

# Semantische Integration



# Schemaintegration: Motivation



- Neben der Anfrageübersetzung ist die Herstellung eines einheitlichen Informationsmodells/Schemas das wichtigste Problem
  
- Zwei Schritte
  - ▶ Definition des integrierten Schemas
  - ▶ Definition der Abbildungsregeln («mappings»)
    - von den Quellen in das integrierte Schema
    - im virtuellen Fall bzw. bei bidirektionaler Kommunikation: vom integrierten Schema in die Quellschemata

# Arten von Integrationskonflikten

## Datenmodell

Unterschiedliche Ausdrucksmächtigkeit von Datenmodellen

Relational, Objektorientiert, XML, Semistrukturiert

## Logische Ebene: Schema

Freiheitsgrade in der Modellierung der realen Welt  
klassische »Schemaintegration«

## Daten Instanzen

Zusammenführen von Instanzen  
(Objektidentität/Duplikaterkennung)

Widersprüchlichkeit, Inkonsistenzen, Subjektivität

# Ebenen der semantischen Integration

## ■ Schemaintegration

(bzw. Integration auf Schemaebene)

- ▶ Zusammenführen und semantische Vereinheitlichung allein auf der Basis von Schemainformationen
  - Typen, Klassen, Attribute, Vererbungsrelation etc.
  - Abbildung wird auf alle Instanzen angewendet

## ■ Instanzintegration (bzw. Integration auf Instanzebene)

- ▶ Zusammenführen von Informationen über einzelne Instanzen (oder Objekte)
  - auf der Basis von Werten
  - typisch: Ähnlichkeitsmaße, paarweise Vergleich, Objektidentität, wertabhängige Abbildungsregeln
  - Schemainformation reicht als Information nicht aus!

# Arten von Integrationskonflikten

## Datenmodell

Unterschiedliche Ausdrucksmächtigkeit von Datenmodellen

Relational, Objektorientiert, XML, Semistrukturiert

## Schema

Freiheitsgrade in der Modellierung der realen Welt  
klassische »Schemaintegration«

## Daten Instanzen

Zusammenführen von Instanzen  
(Objektidentität/Duplikaterkennung)

Widersprüchlichkeit, Inkonsistenzen, Subjektivität

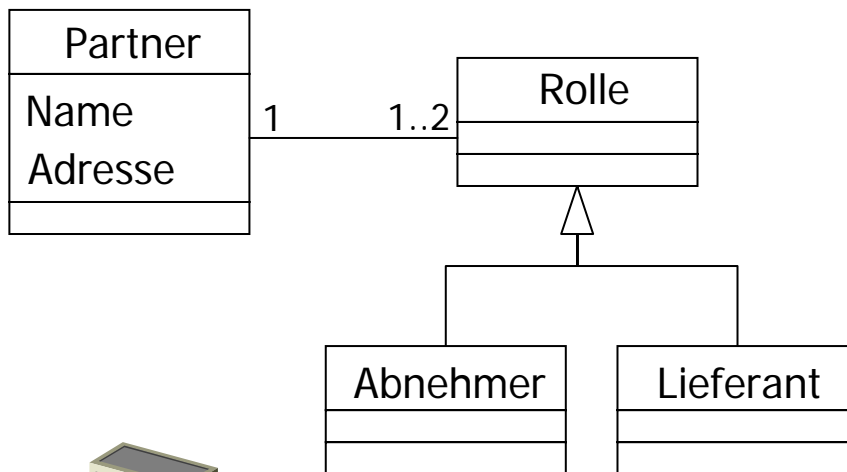
# Datenmodellkonflikte (1)

- Datenmodellkonflikte kommen durch unterschiedliche Ausdrucksmächtigkeit von Datenmodellen zustande
  - ▶ unterschiedliche Strukturbeschreibungen
  - ▶ unterschiedliche Integritätsbedingungen (Kardinalitäten, referentielle Integrität etc.)
  
- In der Praxis wichtige Datenmodelle sind
  - ▶ relationales Modell
  - ▶ objektorientiertes Modell
  - ▶ objektrelationales Modell
  - ▶ XML-Datenmodell(e)

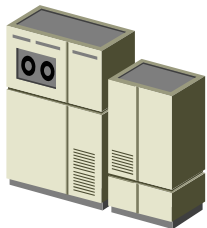
# Datenmodellkonflikte (2)

```
<!ELEMENT hersteller (name, adresse, ware*) >  
<!ELEMENT name (#PCDATA) >  
<!ELEMENT adresse (strasse, stadt, plz) >  
<!ELEMENT ware (nummer, preis) >
```

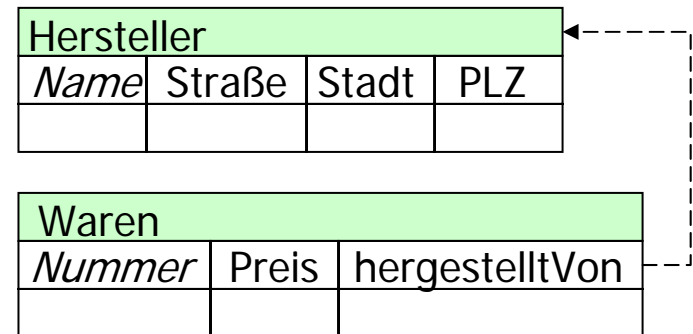
**XML**



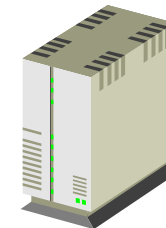
**Objektorientiert**



**System 1**



**Relational**



**System 2**

# Datenmodellkonflikte (3)

- Datenmodellkonflikte treten sehr häufig auf
  - ▶ Mehrschichtenarchitekturen:  
RDBMS ↔ EntityBeans ↔ Präsentation HTML
  - ▶ in virtuellen Integrationsansätzen:  
meistens die Aufgabe von Wrappern
- Für »klassische Anwendungen« sehr viele Lösungsansätze
  - ▶ OR-Mapping-Tools
    - JDO (Java Data Objects)
    - Hibernate
  - ▶ XML-Export aus Datenbanken
    - siehe Vorlesung „XML und Datenbanken“
  - ▶ JAXB (Java XML-Binding)

# Arten von Integrationskonflikten

## Datenmodell

Unterschiedliche Ausdrucksmächtigkeit von Datenmodellen

Relational, Objektorientiert, XML, Semistrukturiert

## Schema

Freiheitsgrade in der Modellierung der realen Welt  
klassische »Schemaintegration«

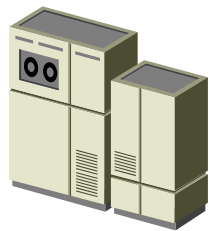
## Daten Instanzen

Zusammenführen von Instanzen  
(Objektidentität/Duplikaterkennung)

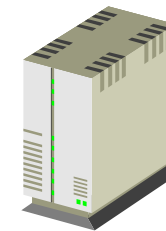
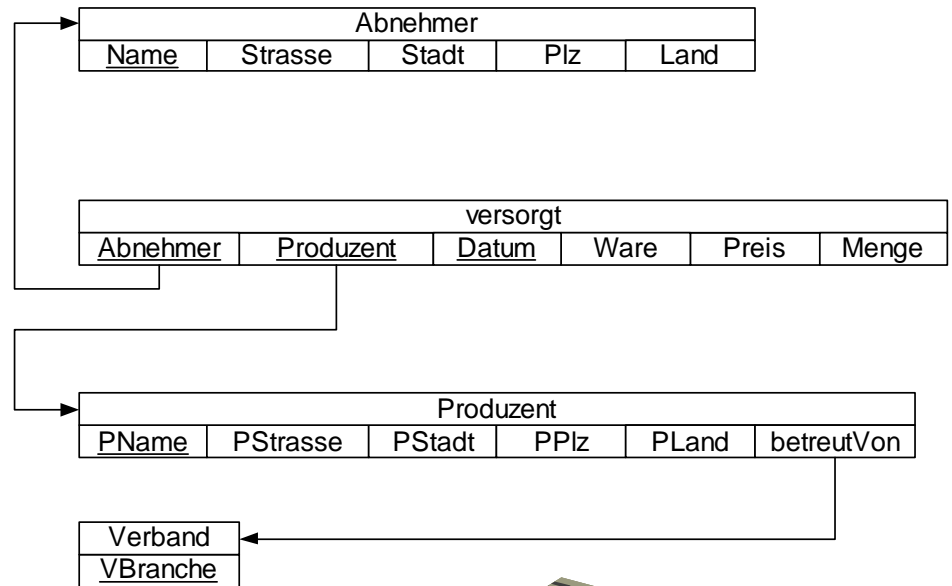
Widersprüchlichkeit, Inkonsistenzen, Subjektivität

- kommen durch Freiheitsgrade in der Modellierung der Realwelt durch Schemakonstrukte zustande
  - ▶ Beschreibung
    - unterschiedliche Benennung
    - unterschiedliche Wertdarstellung
  - ▶ Struktur
    - unterschiedliche Schemakonstrukte
  - ▶ Semantik
    - implizites Wissen
    - ...

# Schemakonflikte: Beispielschemata



System 1

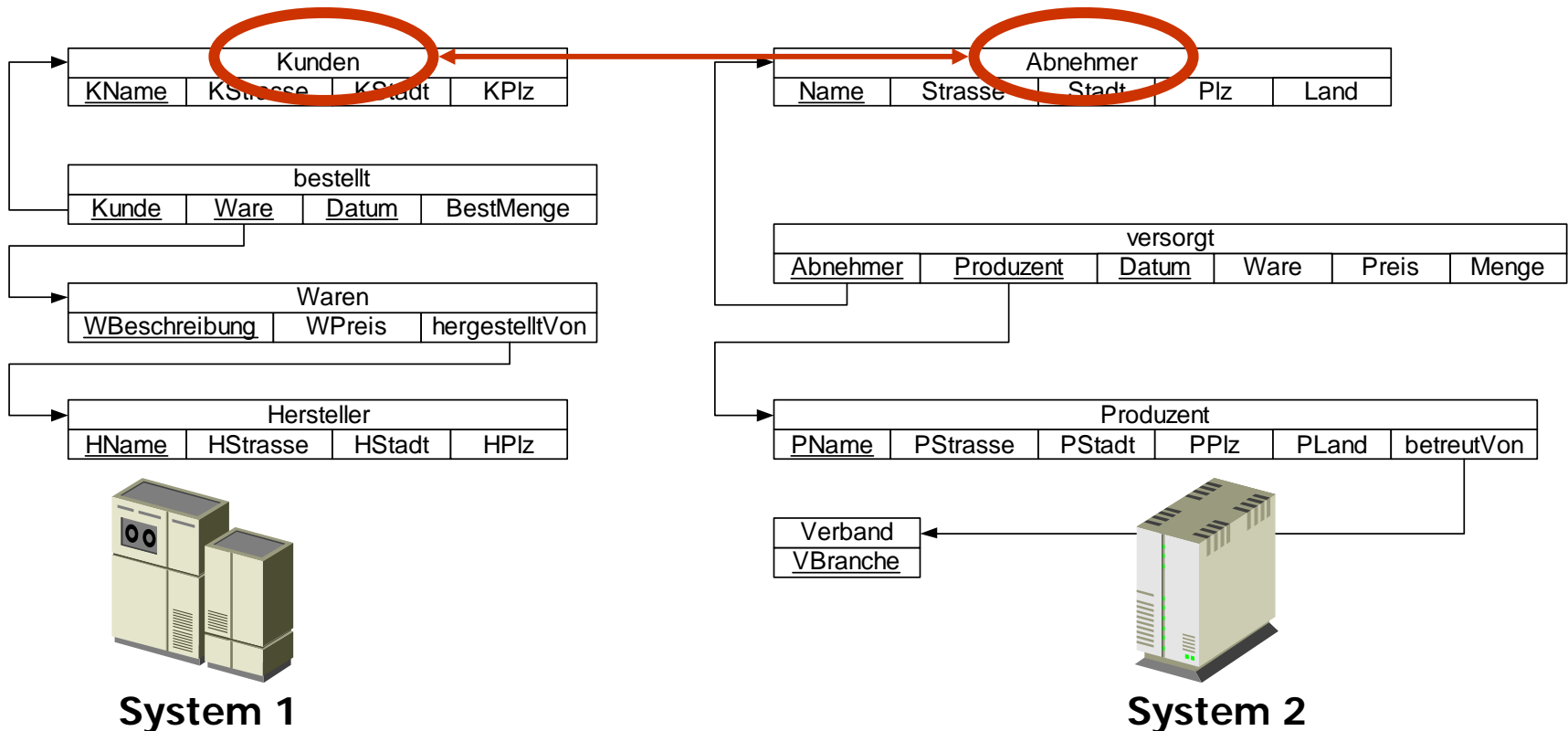


System 2

# Schemakonflikte: Beschreibungskonflikte (1)

## ■ Tabellennamenkonflikte

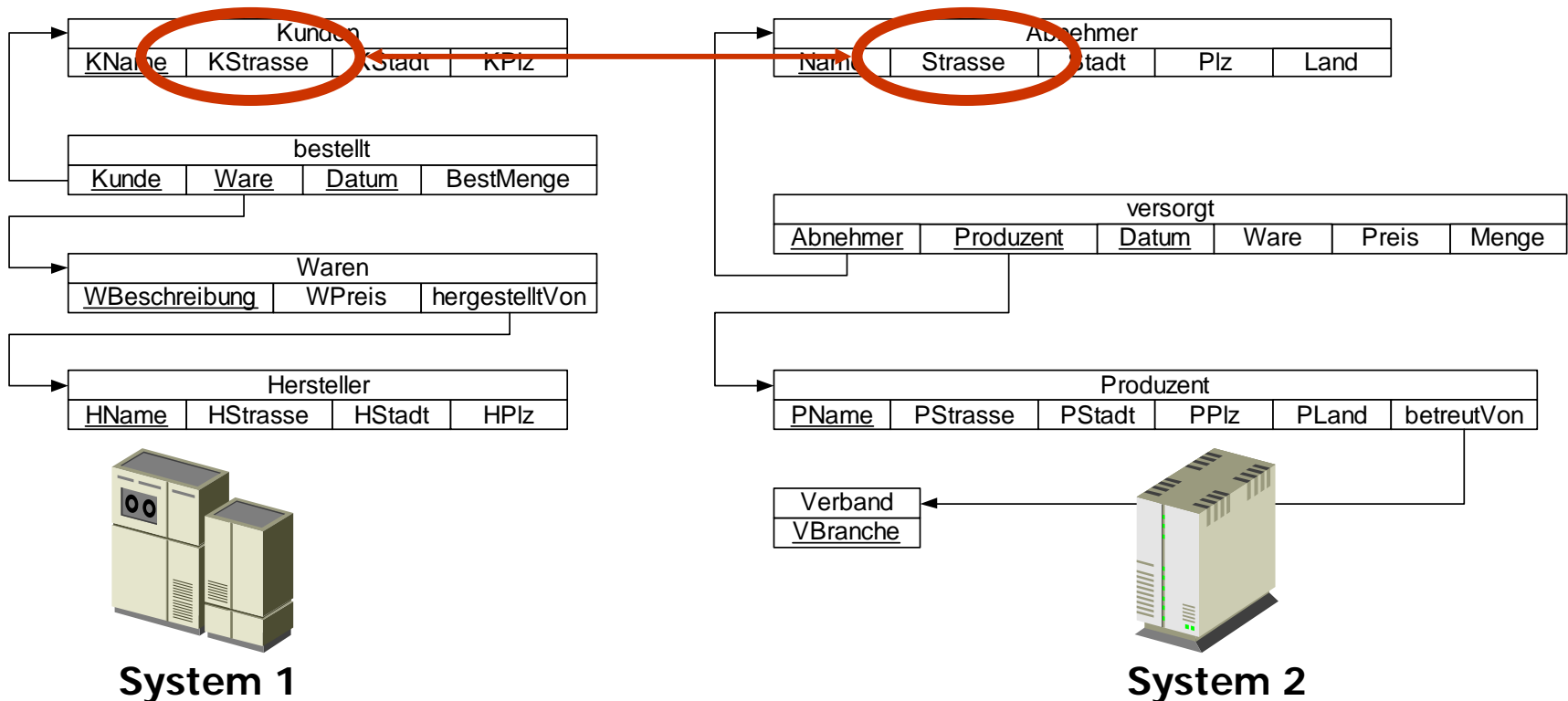
- Verschiedene Namen für gleiche Tabellen
- Gleiche Namen für verschiedene Tabellen



# Schemakonflikte: Beschreibungskonflikte (2)

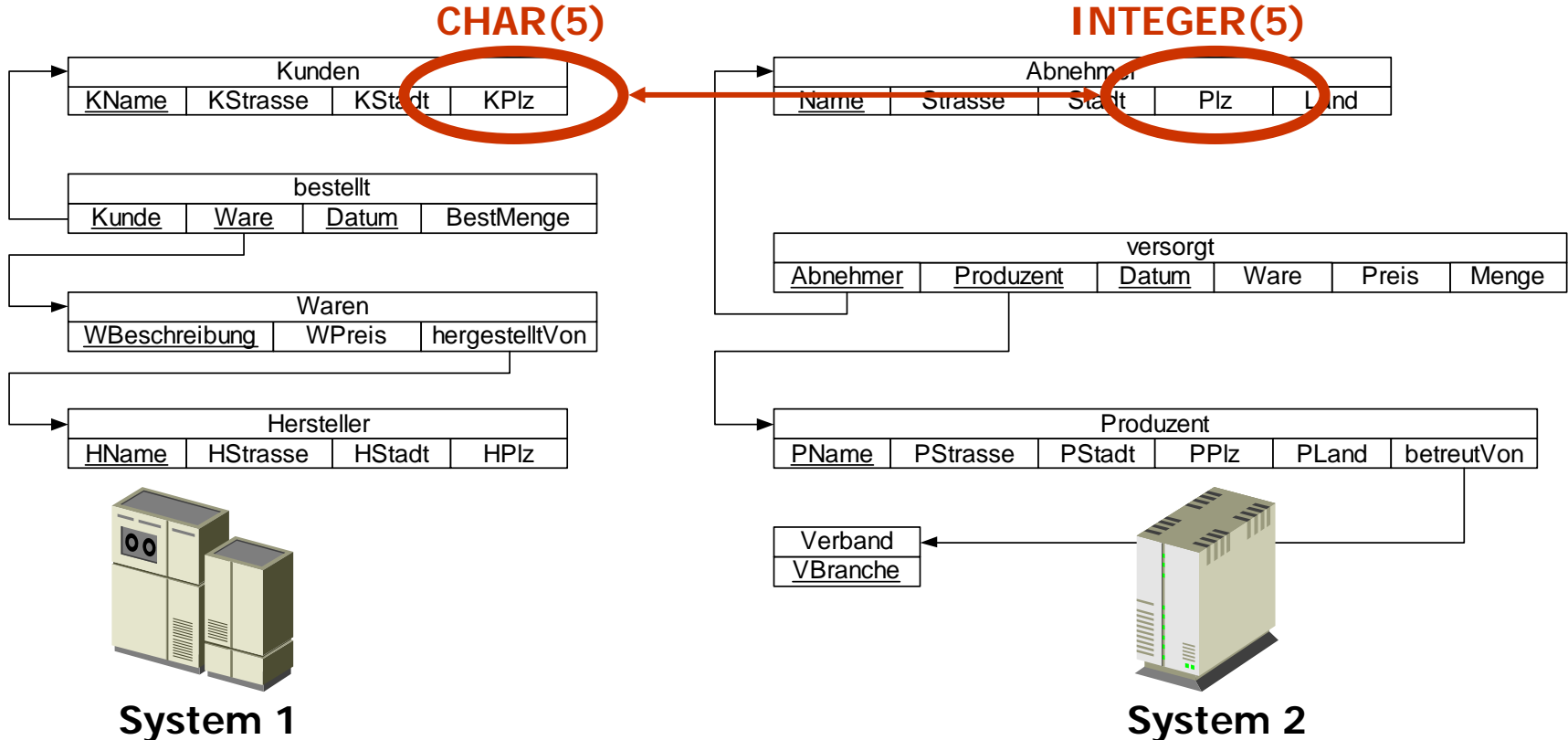
## ■ Attributnamenskonflikte

- Verschiedene Namen für gleiche Attribute
- Gleiche Namen für verschiedene Attribute



# Schemakonflikte: Beschreibungskonflikte (3)

- Integritätsbedingungskonflikte
  - ▶ Datentypkonflikte

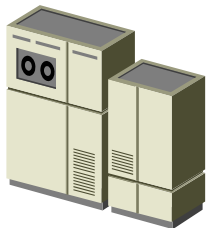


# Schemakonflikte: Beschreibungskonflikte (5)

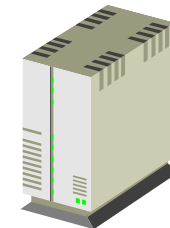


## ■ Unterschiedliche Granularität

- ▶ Beispiel: „Hersteller.Groesse“:
  - Eins aus { groß, mittel, klein }
  - Jahresumsatz in Mio. €
- ▶ Beispiel: „Kleidung.Groesse“:
  - { 24, 25, 26, ..., 48, 50, 52, ... }
  - { XS, S, M, L, XL, XXL }



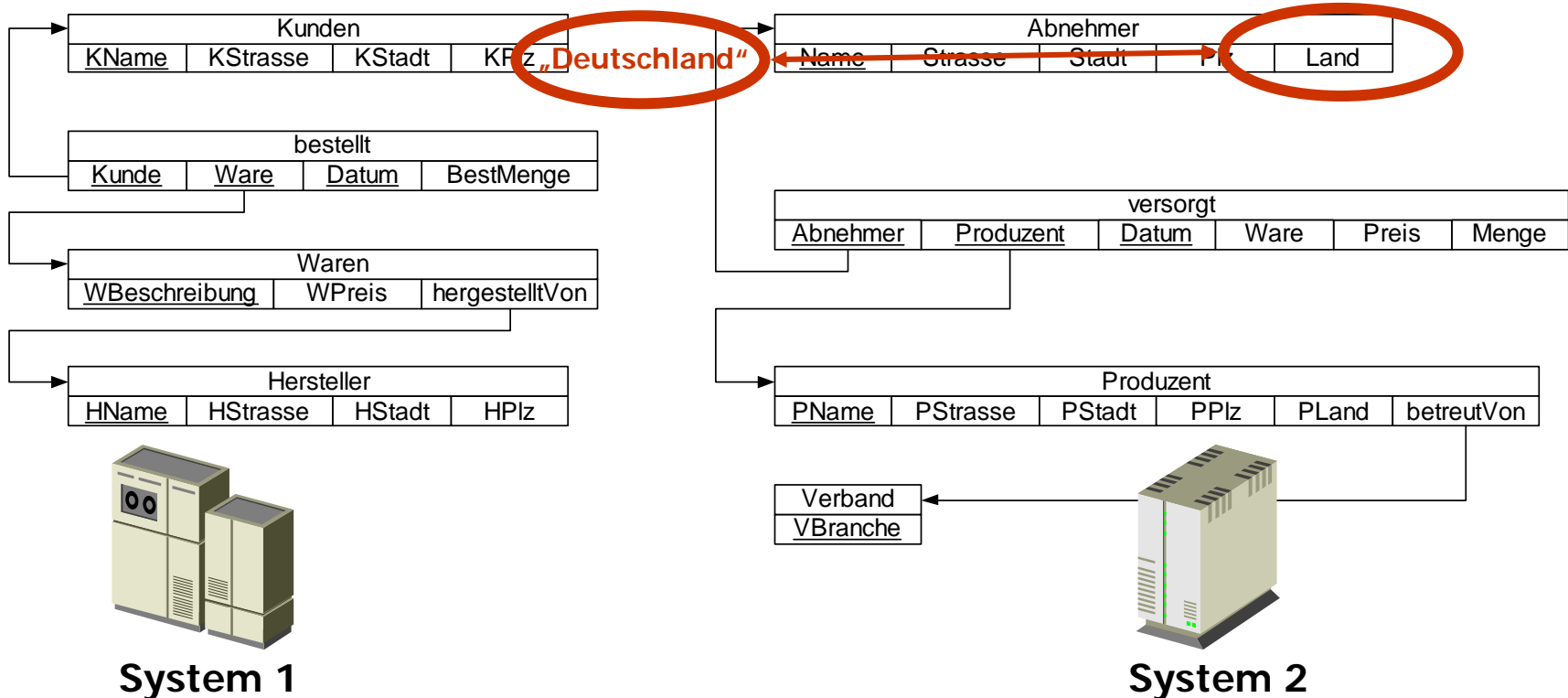
**System 1**



**System 2**

# Schemakonflikte: Strukturkonflikte (1)

- Tabellenstrukturkonflikte
  - ▶ Fehlende, aber implizite Attribute



# Schemakonflikte: Strukturkonflikte (2)

## ■ Meta-Konflikte



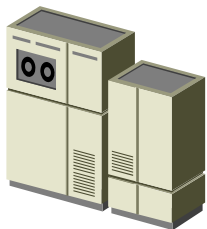
„Status“ eines Kunden ist  
ein Wert in der Datenbank

Status eines Kunden ist  
im Schema kodiert

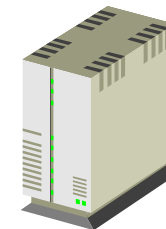
Kunden		
<u>Name</u>	Adresse	Status
Carl	Mannheim	privat
Katja	Karlsruhe	geschaeftlich

Privatkunden	
<u>Name</u>	Adresse
Paul	76131 Karlsruhe, ...

Geschäftskunden	
<u>Name</u>	Adresse
Inge	69115 Heidelberg, ...



**System 1**



**System 2**

# Arten von Integrationskonflikten

## Datenmodell

Unterschiedliche Ausdrucksmächtigkeit von Datenmodellen

Relational, Objektorientiert, XML, Semistrukturiert

## Schema

Freiheitsgrade in der Modellierung der realen Welt  
klassische »Schemaintegration«

## Daten Instanzen

Zusammenführen von Instanzen  
(Objektidentität/Duplikaterkennung)

Widersprüchlichkeit, Inkonsistenzen, Subjektivität

# Konflikte auf Instanzebene



- Konflikte lassen sich nicht durch die Betrachtung der Struktur lösen
  - ▶ Erkennen von Objektidentität
  - ▶ Behandlung von Widersprüchen
  - ▶ implizites Zusatzwissen

# Konflikte auf Instanzebene (2)

## ■ Feststellung der Objektidentität

- ▶ „Object Fusion“
- ▶ Duplikaterkennung

Kunden			
<u>KName</u>	<u>KStrasse</u>	<u>KStadt</u>	<u>KPlz</u>
Blau AG	D...str. 3	Köln	50033
...	...	...	...

Identisch?

Abnehmer				
<u>Name</u>	<u>Strasse</u>	<u>Stadt</u>	<u>Plz</u>	<u>Land</u>
BLAU				
...	...	...	...	...

bestellt			
<u>Kunde</u>	<u>Ware</u>	<u>Datum</u>	<u>BestMenge</u>
Blau AG	Zement	20.05.01	100
...	...	...	...

versorgt					
<u>Abnehmer</u>	<u>Produzent</u>	<u>Datum</u>	<u>Ware</u>	<u>Preis</u>	<u>Menge</u>

Waren		
<u>WBeschreibung</u>	<u>WPreis</u>	<u>hergestelltVon</u>

Produzent					
<u>PName</u>	<u>PStrasse</u>	<u>PStadt</u>	<u>PPlz</u>	<u>PLand</u>	<u>betreutVon</u>

Hersteller			
<u>HName</u>	<u>HStrasse</u>	<u>HStadt</u>	<u>HPlz</u>

Verband
<u>VBranche</u>

# Konflikte auf Instanzebene (3)

## ■ Falsche Daten

- Nicht korrekte Einträge
- Nicht identisch? Identisch aber eins fehlerhaft?

Kunden			
KName	KStrasse	KStadt	KPlz
Blau AG	Domstr. 3	Köln	50033
...	...	...	...

Abnehmer				
Name	Strasse	Stadt	Plz	Land
BLAU			50044	
...	...	...	...	...

bestellt			
Kunde	Ware	Datum	BestMenge
Blau AG	Zement	20.05.01	100
...	...	...	...

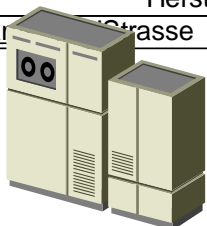
versorgt					
Abnehmer	Produzent	Datum	Ware	Preis	Menge

Waren		
WBeschreibung	WPreis	hergestelltVon

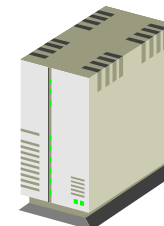
Produzent					
PName	PStrasse	PStadt	PPlz	PLand	betreutVon

Hersteller			
HNar	HStrasse	HStadt	HPlz

Verband
VBranche



**System 1**



**System 2**

# Konflikte auf Instanzebene (4)

## ■ Falsche Daten

### ▶ Veraltete Daten

Kunden			
<u>KName</u>	<u>KStrasse</u>	<u>KStadt</u>	<u>KPlz</u>
Blau AG	Domstr. 3	Köln	50033
...	...	...	...

Abnehmer				
<u>Name</u>	<u>Strasse</u>	<u>Stadt</u>	<u>Plz</u>	<u>Land</u>
BLAU	Neustr. 8, Heidelberg	69115	Deutschland	
...	...	...	...	...

bestellt			
<u>Kunde</u>	<u>Ware</u>	<u>Datum</u>	<u>BestMenge</u>
Blau AG	Zement	20.05.01	100
...	...	...	...

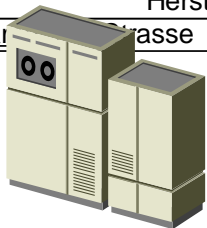
versorgt					
<u>Abnehmer</u>	<u>Produzent</u>	<u>Datum</u>	<u>Ware</u>	<u>Preis</u>	<u>Menge</u>

Waren		
<u>WBeschreibung</u>	<u>WPreis</u>	<u>hergestelltVon</u>

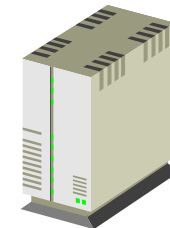
Produzent					
<u>PName</u>	<u>PStrasse</u>	<u>PStadt</u>	<u>PPlz</u>	<u>PLand</u>	<u>betreutVon</u>

Hersteller			
<u>HNar</u>	<u>HStrasse</u>	<u>HStadt</u>	<u>HPlz</u>

Verband	
<u>VBranch</u>	



**System 1**



**System 2**

# Konflikte auf Instanzebene (4)

- „Implizites semantisches Wissen“
- Zwei Schemata mit: Hersteller.Sitz
  - ▶ Schema A verwendet „Stadt“
  - ▶ Schema B verwendet „Land“
- Gesucht: „Alle französischen Hersteller“

**Man muss wissen, welche Städte in Frankreich liegen!**

**Fazit**



# Zusammenfassung: Informationsintegration

- Unterschiedliche methodische Herangehensweisen an das Problem der Informationsintegration
  - ▶ materialisiert vs. virtuell
  
- Zentrales Problem: Semantische Integration
  - ▶ materialisierter Ansatz (Globales Schema)
  - ▶ virtueller Ansatz (Domänenmodell bzw. Globales Schema)
  - ▶ Datenaustauschvokabular (vgl. nächste Vorlesung)

# Grenzen des Ansatzes auf Informationsebene

- Integration auf Informationsebene ist hervorragend geeignet für Nur-Lese-Zugriffe nach dem Anfrage-Ergebnis-Paradigma
  - ▶ z.B. Integration von Katalogen
- Probleme jedoch bei Schreibzugriffen
  - ▶ z.B. Übertragung von Bestelldaten
  - ▶ Aktualisierungsoperationen auf heterogenen Systemen sehr schwierig auf Datenmodell-/Schemaebene zu lösen
  - ▶ Oft auch Geschäftsprozesse von entscheidender Bedeutung
- Hier braucht man (ergänzend) einen dienstorientierten Ansatz

# Datenaustausch



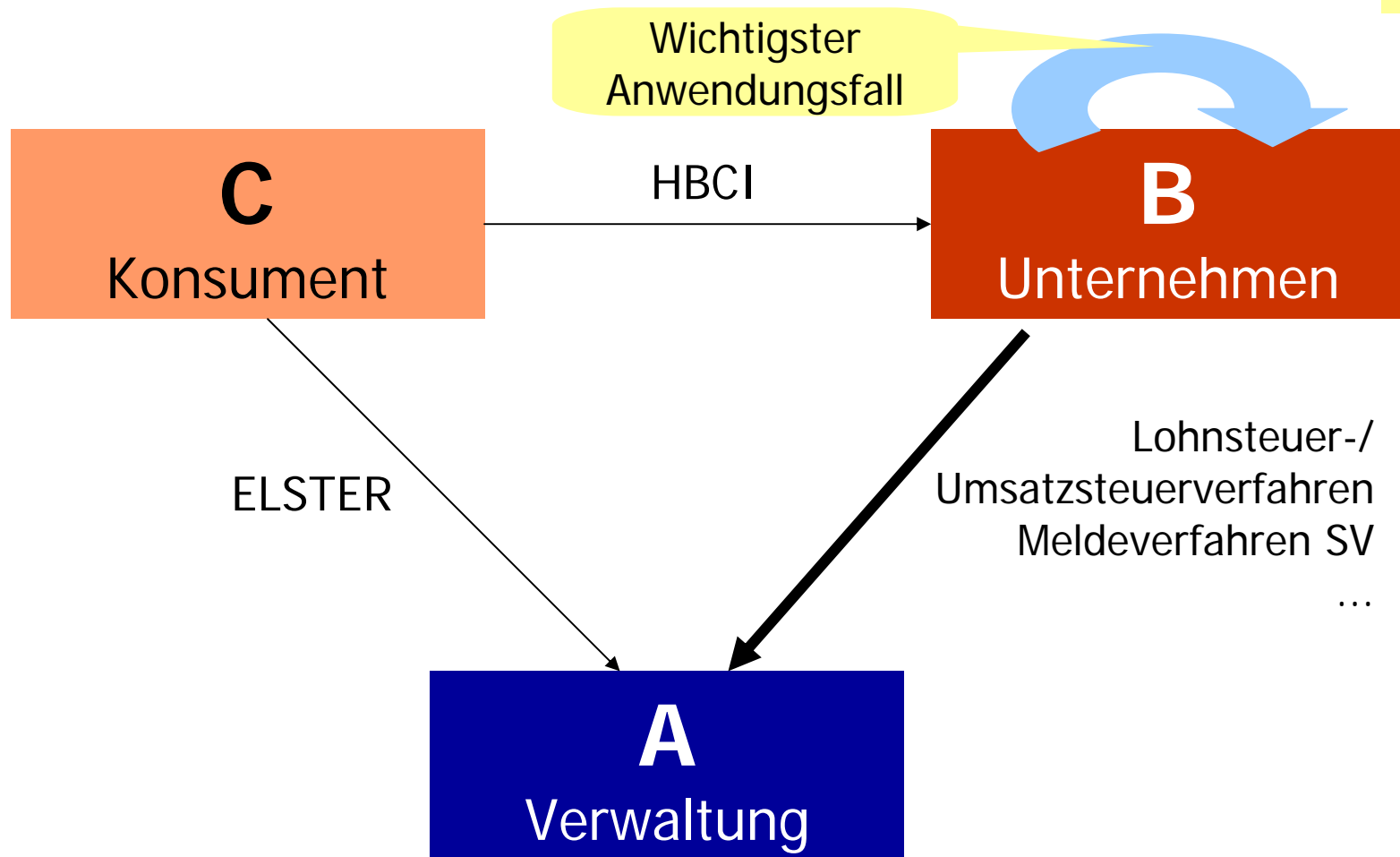
Andreas Schmidt



- Datenaustausch
  - ▶ Anforderungen und Probleme
  - ▶ Traditionelle EDI-Lösungen
  - ▶ XML-basierter Datenaustausch
  
- Validierung als wichtigstes (technisches) Teilproblem
  - ▶ XML Schema
  - ▶ Andere Validierungstechniken für XML

- bisher: Informationsintegration aus Konsumentensicht (B2C)
  - ▶ für eine spezielle (Portal-) Anwendungen werden Informationen aus Informationsquellen extrahiert
  - ▶ Asymmetrie: es existiert eine **zentrale** Stelle, von der aus alles koordiniert wird
    - Ausgangs- und Endpunkt aller Interaktionen
  - ▶ Primäres Interaktionsmuster: Anfrage-Ergebnis
- jetzt: allgemeinerer Ansatz
  - ▶ Symmetrie: **keine vorherrschende Interaktionsrichtung**
  - ▶ weiterhin: Autonomie der technischen Systeme
  - ▶ aber teilweiser Verzicht auf organisatorische Autonomie

# Beispiele nach Akteuren



# **Datenaustausch: Grundlagen**



- Grundmodell: Kooperation ist der Austausch
  - ▶ von Daten
  - ▶ zwischen einer Gruppe von Partnern
  - ▶ für einen bestimmten Kooperationszweck
- Austausch ist dabei geregelt
  - ▶ entweder zwischen einzelnen Partnern,
  - ▶ branchenweit
  - ▶ oder gar als internationaler Standard

= > **Spezifikation**

# Spezifikation: Bestandteile

- Abgrenzung des **Einsatzbereiches**
  - ▶ Was ist Hintergrund und Zweck?
  - ▶ evtl. „Use Cases“
- **Informationsmodell**
  - ▶ Festlegung der Datenstrukturen
  - ▶ Welche Entitäten?
  - ▶ Wie werden die Entitäten beschrieben?
    - insbesondere: Identifikation von Entitäten
- **Prozessmodell**
- **Kommunikationsprotokoll**

# Spezifikation: Informationsmodell

- **Informationsmodell** ist typischerweise Kern der Spezifikation
  - ▶ entspricht der Entwicklung des Domänenmodells bei der virtuellen Integration
  - ▶ sehr aufwendig und langwierig
  - ▶ letztlich Definition eines gemeinsamen Vokabulars
    - Ontologien

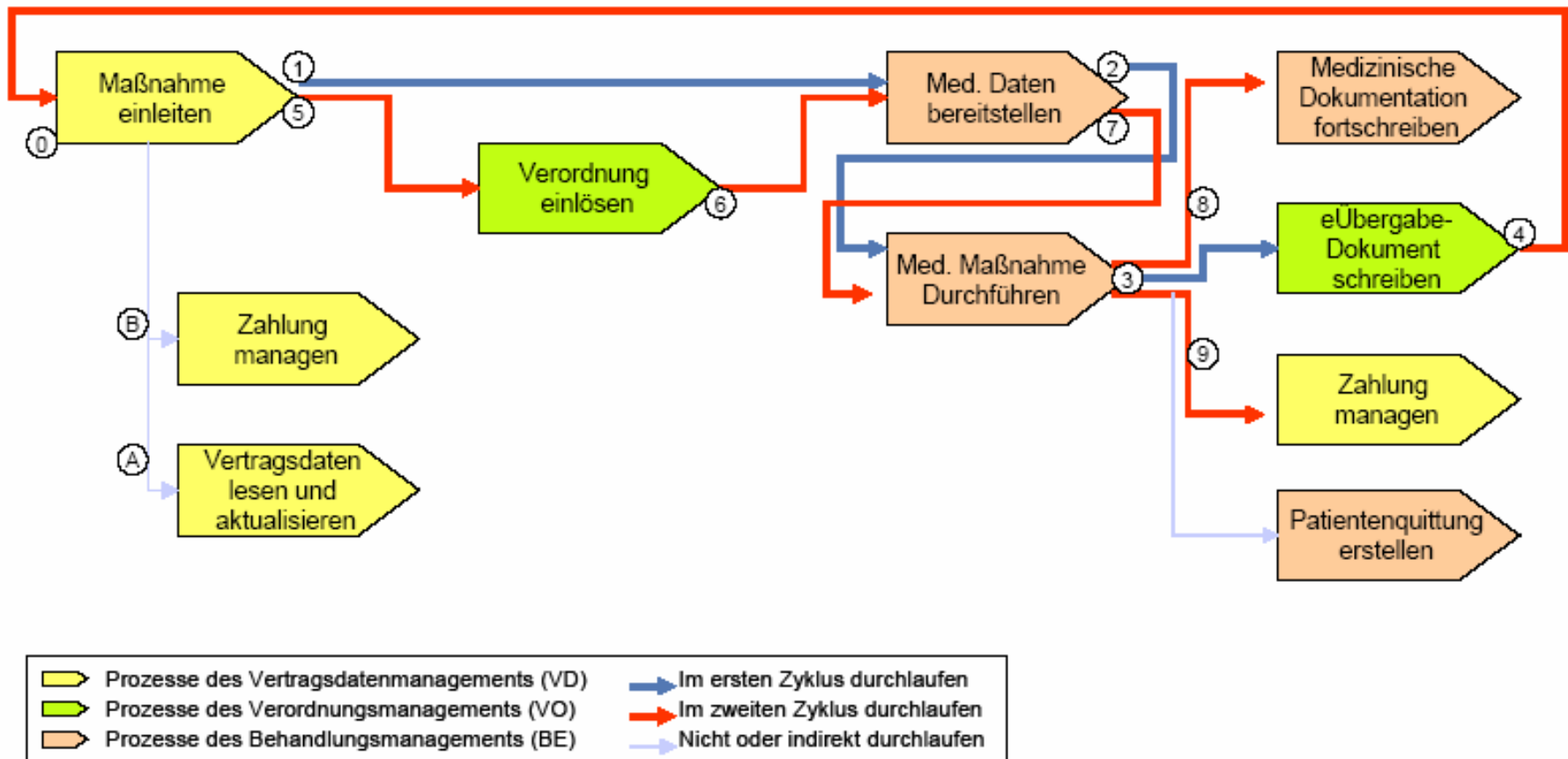
# Spezifikation: Prozeßmodell

- **Prozeßmodell** hat zwei Aspekte
  - ▶ dahinterstehende Geschäftsprozesse
    - bildet den Hintergrund
    - umfassen insbesondere die internen Abläufe und die Schnittstellen nach außen
    - in diese muß das Austauschzenario integriert werden
    - Beispiel: typische Beschaffungs- und Lieferungsprozesse
  - ▶ abstraktes Austausch-„Protokoll“
    - das wird zwischen den Partnern spezifiziert
    - keine internen Prozesse

# Spezifikation: Prozeßmodell (2)

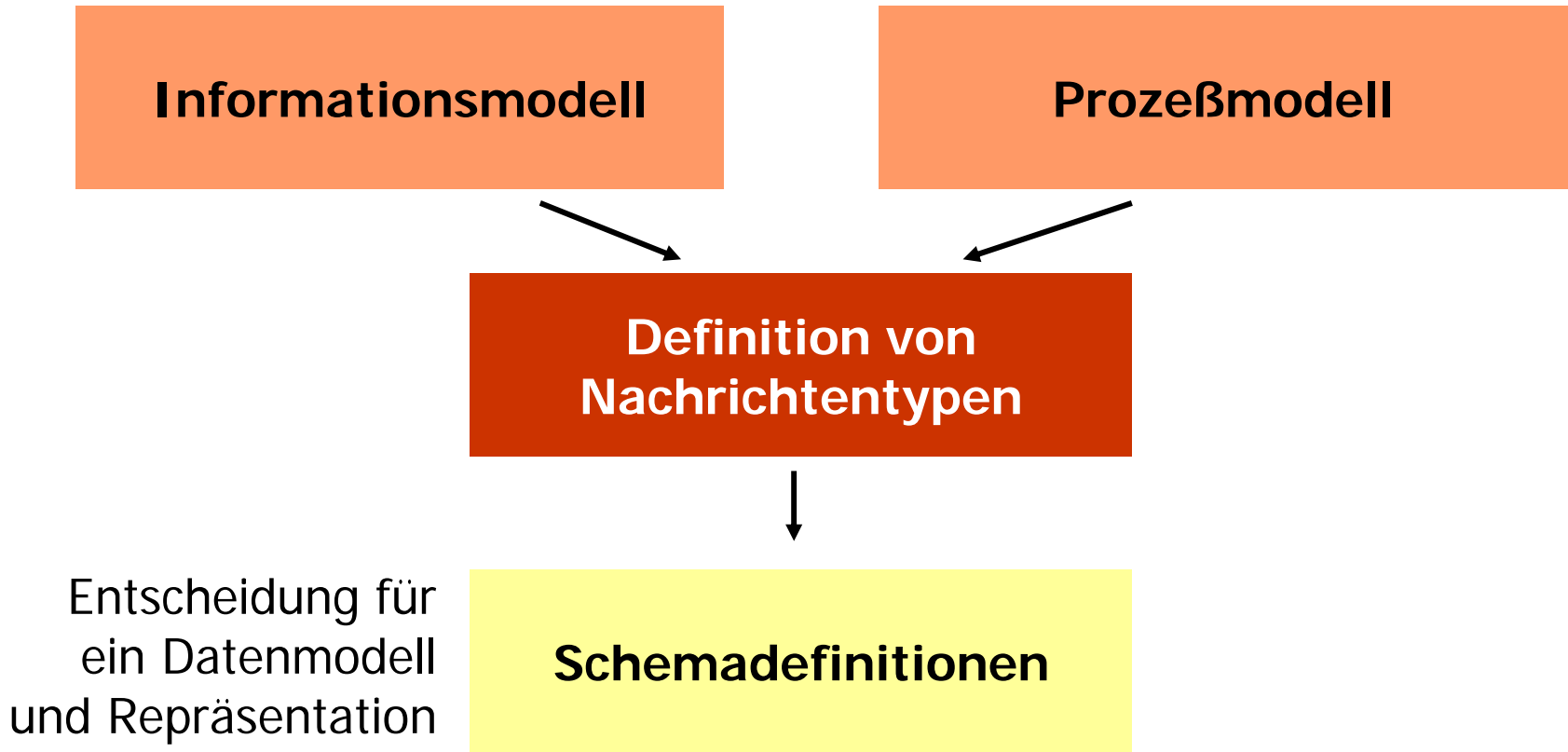
- In jedem Fall wichtig:
  - ▶ Festlegung von Rollen
- Formale Prozeßmodellierung
  - ▶ mit zunehmender „Integrationstiefe“ und Komplexität anzuraten, obwohl derzeit noch kaum praktiziert
  - ▶ z.B. UML Interaktionsdiagramme, Zustandsdiagramme
  - ▶ andere Workflow-Modellierungstechnik vgl. Vorlesung WFMS

# Spezifikation: Prozeßmodell (3)



Beispiel-Prozess aus der bit4health-Rahmenarchitektur

# Datenaustauschformat



# Spezifikation: Nachrichten

- Wie setzt man diese Prozesse um?
  - ▶ Repräsentation des Prozeßkontextes als semantische Zusatzinformation (welchen Zweck hat die Übermittlung)
    - Katalogdaten, Bestellungen, Rechnungen, ...
  
- **Nachrichtenbasierte Kommunikation**
  - ▶ Definition von Nachrichtentypen wie Bestellung, Reklamation
  - ▶ enthalten Informationsobjekte aus dem Informationsmodell als Parameter
  - ▶ Protokollautomat zur Abfolgeregelung
  - ▶ i.d.R. keine synchrone Verarbeitung (außer Quittungen)
    - separate Ergebnismnachrichten
  
- Vorstufe zur dienstbasierten Integration

# Kommunikationsprotokoll

## ■ Technische Perspektive

## ■ Entwurfsentscheidungen

- ▶ Austausch über Datenträger (z.B. Gesundheitskarte) vs. Netz
- ▶ Zentrale Sammel- und Verteilstellen vs. Punkt-zu-Punkt
- ▶ Transportprotokolle (FTAM, SMTP, FTP, HTTP, ...)
- ▶ Sicherheit:
  - Authentifizierung (Gesundheitskarte, HealthProfessionalCard)
  - Verschlüsselung
  - Signatur
- ▶ ...

# Was müssen die Partner tun?

- Informationsebene
  - ▶ Bereitstellung der Daten im gewünschten Schema
  - ▶ Verarbeitung der ausgetauschten Daten und Einbringung in das eigene System
  
- Anwendungs-/Prozeßebene (meist komplexer!)
  - ▶ Einbindung der Datenaustauschschritte in die eigenen Geschäftsprozesse
    - müssen nicht den der Spezifikation zugrundegelegten Prozessen entsprechen
  - ▶ Abstimmung der Anwendungslogik

# Datenaustausch vs. Informationsintegrationsansatz

- Generell: drei getrennte Welten
  - ▶ **Datenaustausch**: pragmatischer, aus der Praxis kommender Ansatz zum Austausch von Daten zwischen unabhängigen, aber kooperierenden Einrichtungen  
=> Schnittstellen nach außen
  - ▶ **Data Warehousing (materialisierte Integration)**: unternehmensinterner Aufbau einer integrierten Datenbasis zur Verbesserung der Auswertung, zur Wissensgewinnung etc., dabei Entkopplung der Auswertung vom operativen Geschäft => Integration im Innern
  - ▶ **Virtuelle Informationsintegration**: Ansatz, der sich auf die Übersetzung von Anfragen und Ergebnissen zur Laufzeit kümmert  
=> Integration von externen Informationsquellen

# Datenaustausch vs. Informationsintegrationsansatz (2)

- Zusammenwachsen der unterschiedlichen Problembereiche und Lösungen
  - ▶ Enterprise Application Integration
  - ▶ Web-Service-basierte Lösungen  
=> dienstorientierte Integration
- Deshalb Schwerpunkt dieser Vorlesung auf den Beitrag des Datenaustauschszenarios
  - ▶ Festlegung von Datenformaten
  - ▶ Validierung von außen empfangener Daten

**Klassisches EDI**



# Austauschformate: EDI

## ■ Electronic Data Interchange

- ▶ einigermaßen standardisierter nachrichtenbasierter Datenaustausch
- ▶ unterteilt in unterschiedlichen Nachrichtengruppen, Nachrichtentypen, Segmente, Attribute
- ▶ Definition von speziellen Nachrichtentypen, z.B.
  - Angebot
  - Bestellung
  - Rechnung
  - Bezahlung

## ■ Unterschiedliche Ausprägungen

- ▶ ANSI X.12
- ▶ UN/EDIFACT  
(**E**lectronic **D**ata **I**nterchange for **A**dministration, **C**ommerce and **T**ransport)

# Beispiel: Rechnung

*Absender:*

Klick and Bau  
Haid-und-Neu-Straße 10-14  
76131 Karlsruhe

*Empfänger:*

Huber GmbH  
Obstgasse 2  
81549 München

**Rechnungsnr. 200301 vom 08.01.2005**

Ihre Bestellung O0010001 vom 23.12.2004

1 Bohrmaschine Bosch PSB 500 RE	43,06
Summe netto	43,06
MwSt 16%	6,89
<b>Summe brutto</b>	<b>49,95</b>

# Beispiel: Rechnung (2)

UNA:+,?'

Syntaxinformationen

UNB+UNOA:2+KLICKANDBAU+HUBER+20030108:1001+200301081010'

UNH+INVOIC0001+INVOIC:D:93A:UN'

Sender und Empfänger (IDs), Datum

BGM+380+2003001+9'

Nachrichtentyp Rechnung

DTM+3+20030108+102'

Rechnung, Rechnungsnr., Original  
Rechnungsdatum

RFF+ON+00010001'

DTM+4+20021223:102'

Referenz auf Bestellnummer  
Bestelldatum

NAD+SE++Klick and Bau++Haid-und-Neu-Str.10-14+Karlsruhe++76131'

NAD+BY++Huber GmbH++Obstgasse 2+München++81549'

Adresse des Rechnungsempfängers

Adresse des Rechnungsstellers

# Beispiel: Rechnung (3)

LIN+1++4711.001'	Rechnungsposition 1 mit Artikelnr. 4711.001
IMD+F++::Bohrmaschine Bosch PSB 500 RE'	Beschreibung
QTY+47:1:PCE'	Berechnete Anzahl 1 (PCE=Stück)
MOA+43,06:750'	Nettopreis der Position
PRI+AAA:43,06'	Einzelpreis der Position
UNS+S'	Summensektion
MOA+79:43,06'	Netto-Gesamtsumme
MOA+124:6,89'	(Mehrwert-)steuer
MOA+128:49,95'	Brutto-Gesamtsumme
TAX+7+VAT+++:::16+S'	Steuerart = MwSt, Satz 16%
UNT+18+INVOIC0001'	Abschluß der Rechnung, #Segmente=18
UNZ+1+200301081010'	Abschluß der Nachricht

- EDIFACT-Lösungen haben eine große praktische Bedeutung, vor allem bei
  - ▶ hohen Transaktionsvolumina
  - ▶ stabilen Beziehungen zwischen den Partnern
  
- Z.B.
  - ▶ Einsatz in der Automobilbranche (Odette)
  - ▶ Bankenwesen (SWIFT)
  - ▶ Kommunikation im deutschen Gesundheitswesen (§§301,302 SGB V)
    - Krankenkassen, Krankenhäuser, Arbeitgeber, Sonstige Leistungserbringer
    - mit massiven Einführungsproblemen

# Probleme mit X12/EDIFACT



- Für dynamische Umgebungen schlecht geeignet
  - ▶ Zentralisierte Vergabe von Identifikationsnummern
  - ▶ Mangelnde Erweiterbarkeit der Nachrichten
- Fehlende standardisierte und maschinenverarbeitbare Nachrichtenbeschreibungssprache
  - ▶ Keine standardisierten und nachrichtentypunabhängigen Validierungswerkzeuge
  - ▶ Oft ungenaue Spezifikationen

# Probleme mit X.12/Edifact (2)

- Schwer zu erlernen
  - ▶ deshalb häufige Probleme mit inkompatiblen Implementierungen
  - ▶ Und: teuer
- Branchenspezifische Lösungen
  - ▶ Keine universelle EDIFACT-Lösung (Odette, SWIFT) wie ursprünglich anvisiert
- Für kleinere Unternehmen oder Ad-Hoc-Geschäftsbeziehungen nicht geeignet

# Datenaustausch mit XML



# XML für den Datenaustausch

- Datenaustausch ist **der** Anwendungsfall für XML
  
- Warum?
  - ▶ XML universelle Beschreibungssprache
  - ▶ (fast) beliebiger Grad der Strukturierung von unstrukturiert bis stark strukturiert ausdrückbar
  - ▶ Internationalisierung (Unicode, wohldefinierte Kodierungen)
  - ▶ Verfügbarkeit generischer Werkzeuge
    - Generierung
    - Validierung
    - Transformation
    - Import
    - Präsentation
  - ▶ selbstbeschreibendes Dateiformat

# Schema und Validierung in XML



- Dokumenttypdefinitionen (**DTDs**) als Erbe von SGML
- XML Schema als der neue Standard des W3C
- Alternative Vorschläge
  - ▶ RELAX NG
  - ▶ Schematron

# XML Schema



# Warum XML Schema?

## ■ Klassische Datenbanksicht

- ▶ Garantien für die Anwendung über die Struktur der Daten
- ▶ Sicherung der Konsistenz der Datenbasis
- ▶ Freiräume für die physikalische Schicht (Optimierung!)
- ▶ Verzicht auf Schema verlagert die Komplexität in die Anwendung und die Speicherung!

## ■ Datenaustausch

- ▶ generische Fehler- und Konsistenzprüfung der übermittelten Daten

## ■ Integration

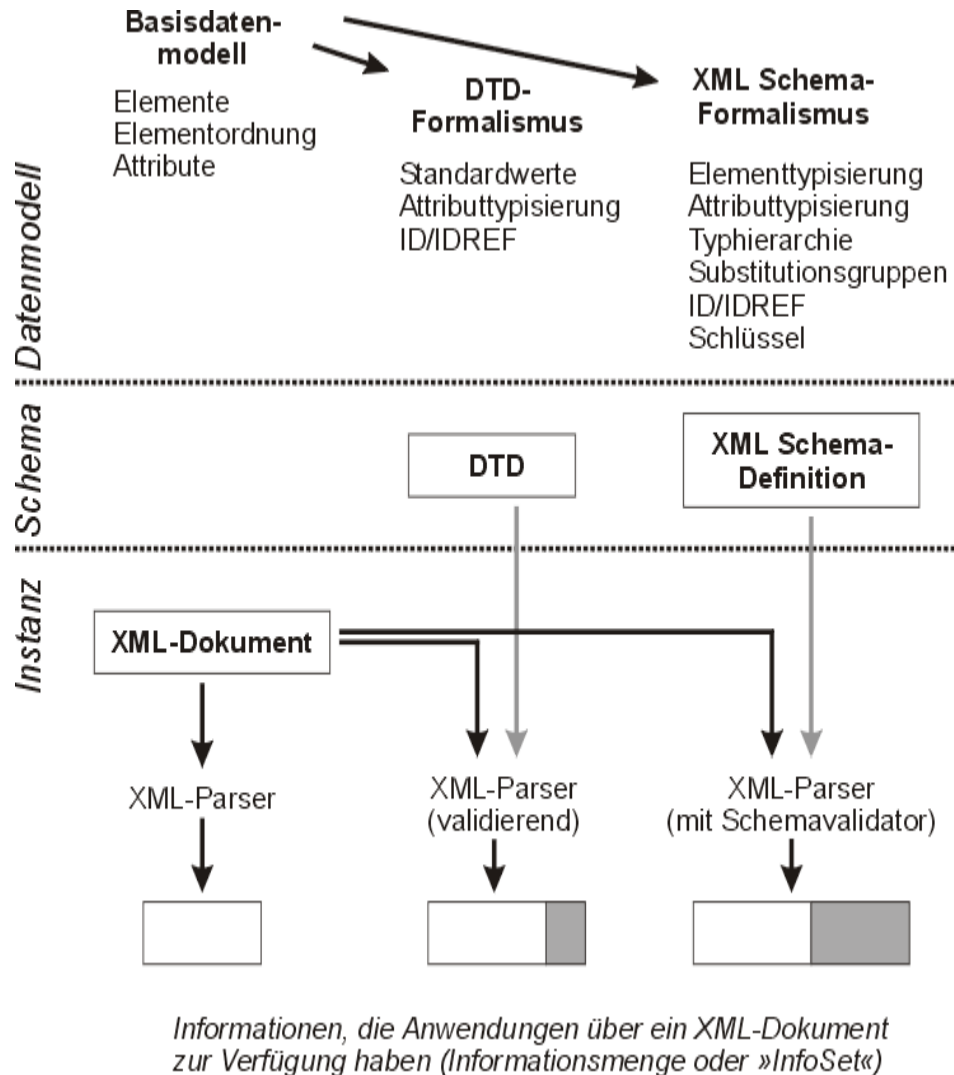
- ▶ Beschreibungssprache für a posteriori Schemata und sonstige Metainformationen über Struktur und Konsistenz

- offizielle XML-Schemadefinitionssprache des W3C
- Seit Mai 2001 Empfehlung (recommendation)
  - ▶ XML Schema 1.1 aktuell in Arbeit
- soll langfristig DTDs ganz ablösen
- Inkorporation der erweiterten Möglichkeiten (Typen, Schlüssel) in andere Standards wie XSL-T oder XPath
  - ▶ XPath 2.0, XSL-T 2.0 und XQuery: Working Draft

# XML Schema: Validierung

- Validierungskonzept ähnlich wie bei DTDs
  - ▶ Schemadefinition weiterhin optional
  - ▶ Verknüpfung mit einer Schemadefinition kann flexibel erfolgen
    - direkter Verweis im Instanzdokument
    - Angabe bei der eigentlichen Validierung
- XML Information Set als repräsentationsunabhängige Sicht auf ein XML Dokument
  - ▶ Validierung vergrößert die Informationsmenge, die Anwendungen über ein Dokument zur Verfügung steht
  - ▶ das Vorhandennsein einer Schemadefinition entscheidet also über das Datenmodell!

# XML Schema: Auswirkung der Validierung



Validierung verändert das Datenmodell und die Informationsmenge, die über ein Dokument verfügbar ist.

# XML Schema: Bausteine

**Wiederverwendung**  
Modularisierung, Erweiterbarkeit

**Konsistenzregeln**  
Schlüssel, Fremdschlüssel

**Struktur**  
für komplexe Elemente

**Datentypen**  
für Elementinhalte und Attribute

# Typen in XML Schema

- Einfache Typen
  - ▶ Atomare Typen
  - ▶ Listen- und Vereinigungstypen
  - ▶ Bildung neuer Typen
  
- Komplexe Typen (Strukturteil!)
  - ▶ all, sequence, choice, any
  - ▶ Ableitung von neuen komplexen Typen

- String, boolean
- Numerische Typen
  - ▶ Beschränkter Zahlenvorrat
    - float, double, long, int, byte, unsigned int, ...
  - ▶ Unbeschränkter Zahlenvorrat
    - number, integer
- Datum/Zeit
  - ▶ Zeitpunkte (dateTime, date, gYearMonth, gYear, ...)
  - ▶ Wiederkehrende Daten (time, gMonthDay, gMonth, ...)
  - ▶ Zeitdauer (duration)
- Binärdaten
- XML-spezifische Typen

# Ableitung neuer atomarer Typen

- Einschränkung existierender atomarer Typen
  - ▶ Datentypen sind Eigenschaften (facets) zugeordnet, die eingeschränkt werden können, z.B.
    - ▶ Aufzählung von möglichen Werte `<xsd:enumeration>`
    - ▶ Strings:
      - Zeichenanzahl
      - reguläre Ausdrücke als Muster `<xsd:pattern>`
    - ▶ Numerische Typen
      - `minInclusive`, `maxInclusive`

# Beispiel für atomare Typen

```
<simpleType name="währung">  
  <restriction base="decimal">  
    <minInclusive value="0" />  
    <fractionDigits value="2" />  
  </restriction>  
</simpleType>
```

```
<simpleType name="kategorieTyp">  
  <restriction base="string">  
    <enumeration value="Bohrmaschinen" />  
    <enumeration value="Akku-Schrauber" />  
    ...  
  </restriction>  
</simpleType>
```

# Struktur: Konstruktoren

- sequence
  - ▶ geordnete Folge von Elementen
- all
  - ▶ beliebige Reihenfolge
  - ▶ AND-Konnektor von SGML kehrt zurück!
  - ▶ Einschränkungen hinsichtlich der Unterstruktur
- choice
  - ▶ entweder - oder
- any
  - ▶ namensraumbezogene Erweiterung um beliebige Elemente

# Struktur: Kardinalitäten

- In DTDs war nur möglich
  - ▶ optional ja/nein      ?, \*
  - ▶ wiederholbar ja/nein      +, \*
  
- XML-Schema unterstützt nun auch allgemeine Kardinalitäten
  - ▶ minOccurs, maxOccurs

# Beispiel für komplexe Typen

```
<complexType name="produkttyp">
  <sequence>
    <element name="bezeichnung" type="string minOccurs="1" />
    <element name="beschreibung" type="string"
      minOccurs="0" maxOccurs="1" />
    <element name="hersteller" type="string minOccurs="1" />
    <element name="preis" type="währung" minOccurs="1" />
    <element name="kategorie" type="kategorieTyp"
      minOccurs="1" maxOccurs="3" />
  </sequence>
</complexType>
```

# Ableitung komplexer Typen



## ■ Restriktion

- ▶ Verschärfung von Kardinalitäten
- ▶ Typisierung von noch nicht typisierten Elementen

## ■ Erweiterung

- ▶ Hinzufügen weiterer Elemente ans Ende der Unterelementliste

# Konsistenzregeln

- Nullwerte
  - ▶ Unterschied zwischen `xsd:nil="true"` und leerem Element
- Eindeutigkeit (unique)
- Schlüsselbedingung (key)
- Fremdschlüsselbedingung (keyref)

## ■ select

- ▶ definiert die Menge, für deren Elemente ein Schlüssel (bzw. eine Eindeutigkeitsbedingung) definiert werden soll (*scope*)

## ■ field

- ▶ definiert die Felder, die zum Schlüssel gehören
- ▶ nur die Kind-Achse von XPath ist erlaubt
  - keine relativen Schlüssel durch Hinzunahme »übergeordneter« Werte möglich

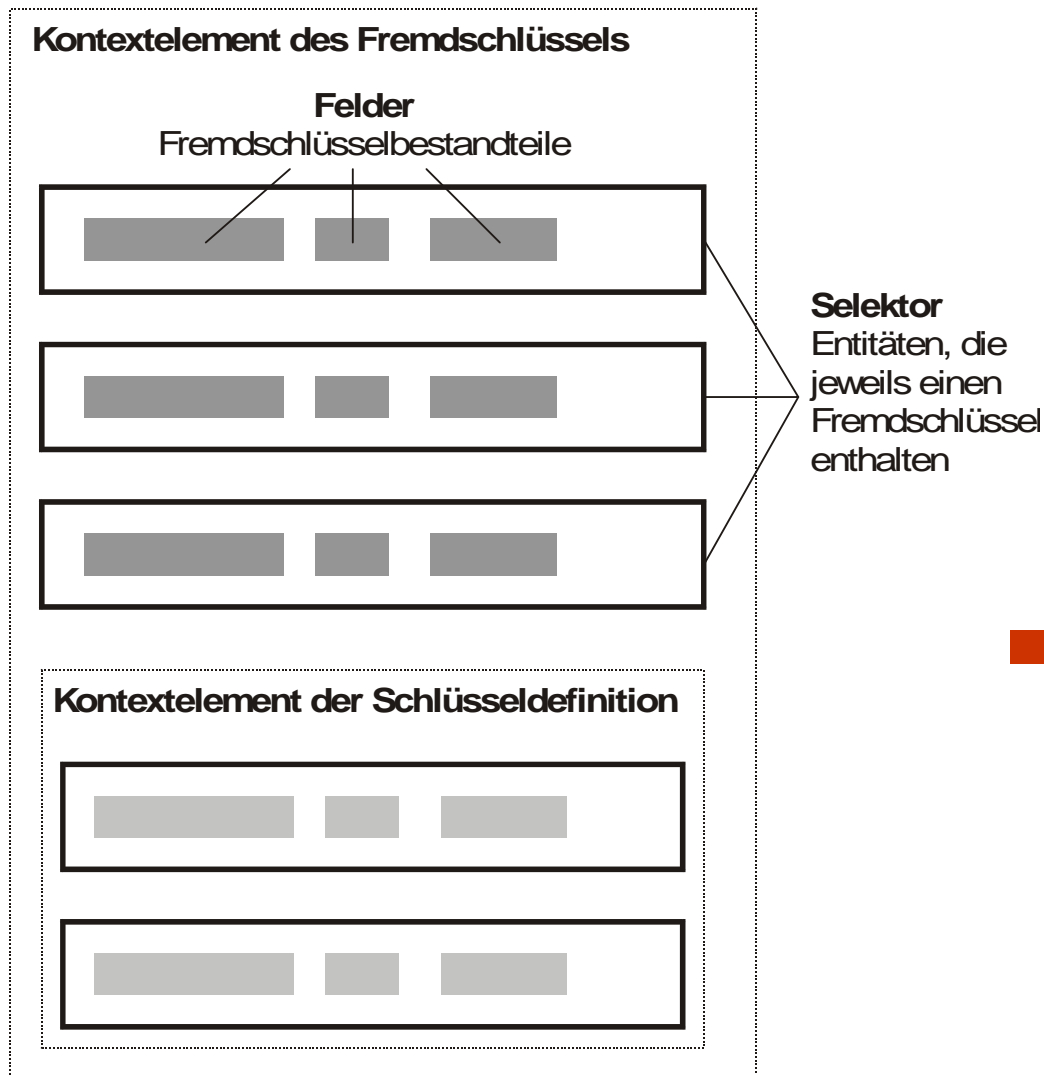
# Schlüsselkonzept: Beispiel

```
<key name="A">  
  <selector xpath="//hersteller" />  
  <field xpath="name" />  
</key>
```

```
<keyref name="B" refer="A">  
  <selector xpath="//produkte" />  
  <field xpath="hersteller" />  
</keyref>
```

```
<herstellerliste>  
  <hersteller>  
    <name>Bosch</name>  
    ...  
  </hersteller>  
</herstellerliste>  
  
<produkte>  
  <produkt>  
    <name>...</name>  
    <hersteller>Bosch</hersteller>  
  </produkt>  
</produkte>
```

# Schlüsselkonzept: Kontextelement



- Rel. Datenbanksicht
  - ▶ Kontext für Fremdschlüssel ist Datenbasis
  - ▶ Schlüsselkontext ist die Relation

# Schlüssel und Datenaustausch

- Das Schlüsselkonzept von XML Schema ist nur eingeschränkt für die Datenaustauschproblematik zu gebrauchen:
  - ▶ Schlüssel sind auf das Dokument als Einheit beschränkt
  - ▶ Validierung von Problemen wie "Artikelnummer in der Bestellung muß einen im Katalog enthaltenen Artikel referenzieren"
    - kann nur bei versuchtem Einbringen in eine Datenbasis kontrolliert werden
    - Keine Validierungsmöglichkeit bei Ankunft der Bestellung
  - ▶ Vorschläge, hierfür XLink zu benutzen
    - Aber: kaum Werkzeuge, eher ein Exot

# Bewertung von XML Schema

- sehr mächtiger Schemaformalismus für XML
  - ▶ fügt flexible Typisierung von Elementinhalten hinzu
  - ▶ erhöht Wiederverwendung
  
- aber: recht komplexer Standard
  - ▶ insbesondere konkurrierende Ausdrucksmittel
  - ▶ nicht mehr so einfach zugänglich
    - sowohl für Implementierungen
    - als auch für Anwendung

# Andere Schemaansätze für XML



# Andere Schemaansätze

## ■ RELAX NG

- ▶ Entstanden aus TREX und RELAX
- ▶ Übernahme des Datentypenteils von XML-Schema
- ▶ Vereinfachung des Strukturteils
- ▶ Außerdem: kompakte Notation
- ▶ inzwischen ISO-Standard

## ■ Schematron

- ▶ basiert auf Zusicherungen
- ▶ kann Fehlerberichte einfach erzeugen
- ▶ soll von der ISO standardisiert werden
- ▶ versteht sich als orthogonal zu anderen Schemaansätzen

- Regelparadigma statt Grammatikorientierung
  - ▶ Strukturen werden nicht »erlaubt« durch das Angeben von Produktionsregeln einer Grammatik,
  - ▶ sondern beschränkt durch Validierungsregeln, die erfüllt sein müssen
- Grundelement von Schematron ist daher die Zusicherung (assertion)
  - ▶ als Argument ein XPath-Test
  - ▶ Verknüpfung mit Berichtstexten für Fehlerberichte
- Kann vollständig mittels XSL-T realisiert werden
  - ▶ einfach und kompakte Validierungssprache

# Schematron: Beispiel

```
<schema>
  <pattern name="Produkte überprüfen">
    <rule context="produkt">
      <assert test="bezeichnung">Produktbezeichnung fehlt</assert>
      <assert test="//herstellerliste/hersteller[name=current()/hersteller]">
        Hersteller nicht in der Herstellerliste</assert>
      <assert test="floor(lagerbestand) = number(lagerbestand)">
        Anzahl nicht ganzzahlig</assert>
    </rule>
  </pattern>
</schema>
```

# Schematron (2)

- Es lassen sich auch positive Prüfergebnisse ausgeben
  - ▶ report-Element
- Schematron will andere Schemaansätze nicht ersetzen, sondern ergänzen
  - ▶ Einbettung in <appinfo> von XML Schema
  - ▶ <xsd:annotation>
    - <xsd:appinfo>
    - <sch:pattern name="...">
    - <sch:rule context="d:Demo">
    - </xsd:appinfo>
    - </xsd:annotation>
  - ▶ Analoge Einbettung in RELAX NG Schemata

# Schematron: Bewertung

- Wachsende Beliebtheit von Schematron
  - ▶ Zahlreiche Validatoren sind bereits verfügbar
  
- Sehr einfach
  
- Hervorragend geeignet für Validierungsberichte
  
- Mögliche Perspektive
  - ▶ Formulierung der Grobstruktur mit XML Schema unter Verzicht zu starker Einschränkungen
  - ▶ Formulierung zusätzlicher Einschränkungen über Schematron

**Fazit**



- XML hat sich im Bereich des Datenaustausches für neue Projekte als die Technologie der Wahl herauskristallisiert
  - ▶ standardisierte Spezifikations Sprachen für Schemata
  - ▶ generische Werkzeuge (davon viele frei verfügbar) zu ihrer Verarbeitung
    - insbesondere Kombination mit Transformation
  - ▶ (leichte Integration/Migration in Web Service-Infrastrukturen)
  
- aber: XML ist nicht die Lösung aller Probleme
  - ▶ Verdrängt auf absehbare Zeit klassische EDI-Lösungen nicht
  - ▶ Die Verwendung von XML macht nicht die Systeme von sich aus interoperabel!
  - ▶ Insbesondere die Entwicklung der Vokabulare ist schwierig  
=> Ontologieproblematik

# Weiteres Programm



- Dienstorientierte Sicht auf die Integrationsproblematik
  - ▶ nächste Vorlesung:  
Dienstorientierte Integration mit **Web Services**
- Ontologien als Werkzeug für die Informationsintegration
  - ▶ konzeptuelle Modelle
  - ▶ logische Schlußfolgerungen
  - ▶ Stichwort: **Semantic Web**

- Wassili Kazakos, Andreas Schmidt, Peter Tomczyk:  
Datenbanken und XML, Springer, März 2002
  - ▶ Kapitel 4
- W3C <http://www.w3c.org>
- RELAX-NG
  - ▶ <http://www.relaxng.org/>
- Schematron
  - ▶ <http://www.schematron.com>