

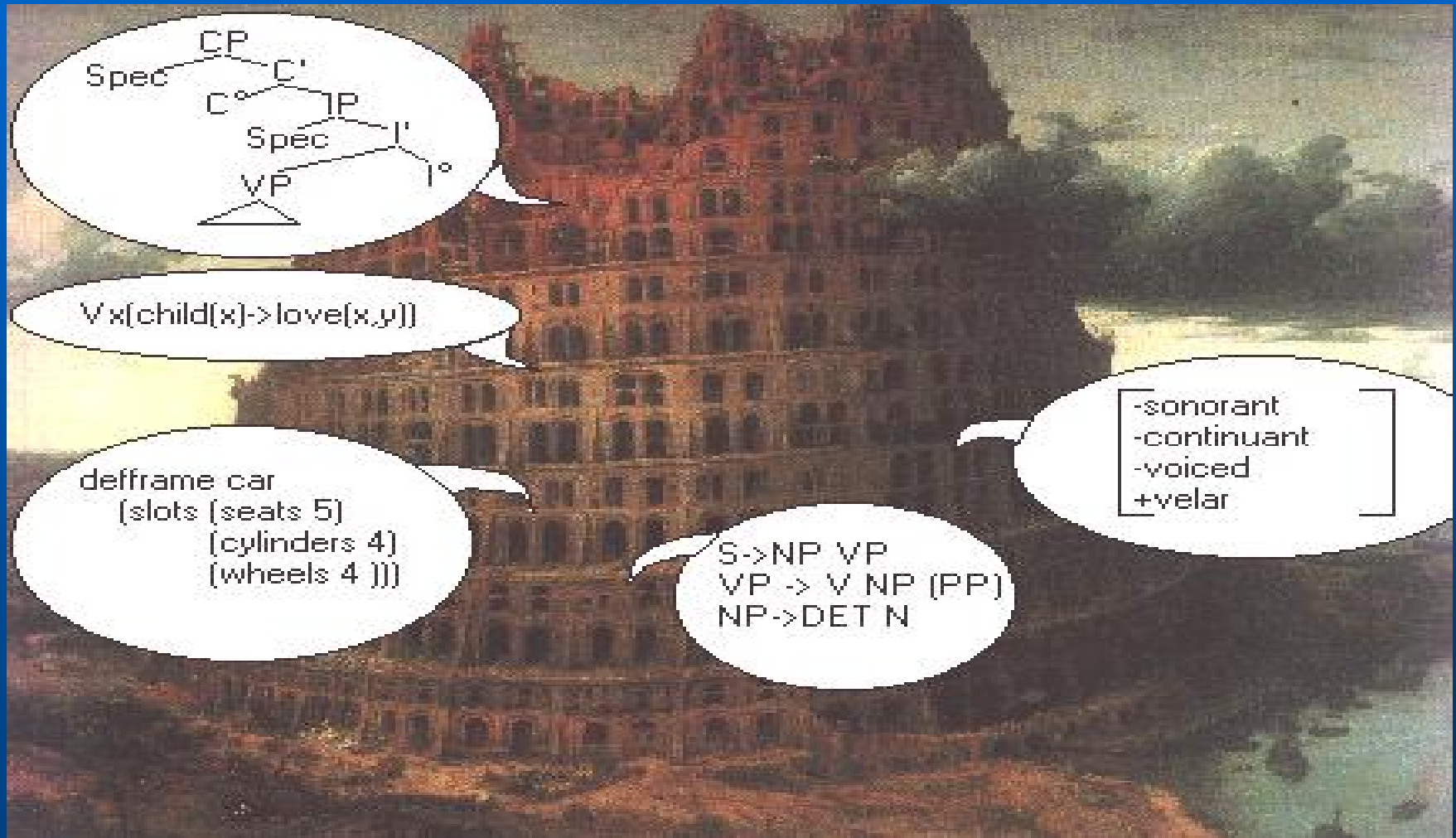
Kopplung Verteilter Datenbanksysteme

Eric Ndengang

21.06.2004

Seminar SS 2004 Universität
Karlsruhe

Babel



Übersicht

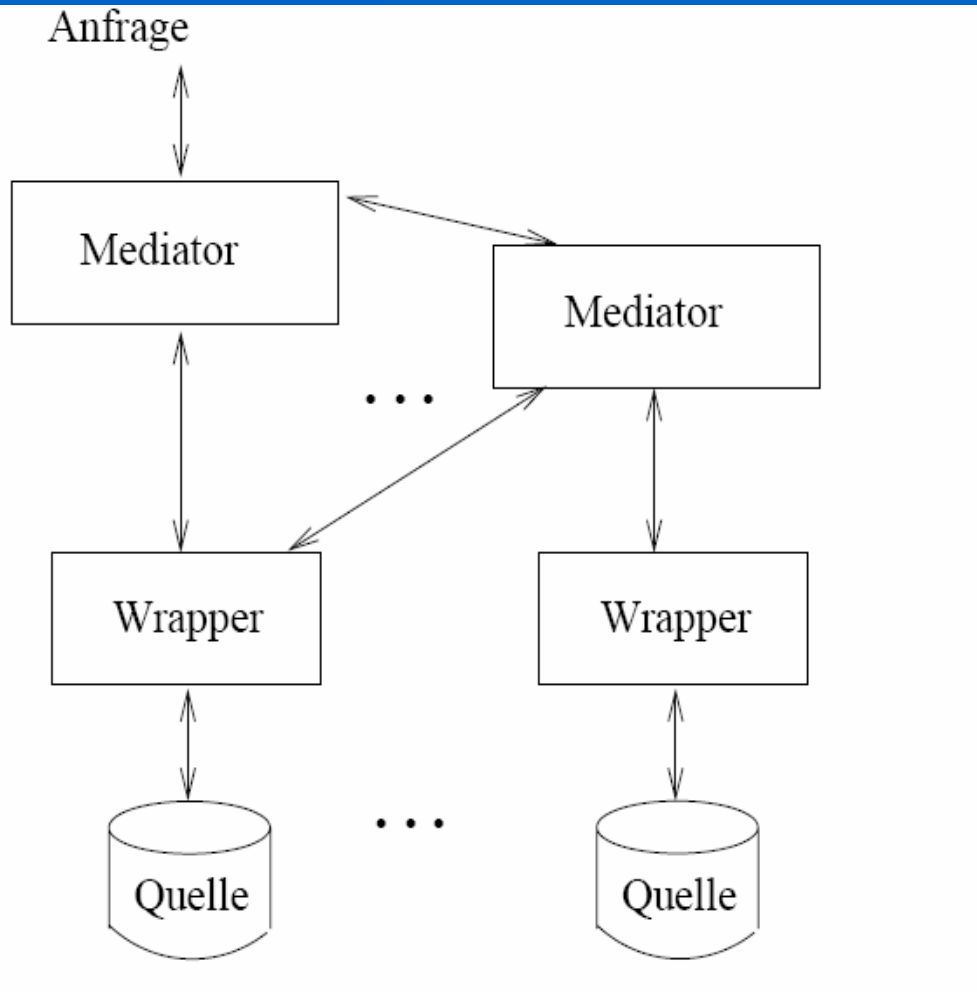
- **Einleitung**
 - **Problematik**
 - **Wrapper / Mediator-basierte Architekturen**
- **Semistrukturierte Datenmodelle**
 - **Das YAT Datenmodell**
 - **YAT System und Problemvorstellung**
 - **Modellierung**
 - **Sprache**
 - **Das OEM Datenmodell**
 - **Tsimmis:Konzept und Architektur**
 - **Das Object-Exchange Model**
 - **Abfragesprachen**
- **Unterschiede und Gemeinsamkeiten der Datenmodelle.**

Einleitung

➤ Problematik

- Die Zahl der Intra/Internet -Anwendungen, die man als heterogene Daten integrieren muss, hat sich schnell erhöht .
- Zentral bei diesen Anwendungen ist die Umwandlung von Daten von einem Format zu anderen .
- viele Datenmodelle und allgemeine Abfragesprachen angeboten, um das Kombinieren der Informationen von vielen unterschiedlichen Quellen zu unterstützen .

Wrapper/Mediator –basierte Architekturen



Eine Möglichkeit, unterschiedliche Informationsquellen zu integrieren, ist die Herstellung eines Mediators, der fähig ist Fragen über Entitäten der realen Welt zu Beantworten.

- ❖ Was ist eigentlich ein Mediator/Wrapper ?
 - Mediator: Ebene zwischen Benutzern und Datenquellen (Middleware).
 - Wrapper (Translatoren) : bieten dem Mediator homogenen Zugriff zu heterogenen Quellen.

2 Architekturen sind bekannt :

- ❖ Bottom-up Ansatz
- ❖ Top-Down Ansatz

Semistrukturierte Datenmodelle

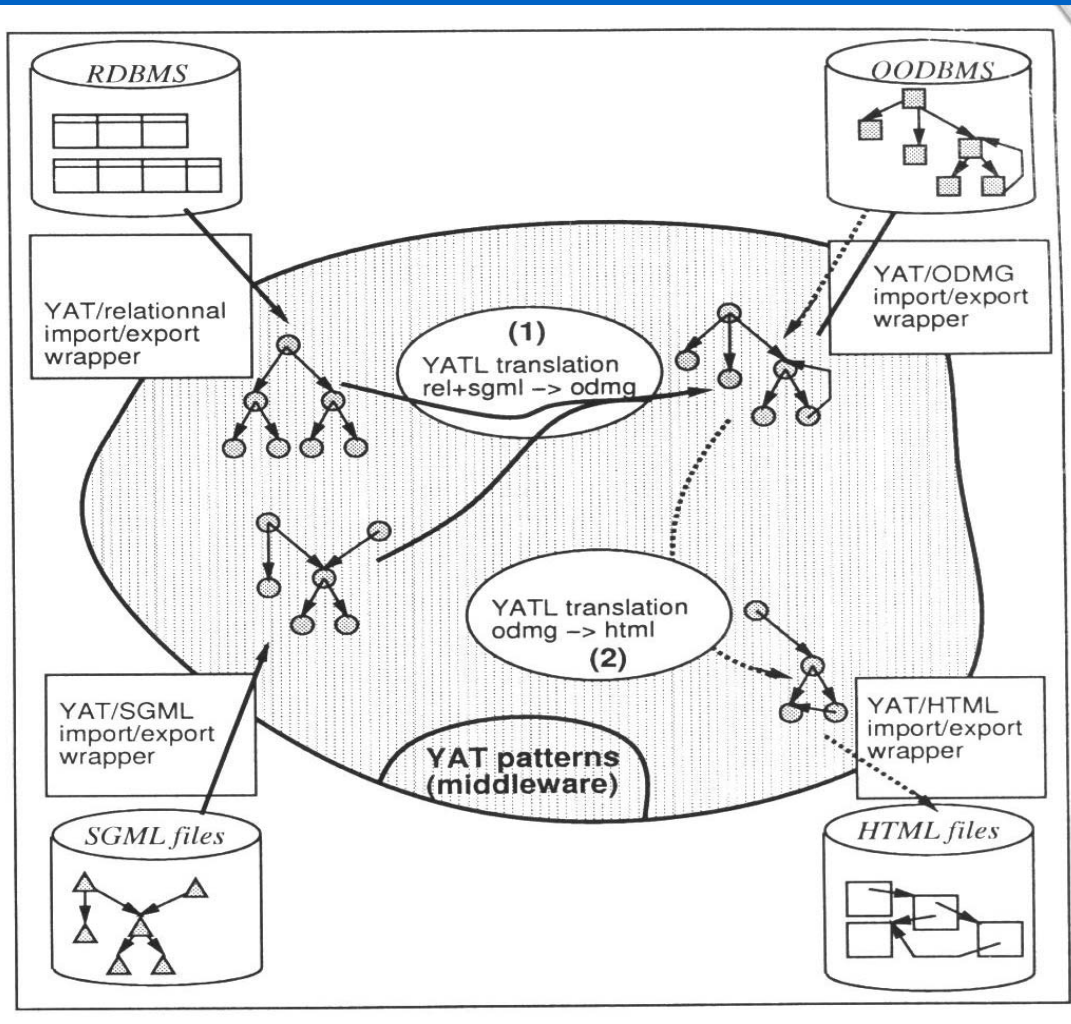
➤ Das YAT Datenmodell

○ Yat System und Problemvorstellung

- Das YAT-System (Yet another tree-based system) wurde am INRIA (Institut National de Recherche en Informatique et Automatique) in Frankreich als Werkzeug zur Spezifikation und Implementierung von Datenkonvertern für heterogene Datenquellen entwickelt.

- Beispiel :

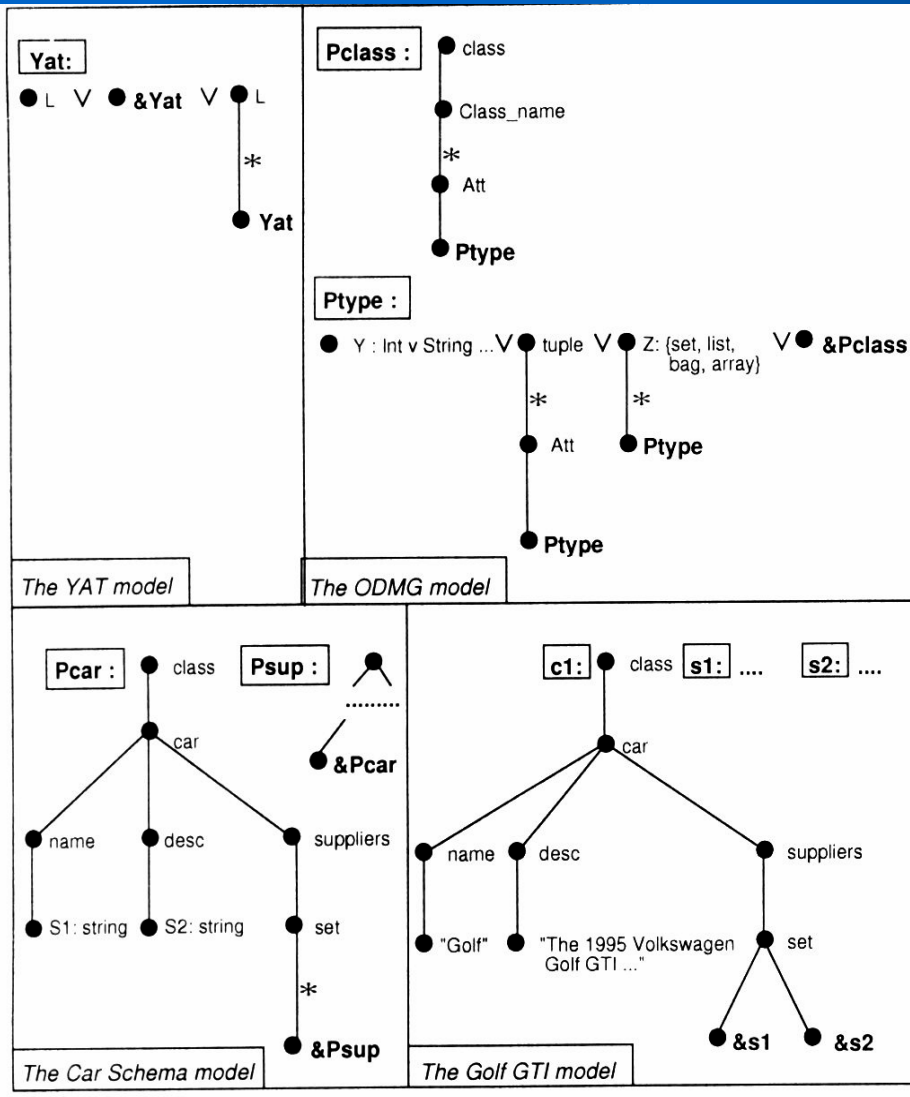
- Eine Autohändlerfirma möchte eine Intranet Anwendung errichten.
- Die Firma speichert Informationen über seine Händler in einem RDBS und die verkauften Autobeschreibungen in SGML Dokumenten.
- Das Ziel ist es, alle Informationen in einer Objektdatenbank zu integrieren und eine HTML Schnittstelle zur Verfügung zu stellen, damit die Angestellten sie auf dem Netz ansehen können.



```

< ! DOCTYPE brochure [
< ! ELEMENT brochure -- (number, title,
                        model, desc, spplrs) >
< ! ELEMENT number -- (#PCADATA) >
< ! ELEMENT title -- (#PCADATA) >
< ! ELEMENT model -- (#PCADATA) >
< ! ELEMENT desc -- (#PCADATA) >
< ! ELEMENT spplrs -- (sup)* >
< ! ELEMENT supplier -- (name,address) >
< ! ELEMENT name -- (#PCADATA) >
< ! ELEMENT address -- (#PCADATA) > ] >
    
```

Modellierung mit YAT /Schema



- Pattern (Muster) werden hier verwendet
- Ein Muster besteht aus einer Menge von Bäumen, auf denen eine Ordnung definiert ist.
- Die Knoten eines Baumes sind mit Variablen oder Konstanten beschriftet .
- Domain einer Daten/Mustervariable.
- Die Instanziierung von Mustern geschieht durch
 - Benennung von Knoten eines Musters mit Namen von anderen Mustern
 - Oder Referenzen darauf
- Zur Definition der Anzahl der erlaubten Ausgangskanten aus einem Knoten, wird den Kanten der *-Operator zugeordnet.

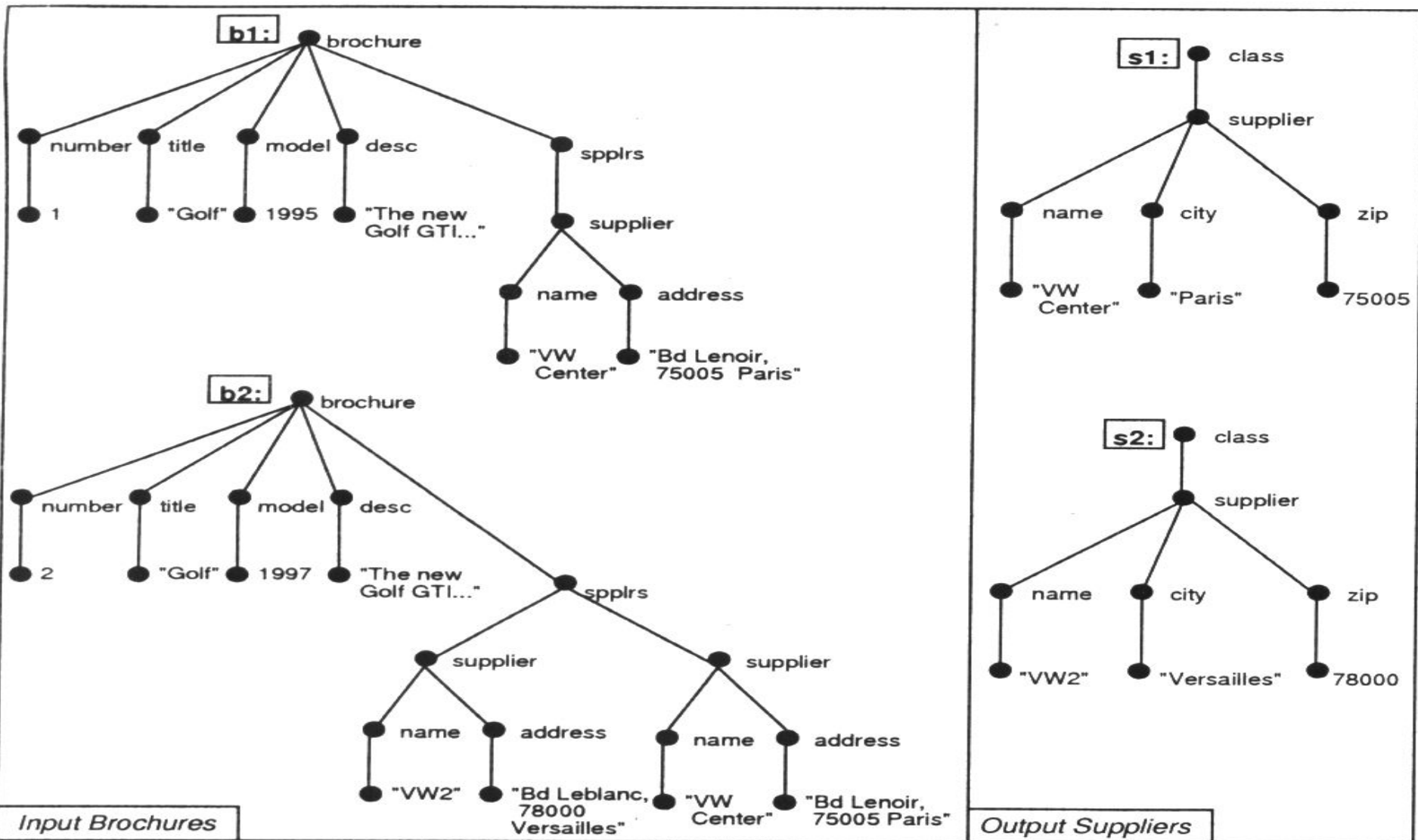
YAT Anfragesprache

- YATL ist eine regelbasierte Sprache bei der, jede Regel aus Kopf und Rumpf besteht.
- Der Kopf der Regel enthält Informationen, wie die gefundenen Daten neu strukturiert werden müssen.
- Der Rumpf enthält Muster und Prädikate.

```
Psup(SN) :  
class → supplier  
  ⟨ → name → SN, ⇐  
    → city → C,  
    → zip → Z⟩  
  
Pbr :  
brochure⟨ → number → Num,  
            → title → T,  
            → model → Year,  
            → desc → D,  
            → suppls *→ supplier  
              ⟨ → name → SN,  
                → address → Add⟩⟩,  
  
Year > 1975,  
C is city(Add),  
Z is zip(Add)
```

Ein YAT Programm das
Für jede SGML Broschüre ein
Lieferantenobjekt erzeugt.

Anwendung von dem Program an 2 Broschüre



YAT Anfragesprache

Ein Yat Programm das für jede SGML Broschüre ein Autoobjekt erzeugt

```
Pcar(Pbr) :
class → car
⟨ → name → T,
  → desc → D,
  → suppliers
    → set  $\{\}$  &Psup(SN)⟩

Pbr :
brochure
⟨ → number → Num,
  → title → T,
  → model → Year,
  → desc → D,
  → suppls  $\overset{*}$  supplier
    ⟨ → name → SN,
      → address → Add ⟩⟩
```

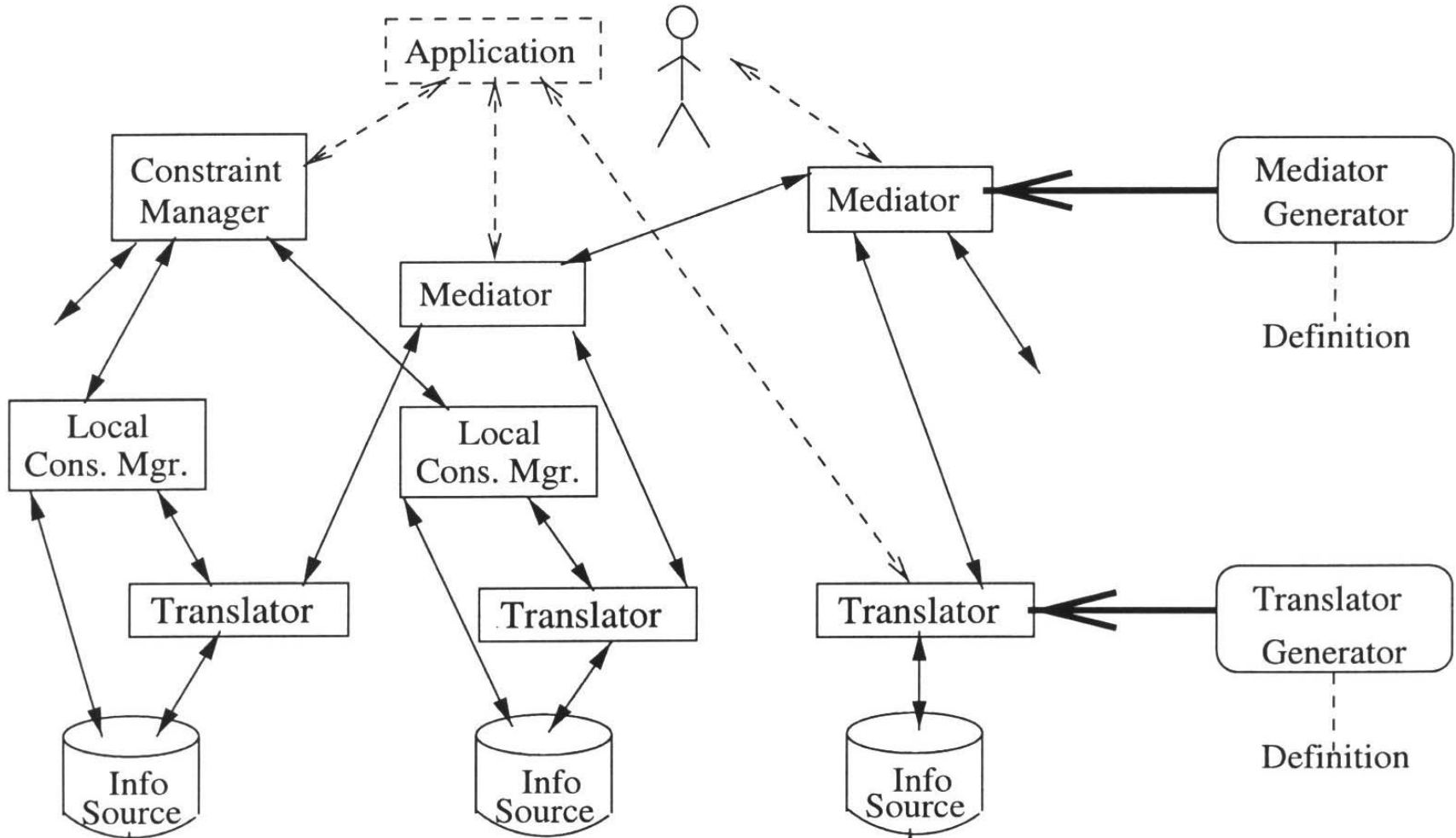
- Jedes Auto referenziert seine Lieferanten
- Die Erzeugung der Verbindungen von Autos zu Lieferanten wird durch die Nutzung von dem parametrisierten Musternamen *Psup(SN)* angefasst .
- Das Symbol $\{\}$ wird für die Erzeugung eines Knoten mit Mehrere Sohne benutzt.

Das OEM Datenmodell

o TSIMMIS : Konzept

- Top Down Ansatz:
- Benutzer stellt Anfrage an einen Mediator
- Mediator gibt Anfrage an die Translatoren weiter
- Translatoren sind Programme, die die Datenobjekte der Quellen 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.

Tsimmis : Architektur



Das Objekt Exchange Modell

- Datenmodell zur Kommunikation zwischen Translatoren und Mediatoren in TSIMMIS
- sehr einfaches „selbstbeschreibendes“ Objektmodell
- kennt nur Objektidentitäten und Schachtelung als Konzept
- Jedes Objekt besteht aus :
 - **OID**: Ein Wert, der das Objekt eindeutig identifiziert
 - **Label**: Beschreibung des Objektinhalts. Damit kann das Objekt identifiziert werden und die Bedeutung des Objekts beschrieben werden.
 - **Typ** : Datentyp des Werts. Verwendung findet hier entweder ein atomarer Typ oder ein Set-Typ
 - **Wert** : Der Wert des Objekts.

Versammlung von OEM Objekten

Buchhandlung	set	
--------------	-----	--

Buch	set	
------	-----	--

Autor	String	Aho
-------	--------	-----

Titel	String	Compilers ...
-------	--------	------------------

OID:

Label	Type	Wert
-------	------	------

Tsimmis: Sprachen

❖ The Mediator Specification Language (MSL)

- Mediatoren werden in MSL (eine regelbasierte Anfragesprache an OEM) programmiert
- MSL-Regeln :

```
Head(Vars) :- body(Vars, databases)
```

- body : Pattern, das mit geeigneten Variablenbindungen erfüllt werden muß
- Head beschreibt die Struktur des OEM-Objekts, das generiert werden soll .
- Objekte der verschiedenen Quellen werden durch `<objekt>@source` identifiziert

Beispiel

- Anfrage: „Finde alle Bücher deren Autor Aho ist“

<Buchtitel X> :-

<Buchhandlung {<Buch {<Titel X> <Autor „Aho“>}> }>@ S1

- S1 könnte hier entweder einen Mediator oder einen Translator sein .
- Der Head der Frage zeigt an, daß jeder Wert, den X bindet, im Resultat als Wert eines Objekts eingeschlossen ist, der Buchtitel beschriftet wird .
- Technischerweise werden diese Objekte subobjekte eines Objekts mit Label **Antwort**, die durch die Frage produziert wird.

The Wrapper Specification Language(WSL)

- Wrapper werden in WSL programmiert
- Jede WSL-Regel besteht aus einem
 - „WSL-template“, und
 - einer Aktion in der Anfragesprache der zugrunde liegenden Datenbank
- Bekommt der Wrapper eine MSL-Anfrage, die auf das WSL-template paßt, wird die Aktion mit der entsprechenden Variablenbindung ausgeführt .
- Beispiel: wir möchten einen Wrapper für eine Quelle errichten, der ein bibliographisches Suchsystem ist.

```
<books X> :-
```

```
  <library { X:<book {<title X> <author $AU>}}>@ S1
```

```
// sprintf( lookup-query, „find author %s“, $AU ) //
```

Semistrukturierte Datenmodelle

- Der erzeugte Wrapper überprüft eine Frage und vergleicht sie mit diesem und anderen Mustern, die in der Wrapper Spezifikation File gegeben werden.
- So, die Frage

```
<books B> :-
```

```
<library { B:<book {<title X> <author „Aho“>}}>@S1
```

würde eine „native“ Frage zur Quelle S1 erzeugen, die Bücher anfordert, die von Aho geschrieben wurden.

- Die in der Aktion oben beschriebene C Funktion sprintf dient dazu, dem variablen lookup-query den String „Find author Aho“ zuzuweisen .
- Schließlich wird den String dann zur quelle S1 geführt .

The LOREL query Language

- LOREL(Lightweight Objekt Repository Language) , basierte OQL Abfragesprache für das OEM Modell.
- Die semistrukturierten Teilen der Abfragen mit Lorel sind Pfadorientiert
- Es ist auch eine Abfragesprache für den LORE.
- Explizite Erklärungen in Quass, Rajaraman, Sagiv, Ullman und in Widom.

Frage :“Find the books of which Aho is an Author“

```
Select library.book.title Where library.book.author = „Aho“
```

Gemeinsamkeiten und Unterschiede zwischen den Datenmodellen

- OEM wurde für den Datenaustausch entwickelt und dienen zur Kommunikation zwischen Wrappern und Mediatoren in Tsimmis
- Damit konnten unstrukturierte Datenmengen einfach in einer gesamten Sicht der Daten integriert werden.
- Der Einsatz von YAT wurde in Bezug auf den Datenaustausch zwischen verschiedensten Datenbanken ,die beliebige,festgelegte Datenmodelle einsetzen ,optimiert.
- YAT ist ein System zum Erstellen von Konvertierern
- Tsimmis ,ein System zur Integration von Informationen von verschiedenen Datenquellen .

Fazit

- Das Wrapper /Mediator Konzept
- Das YAT-Modell , eine Sammlung von Mustern , die die ankommenden Daten und die Ergebnisse der Konvertierung beschreiben.
- YATL, die Sprache für die Spezifikation der Datenumsetzung.
- Das Tsimmis Konzept und Architektur.
- Das OEM Datenmodell , ein Objektorientiertes Modell, das Objekt Label verwendet um Attribute - und Informationsklassen von Objekten zu darstellen
- Die Sprache LOREL für OEM Objekten .
- MSL , eine für Mediatoren regelbasierte Abfragesprache , die an OEM gezielt ist.
- WSL, eine zu MSL erweiterte Sprache , die die Beschreibung der Datenquelleninhalte erlaubt und Anfragenfähigkeiten erleichtert .

Question

