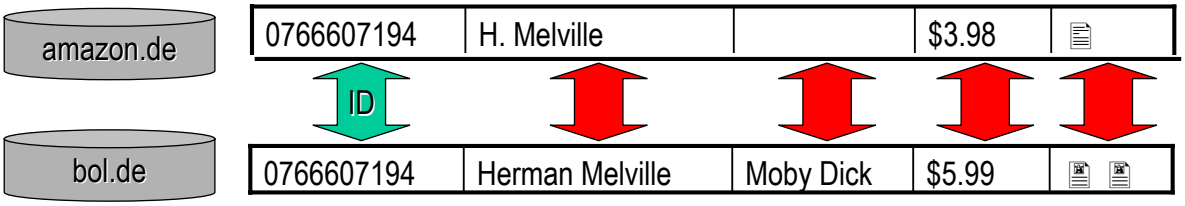
Datenkonflikte und Konfliktlösung

- Datenkonflikt: Duplikate haben unterschiedliche Werte für ein semantisch gleiches Attribut
- Datenkonflikte sind
  - Intra-Source: Innerhalb eines Informationssystems
  - Inter-Source: Bei der Integration mehrerer Informationssysteme



- Datenauswahl: Präferenzordnung
  - Aktualität der Quelle
  - Vertrauen in Datenquelle
  - Quelle mit mehr Informationen
  - Mehr als 2 Quellen: Majority Voting
- Datenfusion
  - Verwendung von Konfliktlösungsfunktionen
  - „Aggregation“ der Konfliktwerte in einen
  - Hochgradig domänen-/attributabhängig

SQL: Standard Union

- Unbefriedigend
  - Keine Duplikaterkennung, nur Eliminierung identischer Tupel
  - Schemata müssen identisch sein, keine strukturelle Heterogenität
  - Kein Füllen von Lücken: NULL ≠ Wert
  - Keine Datenfusion (Probleme mit PK‘s)

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
2	Stefanie	Meier	Null

m ∪ e

p_id	vorname	nachname	alter
1	Peter	Müller	32
2	Stefanie	Meier	34
5	Petra	Weger	28
2	Stefanie	Meier	Null

Duplikatintegration

Standard Union

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, -
a, b a, c	⇨	Fehler

"Gute" Duplikatintegration

a, b a, b	⇨	a, b
a, b a, -	⇨	a, b
a, b a, c	⇨	a, f(b,c)
a, -, d a, c, -	⇨	a, c, d
a, b a, c	⇨	a, f(b,c)

Subsumption

Konfliktlösungs-funktion

Annahme: a identifiziert Objekte (und damit Duplikate)

Outer Union

- R1 und R2 Relationen mit Schemata S1 bzw. S2
- Outer Union füllt R1 und R2 mit Attributen und NULL-Werten auf, so dass beide dem Schema S1 ∪ S2 entsprechen
- Dann berechne die normale Union
- Kann also mit strukturell heterogenen Schemata umgehen
- Attribute in S1 und S2 mit identischem Namen sind nur einmal im Resultat
- Aber keine Datenfusion

R

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

S

B	D
2	U
3	V

$R \uplus S$

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

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

Subsumption

- Subsumption: Tupel t_1 subsumiert t_2, wenn
  - Es die gleichen Attribute hat und
  - t_2 mehr NULL-Werte hat als t_1 und
  - $t_1.A = t_2.A$ für alle Attribute A mit t2.A ≠ NULL
- R↓ ergibt die Tupel aus R, die nicht durch andere Tupel in R subsumiert werden
- Semantik der NULL?
  - Wert unbekannt ("*unknown*")
    - Bsp.: Unbekannter Geburtstag
  - Wert nicht anwendbar ("*inapplicable*")
    - Bsp.: Ehemann/-frau für Ledige
  - Wert zurückbehalten ("*withheld*")
    - Bsp.: Geheime Telefonnummer

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

R↓

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

Ullman: Principles of Database and Knowledge-Base Systems, Volume II., 1989
Garcia-Molina, Ullman, Widom: Database Systems, Prentice Hall, 2002

Minimum Union

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

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	⊥
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	⊥
3	Wiebke	Meyer	⊥
3	⊥	⊥	56

Cesar A. Galindo-Legaria: Outerjoins as Disjunctions,, SIGMOD 1994 Conference

Join

- Ist Join geeignet zur Integration?

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

$K \bowtie_{p\_id} C$

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

- 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 )
```

Customer C

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

Kunde K

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

$C \boxtimes K$

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

Konfliktlösung

- NULL-Werte nur ein (einfacher) Teil der Konfliktlösung
  - Allgemein: Konfliktlösungsfunktion
- Idee zur Implementierung in SQL
  - Duplikatfindung als Gruppierung
  - Konfliktlösung mit Aggregatfunktionen

$$f(x,y) := \begin{cases} \perp & if\ x = \perp\ and\ y = \perp \\ x & if\ x \neq \perp\ and\ y = \perp \\ y & if\ x = \perp\ and\ y \neq \perp \\ g(x,y) & else \end{cases}$$

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
….	….

Gruppierung/Aggregation zur Integration (1)

- Allgemeines Vorgehen
  - Outer-Union auf alle Quellen
  - Mit einer SQL GROUP BY Anfrage umschließen
    - Gruppierung nach “Duplikat-Id”
  - Aggregat-Funktionen als Konfliktlösungsfunktion für alle anderen Attribute
  - Konfliktlösungsfunktion manuell bestimmen

$K \uplus C$

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

p_id	vorname	nachname	alter
1	Peter A.	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

größter Wert

C ist bevorzugte Quelle

Gruppierung/Aggregation zur Integration (2)

- Vorteile
  - Eleganter Rahmen
  - Effizient (durch Sortierung)
  - Simpel, kurz, deklarativ
- Nachteile
  - Outer Union meistens nicht implementiert
    - Kann durch Sichten simuliert werden
  - Konfliktresolution beschränkt auf eingebaute Aggregatfunktionen
    - MAX, MIN, AVG, VAR, STDDEV, SUM, COUNT
    - User-defined aggregate functions?
  - Duplikaterkennung nur nach Gleichheit
    - Besser: GROUP BY similar(id,name,birthdate)
    - User-defined grouping functions?

Zusammenfassung

- Datenqualität entscheidend für Informationssysteme
  - schwieriges Problem für integrierte Informationssysteme
- Object Matching entscheidender Aspekt
  - Erkennung von Instanzen des gleichen Realweltobjekts
  - Viele Algorithmen (u.a. Ähnlichkeitsmaße) und Frameworks
  - Qualität abhängig von Domäne, Parametern, Hintergrundwissen, ...
- Instanzintegration
  - Konfliktlösung für widersprüchliche / fehlende Werte
  - Konfliktlösungsfunktionen zur Definition, welche Attributwerte im fusionierten/aggregierten Ergebnisobjekt auftreten