objekt als Ansammlung von Eigenschaften
- selbstbeschreibende Daten
  - Datenmodell an sich
  - Datenaustausch

⇒ flexibel

ANFRAGESPRACHEN

- deklarativ
- klauselbasiert; Iteratorvariablen und SELECT-FROM-WHERE-Klausel
  (SQL, OQL, Lorel)
- logisch/deduktiv; Bindung von Variablen durch Patterns; Konstruktionssemantik im Regelkopf
  - Patterns in Form von Termen
    (WSL/MSL)
  - Patterns in Form von erweiterten Pfadausdrücken
    (F-Logic)
  - als Programmiersprachen: Fixpunktsemantik
- mehrwertige vs. mengenwertige Semantik

3.4 Gesamtsituation 1996

- Erfahrungen mit SQL (und ODMG/OQL) als Datenbanksprache
  - Standardisierung vs. Produkte
- Dokumentenmanagement mit SGML (Structured Generic Markup Language), CSS (Cascading Stylesheets) und DSSSL.
- Datenaustausch/zugriff per Internet/Web:
  - homogene Lösung notwendig
  - Bereitstellung von Dokumenten und Daten in HTML
    * sehr einfache Variante von SGML
    * Abbildung von SGML mit CSS nach HTML
    * HTML-Web-Server auf relationalen Datenbanken

⇒ “Globaler” Ansatz durch W3C (World Wide Web Consortium) als Koordinationsinstanz: Gesucht wird ein Datenmodell (+ eine Sprache), das (Legacy-)Datenbanken, Dokumente und Web (=HTML) vereinigt

Das W3C (World Wide Web Consortium)

- www.w3.org.
- Gegründet in 1994 um allgemeine Protokolle und Sprachen für das World Wide Web zu entwickeln um die Interoperabilität von Anwendungen im Web zu sichern.
  (Tim Berners-Lee, MIT, CERN)
- Vorbilder: u.a. OMG/ODMG die z.B. den CORBA-Standard und ODL/OIF/OQL entwickelt haben
- Mitglieder: Firmen und Forschungseinrichtungen
- Bildung von Working Groups
- Notes → Working Drafts → Recommendations
- nicht nur XML, sondern auch viele andere Dinge

3.5 Dokumente: SGML und HTML

- Gliederung (und eventuelle Präsentationsbefehle) werden als (logisches und optisches) *“Markup”* bezeichnet. (Dokument = Inhalt + Markup)
- SGML (Standard Generalized Markup Language),
  Entwicklung (IBM) seit 1979, Standard 1986.
  Strukturierung und Auszeichnung von Dokumenten,
  im Verlagswesen weit verbreitet.
- für Darstellung im WWW:
  HTML (Hypertext Markup Language), Entwicklung seit 1989 (CERN), Standard 1991.

⇒ HTML ist *eine* SGML-Applikation zu einer festgelegten Syntax
(Tags, Attribute, siehe später: DTD).
Ziel: Optisches Markup, als Nebeneffekt auch Strukturierung der Dokumente (vgl. <P>-Tag).

- SGML sehr viel flexibler als HTML → komplexer → nicht browsertauglich (HTML ermöglicht nahezu idiotensicheres Parsen)
- SGML-Sourcen können mit Stylesheets (CSS: Cascading Style Sheets) nach HTML transformiert werden.