Informationsintegration

Schemaintegration

Ulf Leser

Inhalt dieser Vorlesung

- Schemaintegration
- Correspondence Assertions
- Generic Integration Model

Schemaintegration

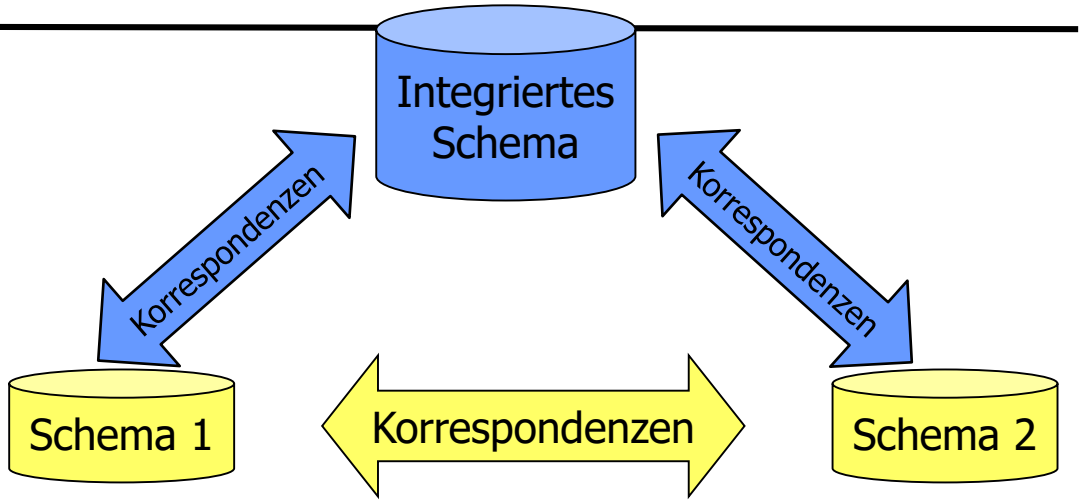
- Erzeugung eines integrierten Schemas aus verschiedenen (ggf mehr als zwei) Quellschemata
- Forderungen
  - Vollständigkeit: Alle Daten der Quellsysteme sind repräsentierbar
    - Nicht immer notwendig
  - Korrektheit: Alle Daten werden semantisch korrekt repräsentiert
    - Integritätsconstraints, Kardinalitäten, …
  - Minimal: Minimales integriertes Schema (Relationen / Attribute)
    - Es gibt kein kleineres Schema, dass vollständig und korrekt ist
    - Ohne diese Forderung wäre Schemaintegration einfach – wie?
  - Verständlich (ganz schwer zu messen)
- Keine dieser Forderungen ist algorithmisch prüfbar

Grundidee

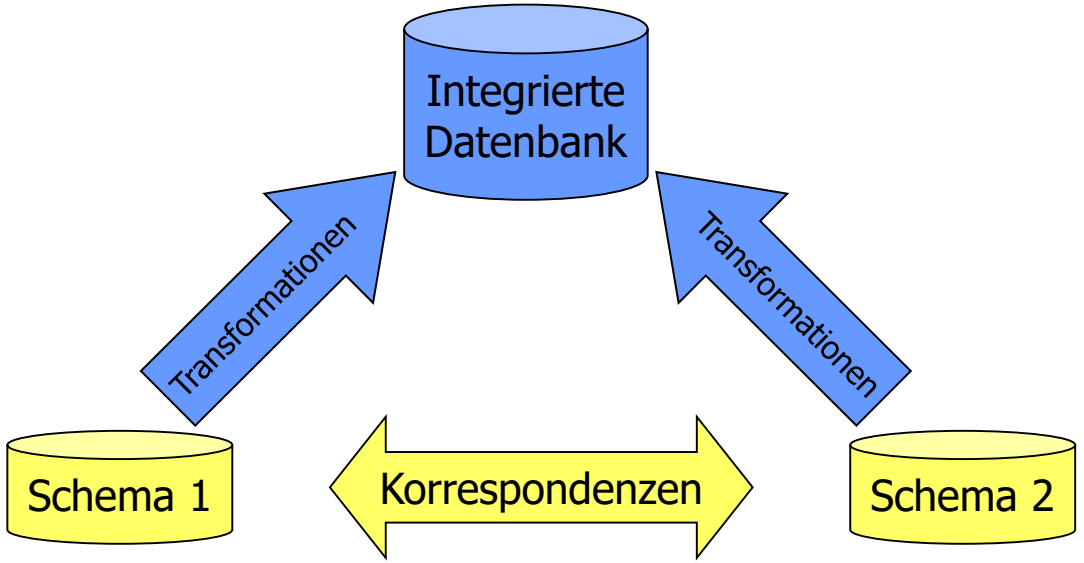
- Semantisch gleiche Sachverhalte nur einmal aufnehmen
- Subsumptionsbeziehungen ggf. durch Vererbung ausdrücken
  - Wenn man ein Datenmodell mit Vererbung benutzt
- Sachverhalte in nur einer Quelle werden idR übernommen
  - Um Vollständigkeit zu gewährleisten
  - Führt zu vielen NULL-Werten bei der Datenmigration
- Wenn feinere und gröbere Auflösungen vorhanden – die feinere übernehmen
- All diese Entscheidungen benötigen Wissen über Beziehungen zwischen Schemaelementen
  - Korrespondenzen

Szenarien

Virtuelle Integration

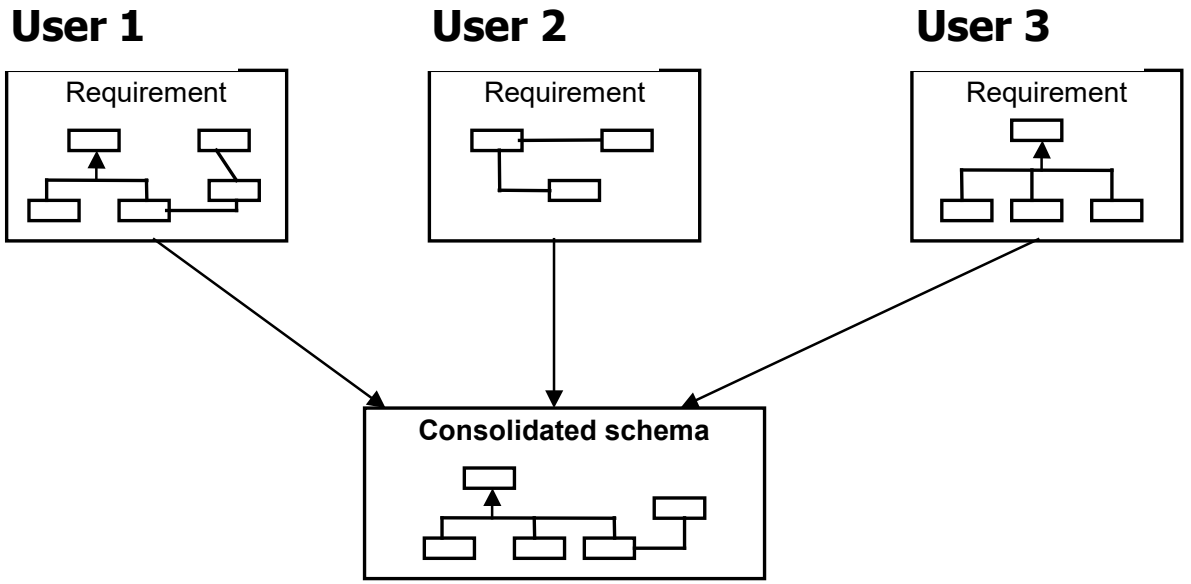


Materialisierte Integration



Verwandtes Problem

Sichtintegration beim Design großer Systeme



Unterschied: Keine Daten

Integrationsschritte

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

Methoden und Verfahren

- In der Literatur wurden diverse Verfahren vorgestellt [Con98]
  - Upward inheritence
  - Assertion-based integration
  - Object-oriented schema integration
  - Generic Integration Model
  - Carnot
  - Pegasus
  - ...
- Keine mir bekannten kommerziellen Implementierungen
- Wir besprechen zwei Beispiele

Inhalt dieser Vorlesung

- Schemaintegration
- Correspondence Assertions
- Generic Integration Model

Korrespondenzbasiertes Verfahren [SPD92]

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

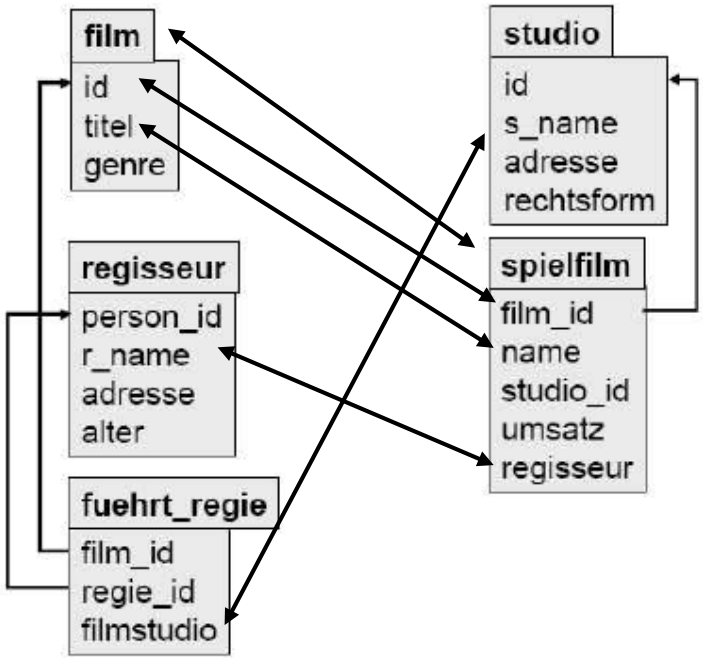
Integrationsregeln

- Nach S übernommen werden
  1. Klassen, die in keiner CA alleine (ohne Pfade) auftauchen
  2. Vereinigung der Attribute äquivalenter Klassen
  3. Äquivalente direkte Beziehungen zwischen äquivalenten Klassen

     A ≡ A‘, B ≡ B‘, A-B ≡ A‘-B‘: Dann wird A-B übernommen
  4. Pfade zwischen äquivalenten Klassen
     - A≡A‘, B ≡ B‘, A-B ≡ A‘-A_1‘-…-A_m‘-B‘: Nur der längere Pfad
       - Der kürzere Pfad wird subsumiert, weil redundant
     - A≡A‘, B ≡ B‘, A-A_1-…-A_n-B ≡ A‘-A_1‘-…-A_m‘-B‘: Beide Pfade
  5. Äquivalenzen zwischen einer Klasse und einem Attribut werden als Beziehungen übernommen
     - Eingeschränkte Behandlung schematischer Heterogenität

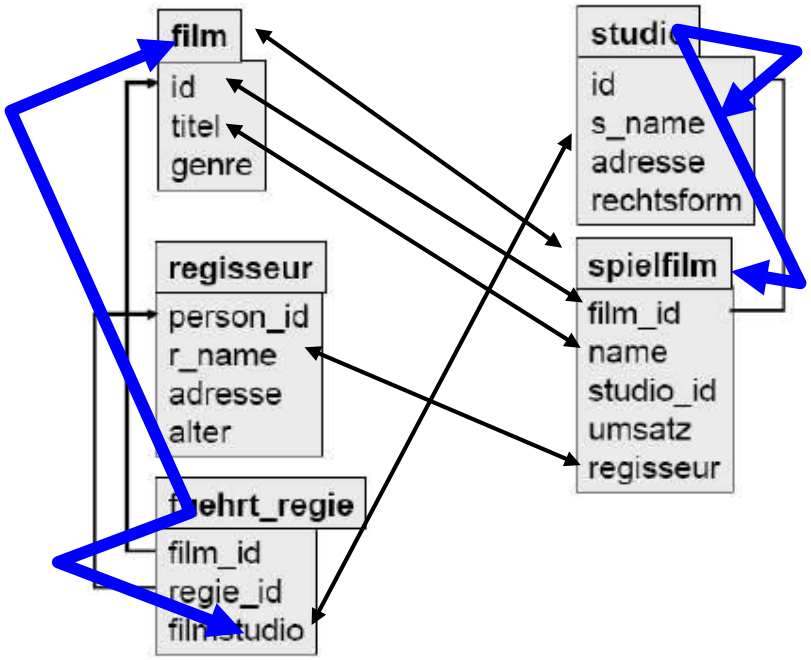
Beispiel

- Input: Eine Menge von CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**



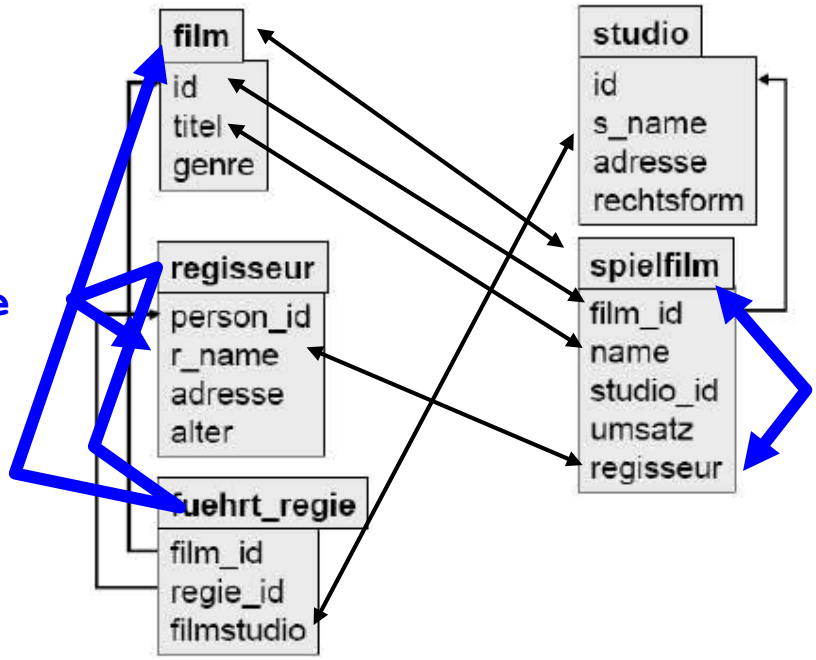
Beispiel

- Input: Eine Menge von CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**
  - **filmstudio-fuehrt_regie-film ≡ s_name-studio-spielfilm**



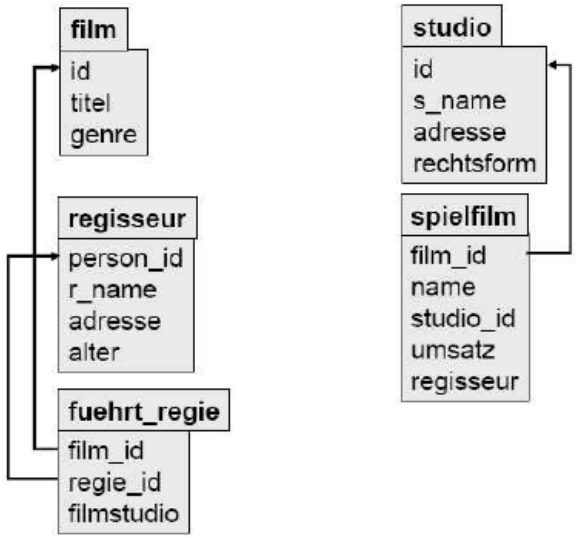
Beispiel

- Input: Eine Menge von CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**
  - **filmstudio-fuehrt_regie-film ≡ s_name-studio-spielfilm**
  - **r_name-regisseur-fuehrt_regie-film ≡ regisseur-spielfilm**



Beispiel

- CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**
  - **filmstudio-fuehrt_regie-film ≡ s_name-studio-spielfilm**
  - **r_name-regisseur-fuehrt_regie-film ≡ regisseur-spielfilm**
- Integriertes Schema
  - Regel 2: äquivalente Klassen **film** und **spielfilm** verschmelzen (zu **film**)
    - Mit der Union ihrer Attribute
  - Regel 1: Klassen ohne direktes Äquivalent werden in integriertes Schema übernommen



film
ID
titel
genre
umsatz
regisseur
studio_id

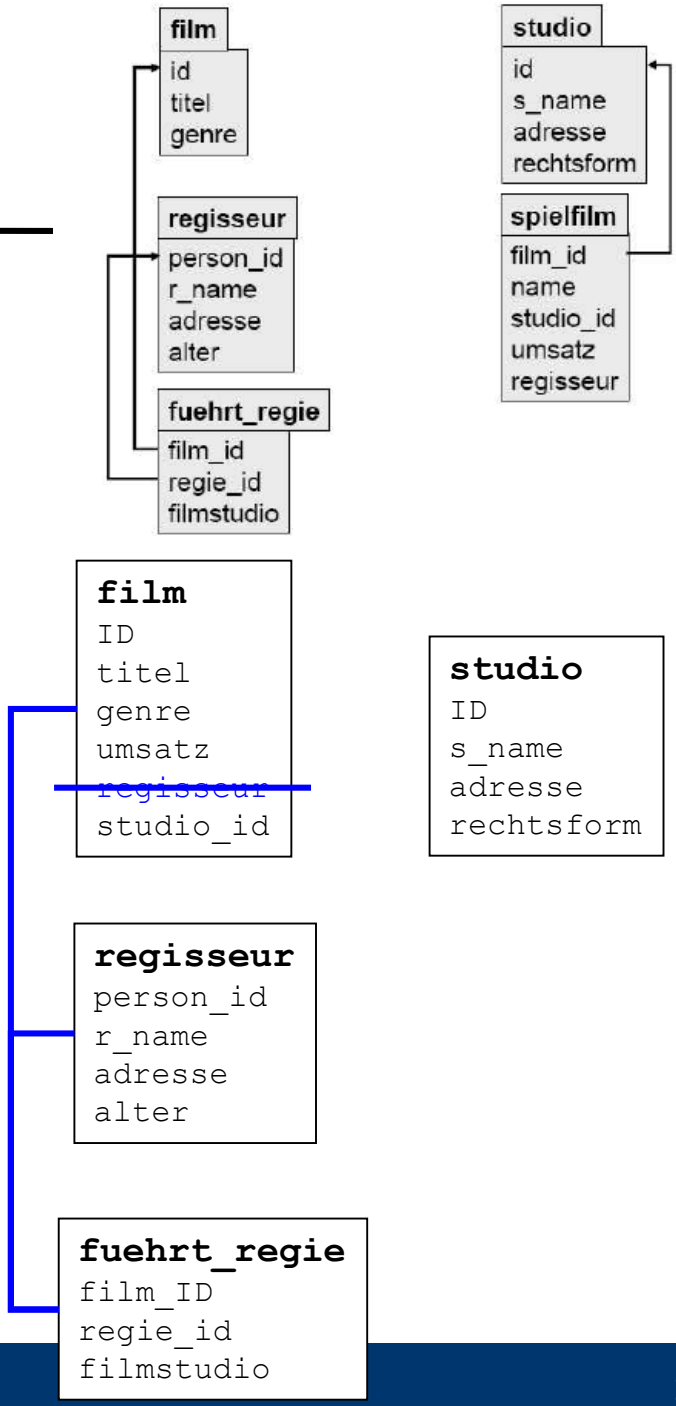
studio
ID
s_name
adresse
rechtsform

regisseur
person_id
r_name
adresse
alter

fuehrt_regie
film_ID
regie_id
filmstudio

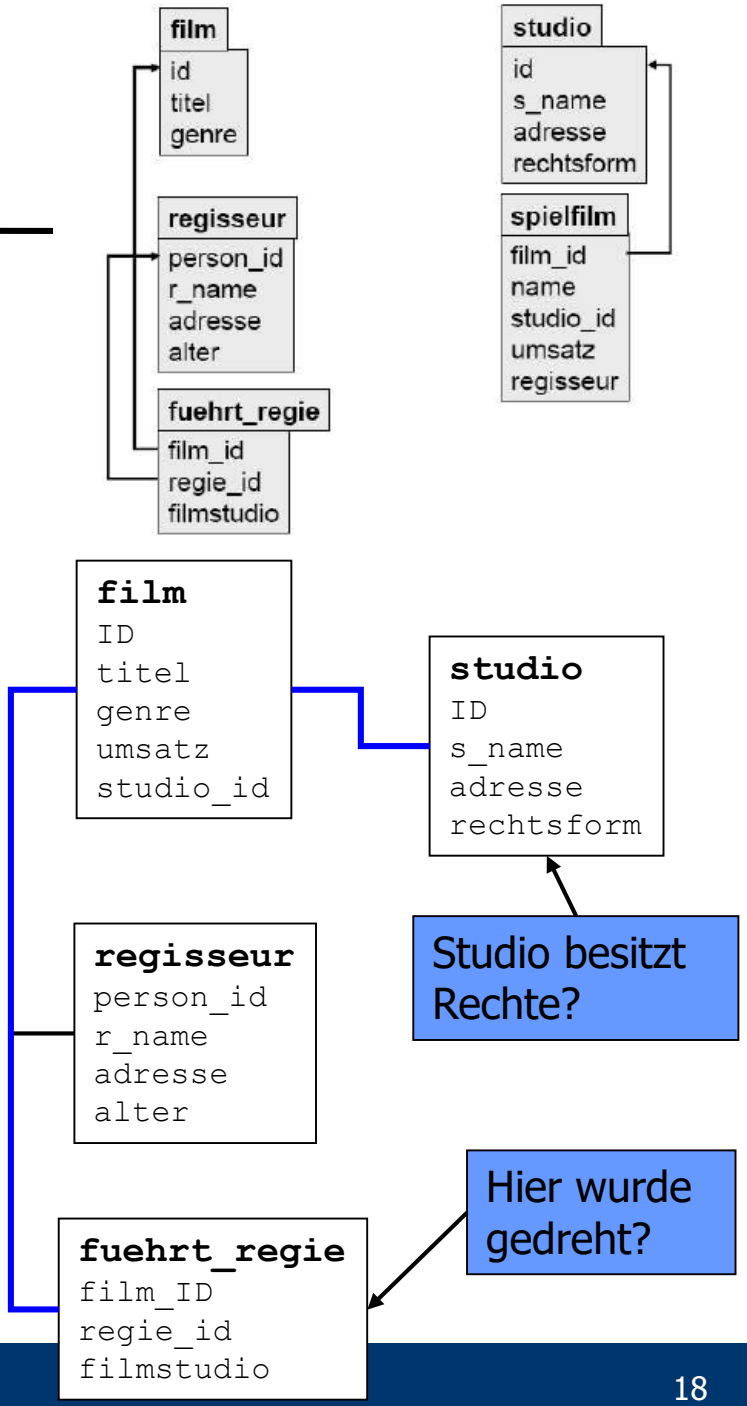
Beispiel

- CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**
  - **filmstudio-fuehrt_regie-film ≡ s_name-studio-spielfilm**
  - **r_name-regisseur-fuehrt_regie-film ≡ regisseur-spielfilm**
- Integriertes Schema
  - Regel 4a: Der Pfad **r_name-regisseur-fuehrt_regie-film** wird übernommen, der Pfad **regisseur-spielfilm** fällt weg
    - **r_name** und **regisseur** können verschmolzen werden



Beispiel

- CAs
  - **film ≡ spielfilm**
    - **id ≡ film_id** und **titel ≡ name**
  - **r_name ≡ regisseur**
  - **filmstudio ≡ s_name**
  - **filmstudio-fuehrt_regie-film ≡ s_name-studio-spielfilm**
  - **r_name-regisseur-fuehrt_regie-film ≡ regisseur-spielfilm**
- Integriertes Schema
  - Regel 4b: Beide Pfade werden übernommen
    - **filmstudio** und **s_name** können nicht verschmolzen werden



Bewertung

- Behandelt nur Äquivalenzkorrespondenzen
  - Weiterentwicklung ist nicht erfolgt
- Nur teilweise automatisiert
  - Verschmelzung von Attributen
- Generiert komplexe und große Schemata
  - „Beide Pfade werden übernommen“: Übernahme ggf. redundanter Pfade
  - Große Gefahr redundanter Elemente bei nicht sehr sorgfältig spezifizierten Korrespondenzen
- Meines Wissens nie auf komplexeren Schemata angewandt worden

Inhalt dieser Vorlesung

- Schemaintegration
- Correspondence Assertions
- Generic Integration Model

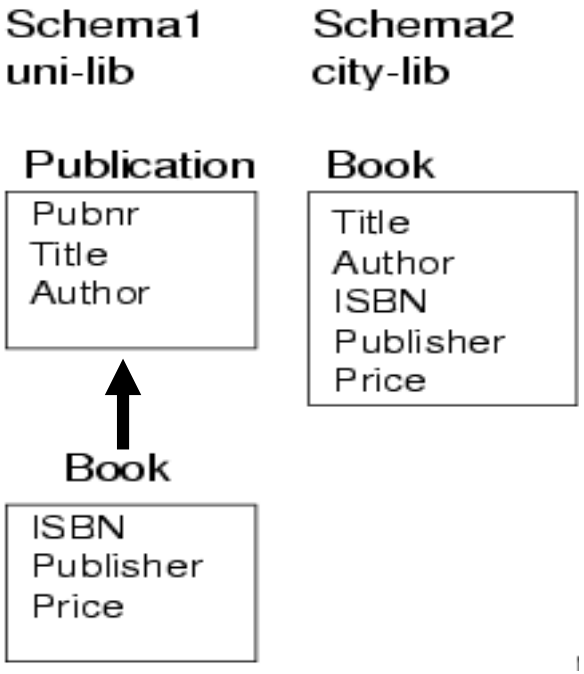
Generic Integration Model (GIM) [Sch98]

- Ziel: Integration von Spezialisierungshierarchien in objektorientierten Schemata
  - Neues Element: Vererbung zwischen Relationen / Klassen
- 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

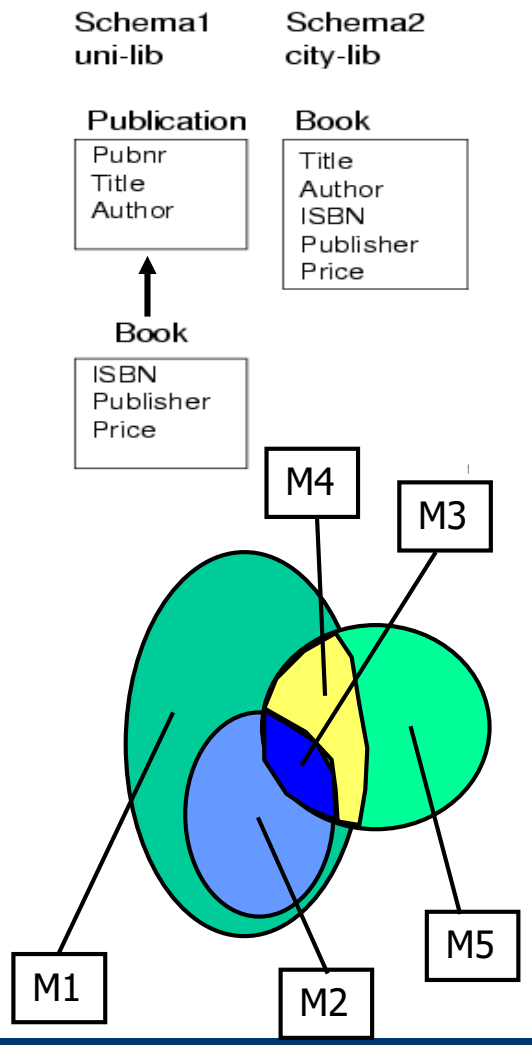
- Schritt 1: Extensionale Zerlegung
  - Finde alle möglichen extensionalen Überlappungen
    - Könnte man auf konkreter Instanz berechnen
    - Aber: Gesucht sind alle möglichen Überlappungen
    - Erfordert manuelle Inspektion (Semantik)
  - Berechne minimale Menge disjunkter Teilextensionen
    - Kleinste Menge disjunkter Teilextensionen so, dass alle realen Extensionen als deren Vereinigungen dargestellt werden können

Beispiel


Schema1
uni-lib
Schema2
city-lib
Publication
Pubnr
Title
Author
Book
Title
Author
ISBN
Publisher
Price
Book
ISBN
Publisher
Price

Extensionale Zerlegung

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



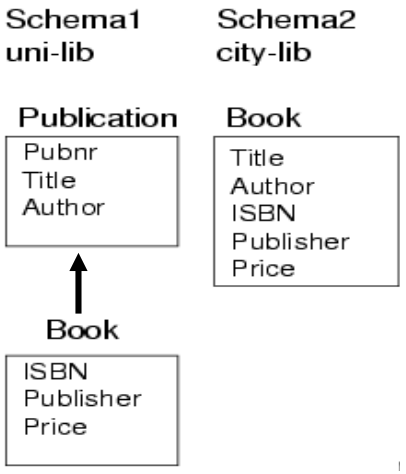
	M1	M2	M3	M4	M5
	S1.Pub, not S1.book or S2.book	not S2.book	Everything	S2.book, not S1.book, but S1.pub	Only S2.book
S1.publicati	X	X	X	X	
S1.book		X	X		
S2.book			X	X	X

Übersicht

- Schritt 2: Intensionale Zerlegung
  - Löse pro Schema syntaktische Heterogenität und Granularitätsprobleme auf
  - Attribute entsprechen sich danach 1:1 (oder gar nicht)
  - Dann: Finde alle intensionalen Überlappungen
    - Kann berechnet werden

Intensionale Zerlegung

- Wir sparen uns die Angleichung und nehmen an: Gleiche Attribute – gleiche Semantik
- Minimale intensionale Zerlegung: Drei Attributmengen notwendig



	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

Übersicht

- Schritt 3: Berechnung der GIM Matrix
  - Beide Zerlegungen kombinieren
- Schritt 4: Ableitung des integrierten Schemas
  - Nicht eindeutig: Freiheitsgrade vorhanden

Schritt 3: GIM Matrix

- Kombiniere intensionale und extensionale Zerlegungen
  - Welche Extensionen brauchen welche Intensionen?
  - Berechnung aller atomaren Kombinationen 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 Vererbung 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

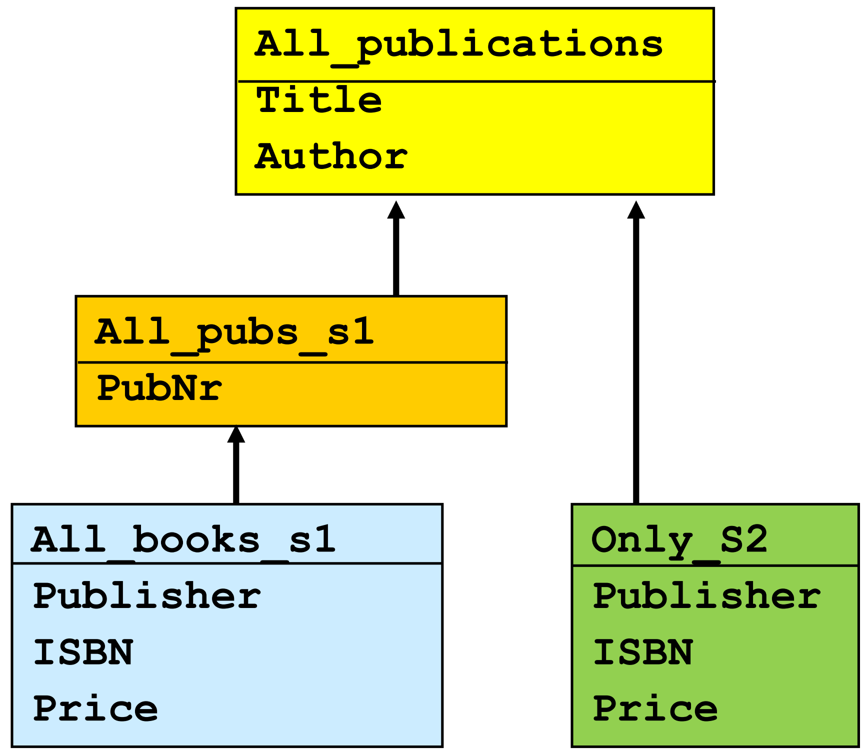
- Blöcke der Matrix ergeben Klassen
  - Mehr Zeilen und weniger Spalten– Unterklasse
  - Lücken in den Blöcken bestimmen Eltern

	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



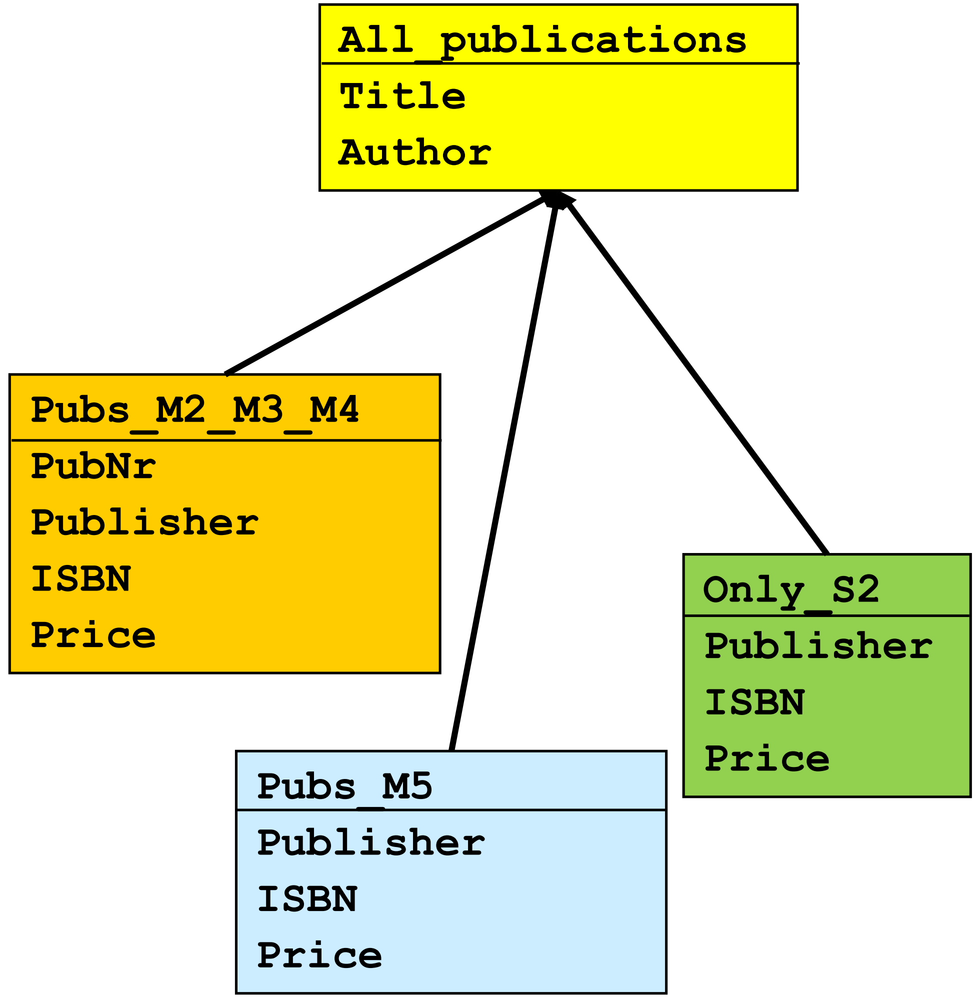
Andere Sortierung / Schemaableitung

	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	

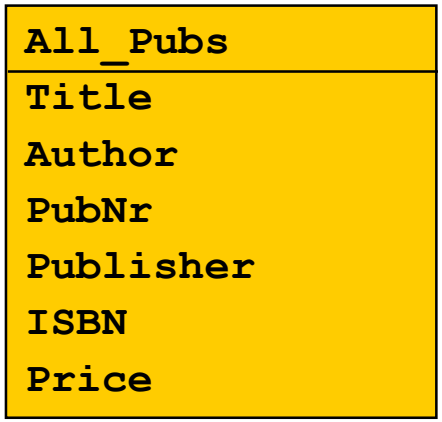
	M2	M3	M4	M5	M1
X2	X	X	X	X	X
X1	X	X	X		X
X3	X	X	X	X	

	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	



- 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
- Freiheitsgrade bei Schemaableitung
- Versucht immer, eine einzige 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