HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

VL Informationsintegration

Verteilung, Autonomie und Heterogenität

19.4.2007

Felix Naumann

Überblick

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

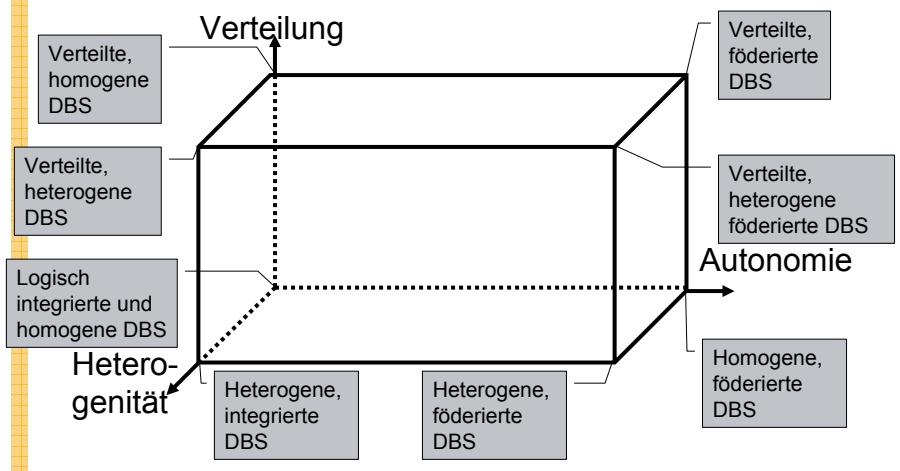
Klassifikation von Informationssystemen [ÖV99]

3

- Drei orthogonale Dimensionen
  - Verteilung
  - Autonomie
  - Heterogenität

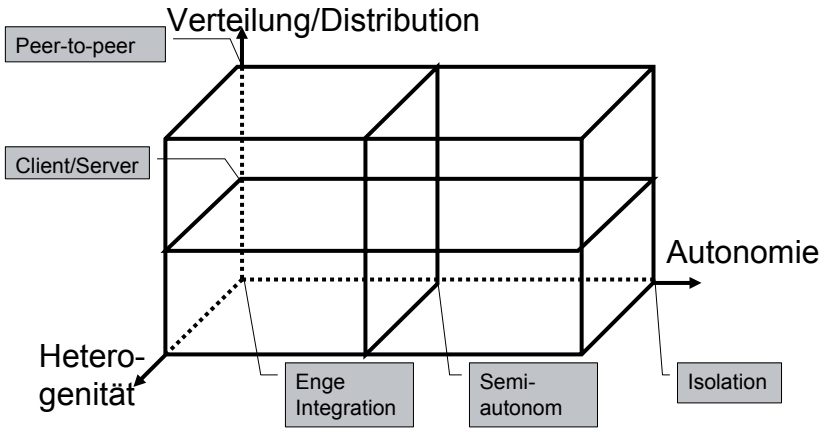
Klassifikation verteilter DBMS [ÖV91]

4



Klassifikation verteilter DBMS
nach [ÖV99]

5



Zusammenhang mit Föderierten DBMS

6

- Verteilung führt zu Autonomie,
  - Intra-Organisation: Historisch
  - Inter-Organisation: Internet & WWW
- und Autonomie führt zu Heterogenität.
  - Verantwortung liegt bei lokalen Administratoren
    - Systempflege
    - Nutzbarkeit und Nützlichkeit
    - Erweiterungen am Informationssystem
    - Design
    - ...
- Diskussion
  - Historischer Entwicklung,
  - aber orthogonale Kriterien!

Verteilung (*Distribution*)

Ein verteiltes Informationssystem ist eine Sammlung mehrerer, logisch verknüpfter Informationssysteme, die über ein gemeinsames Netzwerk verteilt sind. [ÖV91]

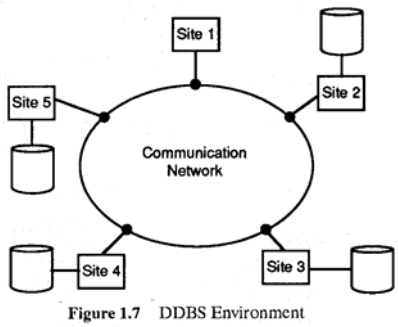


Figure 1.7 DDBS Environment

Physikalische Verteilung

- Motiviert durch Hardwareanforderungen (Hardwarebeschränkungen)
- Server stehen an unterschiedlichen Orten
  - Gleicher Raum, anderer Raum
  - Anderes Gebäude
  - Andere Stadt, anderes Land
- Shared Nothing
  - Server haben keine gemeinsamen, abhängigen Hardwarekapazitäten
    - Memory
    - Disk
    - CPU
  - Mit Ausnahme des Netzwerks
  - Im Gegensatz zu shared-disk und shared-memory

Logische Verteilung

9

- Motiviert durch Anwendungsanforderungen
  - Zuverlässigkeit
    - Bei Ausfall eines Servers
  - Verfügbarkeit
    - Bei Ausfall eines Netzwerkteils
  - Effizienz
- Redundanz
  - Replikation
  - Caching
- Partitionierung
  - Vertikal
  - Horizontal

Verteilung – Vor- und Nachteile

10

Vorteile aus Sicht der Quellen und des IIS

- Autonomie (gleich genauer)
- Performance: Kapazität dort, wo sie gebraucht wird
- Verfügbarkeit: Bei Ausfall eines Standorts
- Erweiterbarkeit
- Teilbarkeit (Verantwortung bei anderen Organisationseinheiten)

Nachteile aus Sicht des IIS

- Komplexität (Verwaltung, Optimierung)
- Kosten
- Sicherheit
- Autonomie

Verteilung – Techniken

HTTP, CORBA, ... nicht hier.

- Anwendungsentwicklung ohne Spezifikation der physikalischen Präsenz der Komponenten

Annahmen an Transparenz

- Datenunabhängigkeit (jedes DBMS)
  - auch Speicherorttransparenz
- Netzwerktransparenz
- Replikationstransparenz
- Fragmentationstransparenz
  - auch Partitionierungstransparenz

Überblick

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Autonomie (*Autonomy*)

13

Der Grad zu dem verschiedene DBMS unabhängig operieren können.

Bezieht sich auf Kontrolle, nicht auf Daten.

Klassen nach [ÖV99]

- Design-Autonomie
- Kommunikations-Autonomie
- Ausführungs-Autonomie

Design-Autonomie

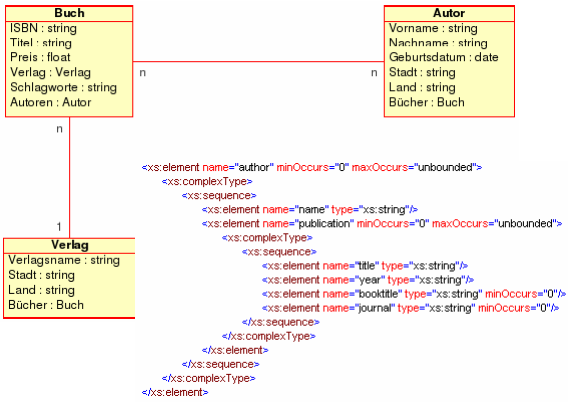
14

- Auch: Entwurfsautonomie
- Freiheit des lokalen DBMS bezüglich
  - Datenmodell
    - Relational, hierarchisch, XML
  - Schema
    - Abdeckung der Domäne (*universe of discourse, miniworld*)
    - Grad der Normalisierung
    - Benennung
  - Transaktionsmanagement
    - Sperrprotokolle
- Freiheit dies jederzeit zu ändern.
  - Besonders problematisch!

Design-Autonomie – Beispiel

15

- Schema und Datenmodell 1
  - (Fast) relational
  - Flach
- Schema und Datenmodell 2
  - XML
  - hierarchisch




```
<xs:element name="author" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="publication" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="title" type="xs:string"/>
                        <xs:element name="year" type="xs:string"/>
                        <xs:element name="booktitle" type="xs:string" minOccurs="0"/>
                        <xs:element name="journal" type="xs:string" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```


Kommunikations-Autonomie

16

- DBMS frei bezüglich
  - Wahl mit welchen Systemen kommuniziert wird
  - Wahl wann mit anderen Systemen kommuniziert wird
    - Jederzeit Eintritt/Austritt aus integriertem System
  - Wahl was (welcher Teil der Information) kommuniziert wird
  - Wahl wie mit anderen Systemen kommuniziert wird
    - Anfragesprache
  - Wahl welcher Teil der Anfragemöglichkeiten zur Verfügung gestellt werden
    - Prädikate
    - Sortierung
    - Write
    - ...

Kommunikations-Autonomie – Beispiel

- Extrem 1: Voller SQL Zugang
  - z.B. via JDBC
  - Transaktionen
  - Optimierung
  - Lesend (und Schreibend?)
  - Schemaveränderungen?
  - Antwort als Ergebnisrelation
- Extrem 2: HTML Formular
  - Nur ein (oder mehr) Suchfelder
  - Antwort als HTML Text
  - Nur Teile der Daten (public area)

Ausführungs-Autonomie

- DBMS frei bezüglich
  - Wahl wann Anfragen ausgeführt werden
  - Wahl wie Anfragen ausgeführt werden
  - Wahl der Scheduling-Strategien
  - Wahl Optimierungs-Strategien
  - Wahl ob globale Transaktionen unterstützt werden

Ausführungs-Autonomie – Beispiel

- Optimierung und Scheduling
  - Behandlung externer vs. lokaler Anfragen
  - *Golden customers*
  - Garantierte Antwortzeiten
- Transaktionen
  - Dirty-read egal?

Autonomie → Heterogenität

- Verteilung als „Ursache“ für Autonomie
- Autonomie als Ursache für Heterogenität:
  - Autonome Systeme
  - ⇒Gestaltungsfreiheit
  - ⇒Unterschiedliche Entscheidungen
  - ⇒Heterogenität

Heterogenität (*Heterogeneity*)

Heterogenität herrscht, wenn sich zwei miteinander verbundene Informationssysteme syntaktisch, strukturell oder inhaltliche unterscheiden.

- Syntaktische Heterogenität
  - Auch: „Technische Heterogenität“
- Strukturelle Heterogenität
- Semantische Heterogenität

Heterogenitäten zu überbrücken ist
die Kernaufgabe der Informationsintegration.

Heterogenitätsklassen

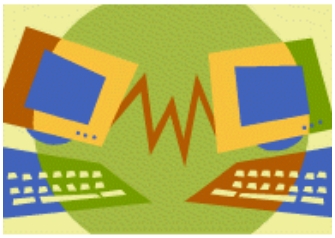
- Auch andere Klassifikationen möglich, z.B. [BKLW99]
  - Syntaktische Heterogenität
  - Datenmodell Heterogenität
  - Logische Heterogenität
- Oder nach [SPD92]
  - Semantische Konflikte
  - Beschreibungskonflikte
  - Heterogenitätskonflikte
  - Strukturelle Konflikte
  - Datenkonflikte

Überblick

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Syntaktische Heterogenität

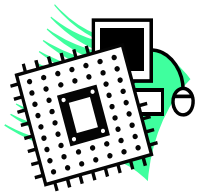
- Hardware-Heterogenität
- Software-Heterogenität
- Schnittstellen-Heterogenität



Hardware Heterogenität

25

- Bandbreite
- Hauptspeicher
- CPU
  - Art
  - Geschwindigkeit



Nicht hier

Software Heterogenität

26

- Betriebssystem
- Dateisystem
- Protokolle
  - HTTP, ODBC, Java API, CORBA, etc.
- Zustandsbehaftet vs. zustandsfrei
- Sicherheit
  - Security level
  - Log-on Prozedur

Software Heterogenität – Beispiel


```
String sqlQuery = „SELECT Name, Strasse FROM Hersteller
WHERE PLZ = 69115“;
...
Connection jdbcCon = DriverManager.getConnection(dbURL, ...);
Statement stmt = jdbcCon.createStatement();
ResultSet table = stmt.executeQuery(sqlQuery);
...
```

```
String webQuery = „plz=69115“;
...
URL url = new URL(„http://www.system2.de/cgi-bin
/search.cgi“ + „?“ + webQuery);
URLConnection urlCon = new url.openConnection();
InputStreamReader reader = new InputStreamReader(
  urlCon.openStream());
...
```

Nicht hier

Quelle: VL: Föderierte Datenbanksysteme Peter Tomczyk, FZI & Uni Karlsruhe

Schnittstellen Heterogenität

Schnittstellen von Informationssystemen sind im wesentlichen deren Anfragensprache:

- HTML Formular,
- „Google“-Sprache (+, - , ...),
- SQL,
- XQuery,
- etc.

Jetzt hier!

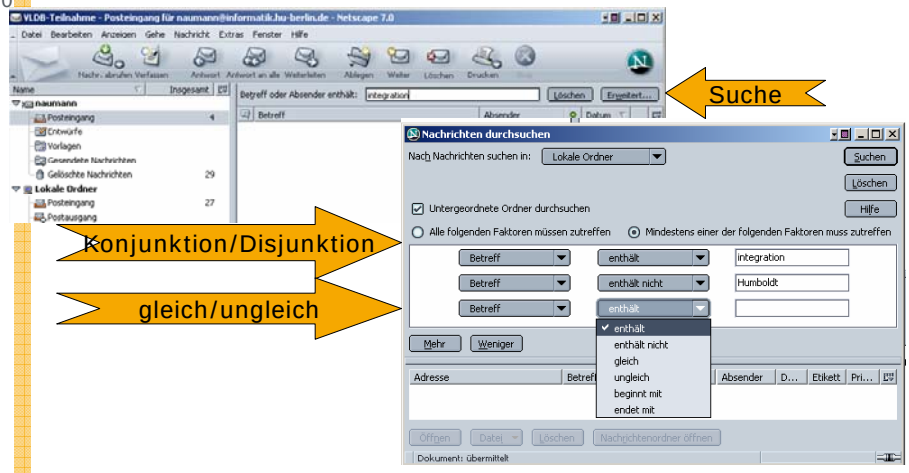
Schnittstellen Heterogenität

29

- Negation vs. keine Negation
  - Oft zu teuer
- Gleichheit / Ungleichheit
  - „=“ oder auch „>,<, ≥, ≤“
- Konjunktion (UND)
  - oder auch Disjunktion (ODER)
- Prädikate nur mit Konstanten (author = „Melville“)
  - Oder auch mit anderen Variablen (ResidenceCountry = Nationality)
- Gebundene und freie Variablen [RSU95,LC00,YLGU99]
  - später
- Andere Einschränkungen
  - Joins über maximal 3 Relationen
  - z.B. Prädikate nur über eine Auswahl von Werten

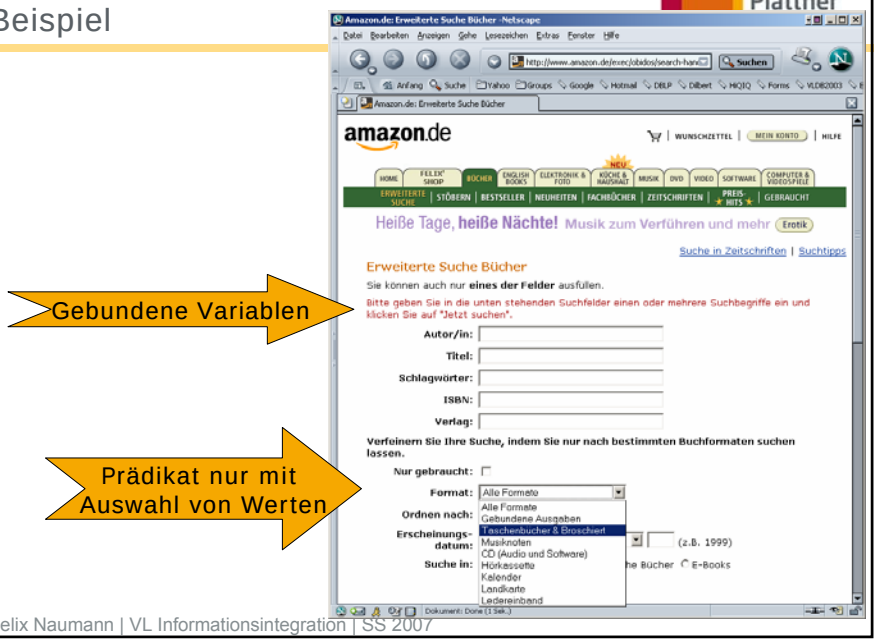
Schnittstellen Heterogenität - Beispiel

30



Schnittstellen-Heterogenität – Beispiel

31



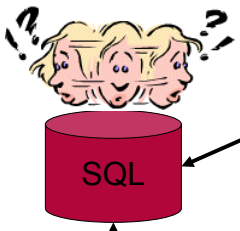
Schnittstellen Heterogenität

32

- In einzelnen Systemen kein Problem
- Probleme für integrierte Systeme
  1. Globale Anfragesprache ist mächtiger als lokale Anfragesprache
     - Anfragen eventuell nicht ausführbar
     - Oder globales System muss kompensieren
  2. Lokale Anfragesprache ist mächtiger als globale Anfragesprache
     - Verpasste Chance, lokale (effiziente) Ausführung auszunutzen
  3. Gebundene und freie Variablen sind inkompatibel
     - Anfragen eventuell nicht ausführbar

Mächtige globale Anfragesprache

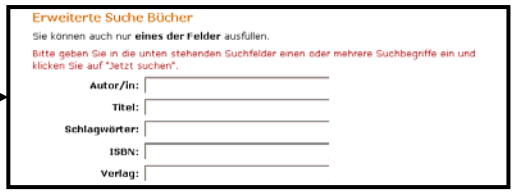
33



```
SELECT      * FROM Books
WHERE       Author = „Defoe“
AND         PubYear = 1979
```

SQL

HTML Form



Mächtige globale Anfragesprache

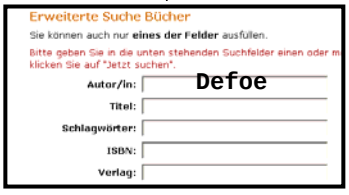
34

```
SELECT * FROM Books
WHERE  Author = „Defoe“
AND    PubYear = 1979
```

```
Daniel Defoe, Robinson Crusoe, 1979
```

PubYear = 1979

```
Daniel Defoe, Robinson Crusoe, 1986
Daniel Defoe, Robinson Crusoe, 1979
Daniel Defoe, Moll Flanders,   1933
```



Mächtige globale Anfragesprache

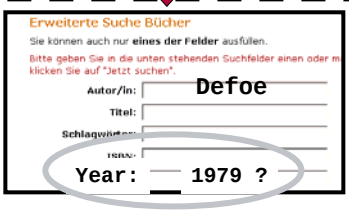
35

```
SELECT * FROM Books
WHERE  Author = „Defoe“
AND    PubYear > 1979
```

```
Daniel Defoe, Robinson Crusoe, 1979
Daniel Defoe, Robinson Crusoe, 1986
```

PubYear > 1979

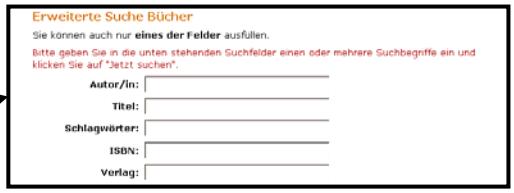
```
Daniel Defoe, Robinson Crusoe, 1986
Daniel Defoe, Robinson Crusoe, 1979
Daniel Defoe, Moll Flanders,   1933
```



Mächtige lokale Anfragesprache

36

HTML Form



SQL

```
SELECT     * FROM Books
WHERE      Author = „Defoe“
```

Verpasste Chancen.

Gebundene & Freie Variablen

37

- **Gebundene Variablen** müssen bei einer Anfrage gebunden werden.
  - z.B.: „Search“-Feld bei Google
- **Freie Variablen** müssen nicht gebunden werden.
  - z.B. „Autor“-Feld bei Amazon.de, falls Titel gebunden ist.

Gebundene & Freie Variablen – Beispiel & Ausblick

38

SONGS	Song	CD
	Friends	Life
	Friends	Love

CDs	CD	Künstler	Preis
	Love	Lucy	15
	Story	Snoopy	14

Künstler	CD	Künstler	Preis
	Story	Lucy	13
	Love	Snoopy	10
	Life	Charlie	8

Bastelaufgabe 1:
Wie teuer ist die billigste CD mit einem Song namens “Friends”?

Quelle: [LC00]

Gebundene & Freie Variablen – Beispiel & Ausblick

SONGS	Song	CD
	Friends	Life
	Friends	Love

CDs	CD	Künstler	Preis
	Love	Lucy	15
	Story	Snoopy	14

Künstler	CD	Künstler	Preis
	Story	Lucy	13
	Love	Snoopy	10
	Life	Charlie	8

Unterstrichen = gebundene Variable

Bastelaufgabe 2:
Welches ist die billigste CD mit einem Song namens “Friends”, *die Sie anfragen können*?

Mehr später…

Syntaktische Heterogenität - Zusammenfassung

- Hardware Heterogenität
  - Bandbreite, CPU, …
- Software Heterogenität
  - Protokolle, Sicherheit, …
- Schnittstellen Heterogenität
  - Mächtigkeit der Anfragesprachen
  - Gebundene & freie Variablen

Überblick

41

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Strukturelle Heterogenität

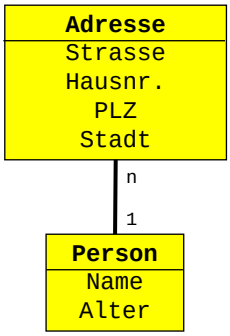
42

- Datenmodell-Heterogenität
  - Unterschiedliche Semantik
  - Unterschiedliche Struktur
- Schematische Heterogenität
  - Integritätsbedingungen, Schlüssel, Fremdschlüssel, etc.
  - Schema (Attribut vs. Relation etc.)
  - Struktur (Gruppierung in Tabellen)

Datenmodell-Heterogenität

Datenmodelle

- Relationales Modell
- XML Modell
- OO Modell
- Hierarchisches Modell




```
Adresse(PersonId, Strasse,
   Hausnr., PLZ, Stadt)

Person( Id, Name, Alter)
```


Schematische Heterogenität

Sprachregelung hier:

Schemas statt Schemata

Alt-Griechisch wird ignoriert...

Schematische Heterogenität

45

- Struktur
  - Modellierung
    - Relation vs. Attribut
    - Attribut vs. Wert
    - Relation vs. Wert
  - Benennung
    - Relationen
    - Attribute
    - Jeweils Homonyme und Synonyme
  - Normalisiert vs. Denormalisiert
  - Geschachtelt vs. Fremdschlüssel
- Diese Probleme sogar bei gleichem Datenmodell!

Schematische Heterogenität

46

Männer(Id, Vorname, Nachname)
Frauen(Id, Vorname, Nachname)

Relation vs. Attribut

Relation vs. Wert

Person(Id, Vorname, Nachname,männlich, weiblich)

Person(Id, Vorname, Nachname, Geschlecht)

Attribut vs. Wert

Schematische Heterogenität

47

Tabellen-Tabellen Konflikte

- Namenskonflikte
  - Semantisch gleiche Tabellen mit verschiedenen Namen (Synonym)
  - Verschiedene Tabellen mit gleichem Namen (Homonym)
- Strukturkonflikte
  - fehlende Attribute
  - fehlende, aber ableitbare Attribute
- IC-Konflikte (Integrity-Constraint = Integritätsbedingungen)

Schematische Heterogenität – Beispiel

48

mitarbeiter

p_id	vorname	nachname	funktion
1	Peter	Müller	Sachbearb.
5	Petra	Weger	Sekr.
...	...	...	...

mitarbeiter (leitend)

p_id	vorname	name
2	Stefanie	Meier
2	Petra	Weger
2	Andreas	Zwickel
...	...	...

Homonym

Fehlendes (ableitbares) Attribut

IC Konflikt (Eindeutigkeit)

Schematische Heterogenität

49

Attribut-Attribut Konflikte

- Namenskonflikte
  - Verschiedene Namen für gleiche Attribute (Synonyme)
  - Gleiche Namen für verschiedene Attribute (Homonyme)
- Default-Wert-Konflikte
- IC-Konflikte
  - Datentypkonflikte
  - Bedingungskonflikte

Schematische Heterogenität – Beispiel

50

mitarbeiter

p_id	Vorname VARCHAR(35)	nachname	alter
1	Wolfgang	Meyer	33
5	Klaus	Schmidt	NULL
...	...	...	..

IC: alter > 18

Synonym

mitarbeiter

p_id	Vorname VARCHAR(20)	name	alter
1	Peter	Müller	0
5	Petra	Weger	17
...	...	...	...

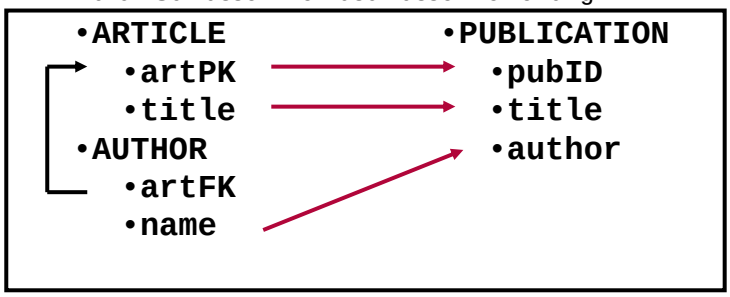
Default Werte

Datentypkonflikt

Schematische Heterogenität - Beispiel

Normalisiert vs. Denormalisiert

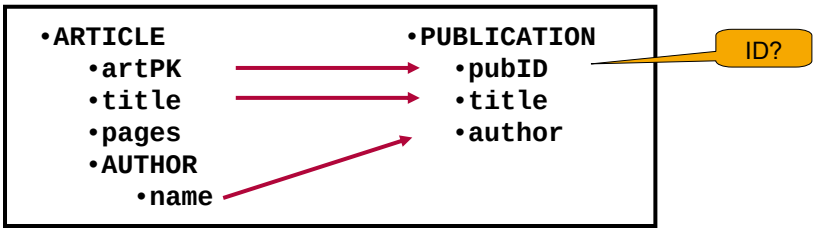
- 1:1 Assoziationen zwischen Werten wird unterschiedlich dargestellt
  - Durch Vorkommen im gleichen Tupel
  - Durch Schlüssel-Fremdschlüssel Beziehung



Schematische Heterogenität - Beispiel

Geschachtelt vs. Flach

- 1:n Assoziationen werden unterschiedlich dargestellt
  - Als geschachtelte Elemente
  - Als Schlüssel-Fremdschlüssel Beziehung



Schematische Heterogenität - Lösungen

- Problem
  - Einheitlich auf beide Schemas zugreifen
    - Auf Schemaebene: Schema Mapping und Schema-Sprachen
    - Auf Datenebene: Virtuelle Integration
  - Beide Schemas in eine gemeinsames neues Schema integrieren
    - Auf Schemaebene: Schemaintegration
    - Auf Datenebene: Materialisierte Integration
- Für die materialisierte Integration
  - Schemaintegration
  - ETL
- Für die virtuelle Integration
  - Schema-Sprachen
    - Z.B. SchemaSQL, MSQL, CPL
  - Schema Mapping
    - Z.B. Clio, RONDO, u.a.

Schematische Heterogenität – Lösungen (Ausblick)

SchemaSQL [LSS96]

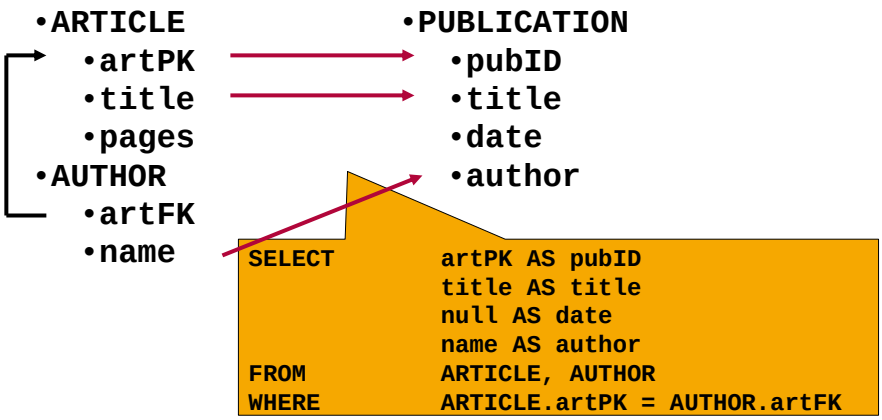
- Erweiterung von SQL
- Daten und Metadaten werden gleich behandelt
- Umstrukturierungen innerhalb der Anfrage
- Dynamische Sicht-Definition
- Horizontale Aggregation

```
SELECT RelA
FROM   uniA->RelA, uniA::RelA A, uniB::grundgehalt B
WHERE  RelA = B.institut
AND    A.Kategorie = „Student“
AND    A.grundgehalt > B.Student
```

High-order Join

Schematische Heterogenität – Lösungen (Ausblick)

Schema Mapping



Zusammenfassung

- Verteilung
- Autonomie
  - Design-Autonomie
  - Kommunikations-Autonomie
  - Ausführungs-Autonomie
- Heterogenität
  - Syntaktische Heterogenität
    - Hardware Heterogenität
    - Software Heterogenität
    - Schnittstellen Heterogenität
  - Strukturelle Heterogenität
    - Datenmodell-Heterogenität
    - Schematische Heterogenität

Überblick

57

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Semantik

58

Fremdwörterduden “Semantik”:

- Teilgebiet der Linguistik, das sich mit den Bedeutungen sprachlicher Zeichen und Zeichenfolgen befasst
- Bedeutung, Inhalt eines Wortes, Satzes oder Textes

„Semantische Heterogenität ist ein überladener Begriff ohne klare Definition. Er bezeichnet die Unterschiede in Bedeutung, Interpretation und Art der Nutzung.“ [ÖV91]

Semantik vs. Struktur

59

Strukturelle Heterogenität

- Betrifft Schemas
- Bedeutung der Labels im Schema egal
- Annahme bisher: Gleiche Label -> Gleiche Semantik

Männer(Id, Vorname, Nachname) **Frauen(Id, Vorname, Nachname)**	**A(Id, X, Y)** **B(Id, X, Y)**
Person(Id, Vorname, Nachname, Männlich, weiblich)	**P(Id, X, Y, a, b)**

Semantische Heterogenität

- Betrifft Daten
- Betrifft „Bedeutung“

Unterschiedliche Namen

60

- Die Probleme
  - Konzept (z.B. Gen)
    - Definition des Konzepts
  - Synonyme (z.B. surname vs. last name)
  - Homonyme (z.B. biweekly)
  - Einheiten (z.B. cm vs. inch)
  - Werte (z.B. „manager“)
- Eher auf Schema Ebene

Konzept

61

- Definition eines Konzepts
  - Noch nicht einmal hier sind sich immer alle einig.
  - Gen, Transaktion, Bestellung, Mitarbeiter
- Semantisch überlappende Weltausschnitte mit einander entsprechenden Klassen
- Korrespondenzarten zwischen Klassenextensionen:
  - A=B Äquivalenz
  - A⊆B Inklusion
  - A∩B Überlappung
  - A≠B Disjunktion

Konzept

62

„Wie viele Mitarbeiter hat IBM?"

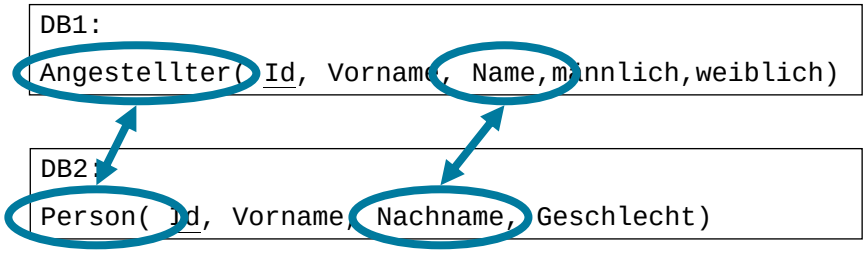
- Definition Mitarbeiter:
  - temporäre MA
  - Diplomanden
  - Berater
  - Studentische Mitarbeiter
  - Stellen oder Köpfe?
- Definition IBM
  - Welche Region? Welcher Geschäftsbereich?
  - Informix?
  - PWC?
- Welcher Zeitpunkt?
- Definition der Zählung:
  - Doppelte Zählung bei mehreren Anstellungen?

„Wieviel Hardware haben wir ans HPI verkauft?"

Synonyme

63

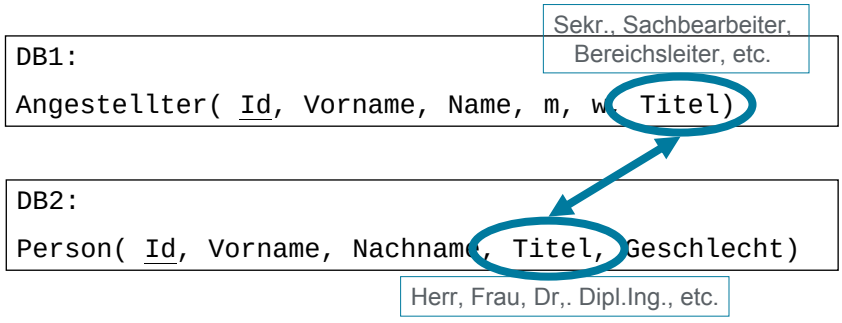
- Verschiedene Worte mit gleicher Bedeutung
- Im Kontext der zu integrierenden Datenbanken



Homonyme

64

- Gleiche Worte verschiedener Bedeutung
- Andere Domäne
- Andere Bedeutung



-nym Wörter

65

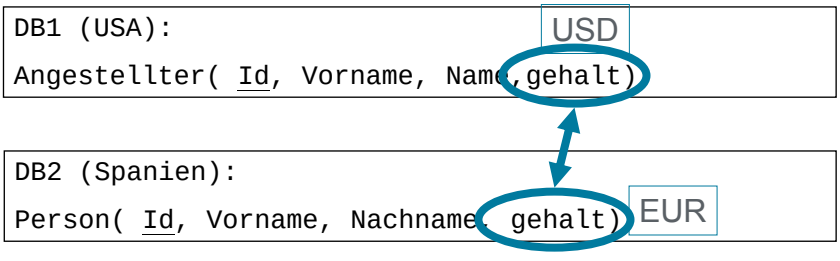
- Synonym
  - Verschiedene Wörter, gleiche Semantik
- Homonym
  - Gleiche Wörter, verschiedene Semantik
- Antonym
  - Verschiedene Wörter, gegenteilige Semantik
- Auto-Antonym:
  - Gleiche Wörter, gegenteilige Semantik
  - Transparenz
  - Overlook
- Heteronym
  - Gleiche Schreibung, verschiedene Aussprache, verschiedene Semantik
- Autonym (selbstbeschreibend, Wort = Semantik, „Substantiv“)
- Pseudonym u.v.a.m.

- http://www.fun-with-words.com/nym_words.html

Einheiten

66

- Gleiche „Bedeutung“ aber anderes Maß.
- Werden auch als Homonym bezeichnet, da anderes Maß eine andere Bedeutung erzeugt.



Überblick

67

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Identität

68

- Drei zentrale Fragen
  - Was ist ein Objekt?
    - XML: Über mehrere Schachtelungsebenen hinweg
    - Relationales Modell: Über mehrere Relationen hinweg
  - Repräsentiert Objekt A die gleiche Entität wie Objekt B?
  - Wie finde ich effizient gleiche Repräsentationen?
- Namen des Problems
  - Duplikaterkennung
  - Objektidentifikation
  - Record Linkage
  - Data Cleansing
  - …
- Auf Datenebene

Typische Anwendungen

69

- Personen- und Adressdaten
  - Volkszählungen
  - Werbeaktionen
  - Kundenpflege
- Molekularbiologische Daten
- Bibliographische Daten
  - Zentrale Register
- Typische Merkmale zur Entstehung:
  - Gleiches Objekt mehrfach beobachtet
  - Manuelle Erfassung der Daten
  - Objekt ändert Eigenschaften von Zeit zu Zeit
  - Keine global konsistente ID
    - ISBN, IBAN, URL, ISO, EAN, SSN, etc.

Duplikaterkennung

70

- Duplikate in Relationen
  - Zwei Tupel, die das gleiche real-world Objekt repräsentieren
  - Semantik!
  - Attributwerte dürfen sich unterscheiden.
- Formales Problem
  - Eine Tabelle (der Größe N), potentiell mit Duplikaten
  - Erzeuge für jedes Tupel einen Identifier, so dass Duplikate gleiche Identifier erhalten
- Problemerweiterungen
  - Zwei Tabellen mit unterschiedlichem Schema
  - Ein XML Dokument mit Duplikaten

Duplikaterkennung

71

- Praktisches Problem
  - Wie entscheide ich, ob zwei Tupel das gleiche Objekt repräsentieren?
  - Ähnlichkeitsmaße und Klassifikation
    - Edit-Distance
    - N-grams
    - IDs
    - Wahrscheinlichkeitstheoretische Ansätze
    - Maschinelles Lernen
    - Augenschein

Duplikaterkennung

72

- Praktisches Problem
  - Sehr große Datenmenge
    - Millionen Tupel
  - Kein quadratischer Algorithmus
  - Kein Hauptspeicher-Algorithmus
- Als SQL Anfrage
  - Sei R die Relation mit Duplikaten
  - SELECT C1.*, genID(C1,C2)
    FROM R as C1, R as C2
    WHERE M(C1,C2)

- Schwieriger als normaler Join
  - Ähnlichkeitsmaß ist nicht nur Gleichheit
- Algorithmen zur Objektidentifikation in VL „Duplikaterkennung“

Überblick

73

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

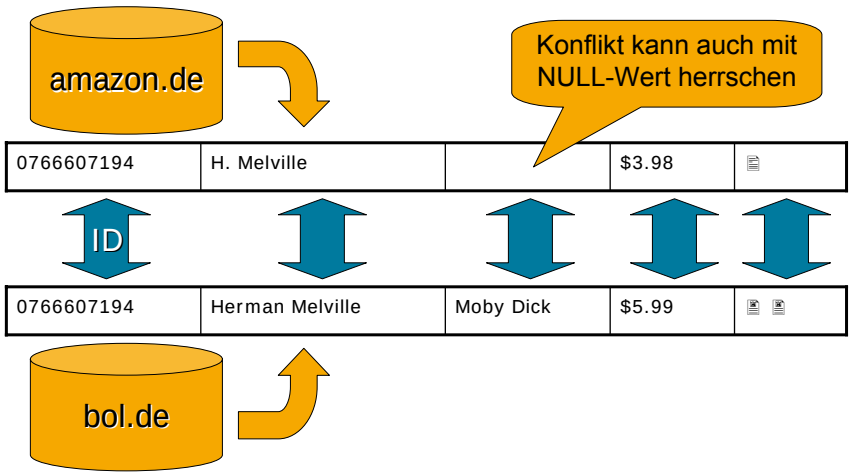
Datenkonflikte

74

- Datenkonflikt:
  - Zwei Duplikate haben unterschiedliche Attributwerte für ein semantisch gleiches Attribut.
  - Im Gegensatz zu Konflikten mit Integritätsbedingungen
- Datenkonflikte entstehen
  - innerhalb eines Informationssystems (intra-source) und
  - bei der Integration mehrerer Informationssysteme (inter-source).
- Voraussetzung:
  - Duplikat!
  - d.h. Identität schon festgestellt.

Datenkonflikte - Beispiel

75



Datenkonflikte – Entstehung

76

Innerhalb eines Informationssystems

- Mangels Integritätsbedingungen oder Konsistenz-Checks
- Bei redundanten Schemata
- Bei Entstehung von Duplikaten
- Nicht korrekte Einträge
  - Tippfehler, Übertragungsfehler
  - Falsche Rechenergebnisse
- obsolete Einträge
  - div. Aktualisierungszeitpunkte
    - ausreichende Aktualität einer Quelle
    - verzögerte Aktualisierung
  - vergessene Aktualisierung

Datenkonflikte – Entstehung

77

Innerhalb eines Informationssystems

- bei div. Datentypen (mit/ohne Codierung)
  - 1,2,...,5 bzw. "sehr gut", "gut", ..., mangelhaft"
- bei gleichem Datentyp
  - Schreibvarianten
    - Kantstr. Kantstrasse Kant Str. Kant Strasse
    - Kolmogorov Kolmogoroff Kolmogorow
  - Typische Verwechslungen U<->V, 0<->o, usw. (OCR)

Datenkonflikte – Behebung

78

- Referenztabellen für exakte Wertabbildung
  - Z.B. Städte, Länder, Produktnamen, Codes...
- Ähnlichkeitsmaße
  - bei Tippfehlern
  - bei Sprachvarianten (Meier, Mayer,...)
- Standardisieren und transformieren
- Nutzung von Hintergrundwissen (Metadaten)
  - bzgl. von Konventionen (landestypische Schreibweisen)
  - Ontologien zur Behandlung von Zusammenhängen
  - Thesauri, Wörterbücher zur Behandlung von Homonymen, Synonymen, ...

Datenkonflikte – Entstehung

79

Bei der Integration von Informationssystemen

- Lokal konsistent aber global inkonsistent
- Duplikate (extensionale Redundanz)
- Andere Datentypen
- Lokale Schreibweisen/Konventionen

Datenkonflikte – Behebung

80

- Präferenzordnung über Datenquellen
  - nach Aktualität, Trust (Vertrauen), Öffnungszeiten usw.
- Informationsqualität
- Konfliktlösungsfunktionen

- Wie implementieren?

Relationale Objektintegration

81

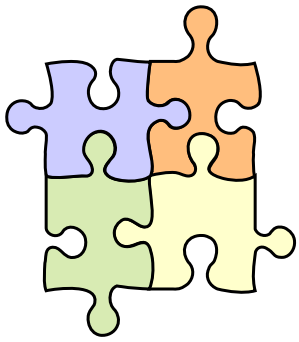
Union (Vereinigung)
- Duplikat-Eliminierung

Minimum Union
- Eliminierung sub-summierter Tupel

...

Aber keine
- Duplikatintegration
- Konfliktlösung

Mehr dazu in VL „Datenfusion“



Überblick

82

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Gebundene & Freie Variablen

Gebundene Variablen müssen bei einer Anfrage spezifiziert werden.

- z.B.: „Search“-Feld bei Google

Freie Variablen müssen nicht gebunden werden.

- z.B. „Autor“-Feld bei Amazon.de, falls Titel gebunden ist.

Einordnung:

- Heterogenität
  - Syntaktische Heterogenität
    - Schnittstellenheterogenität

Gebundene und Freie Variablen – Adornments

- Jede Quelle exportiert eine oder mehrere relationale Sichten.
- IIS erlaubt Anfragen auf diese Sichten mittels Join, Union, Selektion und Projektion.

Quelle: [YLGU99]

Gebundene und Freie Variablen – Adornments

Z gebunden

Beispiel Quelle 1:	Beispiel Anfrage 1:	Beispiel Anfrage 2:
$R_1(X,Y,Z)$	$Q_1(X,Y,z_1)$	$Q_2(X,y_1,Z)$
Daten:	Beispiel Ergebnis:	Beispiel Ergebnis:
(x_1,y_1,z_1)	(x_1,y_1,z_1)	(x_1,y_1,z_1)
(x_1,y_2,z_1)	(x_1,y_2,z_1)	
(x_2,y_2,z_2)		

5 Quellen (für später):

$R_1(X,Y,Z)$
$R_2(X,Y,Z)$
$R_3(X,Y,Z)$
$R_4(Z,U)$
$R_5(U,V,W)$

Quelle: [YLGU99]

Gebundene und Freie Variablen – Adornments

- Anfragefähigkeiten der Quellen als templates
  - Wie ein WWW Formular
  - Templates bestehen aus einem adornment für jedes Attribut
- Anhänge (adornments = Verzierungen) an Attribute schränken ein:
  - f: free
    - Frei: Kann in Anfrage spezifiziert werden, muss aber nicht.
  - u: unspecifiable
    - Unbestimmbar: Kann nicht spezifiziert werden.
    - Ist aber Teil des Ergebnisses.
  - b: bound
    - Gebunden: Muss spezifiziert werden.
  - c[s]: constant
    - Auswahl aus einer Menge s von Konstanten
    - Implizit bound: muss spezifiziert werden
  - o[s]: optional
    - Auswahl aus einer Menge s von Konstanten
    - Implizit free: Muss nicht spezifiziert werden.

Quelle: [YLGU99]

Adornments - Beispiele

Beispiel Quelle 1:
$R_1(X,Y,Z)$

Anfragemöglichkeit 1:
X muss spezifiziert werden
Y kann nicht spezifiziert werden
Z kann spezifiziert werden

Template:
buf

Anfragemöglichkeit 2:
X kann nicht spezifiziert werden
Y kann spezifiziert werden
Z ist entweder z_1 oder z_2

Template:
ufc[z_1, z_2]

Adornments – Anfragebearbeitung

Anfragebearbeitung

- R1(X,Y,Z): bff, ffb
- R2(X,Y,Z): fbf
- Sei M = R1 ∪ R2 eine integrierte Sicht des IIS, gegen die man Anfragen stellen kann.
- Annahme über Anfragebearbeitung:
  - Anfragen werden übersetzt in je eine Anfrage pro Quelle (gebundene Variablen werden weitergereicht)
  - Ergebnisse werden entsprechen der Sicht verknüpft (hier ∪)
- Frage: Was ist das Template der Sicht M?

```
  bff
∪ fbf
=
```

```
  ffb
∪ fbf
=
```

Quelle: [YLGU99]

Adornments – Verknüpfung durch UNION

3 Sichten und deren Adornments:
$R_1(X,Y,Z)$: bff, ffb
$R_2(X,Y,Z)$: fbf
$R_3(X,Y,Z)$: ffc[s_1], c[s_2]ff

$R_1 \cup R_2$:
bff $\cup$ fbf = bbf
ffb $\cup$ fbf = fbb
$(R_1 \cup R_2) \cup R_3$:
bbf $\cup$ ffc[s_1] = bbc[s_1] usw.

	f	**o**[s_3]	**b**	**c**[s_4]	**u**
f	f	o[s_3]	b	c[s_4]	u
o[s_1]	o[s_1]	o[$s_1 \cap s_3$]	c[s_1]	c[$s_1 \cap s_4$]	u
b	b	c[s_3]	b	c[s_4]	-
c[s_2]	c[s_2]	c[$s_2 \cap s_3$]	c[s_2]	c[$s_2 \cap s_4$]	-
u	u	u	-	-	u

Quelle: [YLGU99]

Adornments – Verknüpfung durch Join ($\bowtie$)

- Unterschied zu UNION
  - Nicht jedes Attribut der integrierten Sicht ist auch Attribut jeder beteiligten Quelle.
  - Beispiel: R1(X,Y,Z) und R4(Z,U)
  - Sicht: M(X,Y,Z,U) = R1(X,Y,Z) $\bowtie$ R4(Z,U)
- Berechnung des Templates der Sicht
  - Adornments der nicht-Join-Attribute werden kopiert.
  - Adornments der Join-Attribute werden gemäß der UNION Tabelle vereint.

Adornments – Selektion und Projektion

- Selektion
  - Sicht im IIS selektiert mit Prädikaten.
    - X = `Test´ oder U > 1999
  - Prädikate werden auf Ergebnisse der Quellen angewandt.
  - Deshalb: Kein Einfluss auf adornments
- Projektion
  - Einfach projizierte Attribute weglassen.
  - Aber: Falls Attribut mit b oder c adornment durch Projektion wegfallen soll => Sicht des IIS nicht ausführbar
  - Sonst: Adornments bleiben erhalten

Adornments – Anfragebearbeitung

Problem

- UNION-Matrix zu restriktiv

	f	**o**$[s_3]$	**b**	**c**$[s_4]$	**u**
f	f	o$[s_3]$	b	c$[s_4]$	u
o$[s_1]$	o$[s_1]$	o$[s_1 \cap s_3]$	c$[s_1]$	c$[s_1 \cap s_4]$	u
b	b	c$[s_3]$	b	c$[s_4]$	-
c$[s_2]$	c$[s_2]$	c$[s_2 \cap s_3]$	c$[s_2]$	c$[s_2 \cap s_4]$	-
u	u	u	-	-	u

Idee: Erhöhung der Menge beantwortbarer Anfragen

- durch Post-Processing
- durch Passing Bindings

Adornments und Postprocessing

$R_1(X,Y,Z)$: bfu
$R_2(X,Y,Z)$: buf
$R_1 \cup R_2$ = buu

Anfrage 1: (x_1,Y,Z) beantwortbar?
Anfrage 2: (x_1,y_1,z_1) beantwortbar?

Idee: (x_1,y_1,Z) an R_1
(x_1,Y,z_1) an R_2
Dann im Mediator filtern:
$Z=z_1$ bzw. $Y=y_1$

Was ist neu?
u = f : durch nachträgliches Filtern (postprocessing)
o[s] = f : falls Bindung nicht in s, weglassen und später Filtern
Zusammen: $R_1 \cup R_2$ = bff

Quelle: [YLGU99]

Adornments – Verknüpfung durch UNION

Vorher:

	f	**o**$[s_3]$	**b**	**c**$[s_4]$	**u**
f	f	o$[s_3]$	b	c$[s_4]$	u
o$[s_1]$	o$[s_1]$	o$[s_1 \cap s_3]$	c$[s_1]$	c$[s_1 \cap s_4]$	u
b	b	c$[s_3]$	b	c$[s_4]$	-
c$[s_2]$	c$[s_2]$	c$[s_2 \cap s_3]$	c$[s_2]$	c$[s_2 \cap s_4]$	-
u	u	u	-	-	u

Nachher:

f	f	f	b	c$[s_4]$	f
o$[s_1]$	f	f	b	c$[s_4]$	f
b	b	b	b	c$[s_4]$	b
c$[s_2]$	c$[s_2]$	c$[s_2]$	c$[s_2]$	c$[s_2 \cap s_4]$	c$[s_2]$
u	f	f	b	c$[s_4]$	f

Quelle: [YLGU99]

Adornments und Passing Bindings

JOIN über templates ohne *passing bindings*
$R_1(X,Y,Z)$: fbf
$R_5(Z,U)$: bf
$R_1 \bowtie R_2$ = fbbf

Anfrage 1: (X,y_1,z_1,U) beantwortbar?
Anfrage 2: (X,y_1,Z,U) beantwortbar?

Idee: (X,y_1,Z) an R_1
$(z_1,U)\ldots(z_n,U)$ an R_5

Passing Bindings: Ergebnisse einer Sicht werden vom Mediator in die gebundene Variable der nächsten Sicht eingetragen.

JOIN über templates mit *passing bindings*:
$R_1 \triangleright\!\triangleleft R_5$ = fbff

Quelle: [YLGU99]

Adornments und Passing Bindings

Vorher:

	f	**o**$[s_3]$	**b**	**c**$[s_4]$	**u**
f	f	o$[s_3]$	b	c$[s_4]$	u
o$[s_1]$	o$[s_1]$	o$[s_1 \cap s_3]$	c$[s_1]$	c$[s_1 \cap s_4]$	u
b	b	c$[s_3]$	b	c$[s_4]$	-
c$[s_2]$	c$[s_2]$	c$[s_2 \cap s_3]$	c$[s_2]$	c$[s_2 \cap s_4]$	-
u	u	u	-	-	u

Zweite Quelle

Nachher:

	f	**o**$[s_3]$	**b**	**c**$[s_4]$	**u**
f	f	f	f	c$[s_4]$	f
o$[s_1]$	f	f	f	c$[s_4]$	f
b	b	b	b	c$[s_4]$	b
c$[s_2]$	c$[s_2]$	c$[s_2]$	c$[s_2]$	c$[s_2 \cap s_4]$	c$[s_2]$
u	f	f	f	c$[s_4]$	f

Erste Quelle

Quelle: [YLGU99]

Adornments – Selektion mit Postprocessing

Variante 1: $R_1(X,Y,Z)$, $X < x_1$: bfu

$Q(x_2, Y, Z)$ beantwortbar?
$Q(x_2, Y, z_1)$ beantwortbar?
$Q(X, y_1, z_1)$ beantwortbar?

bfu wird zu bff mit postprocessing

Variante 2: $R_1(X,Y,Z)$, $X = x_1$: bfu

$Q(X, y_1, z_1)$ beantwortbar?

$Q(X, y_1, z_1) = Q(x_1, y_1, z_1)$ wegen Prädikat
bfu wird zu bff wird zu fff

Quelle: [YLGU99]

Adornments – Selektion mit Postprocessing

Vorher Nachher

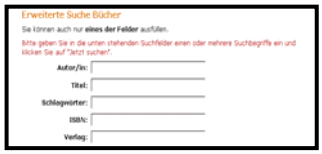
Base View Adornment	Sel. Attribute Adornment
f	f
o[s_1]	f
b	f or b
c[s_1]	f or c[s_1]
u	f

Quelle: [YLGU99]

Viele Templates

Problem: Quellen exportieren oft mehrere templates

- Beispiel: Amazon (Autor Titel, Schlagwort, ISBN, Verlag)
- bffff, fbfff, ffbff, fffbf, ffffb
- Beispiel: Verlage(Verlag, Ort)
- bf, fb
- Sicht im IIS: Amazon ⋈Verlag Verlage
- Templates der Sicht aus jeder Kombination:
  - bfffff, fbffff, ffbfff, fffbff, ffffbf
  - bffffb, fbfffb, ffbffb, fffbfb, ffffbb
  - + fffffb (ffffb ⋈ fb mit passing binding)

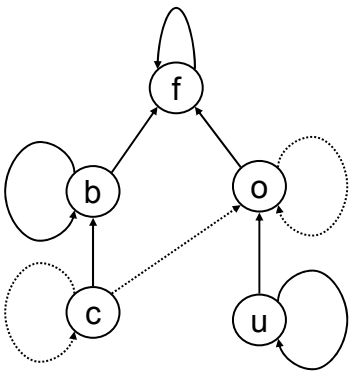


Lösung:

- Einige templates sind redundant

Quelle: [YLGU99]

Redundanz in Templates



Weniger restriktiv

Weniger restriktiv falls Auswahllisten Teilmengen sind

- bfffff, fbffff, ffbfff, fffbff, ffffbf
- bffffb, fbfffb, ffbffb, fffbfb, ffffbb

Algorithmus zur Entfernung redundanter templates.

Quelle: [YLGU99]

Adornments – Fallbeispiel

- Amazon
  - Formular 1: Mindestens eine Spezifikation aus author, title, subject, format (format aus Auswahlliste)
  - Formular 2: ISBN spezifizieren
  - Formular 3: Mindestens eine Spezifikation aus keyword, publisher, date
  - Antwortrelation: author, title, ISBN, publisher, date, format, price, shipping info
- Barnes & Noble
  - Formular 1: Mindestens eine Spezifikation aus author, title, keywords; optionale Spezifikation in format subject, price, age (alles aus Auswahllisten)
  - Formular 2: ISBN spezifizieren

Quelle: [YLGU99]

Adornments - Fallbeispiel

Amazon

author	title	format	subject	KW	ISBN	pub	date	price	ship
b	f	o	f'	u'	u	u	u	u	u
f	b	o	f'	u'	u	u	u	u	u
f	f	c	f'	u'	u	u	u	u	u
f	f	o	b'	u'	u	u	u	u	u
u	u	u	u'	u'	b	u	u	u	u
u	u	u	u'	f'	u	b	f	u	u
u	u	u	u'	b'	u	f	f	u	u
u	u	u	u'	f'	u	f	b	u	u

Barnes & Noble

author	title	format	subject	KW	ISBN	pub	date	price	ship	age
f	b	o	o'	f'	u	u	u	o	u	o'
b	f	o	o'	f'	u	u	u	o	u	o'
f	f	o	o'	b'	u	u	u	o	u	o'
u	u	u	u'	u'	b	u	u	u	u	u'

IIS

author	title	format	subject	KW	ISBN	pub	date	price	ship	age
f	f	f	u'	u'	b	f	f	f	f	u'
f	f	f	u'	b'	f	f	f	f	f	u'
b	f	f	o'	u'	f	f	f	f	f	u'
f	b	f	o'	u'	f	f	f	f	f	u'
b	f	f	u'	f'	f	b	f	f	f	u'
f	b	f	u'	f'	f	b	f	f	f	u'
f	b	f	u'	f'	f	f	b	f	f	u'
b	f	f	u'	f'	f	f	b	f	f	u'

Quelle: [YLGU99]

Adornments - Fallbeispiel

IIS

author	title	format	subject	KW	ISBN	pub	date	price	ship	age
f	f	f	u'	u'	b	f	f	f	f	u'
f	f	f	u'	b'	f	f	f	f	f	u'
b	f	f	o'	u'	f	f	f	f	f	u'
f	b	f	o'	u'	f	f	f	f	f	u'
b	f	f	u'	f'	f	b	f	f	f	u'
f	b	f	u'	f'	f	b	f	f	f	u'
f	b	f	u'	f'	f	f	b	f	f	u'
b	f	f	u'	f'	f	f	b	f	f	u'

Ableiten von 4 Formularen im IIS nach [YLGU99]
- Spezifikation der ISBN (template 1)
- Spezifikation des keyword (template 2)
- Mindestens author oder title spezifizieren (templates 3 und 4)
- Mindestens author oder title und mindestens publisher oder date spezifizieren (templates 5-8)

Quelle: [YLGU99]

Überblick

- Verteilung
- Autonomie
- Syntaktische Heterogenität
- Strukturelle Heterogenität
- Semantische Heterogenität
  - Namenskonflikte
  - Identität
  - Datenkonflikte
- Gebundene und Freie Variablen
  - Adornments
  - Anfrageplanung

Gebundene & Freie Variablen – Beispiel

Sources	View Schemas	Must Bind
1	v1(Song, CD)	Song
2	v2(CD, Artist, Price)	CD
3	v3(CD, Artist, Price)	Artist

Query: "Find the prices of CDs containing a song titled ***Friends***."

$$v1(\textit{Friends}, CD) \bowtie v2(CD, Artist, Price)$$
$$v1(\textit{Friends}, CD) \bowtie v3(CD, Artist, Price)$$

Quelle: [LC00]

Gebundene & Freie Variablen – Beispiel

v_1(Song, CD)

<Friends, Life>

<Friends, Love>

v_2(CD, Artist, Price)

<Love, Lucy, $15>

<Story, Snoopy, $14>

v_3(CD, Artist, Price)

<Story, Lucy, $13>

<Love, Snoopy, $10>

<Life, Charlie, $8>

Bastelaufgabe 1:
Wie teuer ist die billigste CD mit einem Song namens "Friends"?

Quelle: [LC00]

Gebundene & Freie Variablen – Beispiel

v_1(Song, CD)

<Friends, Life>

<Friends, Love>

v_2(CD, Artist, Price)

<Love, Lucy, $15>

<Story, Snoopy, $14>

v_3(CD, Artist, Price)

<Story, Lucy, $13>

<Love, Snoopy, $10>

<Life, Charlie, $8>

Bastelaufgabe 2:
Welches ist die billigste CD mit einem Song namens "Friends", *die Sie anfragen können*?

Gebundene & Freie Variablen – Beispiel

v1 ⋈ v2: {$15}

v_1(Song, CD)

<Friends, Life>

<Friends, Love>

v_2(CD, Artist, Price)

<Love, Lucy, $15>

<Story, Snoopy, $14>

v_3(CD, Artist, Price)

<Story, Lucy, $13>

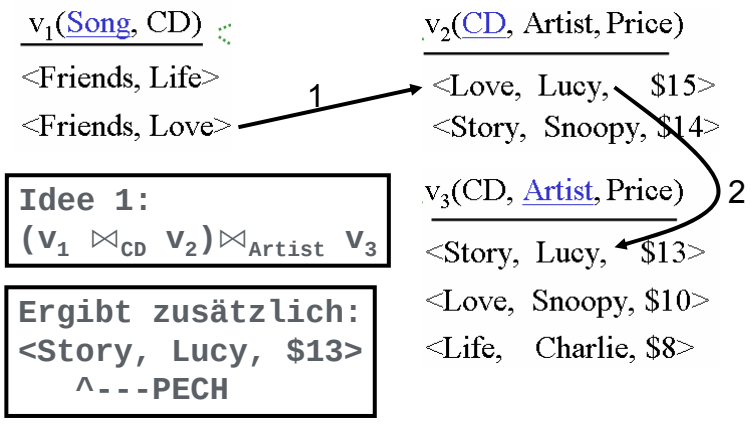
<Love, Snoopy, $10>

<Life, Charlie, $8>

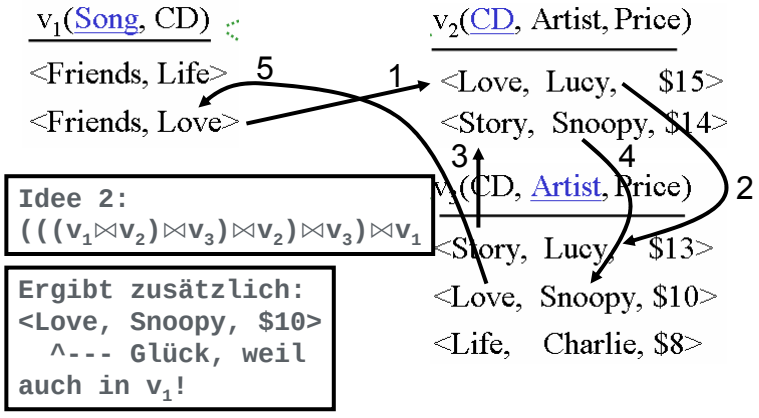
v1 ⋈ v3: empty, no binding for Artist.

Quelle: [LC00]

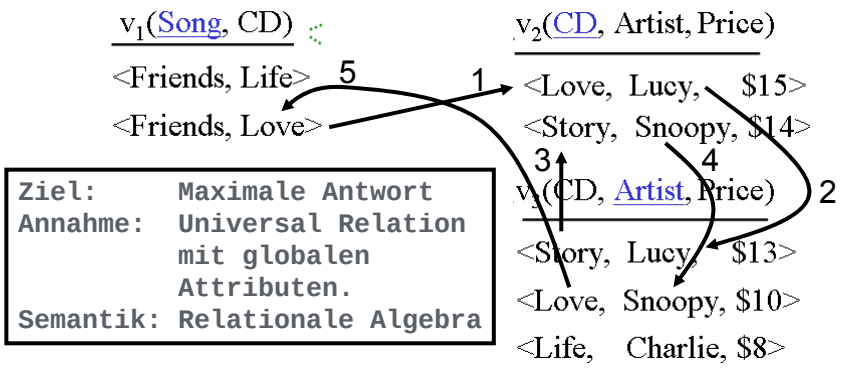
Gebundene & Freie Variablen – Beispiel



Gebundene & Freie Variablen – Beispiel

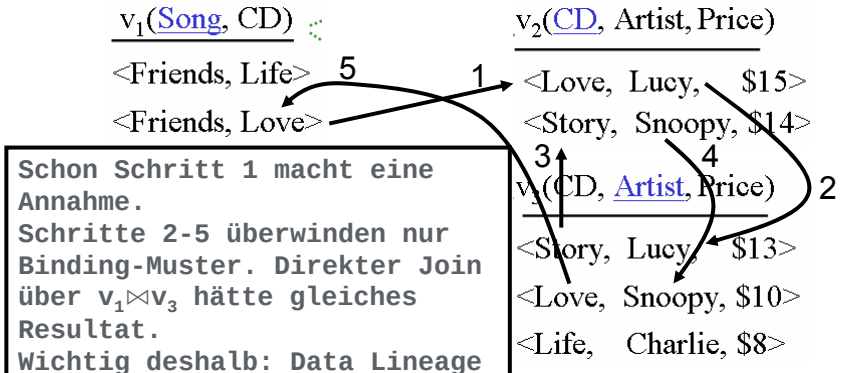


Gebundene & Freie Variablen – Beispiel: Semantik



Ziel: Maximale Antwort
Annahme: Universal Relation mit globalen Attributen.
Semantik: Relationale Algebra

Gebundene & Freie Variablen – Beispiel: Semantik



Schon Schritt 1 macht eine Annahme.
Schritte 2-5 überwinden nur Binding-Muster. Direkter Join über $v_1 \bowtie v_3$ hätte gleiches Resultat.
Wichtig deshalb: Data Lineage und Visualisierung

Literatur

Wichtigste Literatur für heute

- [BKLW99] Busse, Kutsche, Leser, Weber, Federated Information Systems: Concepts, Terminology and Architectures. Forschungsbericht 99-9 des FB Informatik der TU Berlin, 1999.
  Online: http://www.informatik.hu-berlin.de/~leser/publications/tr_terminology.ps
- [ÖV99] Principles of Distributed Database Systems
  M. Tamer Özsu, Patrick Valduriez, Prentice Hall, (1991/)1999.
  Kapitel 1 und 4
- [YLGU99] Ramana Yerneni, Chen Li, Hector Garcia-Molina, Jeffrey D. Ullman, „Computing Capabilities of Mediators“, SIGMOD 1999

Weitere Literatur

- [Con97] Föderierte Datenbanksysteme. Konzepte der Datenintegration
  Stefan Conrad, Springer Verlag, 1997
- [LC00] Chen Li, Edward Chang „Query Planning with Limited Source Capabilities“, ICDE 2000