

Informationsintegration

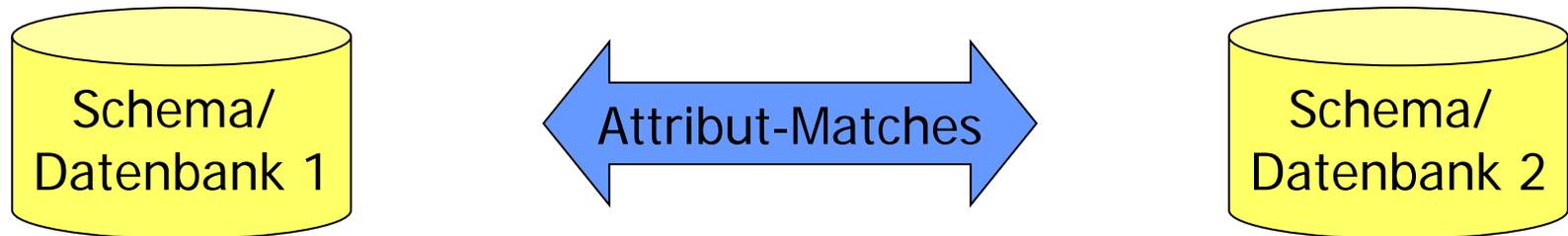
Schemaintegration

Ulf Leser

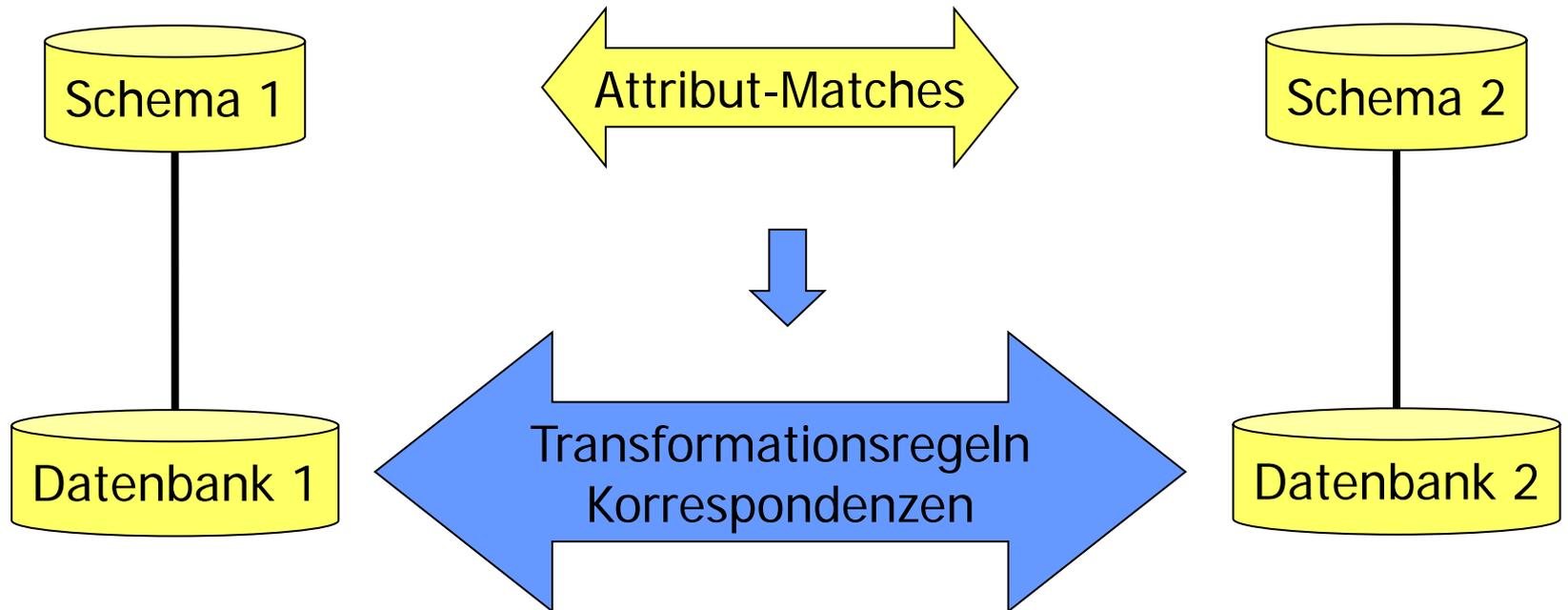
Inhalt dieser Vorlesung

- Schema Management
- Schemaintegration
- Correspondence Assertions
- Generic Integration Model

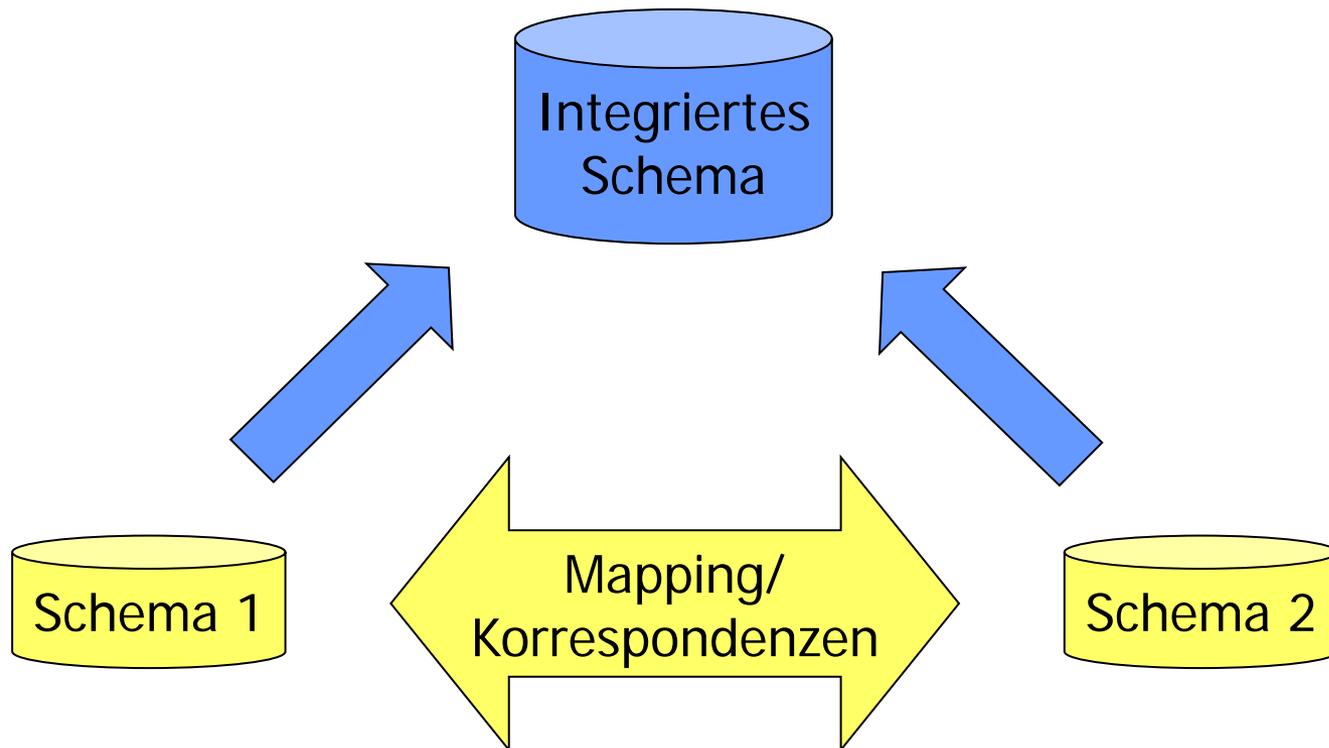
Schema Matching



Schema Mapping



Korrespondenzbasierte Schemaintegration



Inhalt dieser Vorlesung

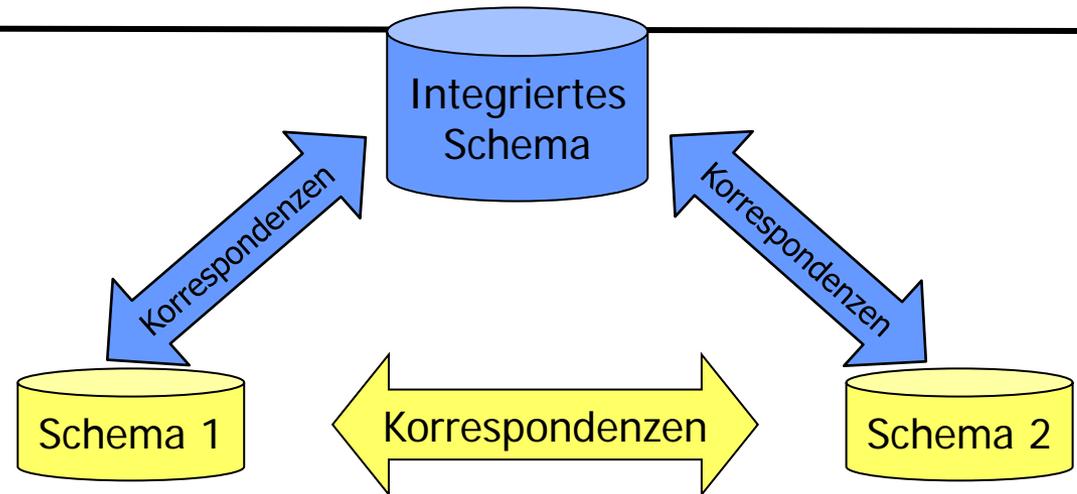
- Schema Management
- **Schemaintegration**
- Correspondence Assertions
- Generic Integration Model

Schemaintegration

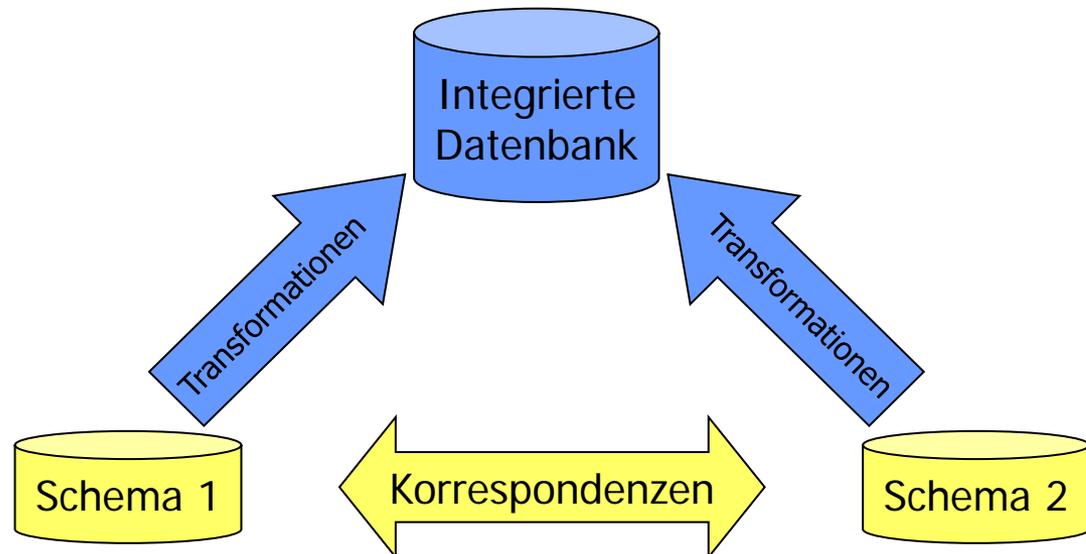
- Erzeugung eines **integrierten Schemas** aus Quellschemata
- Forderungen
 - **Vollständigkeit**: Alle Daten der Quellsysteme sind repräsentiert
 - Nicht immer notwendig
 - **Korrektheit**: Alle Daten werden semantisch korrekt repräsentiert
 - Integritätsconstraints, Kardinalitäten, ...
 - **Minimal**: Minimales integriertes Schema (Relationen / Attribute)
 - Insbesondere redundanzfrei
 - Ohne diese Forderung wäre Schemaintegration einfach – wie?
 - **Verständlich** (ganz schwer zu messen)
- Keine dieser Forderungen ist algorithmisch prüfbar

Szenarien

Virtuelle Integration

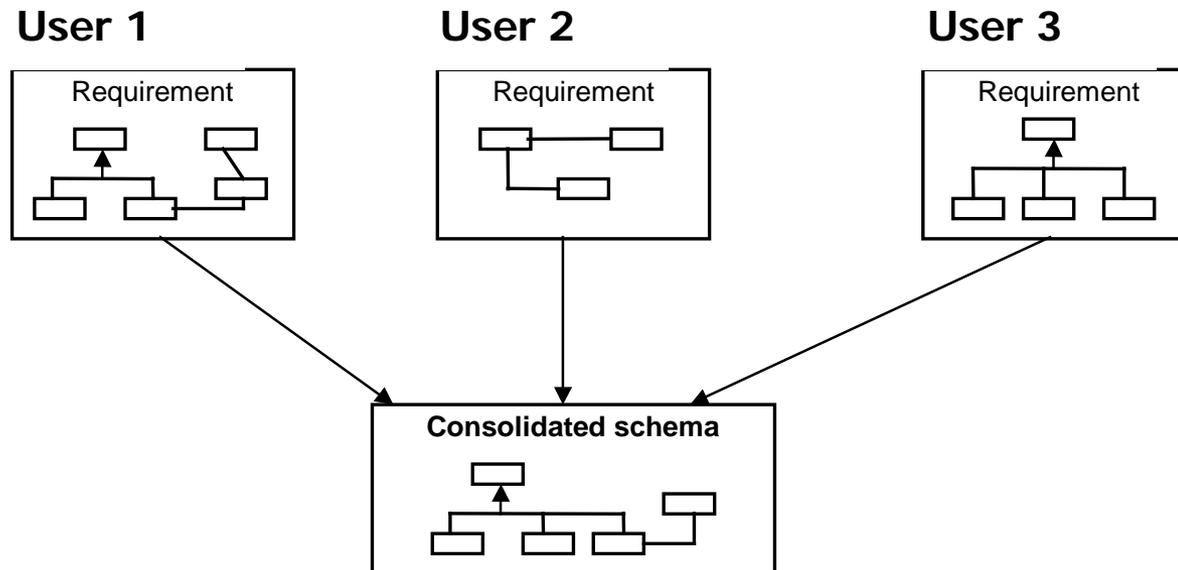


Materialisierte Integration



Verwandtes Problem

Sichtintegration beim Design großer Systeme



Unterschied: Keine Daten

Integrationssschritte

- Vorintegration
 - Auswahl der Integrationsreihenfolge (binäre Einzelschritte)
 - Konvertierung der Daten in kanonisches Datenmodell
- Schemavergleich
 - Ermittlung von **Korrespondenzen** (Schema Matching)
 - Identifikation von Konflikten (schematisch, semantisch, ...)
- Schemaangleichung
 - Erste Umformung der Schemata zur Behebung von Konflikten
- **Schemafusion**
 - Erstellung des integrierten Schemas **aufgrund der Korrespondenzen**
 - Auflösung noch bestehender Konflikte

Methoden und Verfahren

- In der Literatur wurden viele Verfahren vorgestellt [Con98]
 - Upward inheritance
 - Assertion-based integration
 - Object-oriented schema integration
 - Generic Integration Model
 - Carnot
 - Pegasus
 - ...
- Meist eher **Vorgehensbeschreibungen** als Algorithmen
- Keine kommerziellen Implementierungen
- Wir besprechen zwei Beispiele

Inhalt dieser Vorlesung

- Schema Management
- Schemaintegration
- Correspondence Assertions
- Generic Integration Model

Korrespondenzbasiertes Verfahren [SPD92]

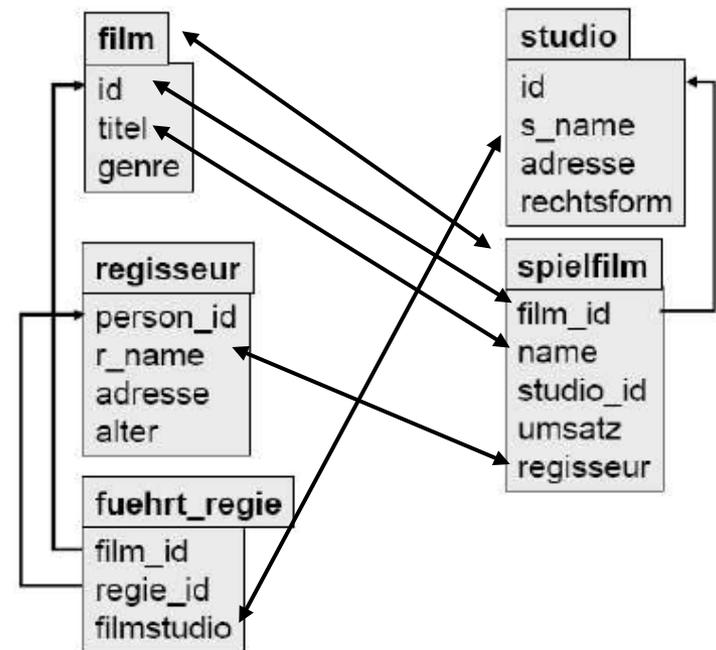
- Eingabe
 - Zwei Schemata im **Generic Data Model**
 - Klassen, komplexe und einfache Attribute, Beziehungen
 - Eng verwandt zu E/R Modellen
 - Subsumiert viele andere Datenmodelle (z.B.: relationales Modell)
 - **Correspondence Assertions (CA)**
 - Korrespondenzen zwischen Klassen und Attributen
 - Korrespondenzen **zwischen Pfaden** (Beziehungen zwischen Klassen)
 - Theoretisch alle Korrespondenztypen
 - Wir (und [SPD92]) behandeln nur Äquivalenz
 - Annahme: **1:1 Beziehungen** zwischen Attributen
- Ausgabe: Integriertes Schema S im GDM

Integrationsregeln

- Nach S übernommen werden
 1. Elemente, die in **keiner CA alleine** auftauchen
 2. Vereinigung der Attribute **äquivalenter Klassen**
 3. **Äquivalente direkte Beziehungen** zwischen äquivalenten Klassen
 $A \equiv A', B \equiv B', A-B \equiv A'-B'$: Dann wird A-B übernommen
 4. **Pfade** zwischen äquivalenten Klassen
 - $A \equiv A', B \equiv B', A-B \equiv A'-A_1'-\dots-A_m'-B'$: Nur der längere Pfad
 - Der kürzere wird subsumiert
 - $A \equiv A', B \equiv B', A-A_1-\dots-A_n-B \equiv A'-A_1'-\dots-A_m'-B'$: Beide Pfade
 5. Äquivalenzen zwischen einer **Klasse und einem Attribut** werden als Beziehungen übernommen
 - Eingeschränkte Behandlung **schematischer Heterogenität**

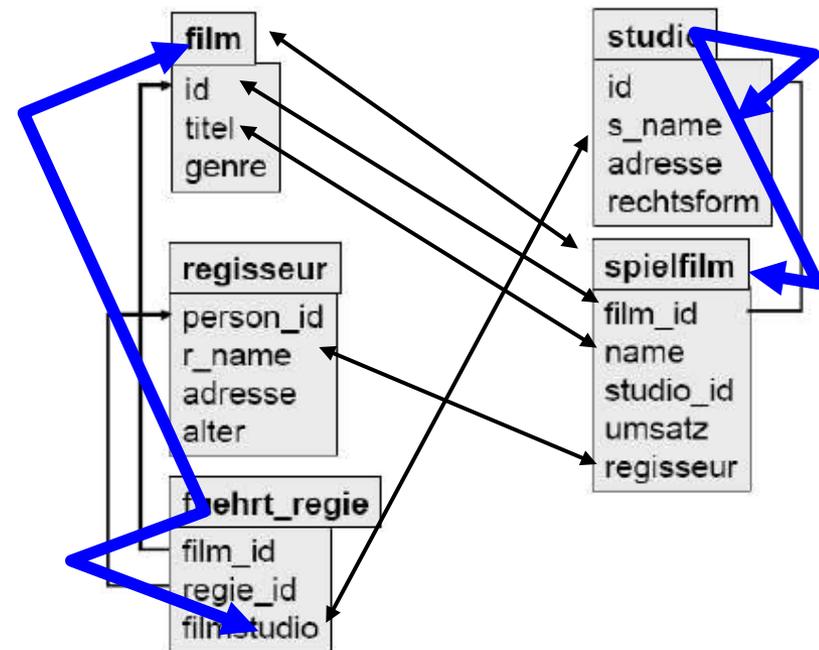
Beispiel

- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`



Beispiel

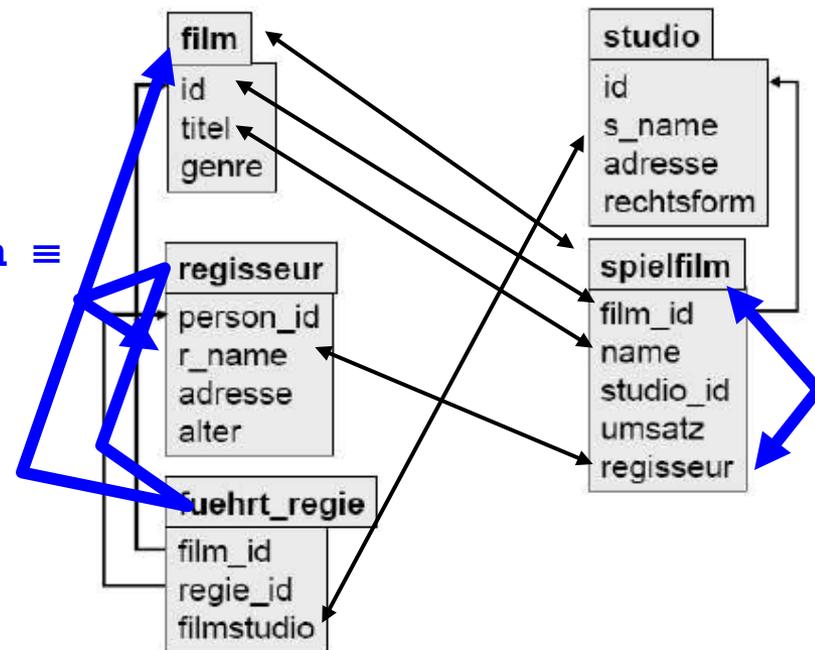
- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`
- `filmstudio-fuehrt_regie-film` \equiv `s_name-studio-spielfilm`



Beispiel

- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`
- `filmstudio-fuehrt_regie-film` \equiv
`s_name-studio-spielfilm`

`r_name-regisseur-fuehrt_regie-film` \equiv
`regisseur-spielfilm`



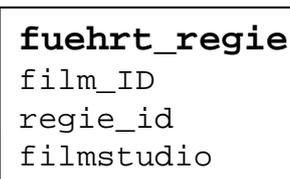
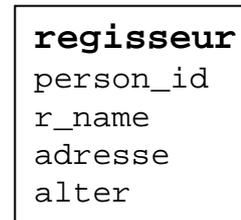
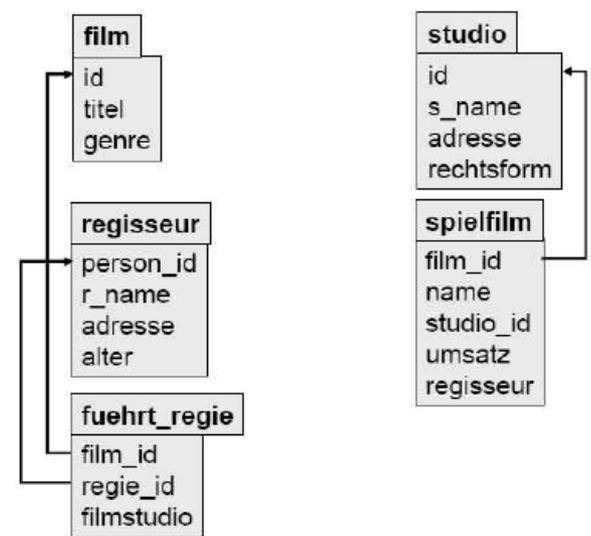
Beispiel

- CAs

- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`
- `filmstudio-fuehrt_regie-film` \equiv `s_name-studio-spielfilm`
- `r_name-regisseur-fuehrt_regie-film` \equiv `regisseur-spielfilm`

- Integriertes Schema

- Regel 2: äquivalente Klassen `film` und `spielfilm` verschmelzen (zu `film`)
- Regel 1: Klassen ohne direktes Äquivalent werden in integriertes Schema übernommen



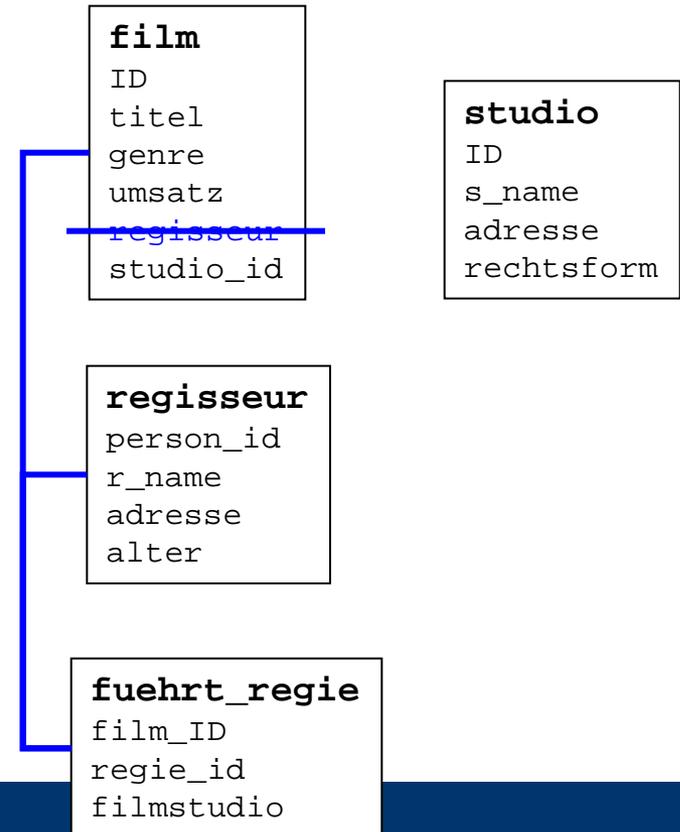
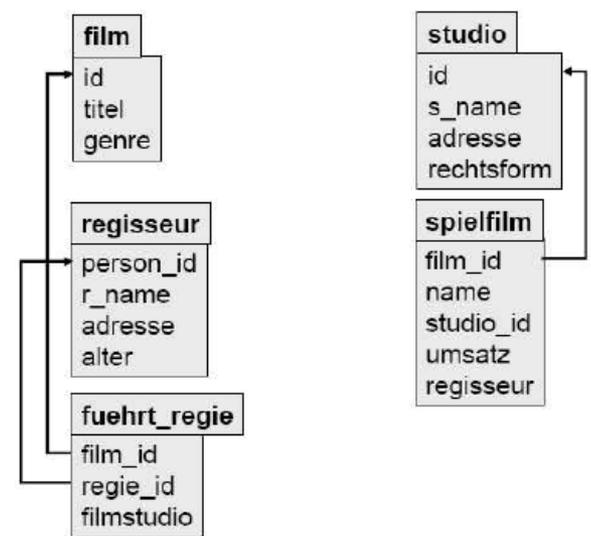
Beispiel

- CAs

- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`
- `filmstudio-fuehrt_regie-film` \equiv `s_name-studio-spielfilm`
- `r_name-regisseur-fuehrt_regie-film` \equiv `regisseur-spielfilm`

- Integriertes Schema

- Regel 4a: Der Pfad `r_name-regisseur-fuehrt_regie-film` wird übernommen, der Pfad `regisseur-spielfilm` fällt weg
 - Also werden `r_name` und `regisseur` verschmolzen



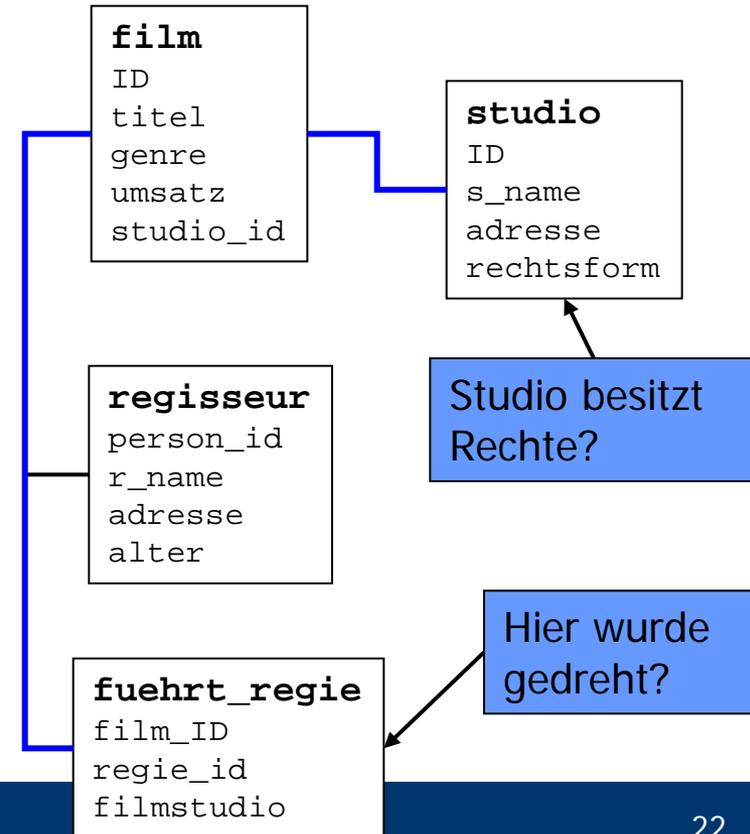
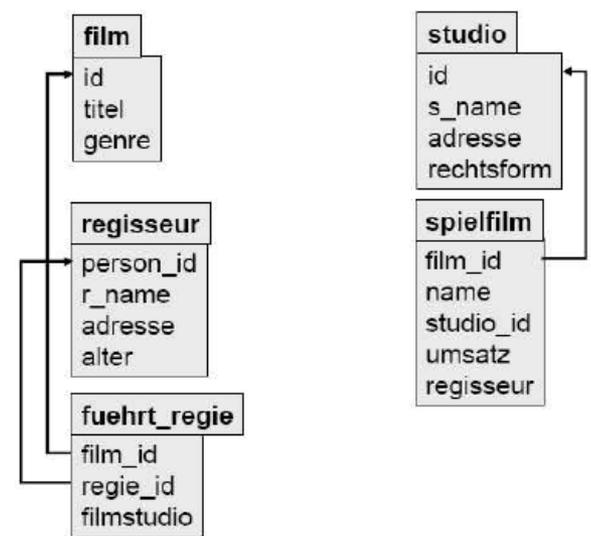
Beispiel

- CAs

- `film` \equiv `spielfilm`
 - `id` \equiv `film_id` und `titel` \equiv `name`
- `r_name` \equiv `regisseur`
- `filmstudio` \equiv `s_name`
- `filmstudio-fuehrt_regie-film` \equiv
`s_name-studio-spielfilm`
- `r_name-regisseur-fuehrt_regie-film` \equiv
`regisseur-spielfilm`

- Integriertes Schema

- Regel 4b: Beide Pfade werden übernommen
 - `filmstudio` und `s_name` werden nicht verschmolzen



Bewertung

- Behandelt **nur Äquivalenzkorrespondenzen**
 - Weiterentwicklung ist nicht erfolgt
- Keine Integration von nicht-binären Beziehungen
- Generiert **komplexe und große Schemata**
 - „Beide Pfade werden übernommen“: Übernahme ggf. redundanter Pfade
 - Große Gefahr redundanter Elemente bei nicht sehr sorgfältig spezifizierten Korrespondenzen
- 1:1 Beziehungen zw Attributen nicht immer vorhanden

Inhalt dieser Vorlesung

- Schema Management
- Schemaintegration
- Correspondence Assertions
- **Generic Integration Model**

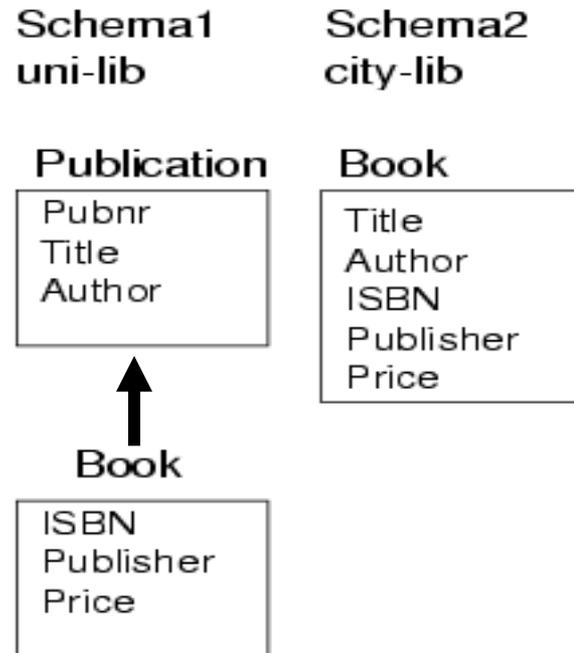
Generic Integration Model (GIM) [Sch98]

- Integration von **Spezialisierungshierarchien** in objektorientierten Schemata
- Verwendet keine expliziten Korrespondenzen, sondern Wissen über äquivalente **(Teil-)Extensionen / Intensionen**
- Integriertes Schema wird so aufgebaut, dass alle (Teil-)Extensionen und Intensionen repräsentiert sind
- Zur Erinnerung
 - Extension: Menge der Objekte einer Klasse
 - Intension: Menge der Attribute einer Klasse

Übersicht

- **Extensionale Zerlegung**
 - Finde alle möglichen extensionalen Überlappungen
 - Könnte man auf konkreter Instanz berechnen (Duplikaterkennung)
 - Aber: Gesucht sind alle **möglichen Überlappungen** (Semantik)
 - Berechne minimale Menge disjunkter Teilextensionen
 - Kleinste Menge disjunkter Teilextensionen so, dass alle realen Extensionen als deren Vereinigungen dargestellt werden können
- **Intensionale Zerlegung**
 - Löse pro Schema syntaktische Heterog. und Granularitätsprobleme
 - Attribute entsprechen sich danach 1:1 (oder gar nicht)
 - Dann: Finde alle intensionalen Überlappungen
- Berechnung der **GIM Matrix**: Beide Zerlegungen kombiniert
- Schritt 4: Ableitung des integrierten Schemas

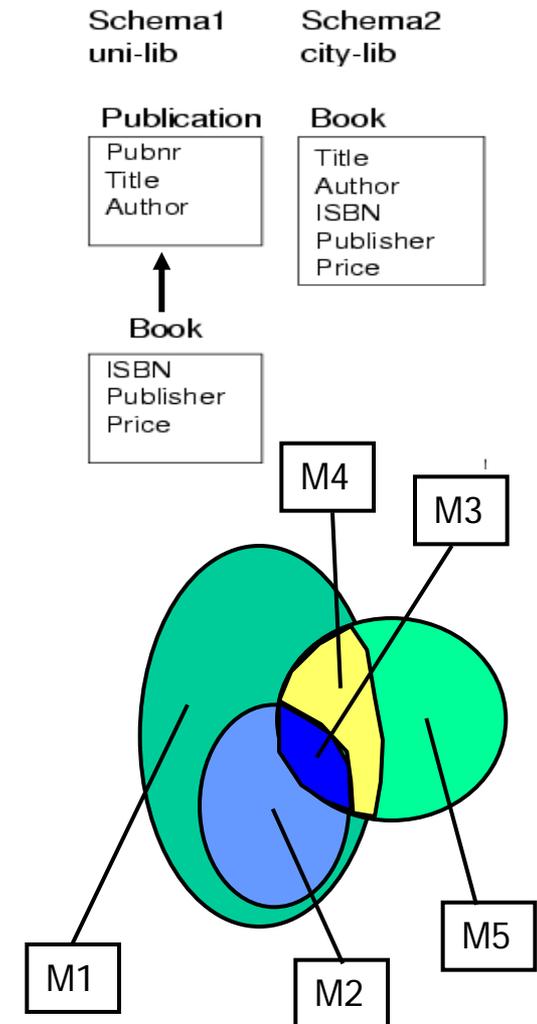
Beispiel



Extensionale Zerlegung

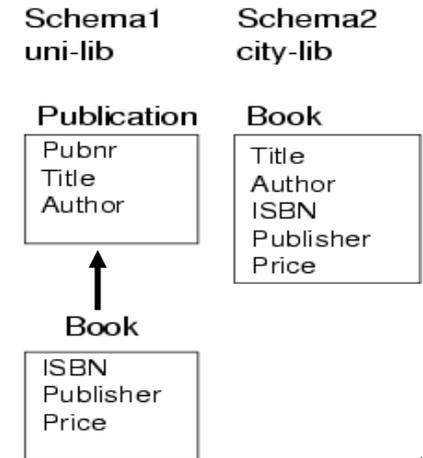
- $S1.books \subseteq S1.publication$
- $S1.books \cap S2.books \neq \emptyset$
- Fünf disjunkte Extensionen

	M1	M2	M3	M4	M5
	S1.Pub, not S1.book or S2.book	not S2.book	Everything	S2.book, not S1.book, but S1.pub	Only S2.book
S1.publicati	X	X	X	X	
S1.book		X	X		
S2.book			X	X	X



Intensionale Zerlegung

- Annahme: Gleiche Attribute – gleiche Semantik
- Drei disjunkte Mengen



	X1	X2		X3		
	PubNr	Title	Author	Publisher	ISBN	Price
S1.publica	X	X	X			
S1.book	X	X	X	X	X	X
S2.book		X	X	X	X	X

Schritt 3: GIM Matrix

- Kombiniere intensionale und extensionale Zerlegungen
 - Welche Extensionen haben (potentiell) welche Intensionen?
 - Berechnung aller atomaren **Paare Extension/Intension**

	M1	M2	M3	M4	M5
	S1.Pub, not S1.book	not S2.book	S1.book and S2.book	S2.book, not S1.book	Only S2.book
S1.publisher	X	X	X	X	
S1.book		X	X		
S2.book			X	X	X

	X1	X2	X3			
	PubNr	Title	Author	Publisher	ISBN	Price
S1.publish	X	X	X			
S1.book	X	X	X	X	X	X
S2.book		X	X	X	X	X

	M1	M2	M3	M4	M5
X1	X	X	X	X	
X2	X	X	X	X	X
X3		X	X	X	X

GIM Matrix

	M1	M2	M3	M4	M5
X1	X	X	X	X	
X2	X	X	X	X	X
X3		X	X	X	X

- Man könnte aus jedem „X“ eine Klasse machen
 - Füllung aus Originaldaten durch Schnitt von Extensionen und Projektion von Intension
 - Führt zu feingranularen, unverständlichen Schemata
- Idee: Finde eine **minimale Menge von vollständigen Klassen**, die die Matrix abdecken
 - Vollständig: Werte für alle Attribute
 - Extensionale Überlappungen in Vreerbungen überführen

Schritt 4a: Sortierung der GIM Matrix

- Permutation der Spalten und der Zeilen
- Ziel: Lücken nach rechts-unten bewegen
- Dabei gibt es viele Freiheitsgrade
 - Mit Auswirkungen auf das integrierte Schema

	M1	M2	M3	M4	M5
X1	X	X	X	X	
X2	X	X	X	X	X
X3		X	X	X	X



Vertausche X1 und X2

	M1	M2	M3	M4	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3		X	X	X	X



Schiebe M1 vor M5

	M2	M3	M4	M1	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3	X	X	X		X

Schritt 4b: Schemaableitung

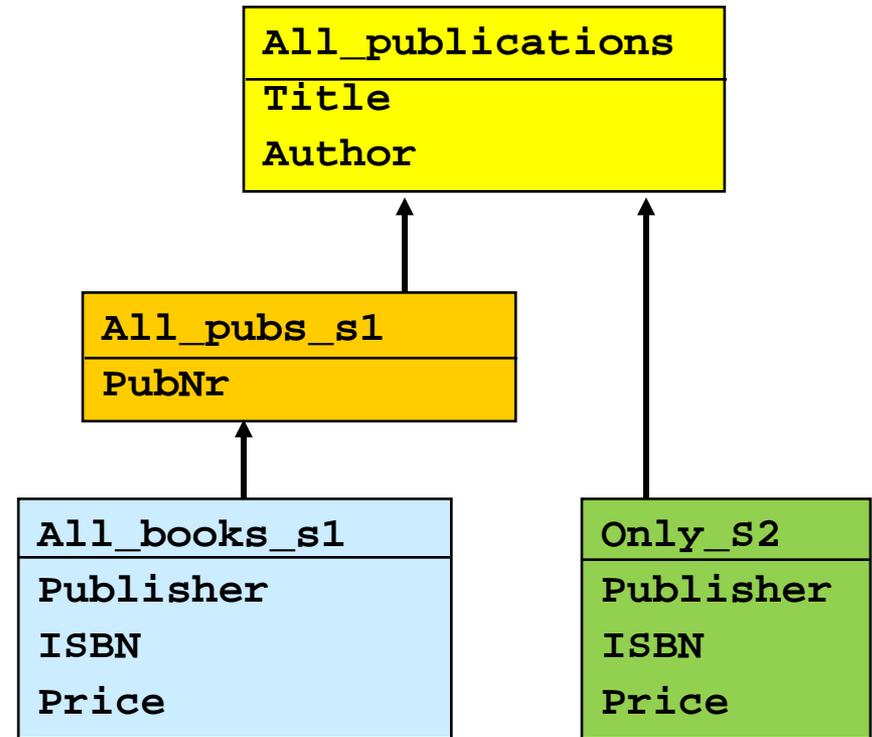
- **Blöcke** der Matrix ergeben Klassen
 - Mehr Zeilen und weniger Spalten– Unterklasse
 - Lücken in den Blöcken bestimmen Eltern

	M2	M3	M4	M1	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3	X	X	X		X

	M2	M3	M4	M1	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3	X	X	X		X

	M2	M3	M4	M1	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3	X	X	X		X

	M2	M3	M4	M1	M5
X2	X	X	X	X	X
X1	X	X	X	X	
X3	X	X	X		X



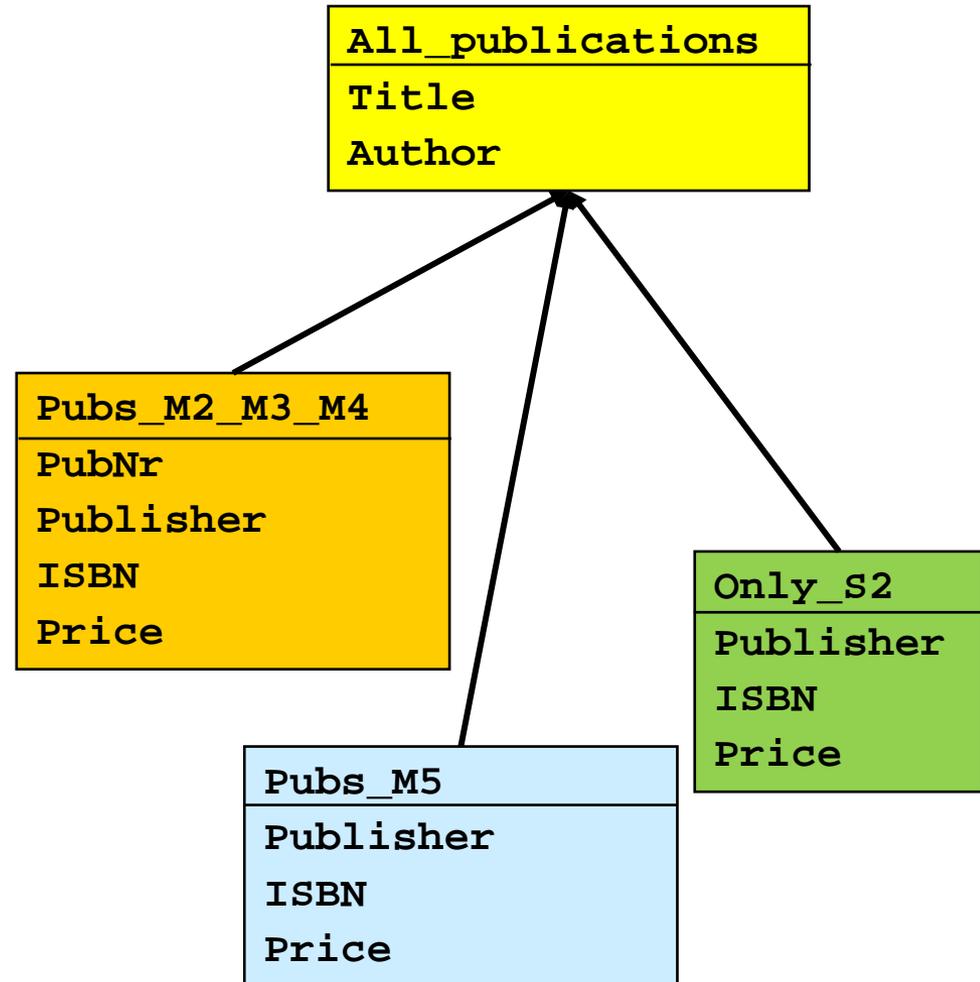
Andere Sortierung / Schemaableitung

	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	

	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	

	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	

	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	



Warum nicht?

	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	

All_Pubs
Title
Author
PubNr
Publisher
ISBN
Price

- Manche Attributwerte werden, je nach Herkunft, immer NULL sein
- Herkunft der Daten ist nicht mehr im Schema ablesbar

Eigenschaften von GIM

- Elegante Methode
- Einige Einschränkungen
 - Keine **Behandlung von Beziehungen**
 - Keine Behandlung schematischer Konflikte
 - Attribute müssen sich 1:1 transformieren lassen
- Korrespondenzen sind in den ersten Schritten verborgen
 - Zerlegung verlangt detailliertes Verständnis beider Schemata
 - Äquivalente / subsumierte Attribute / Klassen gehen in Zerlegung ein
- Diverse Freiheitsgrade bei Schemaableitung
- Versucht immer, einer einzigen Hierarchie zu erzeugen
- Neigt zu **großen und komplexen Hierarchien**

Fazit Schemaintegration

- Korrespondenzen finden – Schema ableiten
- Verfahren nur halbautomatisch
- Generieren alle große und komplexe Schemata
 - Vereinfachung verlangt Anwendungswissen
- Kein Ansatz kann alle Arten von Heterogenität auflösen
- Neue Richtung: Schema Management [BLP00]
 - Operatoren, die auf Schemata operieren und Schemata erzeugen
 - Ziel: Algebra zum Umgang mit Schemata