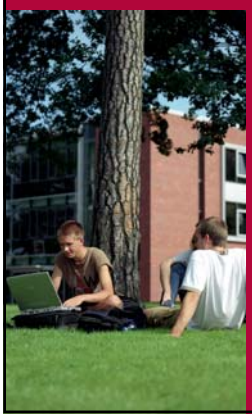
Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam



Informationsintegration

Datenfusion: Union und Co.

12.7.2006

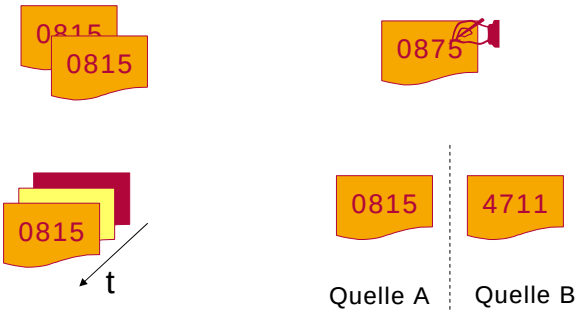
Jens Bleiholder

„When you use information from one source, it's plagiarism;
when you use information from many sources, it's information fusion."
[Belur V. Dasarathy]

„Semantische Heterogenität ist ein überladener Begriff ohne klare Definition. Er bezeichnet die Unterschiede in Bedeutung, Interpretation und Art der Nutzung."
[Öszu, Valduriez 91]

Semantische Heterogenität

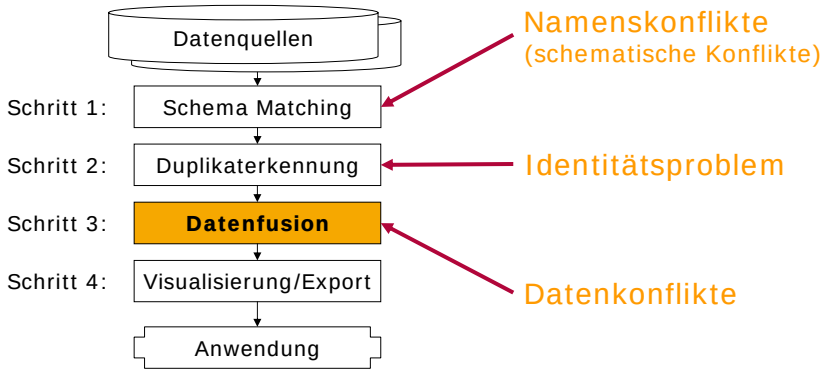
- Gleiches Objekt mehrfach beobachtet
- Manuelle Erfassung der Daten
- Objekt ändert Eigenschaften von Zeit zu Zeit
- Keine global konsistente ID
  - ISBN, SSN, etc.



Typische Anwendungen

- Personen- und Adressdaten
  - Volkszählungen
  - Werbeaktionen
  - Kundenpflege
- Bibliographische Daten
  - Zentrale Register
- Waren/Katalogdaten
  - Einkauf bei mehreren Shops
- Webseiten
  - Metasuchmaschinen
- Molekularbiologische Daten

Überblick



Schema Matching

Findet semantisch gleiche Attribute in Relationen

- Zwei Attribute, die gleiche Bedeutung tragen
- Attributnamen und -werte dürfen sich unterscheiden

Title	Year	Director	Genre
Snatch	2000	Ritchie	Crime
Troy	2004	Petersen	⊥
Vanilla Sky	2001	Crowe	Sci-Fi
Shrek	2001	Adamson	Anim.
The Matrix	1999	Wachowski	Fantasy

Titel	Jahr	Rating	Genre
Snatch	1999	R	Crime
Troja	2004	R	History
Vannila Ski	2001	R	Sci-Fi
Vannile Sky	2000	16	Comedy
Matrix	1999	16	Fantasy

Schema Matching

Formales Problem

- Zwei Tabellen, mit unterschiedlichen Attributen
- Erzeuge eine Abbildung, die jedem Attribut einer Tabelle die semantisch gleichen der anderen Tabellen zuordnet

Problemerweiterungen

- Mehrere Tabellen
- 1:m und n:m Beziehungen
- XML Dokument, also Nesting

Objektidentifikation

Findet Duplikate in Relationen

- Zwei Tupel, die das gleiche *real-world* Objekt repräsentieren
- Attributwerte dürfen sich unterscheiden

Title	Year	Director	Genre	ID
Snatch	2000	Ritchie	Crime	1
Troy	2004	Petersen	⊥	2
Vanilla Sky	2001	Crowe	Sci-Fi	3
Shrek	2001	Adamson	Anim.	4
The Matrix	1999	Wachowski	Fantasy	5

ID	Titel	Jahr	Rating	Genre
1	Snatch	1999	R	Crime
2	Troja	2004	R	History
3	Vannila Ski	2001	R	Sci-Fi
3	Vannile Sky	2000	16	Comedy
5	Matrix	1999	16	Fantasy

Objektidentifikation

Formales Problem

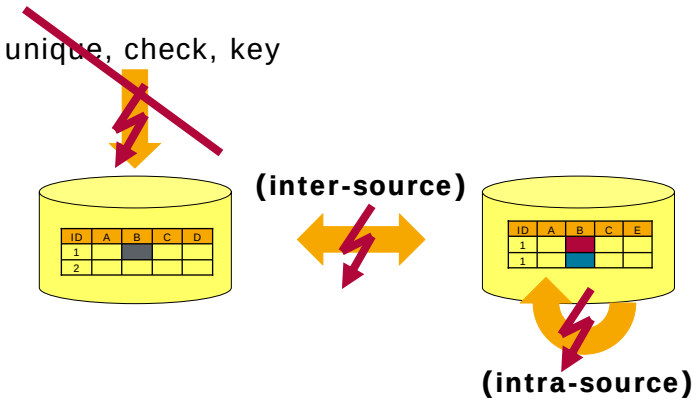
- Eine Tabelle (der Größe *N*), potentiell mit Duplikaten
- Erzeuge für jedes Tupel eine ID, so dass Duplikate gleiche ID's erhalten

Problemerweiterungen

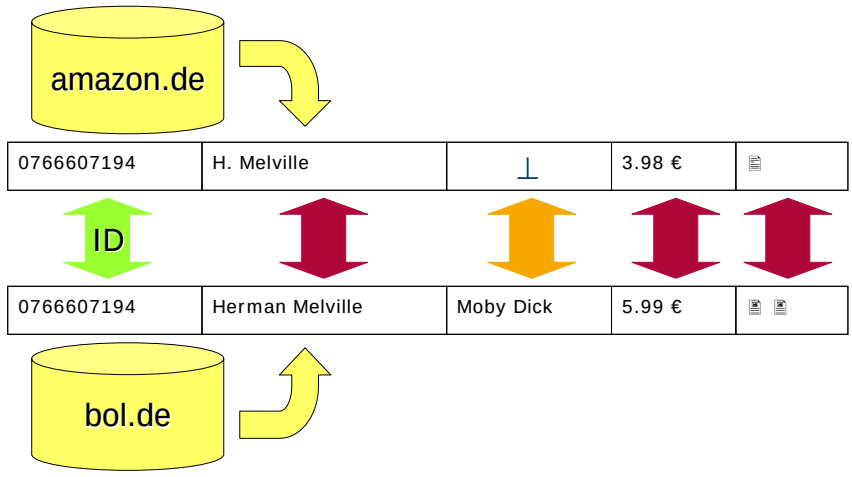
- Zwei Tabellen mit unterschiedlichem Schema
- Ein XML Dokument mit Duplikaten

Datenkonflikte

Zwei Duplikate haben unterschiedliche Attributwerte für semantisch gleiches Attribut.



Datenkonflikte – Beispiele



Datenkonflikte – Entstehung

- Keine Integritätsbedingungen / Konsistenz-Checks
- Redundante Schemata
- Vorhandensein von Duplikaten
- Nicht korrekte Einträge
  - Tippfehler, Übertragungsfehler
  - Falsche Rechenergebnisse
- Obsolete Einträge
  - Verschiedene Aktualisierungszeitpunkte
    - ausreichende Aktualität einer Quelle
    - verzögerte Aktualisierung
  - Vergessene Aktualisierung

Innerhalb eines Informationssystems

Datenkonflikte – Entstehung

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

Innerhalb eines Informationssystems

Datenkonflikte – Behebung

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

Innerhalb eines Informationssystems

Datenkonflikte – Entstehung

- Lokal konsistent aber global inkonsistent
- Duplikate
- Unterschiedliche Datentypen
- Lokale Schreibweisen/Konventionen

Datenkonflikte – Behebung

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

Datenkonflikte – Implementierung

Prozedural

- Programm
- Im DB Kern

✚ Schnell
✚ Gut in materialisierten IS
✖ Schlecht wiederverwendbar

Deklarativ

- Anfragesprache

✚ Funktioniert gut in föderierten IS
✚ Wiederverwendbar
✖ Langsamer

Relationale Objektintegration

Union (Vereinigung), z.B. [GUW02]

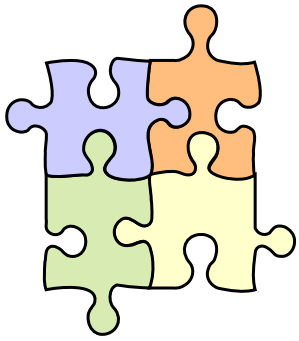
✚ Eliminierung exakter Duplikate

Minimum Union, [GL94]

✚ Eliminierung subsumierter Tupel

Jedoch

✖ Duplikatintegration
✖ Konfliktlösung



Auch: Join, Merge, Group, ...

Relationale Algebra – Vereinigung

Mitarbeiter m

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34

Employees e

p_id	vorname	nachname	alter
5	Petra	Weger	28
7	Andreas	Zwickel	44

Frage: Wie sieht m ∪ e aus?

m ∪ e

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34
5	Petra	Weger	28
7	Andreas	Zwickel	44

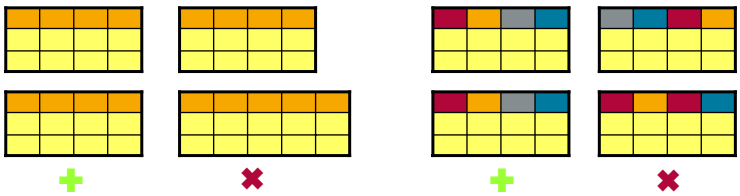
Union zur Informationsintegration?

Verschiedene Schemata?

Null-Werte?

Datenkonflikte?

Was bedeutet *UNION compatible*?



Gleiche Spaltenzahl

Gleiche Datentypen

Relationale Algebra – Vereinigung

Mitarbeiter m

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34

Employees e

p_id	vorname	nachname	geboren
5	Petra	Weger	21.3.76
7	Andreas	Zwickel	12.3.60

Anderes Schema:

m ∪ e nicht definiert!

Problemlösung:

SchemaSQL

Schema Mapping

Relationale Algebra – Vereinigung

Mitarbeiter m

p_id	vorname	nachname	alter
5	Petra	Weger	28
7	Andreas	Zwickel	34

Employees e

p_id	vorname	nachname	alter
5	Petra	Weger	28
7	Andreas	Zwickel	⊥

Null-Werte

m ∪ e

p_id	vorname	nachname	alter
7	Andreas	Zwickel	34
5	Petra	Weger	28
7	Andreas	Zwickel	⊥

„ ⊥“ vs. „34“ ist auch ein Konflikt!

Relationale Algebra – Vereinigung

Mitarbeiter m

p_id	vorname	nachname	alter
1	Peter	Müller	32
7	Andreas	Zwickel	34

Employees e

p_id	vorname	nachname	alter
5	Petra	Weger	28
7	Andreas	Zwickel	44

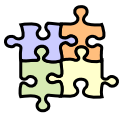
m ∪ e

p_id	vorname	nachname	alter
1	Peter	Müller	32
7	Andreas	Zwickel	34
5	Petra	Weger	28
7	Andreas	Zwickel	44

„44“ vs. „34“ ist erst recht ein Konflikt!

Konfliktbehebung mit Union?

Exakte Duplikate!



Quelle 1
Quelle 2

a, b
a, b
=> a, b

a, b
a, -
=> a, b
a, -

a, b
a, c
=> a, b
a, c

a, -, d
a, c, -
=> a, -, d
a, c, -

Konfliktbehebung mit Union?

Ungeeignet

- Es werden nur „echte“ Duplikate entfernt.
- In SQL nur mit DISTINCT (Aufwand!)

Außerdem

- Es können leicht strukturelle Konflikte auftreten:
  - Primärschlüssel
- Es muss Schemagleichheit herrschen

Minimum Union

Outer Union, [Cod79]

Seien R_1 und R_2 Relationen mit Schemata S_1 bzw. S_2.

Outer Union füllt R_1 und R_2 zunächst mit NULL-Werten auf, so dass beide dem Schema $S_1 \cup S_2$ entsprechen.

Dann wird die Vereinigung der beiden aufgefüllten Relationen gebildet.

R

A	B	C
P	1	2
P	2	1
Q	1	2

S

B	D
2	U
3	V

Frage: Wie sieht $R \uplus S$ aus?

$R \uplus S$

A	B	C	D
P	1	2	⊥
P	2	1	⊥
Q	1	2	⊥
⊥	2	⊥	U
⊥	3	⊥	V

Beispiel aus [Cod79]

Minimum Union

Subsumption, [Ull89]

Ein Tupel t_1 subsumiert ein Tupel t_2, wenn

- es über die gleichen Attribute definiert ist,
- t_2 hat mehr NULL Werte als t_1,
- $t_1 = t_2$ für alle nicht-NULL Werte von t_2.

Schreibweise:

R↓ ergibt die Tupel aus R, die nicht subsumiert sind.

R

p_id	vorname	nachname	alter
1	Peter	Müller	32
1	Peter	Müller	⊥
1	Peter	⊥	⊥
1	Peter	⊥	32
1	Peter	⊥	42
2	Wiebke	⊥	2
2	⊥	Meyer	2

Frage: Wieviele Tupel hat R↓?

R↓

p_id	vorname	nachname	alter
1	Peter	Müller	32
1	Peter	⊥	42
2	Wiebke	⊥	2
2	⊥	Meyer	2

Minimum Union – NULL-Werte

Semantik der NULL? [GUW02]

- Wert unbekannt ("*unknown*")
  - Es gibt einen Wert, ich kenne ihn aber nicht
  - Bsp.: Unbekannter Geburtstag
- Wert nicht anwendbar ("*inapplicable*")
  - Es gibt keinen Wert, der hier Sinn macht
  - Bsp.: Ehemann/-frau für Ledige
- Wert zurückbehalten ("*withheld*")
  - Wir dürfen den Wert nicht erfahren
  - Bsp.: Geheime Telefonnummer

Minimum Union – NULL-Werte

"Value not supplied"

"Value does not exist"

"Value undefined"

„Distinguished" NULL

"Total ignorance" NULL

C.J. Date:
- "Into the Unknown"
- "Much Ado About Nothing"
- "NOT Is Not Not!"
- "Oh No Not Nulls Again"
-...

Im weiteren: "**Unknown**"

Minimum Union – Beispiel

Minimum Union, [GL94]

$$K \oplus C = (K \uplus C)\downarrow$$

Kunde K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥

Customer C

p_id	nachname	alter
1	Müller	32
2	Schmidt	⊥
3	⊥	56

$K \uplus C$

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
1	⊥	Müller	32
2	⊥	Schmidt	⊥
3	⊥	⊥	56

$K \oplus C$

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
3	⊥	⊥	56

Konfliktbehebung mit Minimum Union?

Nicht in Standard-DBMS implementiert!

Quelle 1 → a, b
Quelle 2 → a, b
=> a, b

a, b
a, - => a, b

a, b
a, c => a, b
a, c

Subsumption!

a, -, d
a, c, - => a, -, d
a, c, -

Und was ist mit Join?

Kunde K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
4	Klaus	Lehmann	28

Customer C

p_id	nachname	alter
1	⊥	32
2	Schmidt	⊥
3	Meier	56
5	Weger	47

Frage: Ist Join zur Konfliktbehebung geeignet?

$K \bowtie_{p\_id} C$

p_id	K.vorname	K.nachname	C.nachname	K.alter	C.alter
1	Peter	Müller	⊥	32	32
2	Franz	Schmidt	Schmidt	55	⊥
3	Wiebke	Meyer	Meier	⊥	56

Merge und Prioritized Merge

Merge (⊠), [GPZ01]

- Vermischt Join und Union zu einem Operator
- COALESCE beseitigt NULLs
- Priorisierung möglich (◁)
- Lässt sich mit Hilfe von SQL ausdrücken

```
( SELECT K.p_id, K.vorname, Coalesce(K.nachname, C.nachname), Coalesce(K.alter, C.alter)
FROM K LEFT OUTER JOIN C ON K.p_id = C.p_id )
        UNION
( SELECT C.p_id, K.vorname, Coalesce(C.nachname, K.nachname), Coalesce(C.alter, K.alter)
FROM K RIGHT OUTER JOIN C ON K.p_id = C.p_id )
```


Merge – Beispiel

Kunde K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	⊥
4	Klaus	Lehmann	28

Customer C

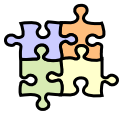
p_id	nachname	alter
1	⊥	32
2	Schmidt	⊥
3	Meier	56
5	Weger	47

Fragen: Wie sieht das Schema aus? Wie das Ergebnis?

C ⊠ K

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meier	56
3	Wiebke	Meyer	56
4	Klaus	Lehmann	28
5	⊥	Weger	47

Konfliktbehebung mit Merge?



Quelle 1 → a, b => a, b
Quelle 2 → a, b

a, b
a, - => a, b

a, b
a, c => a, b
a, c

Keine Subsumption!

a, -, d
a, c, - => a, c, d

Und was gibt es sonst noch?

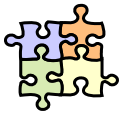
Match Join [YaÖz99]
- Komplexer Operator

ConQuer [FuFM05]
- „Consistent Query Answering“
- Rewriting einer SQL Anfrage

Burdick et. al. [BDJR05]
- Uncertainty in Data Warehouses
- „Possible Worlds“

Probabilistische Modelle [Mich89]
- Erweitern das Schema um Wahrscheinlichkeiten

Was sollte Duplikatintegration können?



Quelle 1 → a, b
Quelle 2 → a, b
=> a, b

a, b
a, - => a, b

a, b
a, c => a, cr(b,c)

a, -, d
a, c, - => a, c, d

Konfliktbehandlung

Konfliktbehandlung

Benutzer

Lösungsidee
(Strategie)

Ausführung
(Funktion)

System

Strategien, um Konflikte zu…

…ignorieren

Trust your friends

Pass it on

…vermeiden

Nothing is older than
the news from yesterday

Take the information
No Gossiping

…lösen

Cry with the wolves
Roll the dice

Meet in the middle

Konfliktlösungsfunktionen

Min, Max, Sum, Count, Avg, StdDev	Standard Aggregationsfunktionen
Random	Zufallswahl
First, Last	Nimmt ersten/letzten Wert, reihenfolgeabhängig
Longest, Shortest	Nimmt längsten/kürzesten Wert
Choose(*source*)	Quellenauswahl
ChooseDepending(*col, val*)	Wahl abhängig von *val* in *col*
Vote	Mehrheitsentscheid
Coalesce	Nimmt ersten nicht-null Wert
Group, Concat	Gruppiert, fügt zusammen
MostRecent	Nimmt aktuellsten Wert
MostAbstract, MostSpecific	Benutzt eine Taxonomie
….	….

Ausführung von Konfliktlösung

Mit SQL Bordmitteln?

~~Union, Join und Co.~~ → Gruppierung (nach ID)

Gruppierung – Beispiel

Bestellungen b

name	datum	wert
Peter	10.11.03	20
Peter	23.4.02	30
Peter	12.7.03	12
Franz	3.3.01	11
Martina	5.4.02	2
Jochen	6.6.99	200
Martina	1.1.95	2

```
SELECT    name, SUM(wert)
FROM      Bestellungen
GROUP BY  name
```

name	wert
Peter	62
Franz	11
Martina	4
Jochen	200

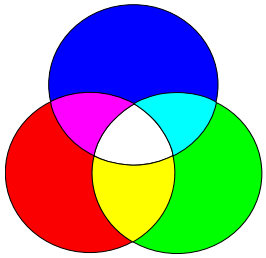
Gruppierung und Aggregation

- Gruppierung weist jedem Attribut zu:
  - Gruppierung oder
  - Aggregation
- Tupel werden gruppiert nach gleichen Gruppierungswerten, alle anderen Attribute werden innerhalb der Gruppen aggregiert.
- NULL-Werte bilden eigene Gruppe

```
SELECT    name, SUM(wert)
FROM      Bestellungen
GROUP BY  name
```


Gruppierung zur Integration

- Outer-Union auf alle Quellen
- Mit SQL GROUP BY umschließen
  - Gruppierung nach ID Attribut
  - Aggregat-Funktionen als Konfliktlösungsfunktion für alle anderen Attribute.



Gruppierung zur Integration

$K \uplus C$

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	55
1	⊥	Müller	32
2	⊥	Schmidt	⊥
3	⊥	Meier	56

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	55

```
SELECT p_id, RESOLVE(vorname), RESOLVE(nachname), RESOLVE(alter)
FROM        K ] C
GROUP BY    p_id
```


Gruppierung zur Integration

$K \uplus C$

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meyer	55
1	⊥	Müller	32
2	⊥	Schmidt	⊥
3	⊥	Meier	56

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Franz	Schmidt	55
3	Wiebke	Meier	56

```
SELECT p_id, MAXLEN(vorname), CHOOSE(nachname,C), MAX(alter)
FROM        K ] C
GROUP BY    p_id
```

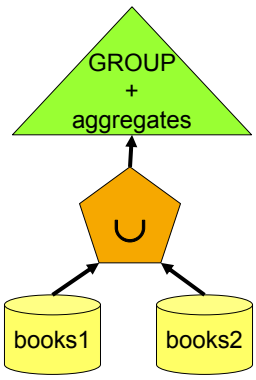
Längster String

C ist bevorzugte Quelle

größter Wert

Gruppierung – Beispiel


```
SELECT books.isbn,
       MAXLEN(books.title),
       MIN(books.price)
FROM   (
            SELECT * FROM books1
                  UNION
            SELECT * FROM books2
       )
       AS books
GROUP BY
       books.isbn
```



Gruppierung – Vor-/Nachteile

✚ Vorteile
- Effizient
  - Implementierung durch Sortierung
- Behandelt Duplikate innerhalb einer Quelle und zwischen Quellen gleich.
- Simpel / kurz

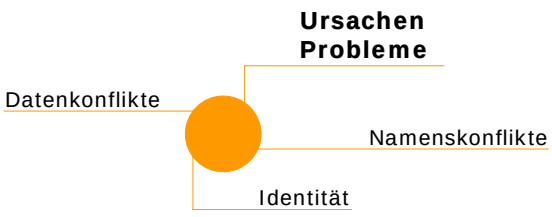
✖ Nachteile
- Beschränkt auf eingebaute Standard Aggregat-Funktionen:
  - MAX, MIN, AVG, VAR, STDDEV, SUM, COUNT
- Gruppierung nur nach Gleichheit
  - ID Attribut ist notwendig
- Outer Union nicht immer implementiert.

Gruppierung – Vor-/Nachteile

- Nachteile genauer:
  - Standard Aggregate
    - MAX, MIN, AVG, VAR, STDDEV, SUM, COUNT
    - Erweiterung um „user-defined aggregates“
      - Optimierung
      - Assoziativität und Kommutativität nicht sichergestellt.
    - Einzig Informix bot dies an.
    - Außerdem zwei Forschungsprojekte
  - Standard Gruppierung
    - Implizit ist Gleichheit gefordert.
    - Besser wäre: GROUP BY similar(id,name,birthdate)
    - Wird nirgends angeboten
      - Effizienz ist schwer für den Optimierer abzuschätzen.
      - Transitivität kann nicht gewährleistet werden.

Dies sollten Sie gelernt haben

Semantische Heterogenität



Masterarbeiten

- GenFuse
  - Automatische Generierung von Fusionsanfragen
  - Nutzung historischer Daten
  - Nutzung von Präferenzen
- DifFuse
  - Erfassung von Differenzen von Fusionen
  - Nutzererfahrungen
  - Maße

Literatur

[GL94] *Outerjoins as Disjunctions*, Cesar A. Galindo-Legaria, SIGMOD 1994 conference

[Cod79] E. F. Codd: *Extending the Database Relational Model to Capture More Meaning*. TODS 4(4): 397-434 (1979)

[Ull89] Jeffrey D. Ullman: *Principles of Database and Knowledge-Base Systems, Volume II*. Computer Science Press 1989

[GUW02] Hector Garcia-Molina, Jeffrey D. Ullman and Jennifer Widom, *Database Systems*, Prentice Hall, 2002

[GPZ01] S. Greco, L. Pontieri, E. Zumpano, *Integrating and Managing Conflicting Data*, Springer, 2001

[YaÖz99] L. L. Yan, T. Özsu, *Conflict tolerant queries in AURORA,* CoopIS 1999 conference

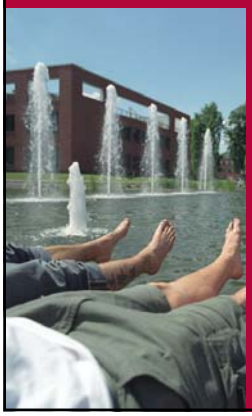
[FuFM05] A. Fuxman, E. Fazli, R. Miller, *Conquer: Efficient Management of inconsistent databases,* SIGMOD 2005 conference

[BDJR05] D. Burdick, P. Deshpande, T. Jayram, R. Ramakrishnan, S. Vaithyanathan, *OLAP Over Uncertain and Imprecise Dat,* VLDB 2005 conference

[Mich89] L. DeMichiel, *Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains,* IEEE Trans. Knowl. Data Eng., 1989(1), p. 485-493

HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam



Vielen Dank fürs Zuhören!

- Zusammenfassung:
  - Datenintegration
  - Relationale Datenfusion
  - Umsetzung
- Thema Dienstag:
  - Hidden Web