# Semistrukturierte Daten und deklarativer Zugriff auf semistrukturierte Daten



## Semistrukturierte Daten

Was sind semistrukturierte Daten?

- Daten, die nicht zu einem apriori definierten Schema konform sein müssen
  (bzw. können Schemata so wenig restriktiv sein, dass sie unwichtig werden - Beispiel: HTML-DTD),
- Daten mit üblicherweise irregulärer Struktur:
  - Daten können fehlen,
  - ähnliche Konzepte werden durch unterschiedliche Typen repräsentiert,
  - Mengen können heterogen sein,
  - Struktur ist nicht vollständig bekannt.

## Semistrukturierte Daten - Motivation

- Daten lassen sich
  i.a. nicht problemlos in Tabellen abbilden,
  Strukturierung ist aber trotzdem wünschenswert.
  Beispiel: Wörterbuch-Einträge Beispiel entwickeln, ggf. Woerterbuch mitnehmen
  - Optionale Bestandteile,
  - Wiederholungen von Strukturelementen,
  - Reihenfolge der Elemente vielleicht relevant.
- Man möchte die Struktur von Daten
  nur teilweise explizit machen.
  Beispiel: Bestellungen haben strukturierten Anteil,
  aber auch frei definierbare Zusätze.



## Semistrukturierte Daten - Motivation (2)

- Warum sind Daten nicht stets explizit strukturiert?
  - nicht sinnvoll,
  - nicht eindeutig,
  - nicht bekannt,
  - Erstellung der Strukturierung zu aufwendig.
- Semistrukturierte Daten sind im Zusammenhang
  'Interoperable Informationssysteme' von Interesse,
  um Interoperabilität möglichst allgemein
  zu gewährleisten.

## Semistrukturierte Daten - Fallunterscheidung

Semistrukturierte Daten -
unterschiedliche Verwendungen des Begriffs:

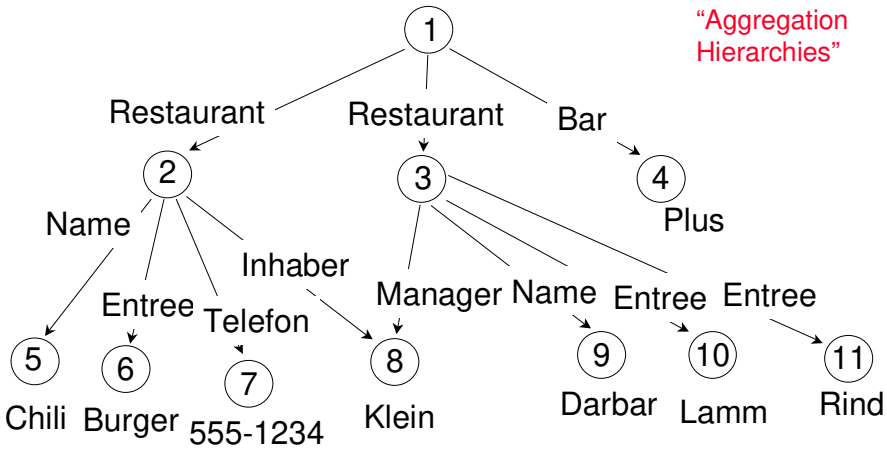
- Schema existiert,
  'Semistrukturiertheit' heisst nur 'Irregularität'
  oder 'teilweise strukturiert, teilweise nicht',
  Beispiel: SGML-/XML-DTDs
- kein Schema,
  aber Struktur der Daten eindeutig erkennbar, z.B.
  valide Dokumente a la XML,
- Struktur nicht einwandfrei identifizierbar,
  kann z.B. in HTML-Dokumenten vorkommen.



## Object Exchange Model (OEM)

- Jedes Objekt hat OID und Wert:
  - Atomarer Wert, z.B. int, string,
  - komplexer Wert:
    Menge von Subobjekten,
    Verknüpfung mit Parent durch Label.
- 'OEM-Objekt' heisst: Objekt + Subobjekte.

# OEM - Beispiel




"Aggregation Hierarchies"

Mehrere Parents und Zykel sind erlaubt.



# Pfade im OEM-Kontext

- *Label path eines OEM Objekts o* -
  Folge von Labels, separiert durch Punkte, $l_1.l_2...l_n$,
  so dass man von o aus den Pfad $(e_1, ..., e_n)$
  traversieren kann, und Kante $e_i$ hat Label $l_i$. Beispiel
- *Data path eines OEM Objekts o* -
  Alternierende Folge von Labels und OIDs,
  separiert durch Punkte, $l_1.o_1.l_2.o_2...l_n.o_n$,
  so dass man von o aus den Pfad $(e_1, ..., e_n)$
  durch Objekte $(x_1, ..., x_n)$ traversieren kann,
  Kante $e_i$ hat Label $l_i$, und Objekt $x_i$ hat OID $o_i$.
- Ein data path d ist Instanz eines label paths l,
  wenn die Folgen der Labels übereinstimmen.

Alle Instanzen von Resaurant.Entree

## Target Set

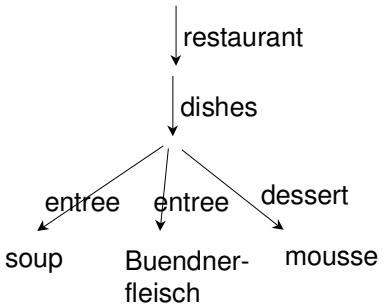
Das *Target Set in einem OEM Objekt s von einem label path l von s* ist die Menge
$t = \{o | l_1.o_1.l_2.o_2 \ldots l_n.o$ ist Instanz von $l\}$

Target Set von Resaurant.Entrée
Label Paths des Target Sets {8}

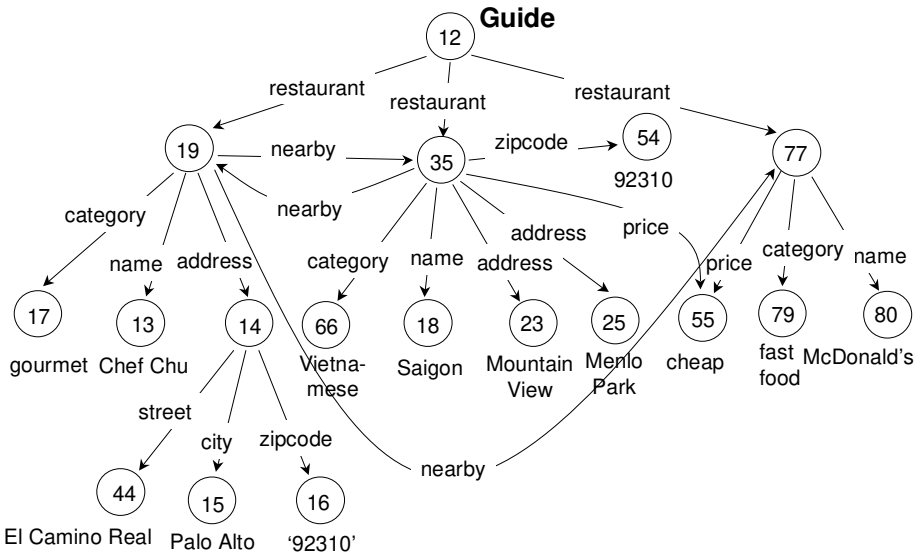


## Deklarativer Zugriff auf semistrukturierte Daten

- Man kann bzw. möchte Struktur der Daten nicht immer genau spezifizieren,
- manchmal haben die Daten aber Struktur, und sie ist dem Benutzer bekannt.



# OEM - Datenbank






# 'Unregelmässigkeiten' in der Beispiel-Datenbank

- Restaurants haben beliebig viele Adressen,
- Adressen sind manchmal Strings, haben manchmal aber auch explizite Struktur,
- zipcode kann String oder Integer sein,
- zipcode ist manchmal direkter Bestandteil von restaurant, manchmal Bestandteil von address.

## Definitionen

- Ein *Schema* ist eine endliche Menge von Namen.

Wirklich sehr vage Definition.

- Eine Instanz eines Schemas besteht aus
  - einem endlichen gelabelten Graphen $(V_a \cup V_c, E)$
    - $V_a$ enthält die atomic objects,
    - $V_c$ enthält die complex objects,
    - Labels der Kanten sind Strings,
  - Abbildung von Namen zu Knoten,
  - Abb. von atomic objects zu atomaren Werten.

  Ausserdem gilt:
  - keine ausgehenden Kanten von atomic objects,
  - jd. Knoten ist von einem mit Namen erreichbar.

Was waere ein Schema fuer das Restaurant-Beispiel?



## Beispiel

- Textuelle Beschreibung: “Finde die Adressen aller Restaurants mit PLZ 92310.”
- Query:
  select Guide.restaurant.address
  where Guide.restaurant.address.zipcode = 93210
- Anmerkung:
  Vorgestellt wird eine spezielle Querysprache, andere Semantik wäre denkbar (Beispiel s.o.).

## Erläuterungen zum Beispiel

- Identische Präfixe von Pfadausdrücken sollen den gleichen Datenpfaden entsprechen,
- Query abstrahiert davon, ob zipcode String oder Integer,
- address ohne zipcode verursacht keine Fehlermeldung, im Gegensatz zu anderen Querysprachen,
- address, zipcode können mengenwertig sein.



## Beispiel 2

- Textuelle Beschreibung: “Finde Namen und Zipcodes aller billigen Restaurants.”
- Probleme - Anfrager kennt/weiss nicht:
  - die genaue Position der zipcode-Objekte,
  - welches Objekt den String “cheap” enthält.
- Query:

```
select  Guide.restaurant.name,
        Guide.restaurant(.address)?.zipcode
where   Guide.restaurant.% grep “cheap”
```

- Erläuterungen:
  - ‘?’ identifiziert optionalen Pfad-Bestandteil,
  - ‘%’ ist Wildcard, ‘grep’ hat übliche Semantik.

# Vergleich von Objekten unterschiedlichen Typs

- Ziel: Man möchte Bedingungen akzeptieren wie Z=1.0 oder Z> “0.9”, egal ob Z mit 1 oder “1” belegt ist.
- Typumwandlung:

| arg2 / arg1 | string | real | int |
|---|---|---|---|
| string | - | string → real | both → real |
| real | | - | int → real |
| int | | | - |

- Beispiel: “4.3” < 5 - beide Werte konvertieren.
- Achtung: Gleichheit ist nicht mehr transitiv: “05” = 5, “5” = 5, aber “05” ≠ “5”



# Vergleich komplexer Objekte unterschiedlichen Typs

- Typumwandlung:

| arg2 / arg1 | value | atomic object | set of objects | complex object |
|---|---|---|---|---|
| value | coerce | de-reference | existential with = | false |
| atomic object | | object = | existential with = | false |
| set of objects | | | set equality | false |
| complex object | | | | object = |

- Beispiel: select X.address
  from Guide.restaurant X
  where X.name = “Chef Chu”

where-Klausel interpret. als ‘where exists Z in X: Z=“Chef Chu”’

Query-Var. wird u.U. fuer Joins etc. gebraucht Was waere, wenn name mengenwertig?

## Pfadausdrücke

- Graph-Struktur der Daten
  erfordert Pfadausdrücke,
- 'Unstrukturiertheit' der Daten
  erfordert Flexibilität
  bei der Formulierung von Pfadausdrücken,
- Pfadausdrücke
  können Bestandteil aller Query-Bestandteile sein,
  d.h. select-, from-und where-Klausel.



## Pfadausdrücke in der from-Klausel

- Pfadausdruck in der from-Klausel:
  from Guide.restaurant.address.zipcode Z,
  Guide.restaurant.name N
- Äquivalente from-Klausel:
  from Guide.restaurant R,
  R.address A,
  A.zipcode Z,
  R.name N

## Pfadausdrücke in der select-Klausel

- Query mit Pfadausdruck in der select-Klausel:
  select Guide.restaurant.address.zipcode
  from Guide.restaurant
- Äquivalente Query:
  select (select Z
  from R.address A,
  A.zipcode Z)
  from Guide.restaurant R



## Pfadausdrücke in der where-Klausel

- Beispielquery:
  select Guide.restaurant.name
  from Guide.restaurant
  where Guide.restaurant.address.zipcode = 93210
  or
  (Guide.restaurant.address.street = “Palm”
  and
  Guide.restaurant.address.city = “Palo Alto”)
- Äquivalente Query:
  select R
  from Guide.restaurant R
  where exists A in R.address:
  ((exists Z in A.zipcode:
  Z = 93210) or
  ((exists S in A.street:
  S = “Palm”) and
  (exists C in A.city:
  C = “Palo Alto”)))

Existenzquantor für address
enthält alle drei Bedingungen,
im Gegensatz zu den anderen Quantoren.

## Auslassen der from-Klausel

- Query ohne from-Klausel:
  select Guide.restaurant.name
  where Guide.restaurant.category = “gourmet”
- Äquivalente Query:
  select Guide.restaurant.name
  from Guide.restaurant
  where Guide.restaurant.category = “gourmet”



## Allgemeine Pfad-Ausdrücke - gpe-Komponenten

- Charakterisierung:
  - Bestandteile allgemeiner Pfad-Ausdrücke,
  - Verallgemeinerung von Labels in einfachen Pfad-Ausdrücken;
- Definition (rekursiv) -
  - wenn l ein Label ist, ist .l eine gpe-Komponente
  - wenn $s_1$, $s_2$ gpe-Komponenten sind, dann sind es die folgenden Ausdrücke auch: $s_1s_2$, $s_1|s_2$, $(s_1)$, $(s_1?)$, $(s_1)+$, $(s_1)^*$
  - wenn X ein String-Objekt ist, dann ist .unquote(X) eine gpe-Komponente.

    Wofür kann das gut sein?

## Übungen

- OEM-Datenbank zum Beispiel vom letzten Vorlesungstermin erstellen.
- Beispielanfragen erstellen:
  - Alle Kategorien von Restaurants in Palo Alto.
  - Alle Paare von Restaurants nahe beieinander, von denen eins billig ist (beide billig sind).