Schema Mapping

Komplexe Informationssysteme

Fabian Panse

panse@informatik.uni-hamburg.de

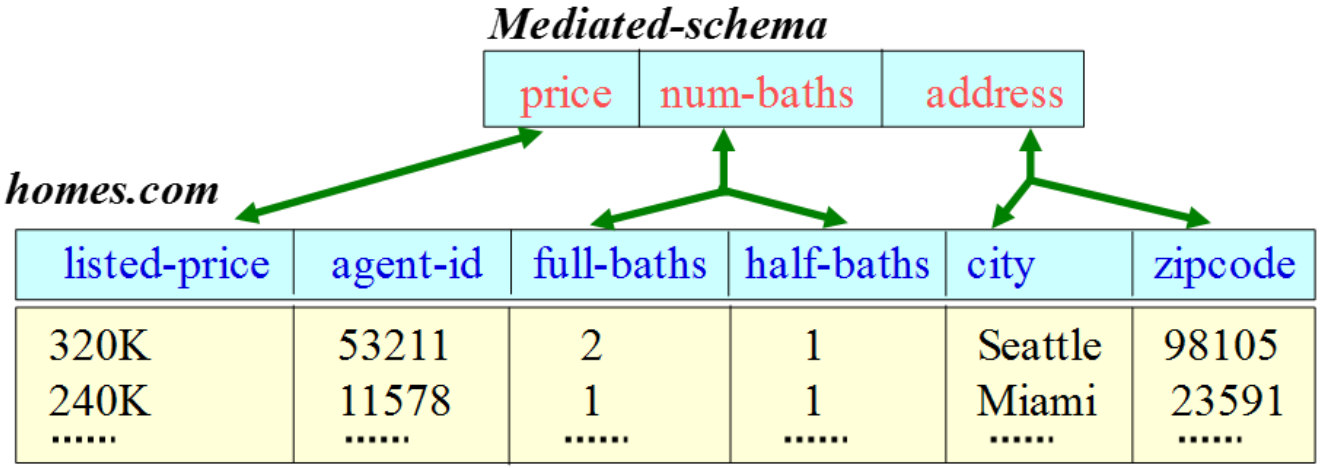
Universität Hamburg

Universität Hamburg

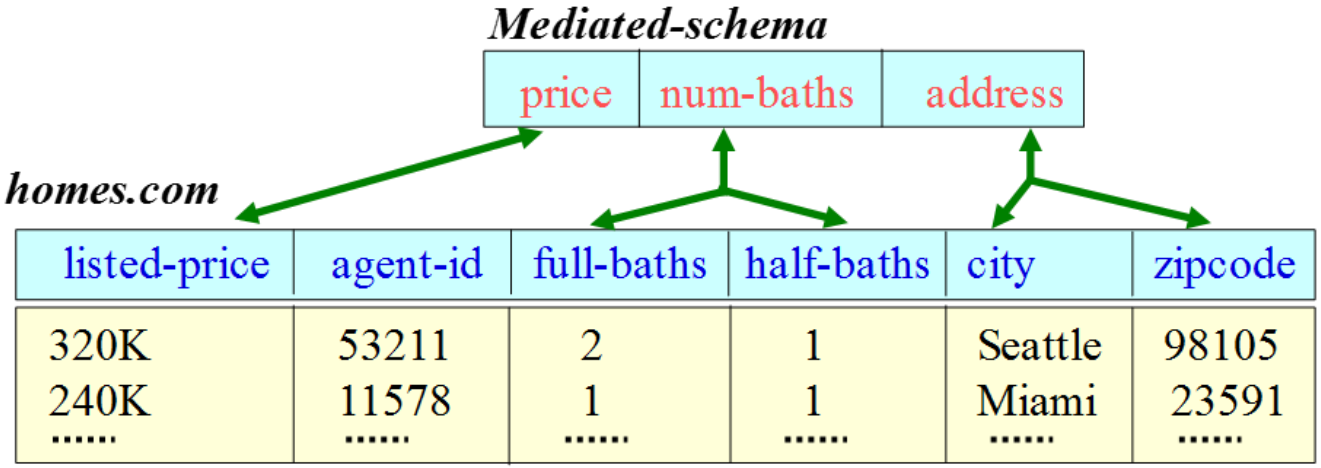
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Wiederholung: Schema Matching

Wiederholung: Schema Matching

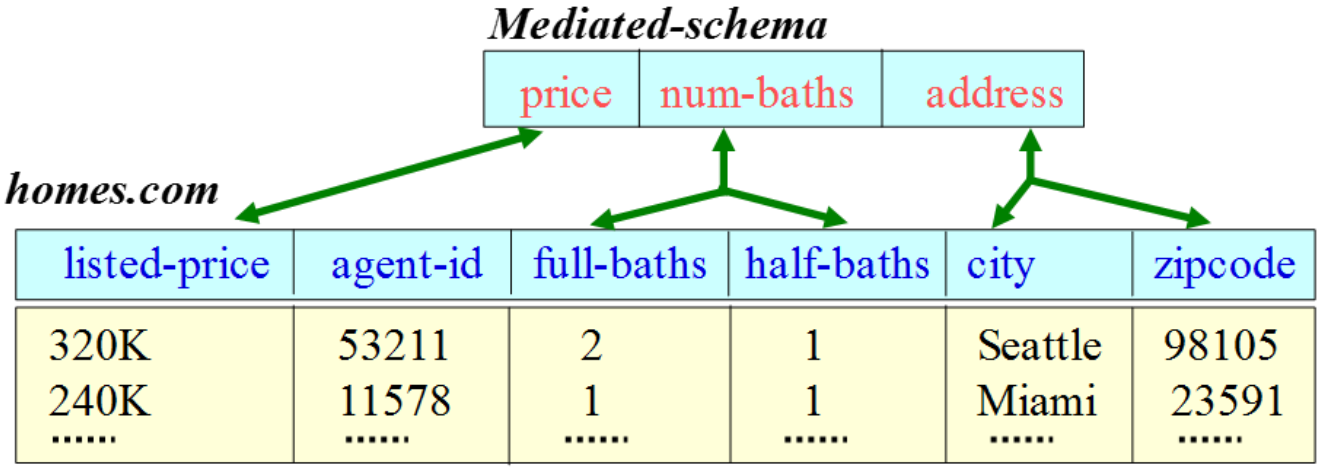


Wiederholung: Schema Matching



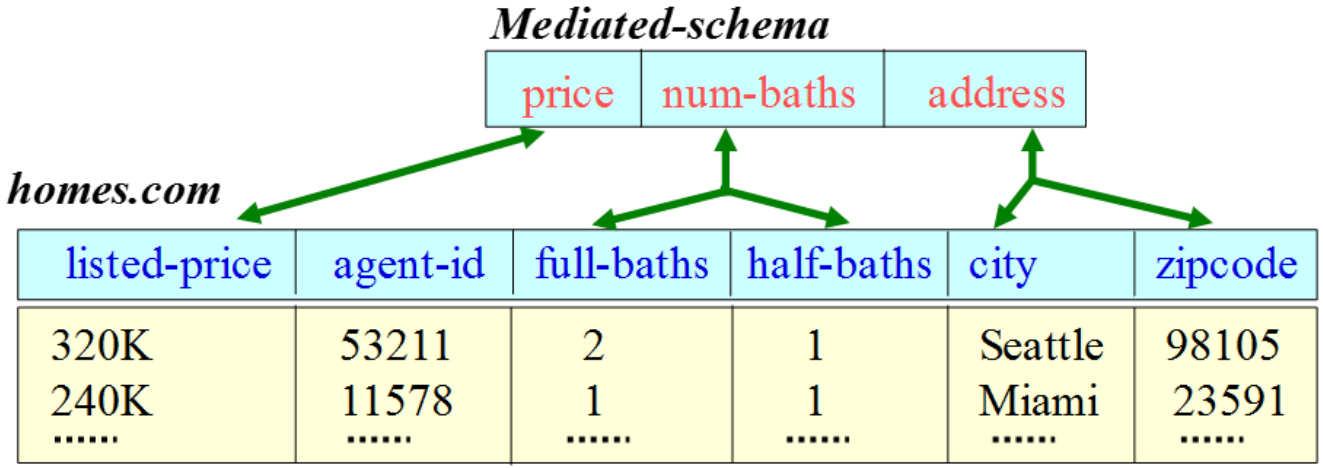
- Vergleich von Schemaelementen verschiedener Abstraktionsebenen auf semantische Gleichheit

Wiederholung: Schema Matching



- Vergleich von Schemaelementen verschiedener Abstraktionsebenen auf semantische Gleichheit
- Ergebnis: Eine Menge von Korrespondenzen (1:1, 1:n, oder n:m) zwischen den Elementen beider Schemata

Wiederholung: Schema Matching



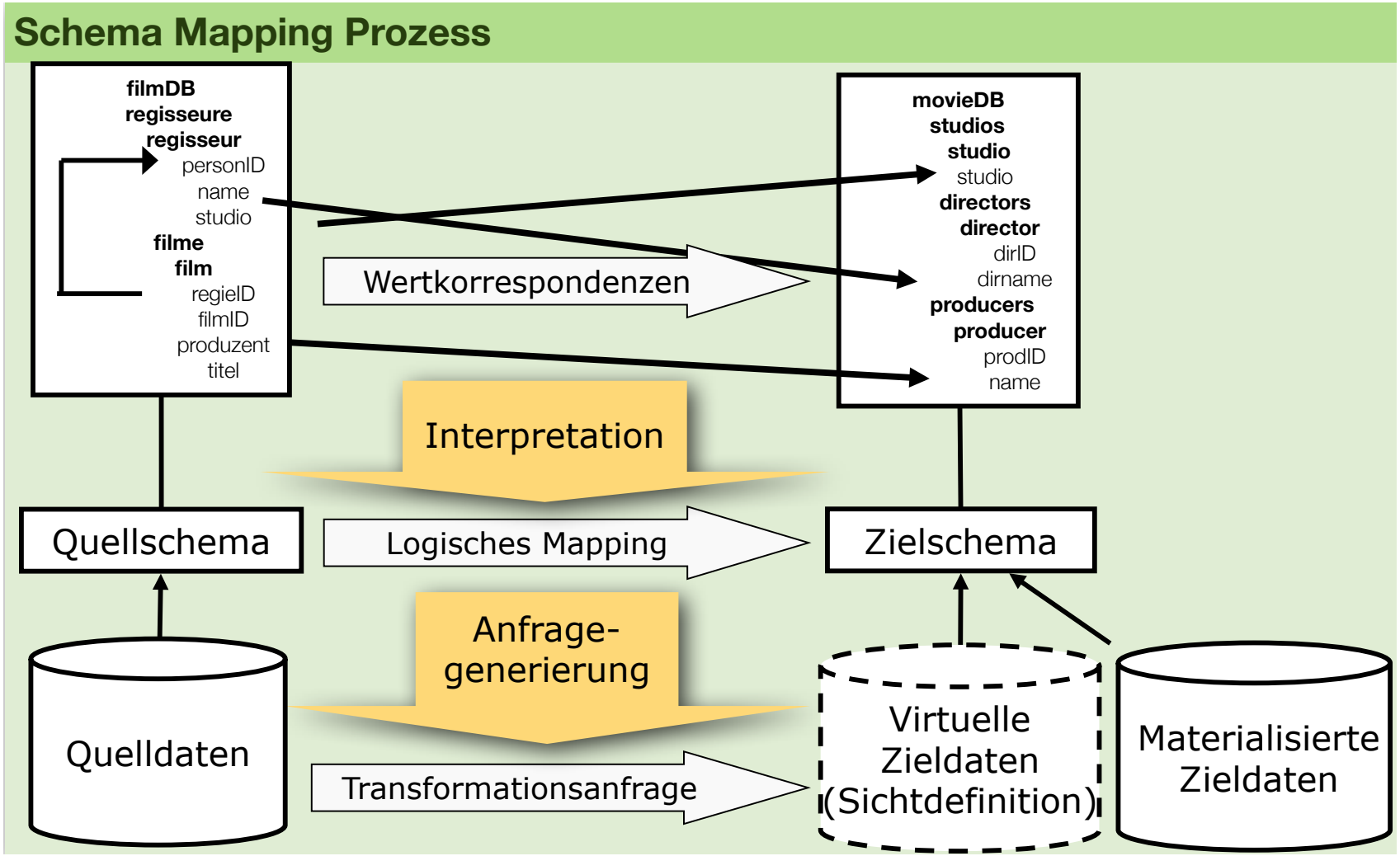
- Vergleich von Schemaelementen verschiedener Abstraktionsebenen auf semantische Gleichheit
- Ergebnis: Eine Menge von Korrespondenzen (1:1, 1:n, oder n:m) zwischen den Elementen beider Schemata
- nächster Schritt: Ableitung eines Mappings von den gefundenen Korrespondenzen

Schema Mapping

Schema Mapping – Definitionen [FHP+02]

- (Inter-Schema) Korrespondenz: Zuordnung von Quellschemaelement(en) zu Zielschemaelement(en)
- (High-level) Mapping: Menge von Korrespondenzen
- (Low-Level) Logisches Mapping: Logische Übersetzung (Interpretation) eines oder mehrerer Mappings, die
  - den Integritätsbedingungen beider Schemas gehorcht und
  - die Intention des Nutzers wiederspiegelt
- Transformationsanfrage: Anfrage in einer Anfragesprache, die Daten der Quelle in Struktur des Zielschemas überführt

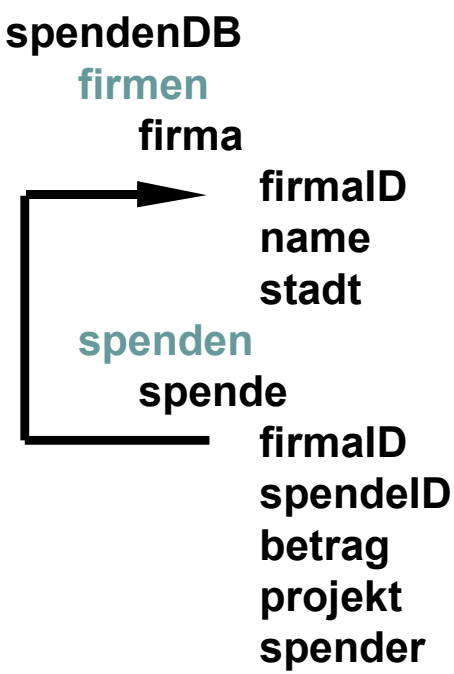
Schema Mapping Prozess

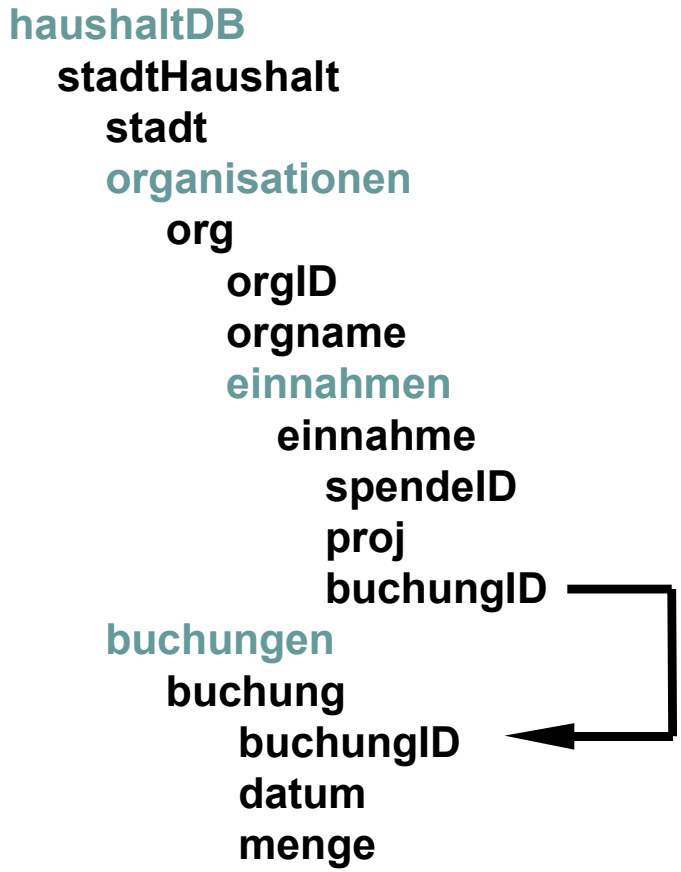


Probleme

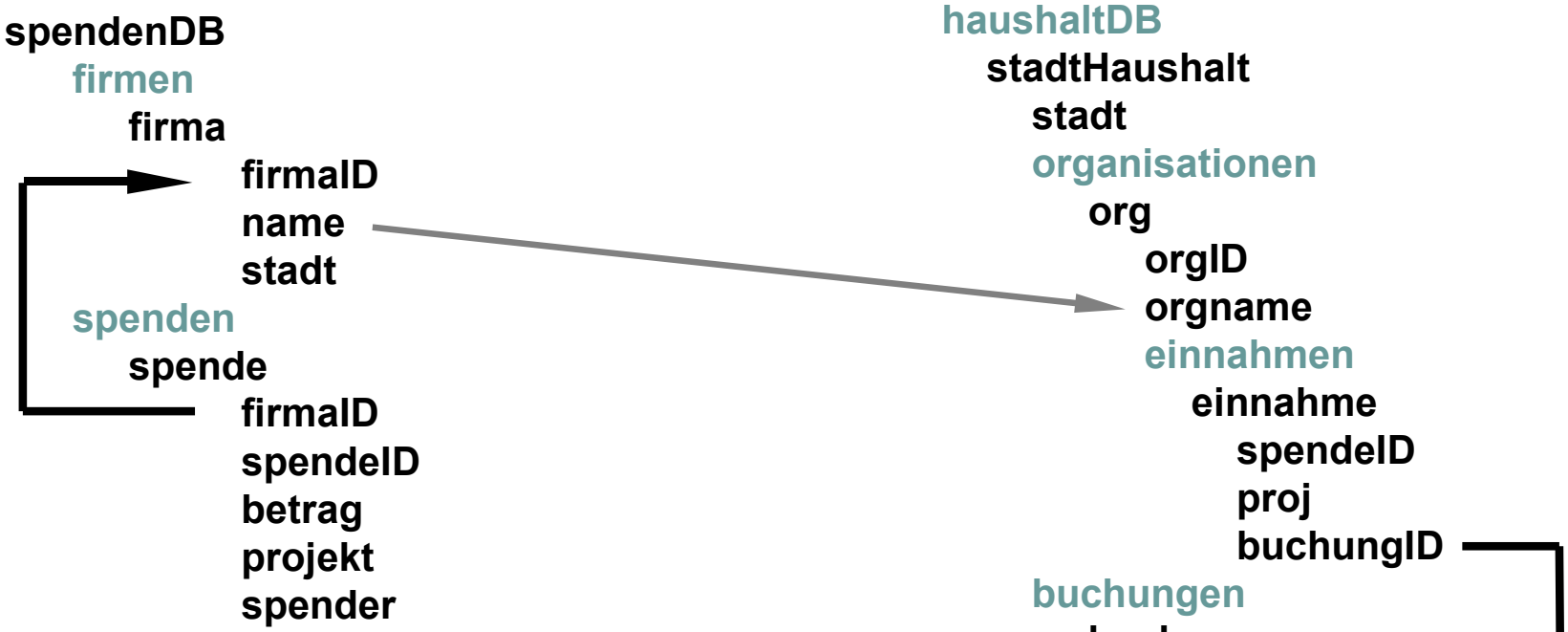
- Geschachtelte Strukturen
  - Geschachteltes relationales Modell
  - XML
  - Geschachtelte Integritätsbedingungen
- Korrespondenzen
  - Nutzerfreundlich
  - Automatische Entdeckung
- Intention des Nutzers erkennen und repräsentieren
- Semantik der Daten erhalten
  - Assoziationen entdecken & erhalten
  - Schemata und Integritätsbedingungen nutzen
- Neue Datenwerte erzeugen
- Korrekte Gruppierungen

Schema Mapping – Beispiel 1 [FHP+02]





Schema Mapping – Beispiel 1



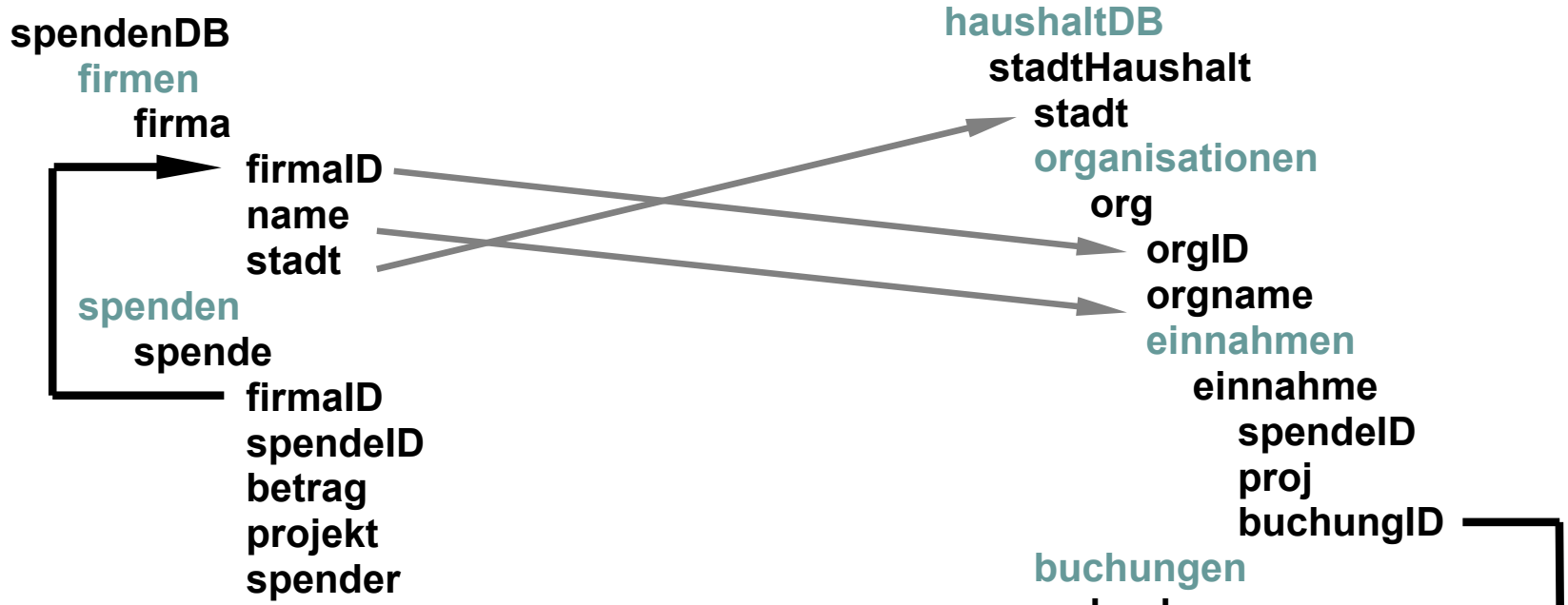
Interpretation:
- Erzeuge für jede **firma** in **spendenDB** eine **org** in **haushaltDB**
- **orgID** muss „erfunden“ werden
- **stadt** muss „erfunden“ werden

Erfinden von Werten

- Zwei Gründe zum Erfinden: non-null und Identität
- Non-null Werte:
  - Erfundener Wert (z.B. 'unbekannt') hat keinen Einfluß auf das log. Mapping nur auf die Korrektheit der resultierenden Daten
- ID Werte: Skolemfunktion
  - Input: *n* Werte (beliebige Domäne)
  - Output: bzgl. Input eindeutiger Wert (beliebiger Domäne)
  - Beispiel: Konkatenation aller Input-Werte als String

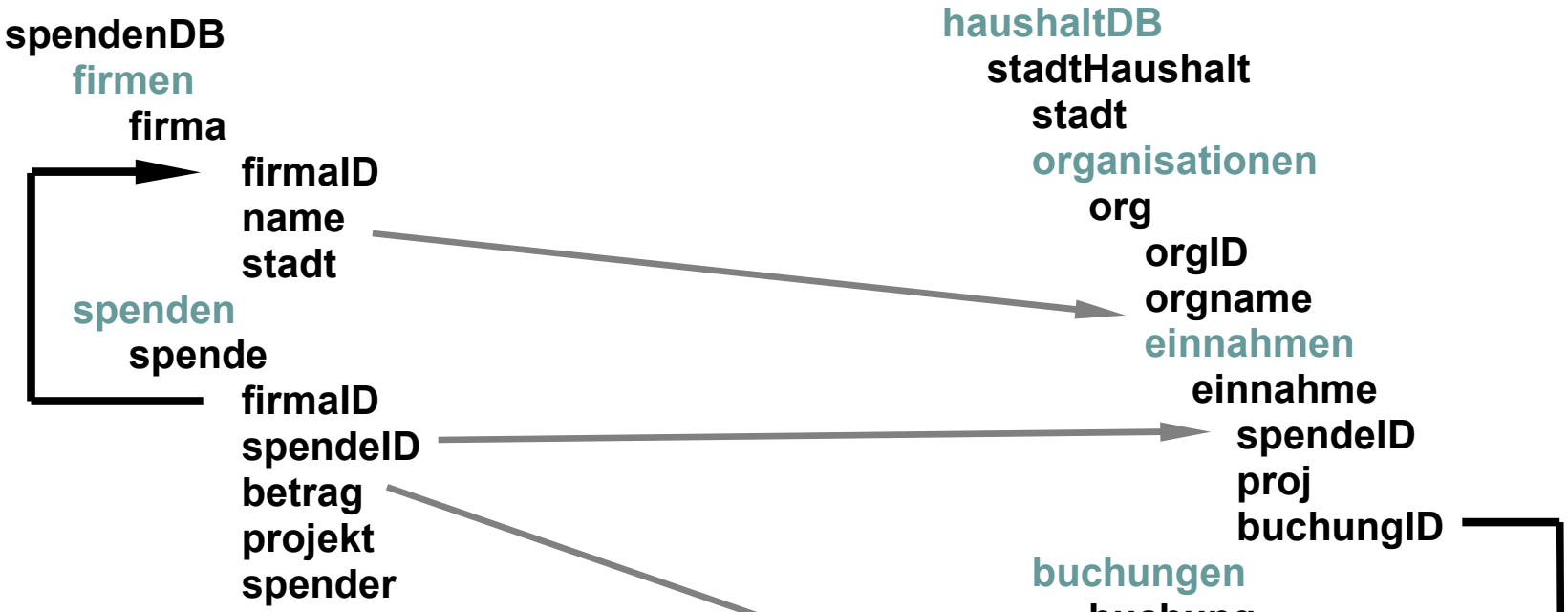
Wert für **org.orgID** nicht egal, sondern je nach **firma.name** eindeutig!

Schema Mapping – Beispiel 2



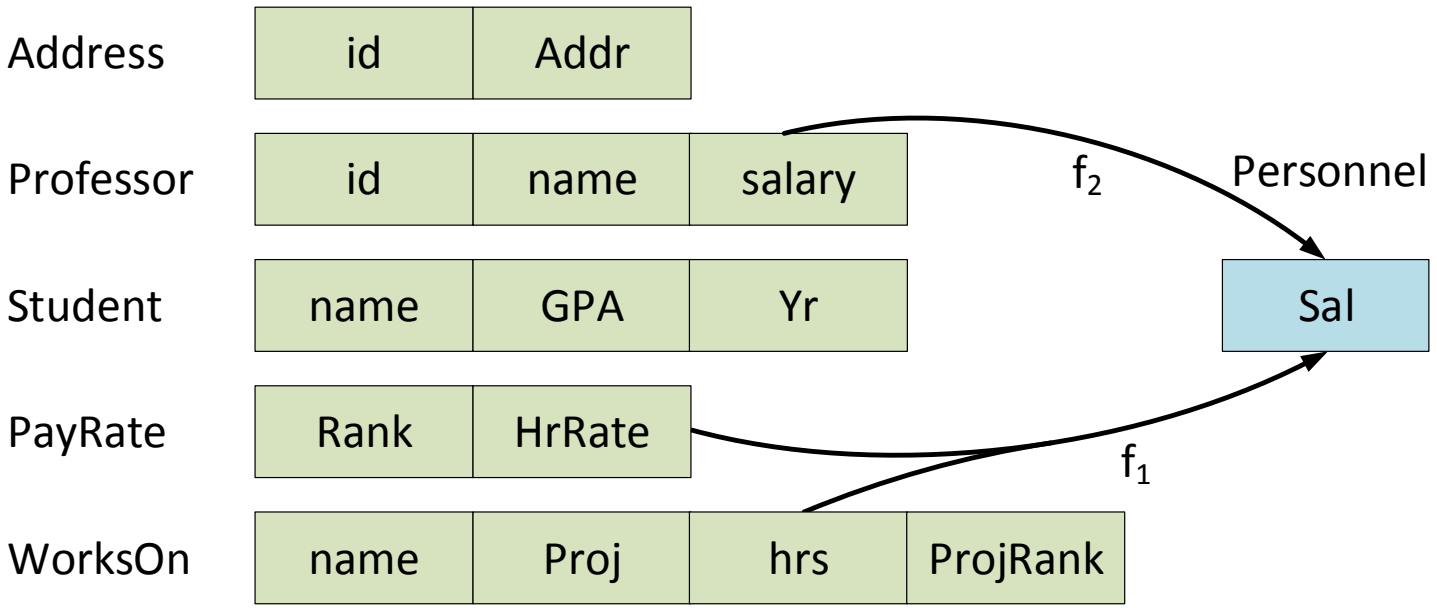
Interpretation:
- Erzeuge für jede **spendenDB.firma.stadt** ein **haushaltDB.stadthaushalt** mit gleichem Namen
- Gruppiere jede **firma** unter den entsprechenden **stadtHaushalt**

Schema Mapping – Beispiel 3

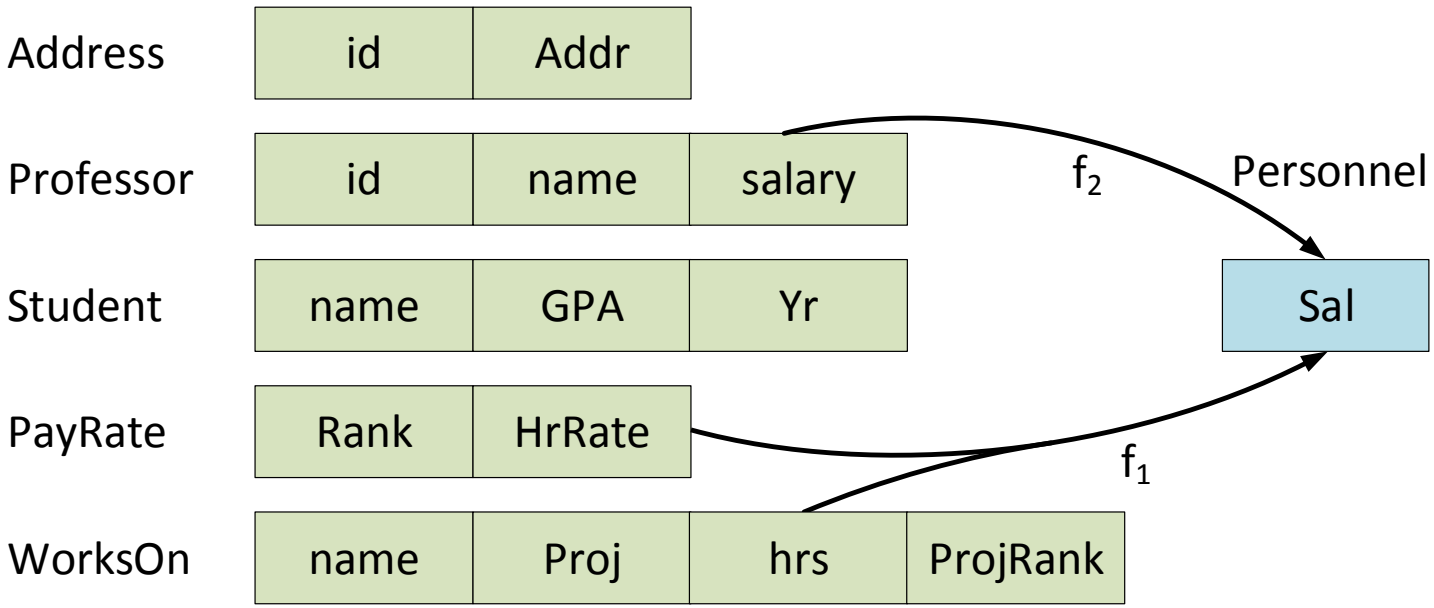


Interpretation:
- Erzeuge für jede **firma** in **spendenDB**.
  eine **org** in **haushaltDB**
- Erzeuge für jede **spende** in **spendenDB**
  eine **einnahme** in **haushaltDB**
- Erzeuge für jede **spende** in **spendenDB**
  eine **buchung** in **haushaltDB**
- Gruppiere korrekt: Schachtelung & Fremdschlüssel!

Schema Mapping – Beispiel 4 [DHI12]

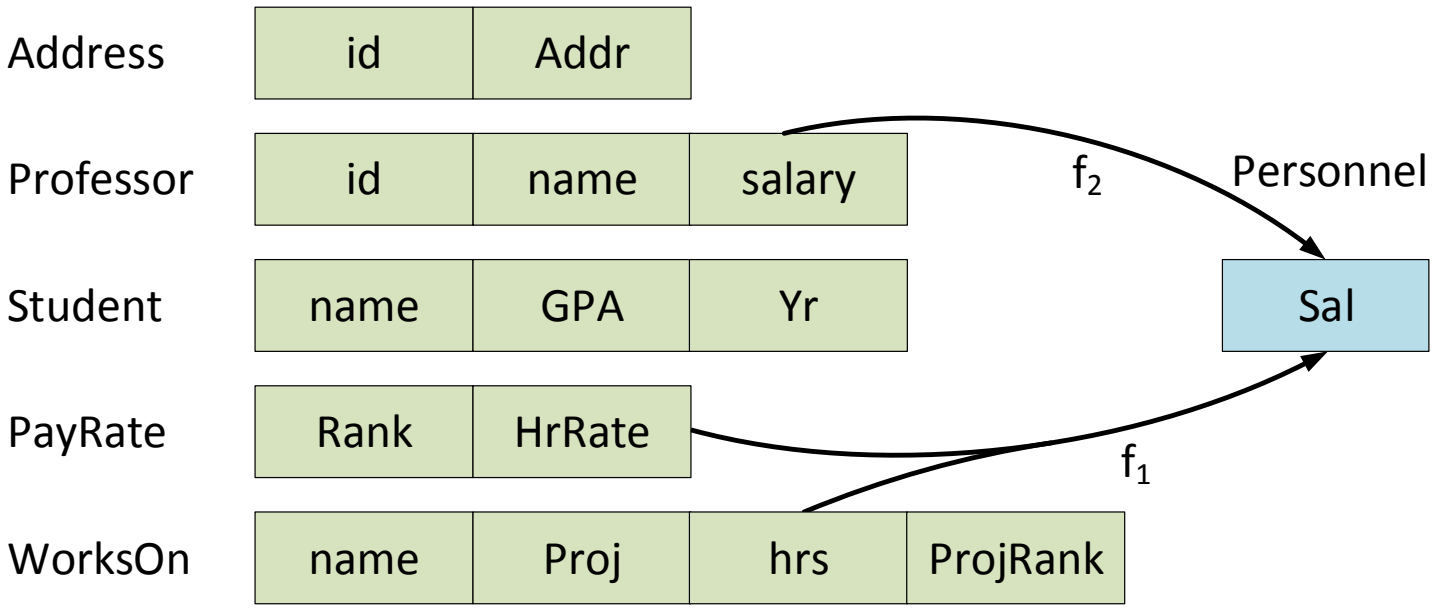


Schema Mapping – Beispiel 4 [DHI12]



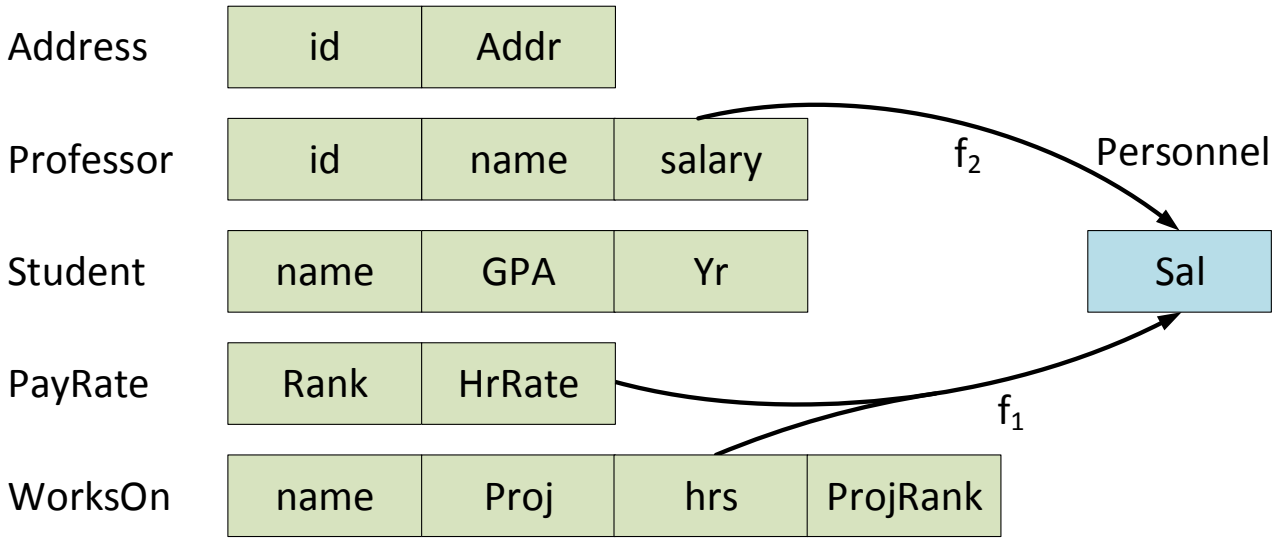
- Zwei Korrespondenzen:
  - $f_1 : PayRate(HrRate) \cdot WorksOn(Hrs) = Personnel(Sal)$
  - $f_2 : Professor(salary) = Personnel(Sal)$

Schema Mapping – Beispiel 4 [DHI12]



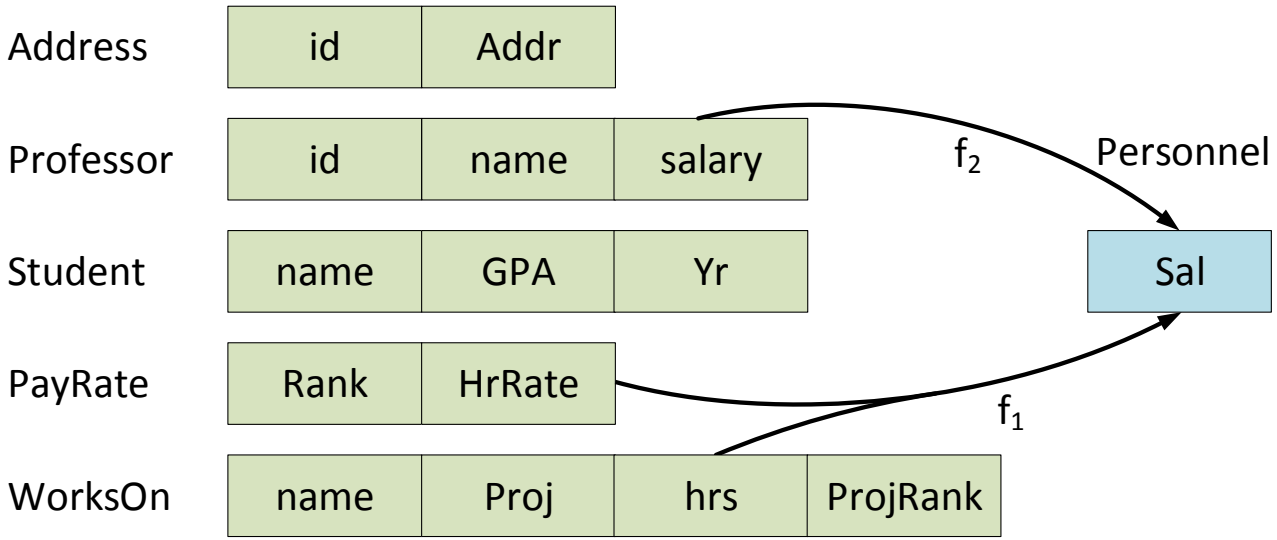
- Zwei Korrespondenzen:
  - $f_1 : PayRate(HrRate) \cdot WorksOn(Hrs) = Personnel(Sal)$
  - $f_2 : Professor(salary) = Personnel(Sal)$
- Frage 1: Wie müssen die Quellrelationen gejoint werden damit *HrRate* and *Hrs* multipliziert werden können?

Schema Mapping – Beispiel 4



Annahme: *WorksOn(ProjRank)* ist ein Fremdschlüssel auf *PayRate(Rank)*

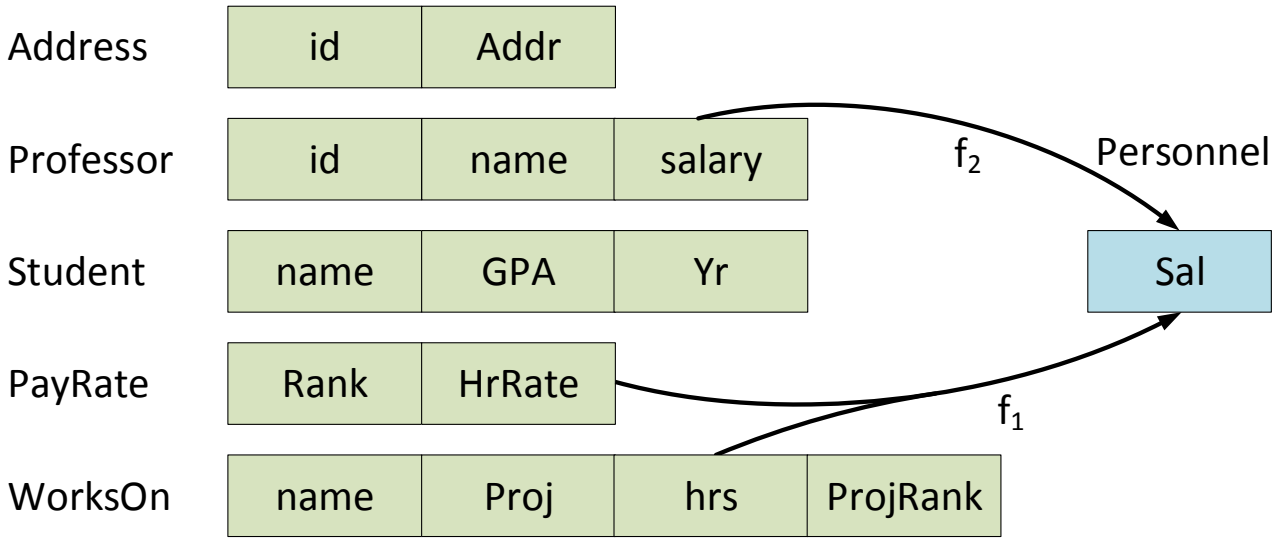
Schema Mapping – Beispiel 4



Annahme: *WorksOn(ProjRank)* ist ein Fremdschlüssel auf *PayRate(Rank)*

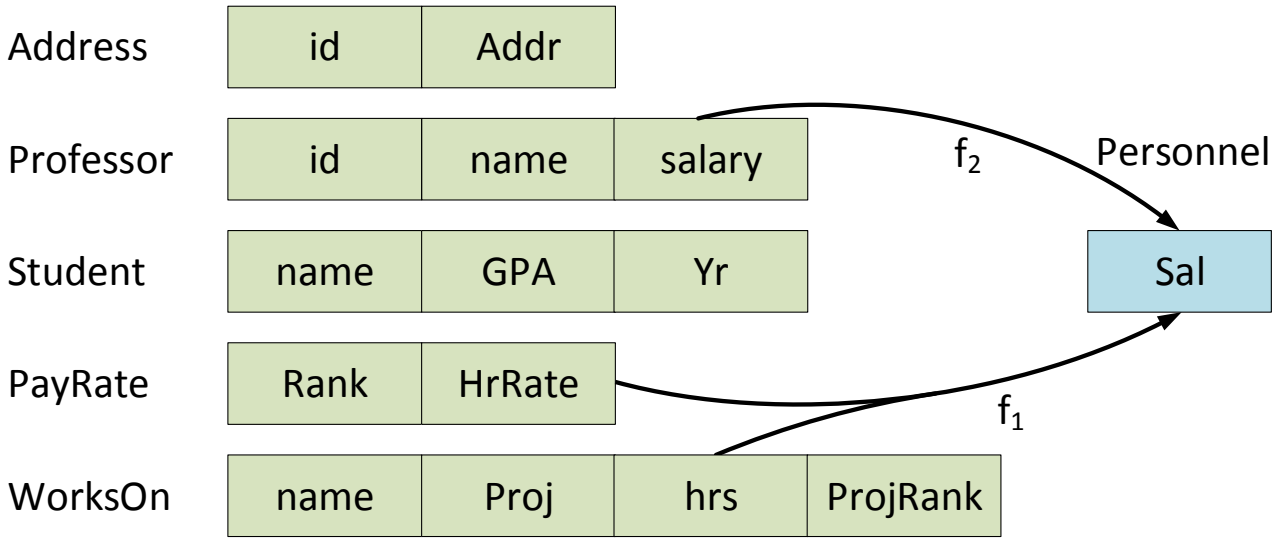
```
SELECT  P.HrRate * W.Hrs
FROM    PayRate P, WorksOn W
WHERE   P.Rank = W.ProjRank
```

Schema Mapping – Beispiel 4



Annahme: *WorksOn(name)* ist ein Fremdschlüssel auf *Student(name)* und *Student(Yr)* ist ein Fremdschlüssel auf *PayRate(Rank)*

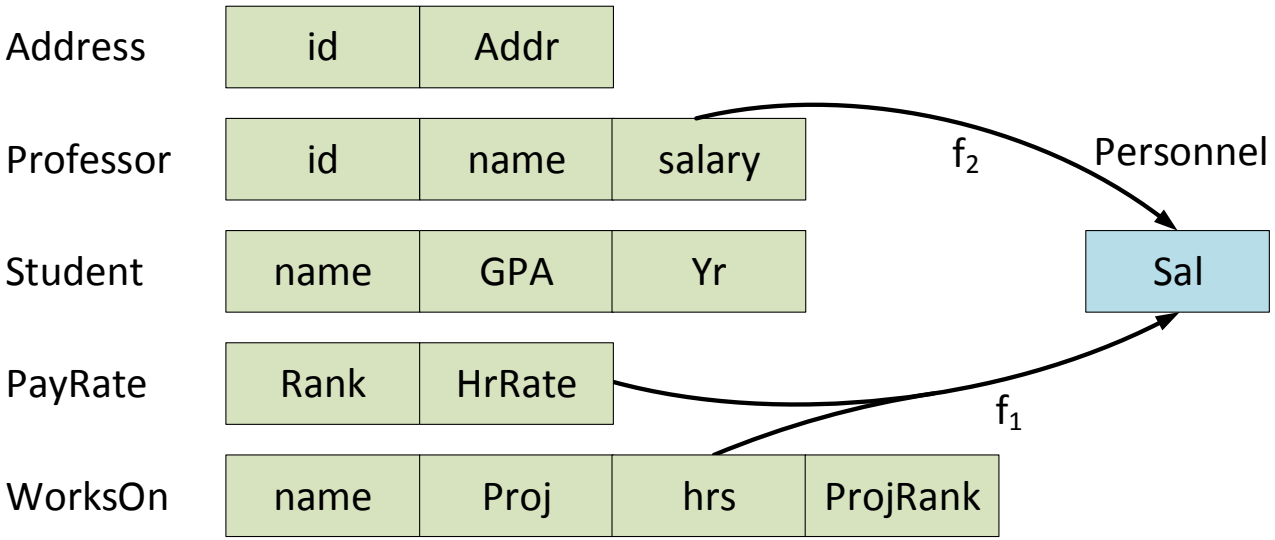
Schema Mapping – Beispiel 4



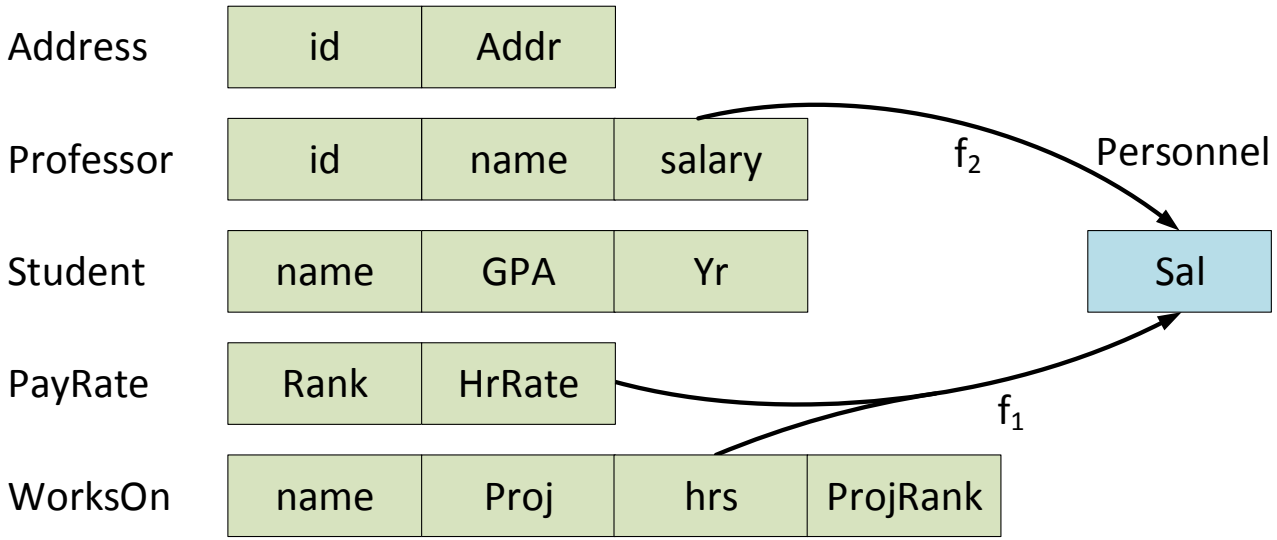
Annahme: *WorksOn(name)* ist ein Fremdschlüssel auf *Student(name)* und *Student(Yr)* ist ein Fremdschlüssel auf *PayRate(Rank)*

```
SELECT  P.HrRate * W.Hrs
FROM    PayRate P, WorksOn W, Student S
WHERE   W.name = S.name
AND     S.Yr = P.Rank
```

Schema Mapping – Beispiel 4

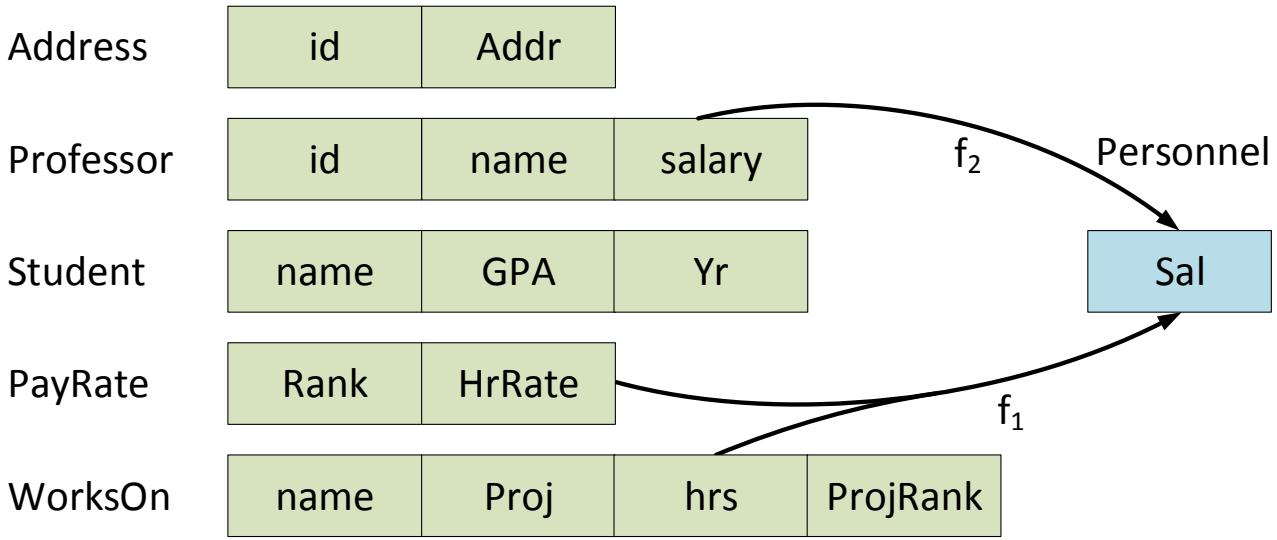


Schema Mapping – Beispiel 4



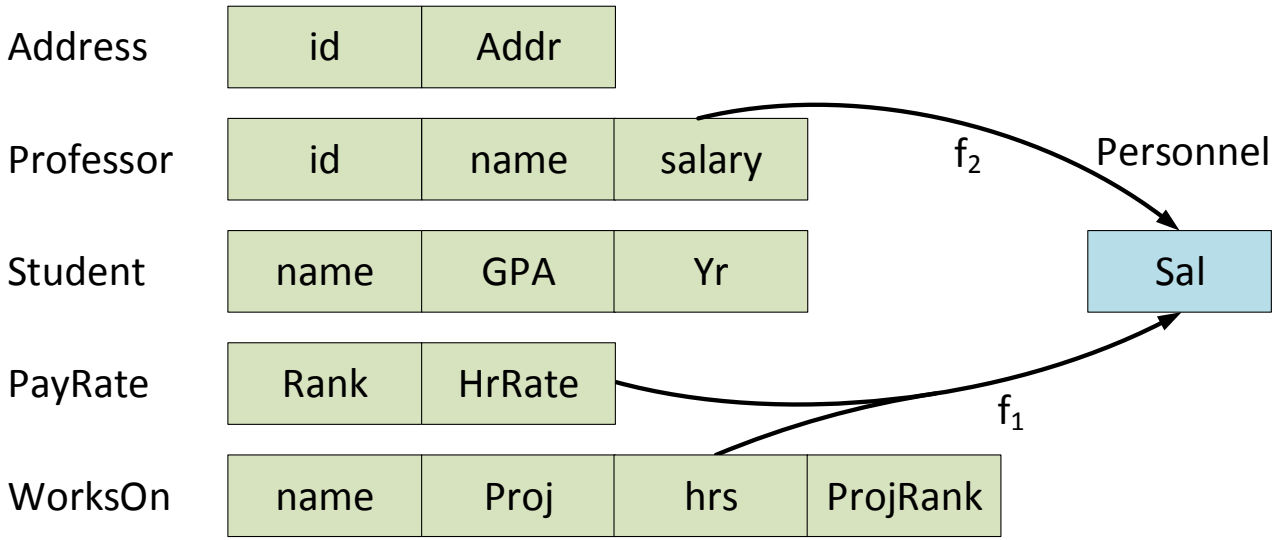
- Frage 2: Verhältnis zwischen den Korrespondenzen f_1 und f_2?

Schema Mapping – Beispiel 4



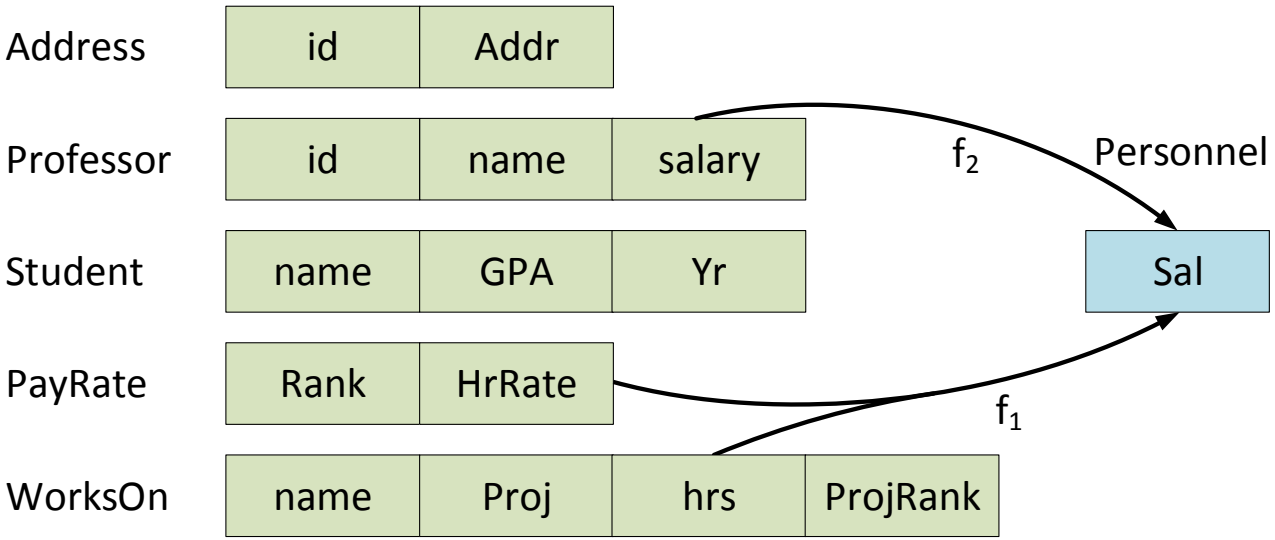
- Frage 2: Verhältnis zwischen den Korrespondenzen f_1 und f_2?
- JOIN der Relationen *Professor* und der generierten Anfrage von f_1 über das Gehalt?

Schema Mapping – Beispiel 4

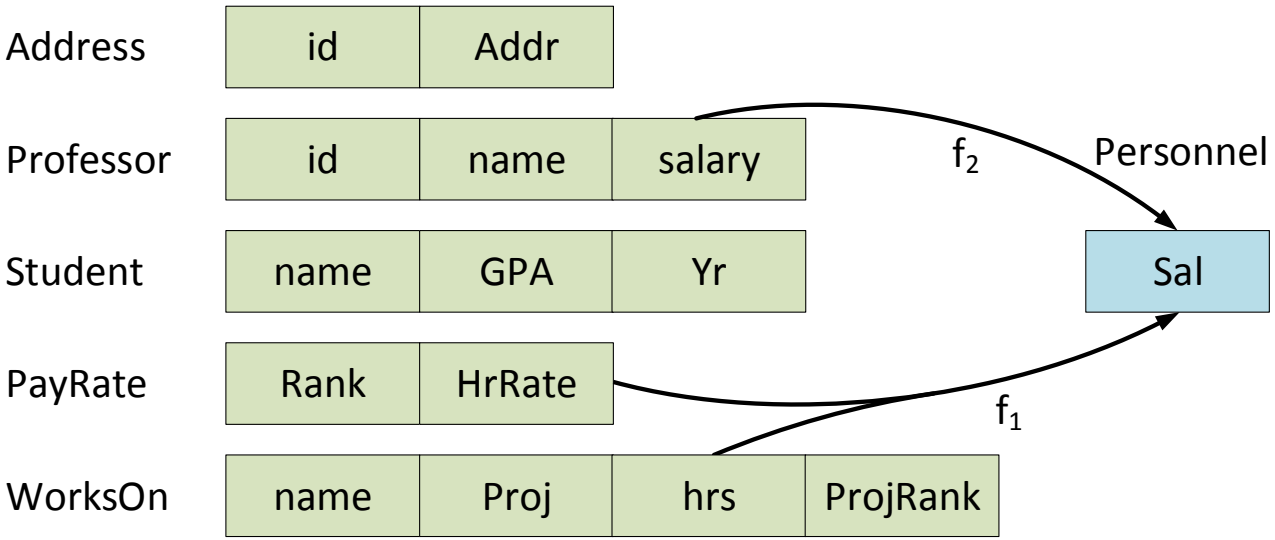


- Frage 2: Verhältnis zwischen den Korrespondenzen f_1 und f_2?
- JOIN der Relationen *Professor* und der generierten Anfrage von f_1 über das Gehalt?
- UNION beider Relationen?
  (Wenn ja normaler UNION oder OUTER UNION?)

Schema Mapping – Beispiel 4

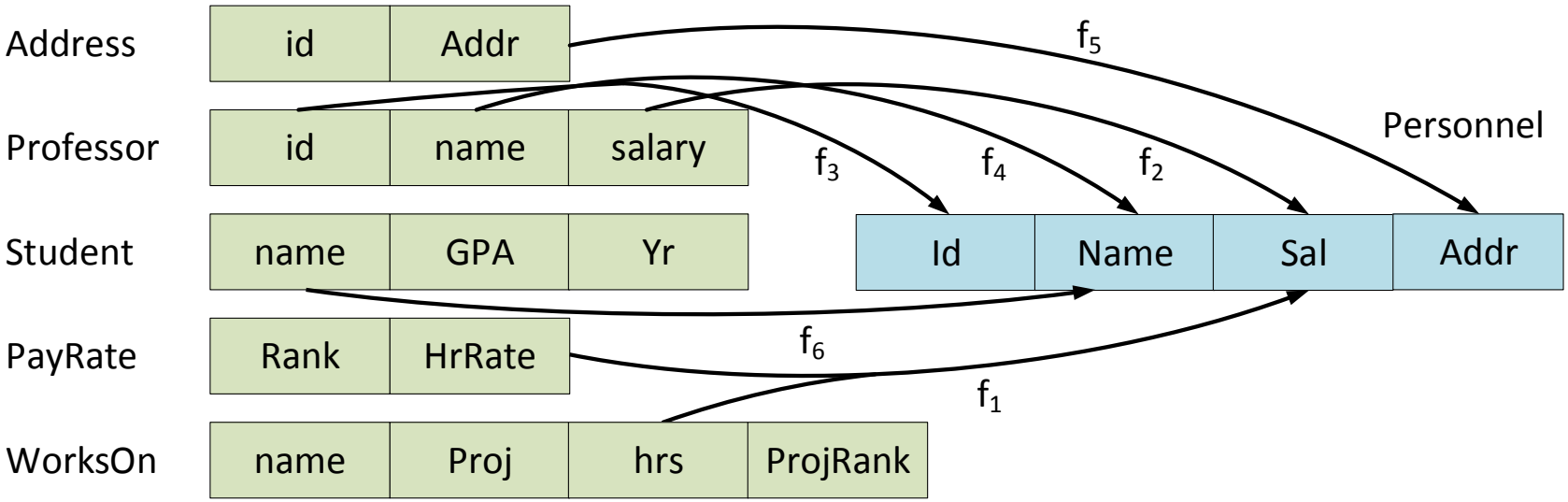


Schema Mapping – Beispiel 4



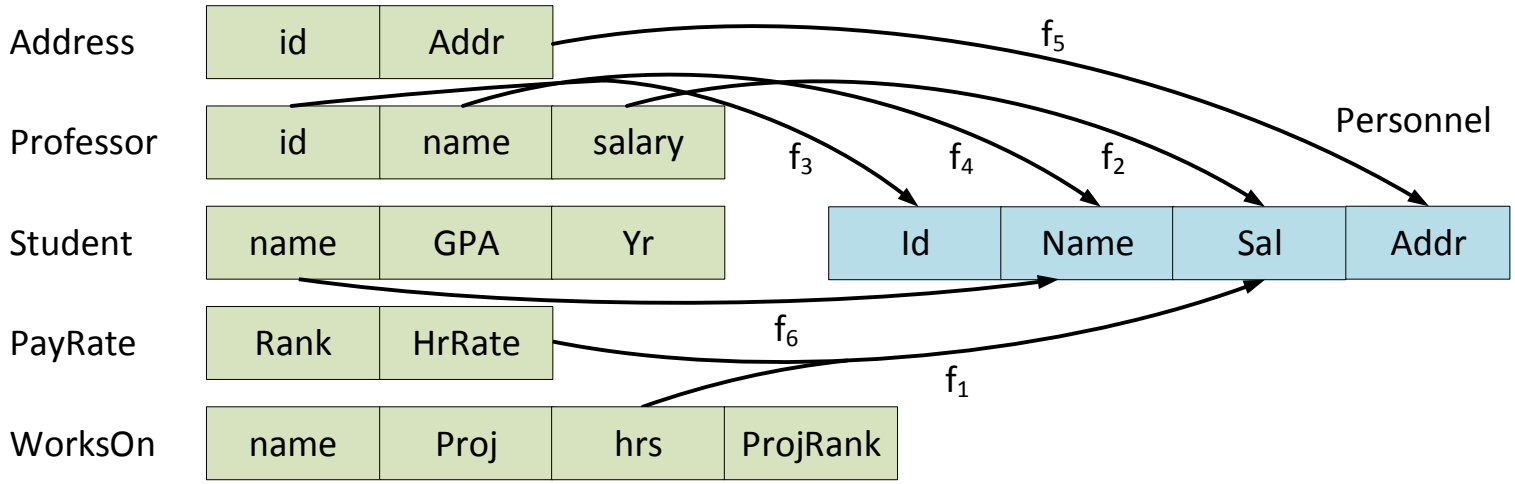

```
(SELECT P.HrRate * W.Hrs
FROM    PayRate P, WorksOn W
WHERE   P.Rank = W.ProjRank)

UNION ALL

(SELECT Salary
FROM    Professor)
```

Schema Mapping – Beispiel 5

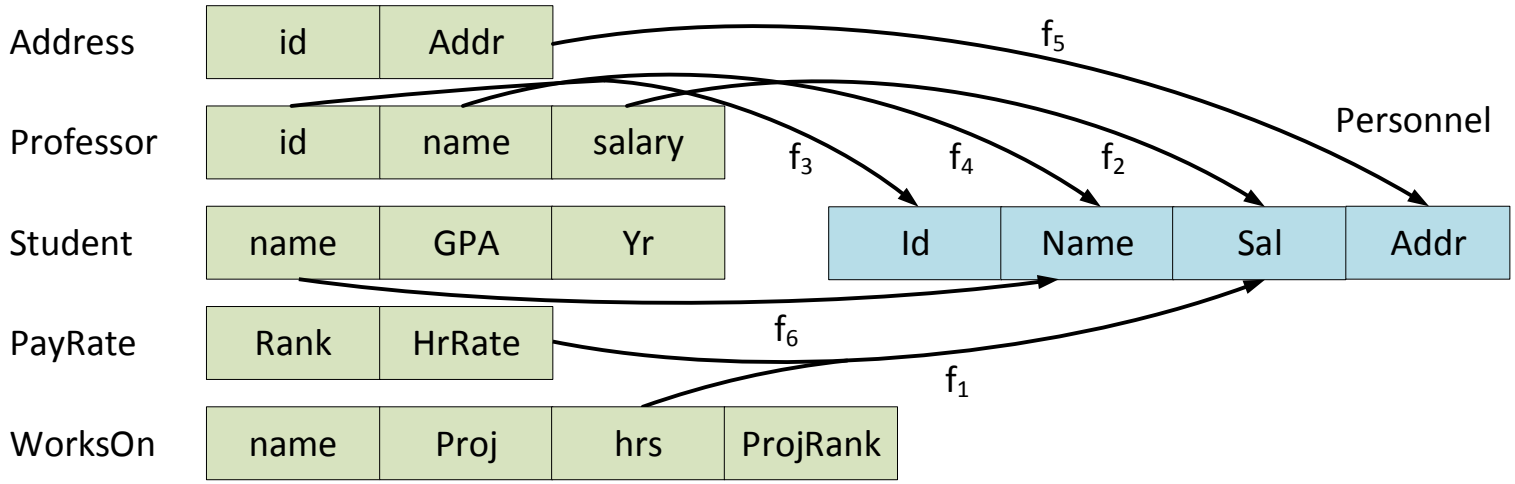


- $f_3 : Professor(id) = Personnel(Id)$
- $f_4 : Professor(name) = Personnel(name)$
- $f_5 : Address(Addr) = Personnel(Addr)$
- $f_6 : Student(name) = Personnel(name)$

Schema Mapping – Beispiel 5



Schema Mapping – Beispiel 5




```
(SELECT NULL AS Id, S.name, P.HrRate * W.Hrs, NULL AS Addr
FROM    PayRate P, WorksOn W, Student S
WHERE   P.Rank = W.ProjRank
And     S.name = W.name)

UNION ALL

(SELECT P.id, P.name, P.Salary, A.Addr
FROM    Professor P, Address A
WHERE   P.id = A.id)
```

Interpretation von logischen Mappings

- Zweistufige Vorgehensweise [DHI12]:
  - Welche Join Pfade sollen gewählt werden?
    (Jede Kombination aus High-Level Mapping und Join Pfad entspicht einem *Candidate Set*)
  - Wie sollen die Ergebnisse der einzelnen Joins kombiniert werden?
- Vermeidung von Informationsverlust:
  - Alle Werte in den Quellen sollten in die Zieldatenbank gemappt werden
  - Jeder Wert sollte nur einmal gemappt werden

Join Pfade

- Auffinden:
  - Fremdschlüsselbeziehungen folgen
  - Pfade aus Anfragen extrahieren (wie werden Relation in typischen Anfragen an die betreffende Quelle gejoint?)
  - Analyse der Datenbank nach joinbaren Attributen (ähnliche Wertmengen)

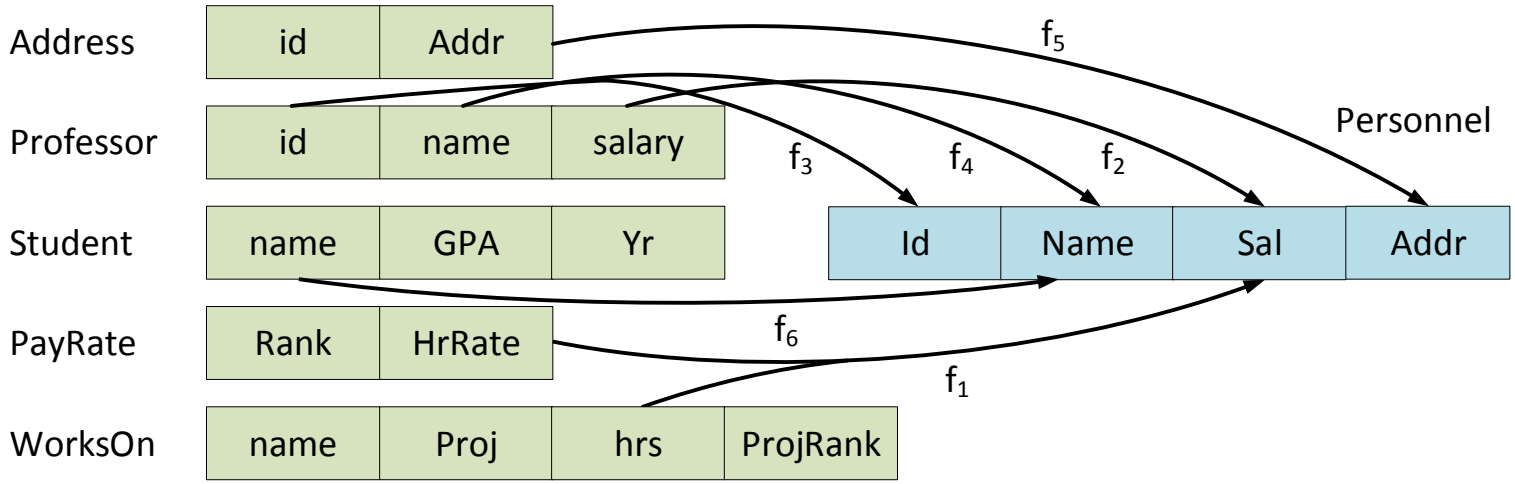
Join Pfade

- Auffinden:
  - Fremdschlüsselbeziehungen folgen
  - Pfade aus Anfragen extrahieren (wie werden Relation in typischen Anfragen an die betreffende Quelle gejoint?)
  - Analyse der Datenbank nach joinbaren Attributen (ähnliche Wertmengen)
- Auswahl:
  - Bevorzuge Fremdschlüsselpfade vor anderen
  - Bevorzuge Pfade bei denen der Unterschied zwischen Inner Joins und Outer Joins am geringsten ist (d.h. die meisten Tupel einen 'Join-Partner' haben)

Join Pfade

- Auffinden:
  - Fremdschlüsselbeziehungen folgen
  - Pfade aus Anfragen extrahieren (wie werden Relation in typischen Anfragen an die betreffende Quelle gejoint?)
  - Analyse der Datenbank nach joinbaren Attributen (ähnliche Wertmengen)
- Auswahl:
  - Bevorzuge Fremdschlüsselpfade vor anderen
  - Bevorzuge Pfade bei denen der Unterschied zwischen Inner Joins und Outer Joins am geringsten ist (d.h. die meisten Tupel einen 'Join-Partner' haben)
- Ergebnis: Candidate Sets

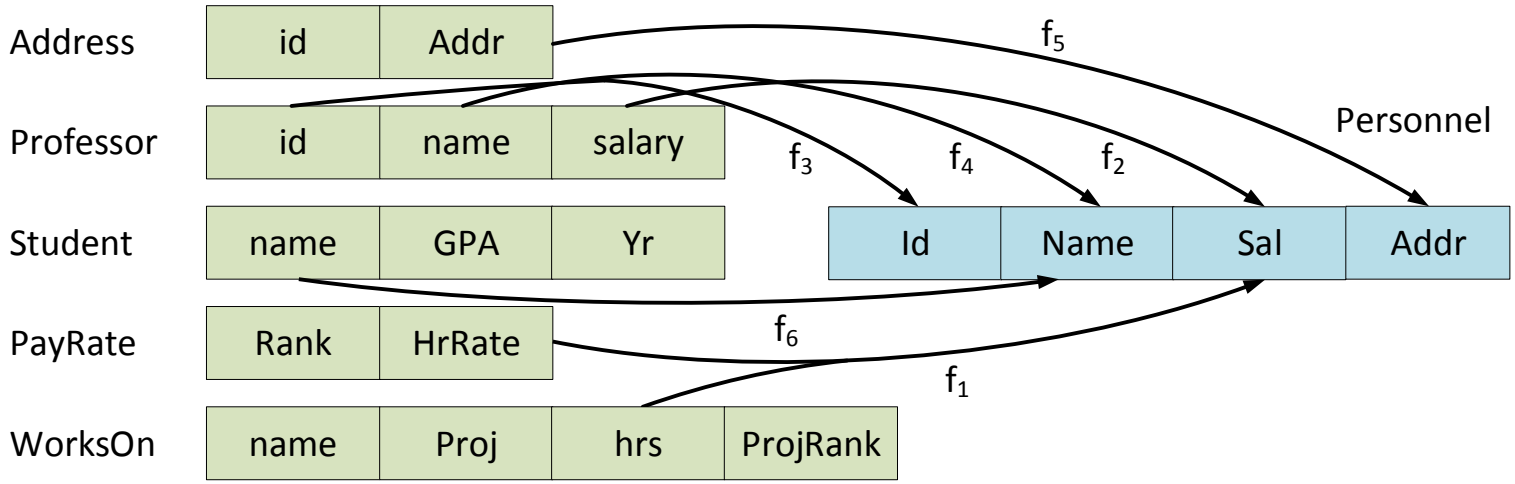
Schema Mapping – Beispiel 5



Plausible Join-Pfade der Länge 1:

- $Professor \bowtie_{id=id} Address$
- $(Professor \bowtie_{name=name} Student)$
- $Professor \bowtie_{name=name} WorksON$
- $Student \bowtie_{name=name} WorksON$
- $PayRate \bowtie_{Rank=ProjRank} WorksON$
- $(PayRate \bowtie_{Rank=Yr} Student)$

Schema Mapping – Beispiel 5



Plausible Join-Pfade der Länge 2:

- $(Address \bowtie_{id=id} Professor \bowtie_{name=name} Student)$
- $Address \bowtie_{id=id} Professor \bowtie_{name=name} WorksON$
- $PayRate \bowtie_{Rank=ProjRank} WorksON \bowtie_{name=name} Student$
- $(PayRate \bowtie_{Rank=Yr} Student \bowtie_{name=name} WorksON)$
- $(PayRate \bowtie_{Rank=Yr} Student \bowtie_{name=name} Professor)$
- $(Professor \bowtie_{name=name} Student \bowtie_{name=name} WorksON)$

Auswahl eines Candidate Covers

- Candidate Cover: Minimale Menge an Candidate Sets die eine maximal große Menge von zueinander konsistenten Eingabekorrespondenzen abdeckt

Auswahl eines Candidate Covers

- Candidate Cover: Minimale Menge an Candidate Sets die eine maximal große Menge von zueinander konsistenten Eingabekorrespondenzen abdeckt
  - Eine konsistente Menge von Eingabekorrespondenzen ist maximal wenn sie nicht erweiterbar ist (d.h. Zunahme einer Korrespondenz führt zu Verlust der Konsistenz)

  $\Rightarrow$ Covers können verschieden viele Korrespondenzen abdecken

Auswahl eines Candidate Covers

- Candidate Cover: Minimale Menge an Candidate Sets die eine maximal große Menge von zueinander konsistenten Eingabekorrespondenzen abdeckt
  - Eine konsistente Menge von Eingabekorrespondenzen ist maximal wenn sie nicht erweiterbar ist (d.h. Zunahme einer Korrespondenz führt zu Verlust der Konsistenz)

  $\Rightarrow$ Covers können verschieden viele Korrespondenzen abdecken
  - Beachte: einzelne Korrespondenzen können zu meheren Candidate Sets gehören und daher mehrmals im Cover vorkommen

Auswahl eines Candidate Covers

- Candidate Cover: Minimale Menge an Candidate Sets die eine maximal große Menge von zueinander konsistenten Eingabekorrespondenzen abdeckt
  - Eine konsistente Menge von Eingabekorrespondenzen ist maximal wenn sie nicht erweiterbar ist (d.h. Zunahme einer Korrespondenz führt zu Verlust der Konsistenz)

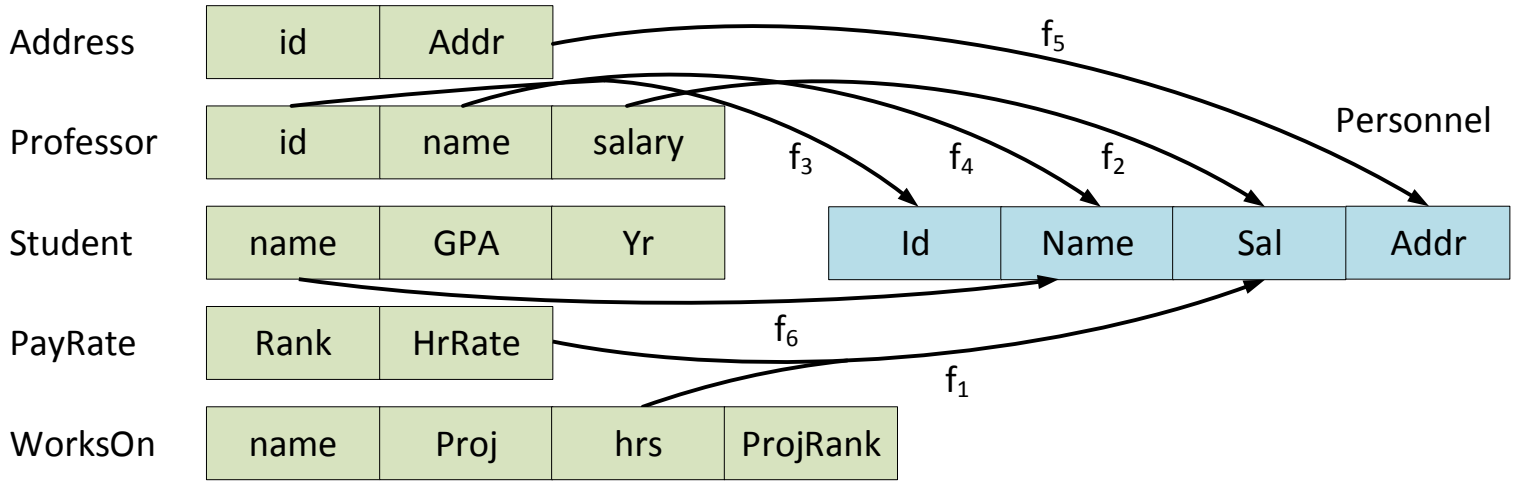
  $\Rightarrow$ Covers können verschieden viele Korrespondenzen abdecken
  - Beachte: einzelne Korrespondenzen können zu meheren Candidate Sets gehören und daher mehrmals im Cover vorkommen
- Auswahl des besten Candidate Covers:
  - Bevorzuge die Covers mit der kleinsten Anzahl an Candidate Sets (Annahme: Einfacherer Mappings sind natürlicher)
  - Bevorzuge die Covers welche die meisten Attribute des Zielschemas abdecken

Auswahl eines Candidate Covers

- Candidate Cover: Minimale Menge an Candidate Sets die eine maximal große Menge von zueinander konsistenten Eingabekorrespondenzen abdeckt
  - Eine konsistente Menge von Eingabekorrespondenzen ist maximal wenn sie nicht erweiterbar ist (d.h. Zunahme einer Korrespondenz führt zu Verlust der Konsistenz)

  $\Rightarrow$ Covers können verschieden viele Korrespondenzen abdecken
  - Beachte: einzelne Korrespondenzen können zu meheren Candidate Sets gehören und daher mehrmals im Cover vorkommen
- Auswahl des besten Candidate Covers:
  - Bevorzuge die Covers mit der kleinsten Anzahl an Candidate Sets (Annahme: Einfacherer Mappings sind natürlicher)
  - Bevorzuge die Covers welche die meisten Attribute des Zielschemas abdecken
- Ergebnis: Ein einzelnes Candidate Cover

Schema Mapping – Beispiel 5



Plausibles Cover:

- $Address \bowtie_{id=id} Professor \bowtie_{name=name} WorksON$
- $PayRate \bowtie_{Rank=ProjRank} WorksON \bowtie_{name=name} Student$

Anfragegenerierung (GaV)

- Gegeben: Candidate Cover (= Menge an Candidate Sets)

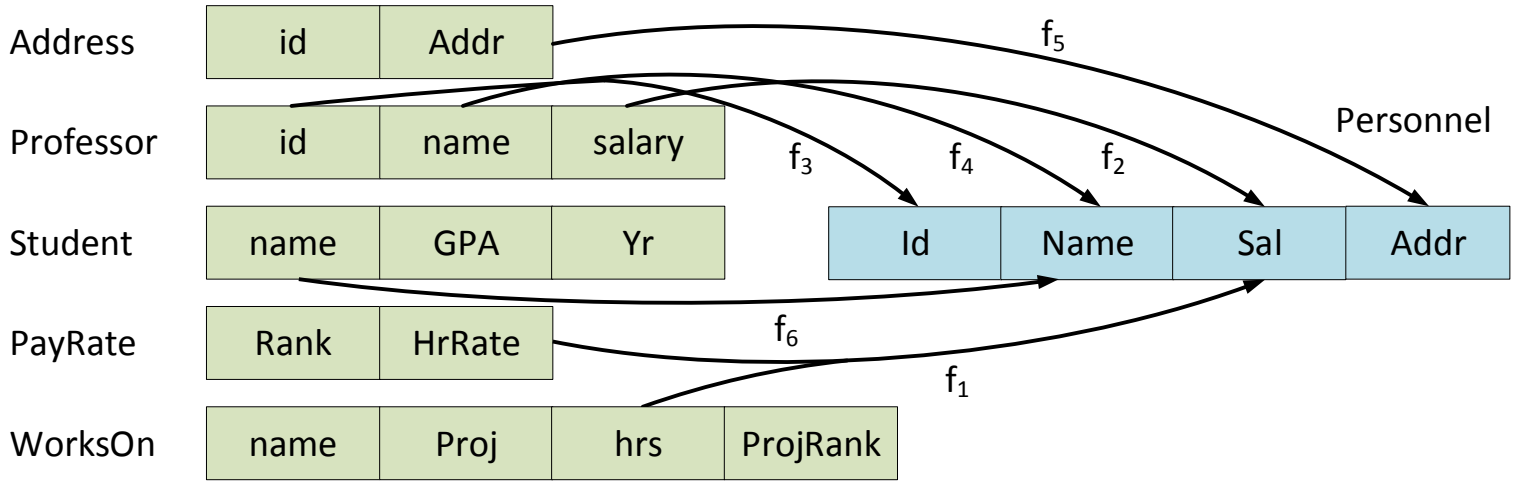
Anfragegenerierung (GaV)

- Gegeben: Candidate Cover (= Menge an Candidate Sets)
- Schritt 1: Erstelle eine Anfrage per Candidate Set
  - SELECT-Klausel: Alle Attribute des Zielschemas welche im Candidate Set vorkommen (bzw. ihre Korrespondenzen aus dem Quellschema)
  - FROM-Klausel: Jede Relation des Join-Pfades
  - WHERE-Klausel: Join Bedingungen + globale Integritätsbedingungen

Anfragegenerierung (GaV)

- Gegeben: Candidate Cover (= Menge an Candidate Sets)
- Schritt 1: Erstelle eine Anfrage per Candidate Set
  - SELECT-Klausel: Alle Attribute des Zielschemas welche im Candidate Set vorkommen (bzw. ihre Korrespondenzen aus dem Quellschema)
  - FROM-Klausel: Jede Relation des Join-Pfades
  - WHERE-Klausel: Join Bedingungen + globale Integritätsbedingungen
- Schritt 2: Kombiniere die erstellten Anfragen durch UNION (bzw. OUTER UNION oder UNION ALL)

Schema Mapping – Beispiel 5



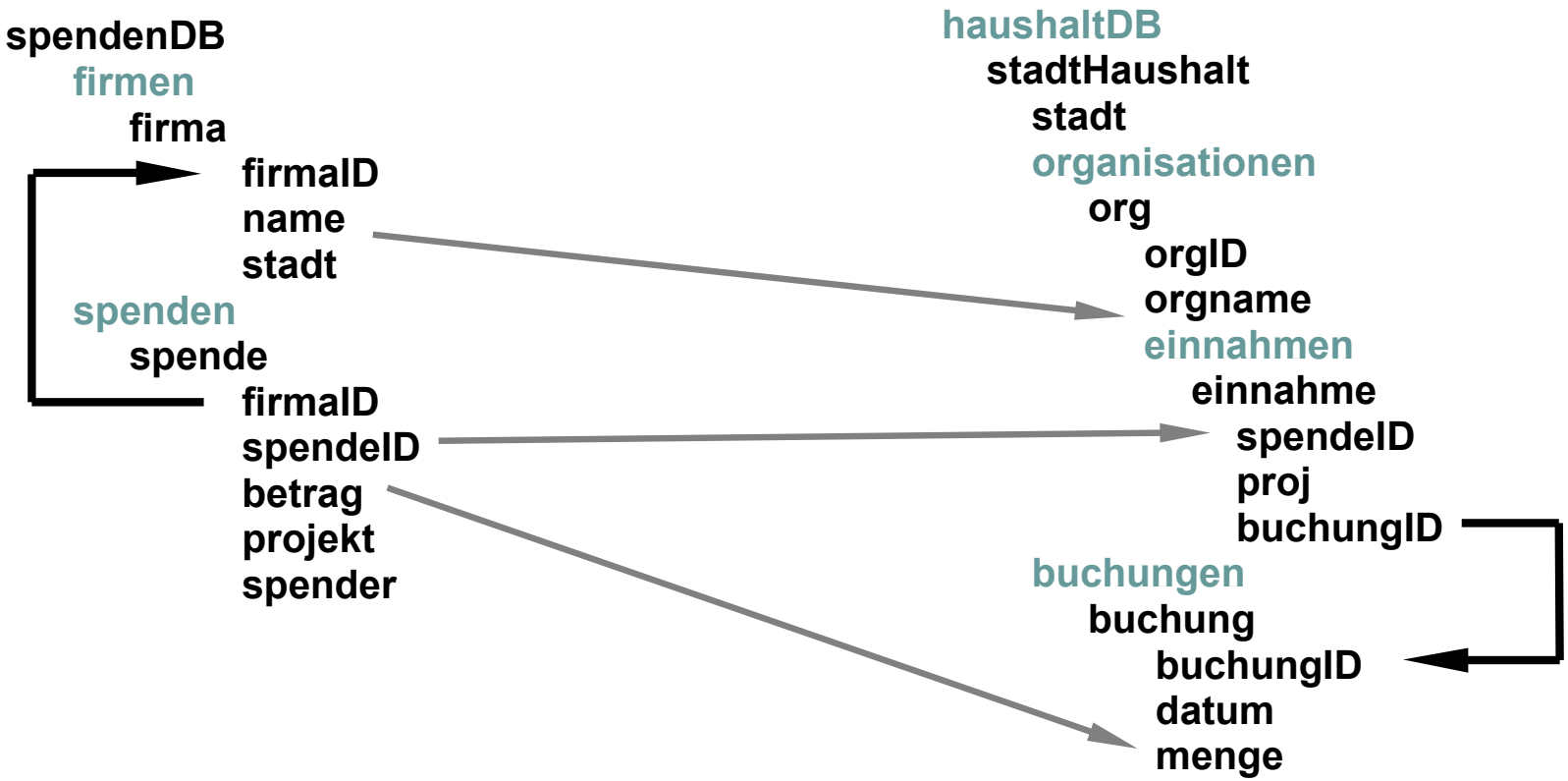
Plausible Anfrage:

$$(\pi_{\ldots}(Address \bowtie_{id=id} Professor \bowtie_{name=name} WorksON))$$
$$\cup \quad (\pi_{\ldots}(PayRate \bowtie_{Rank=ProjRank} WorksON \bowtie_{name=name} Student))$$

Schema Mapping – Assoziations-basierter Ansatz

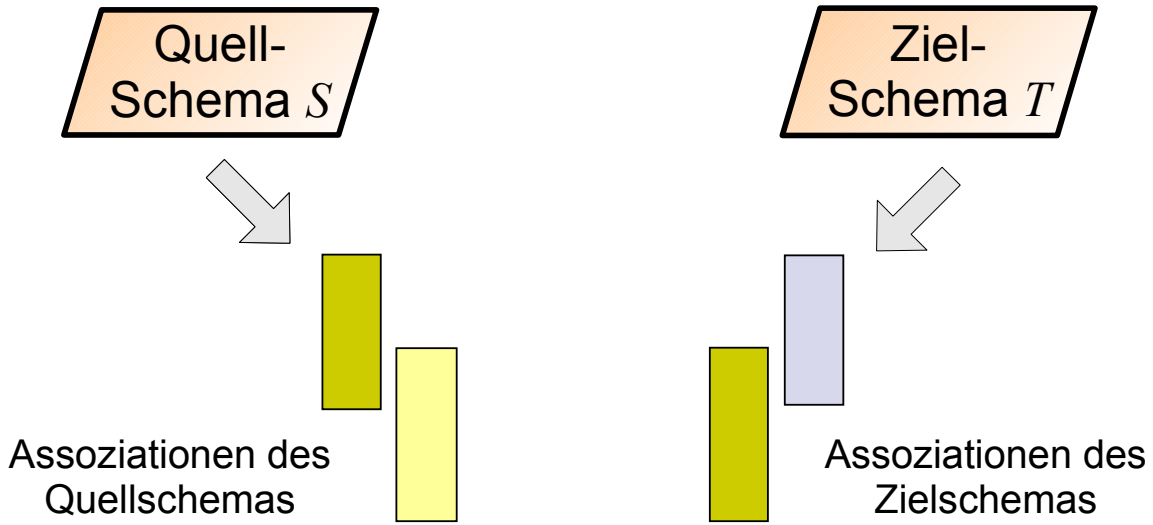
Drei Schritte:

1. Entdeckung von intra-Schema Assoziationen
2. Entdeckung von inter-Schema logischen Mappings
3. Anfrageerzeugung



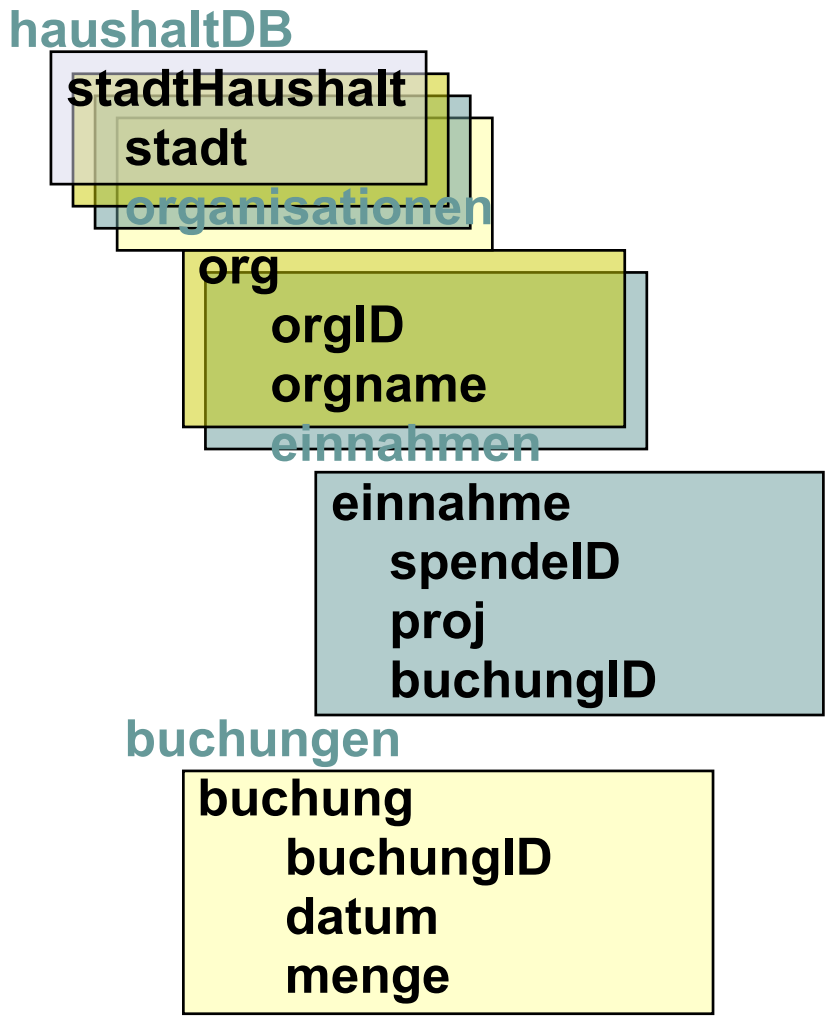
Schritt 1: Entdeckung von Assoziationen

- Intra-schema Assoziationen zwischen Schemaelementen (vergleichbar mit Join-Pfaden)
- Attribute einer Relation sind assoziiert
- Über Fremdschlüssel verbundene Relationen sind assoziiert
- Unabhängig vom Mapping (aber nur 'gemappte' Elemente)



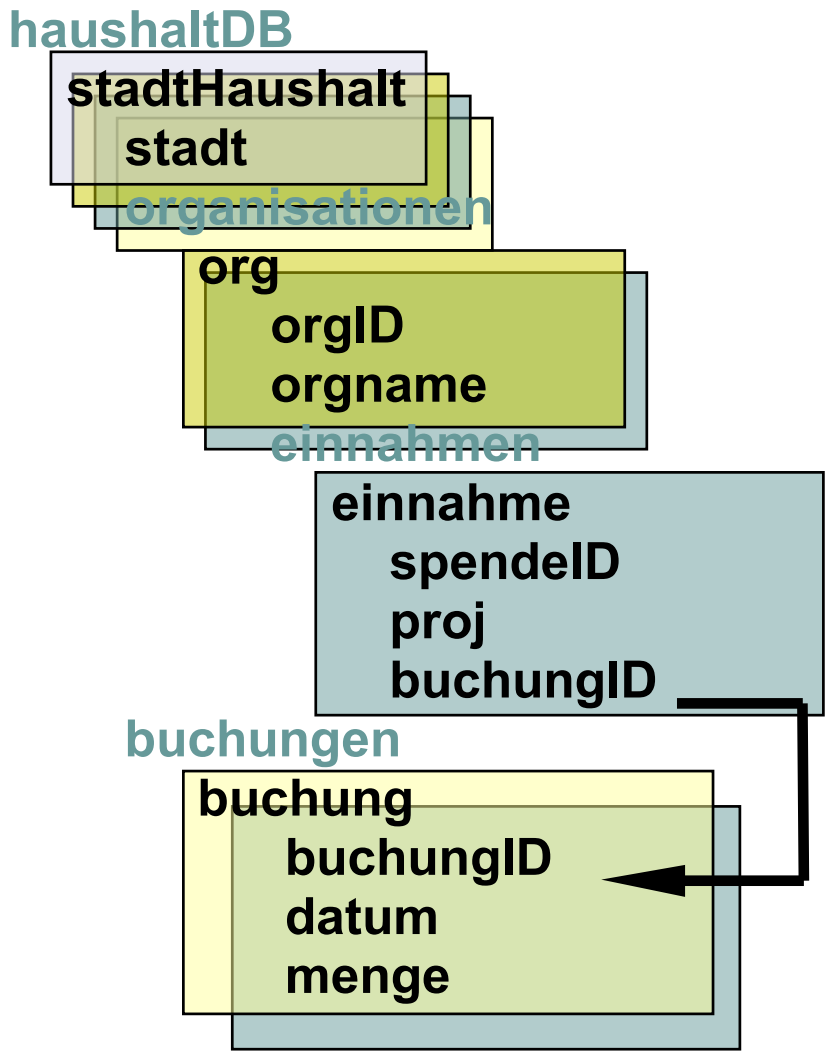
Entdeckung von Assoziationen

- Start: Alle primären Pfade
  - Assoziationen im Schema ohne Integritätsbedingungen
- Relationale Schemas:
  - Jede Relation entspricht einem primären Pfad
- Geschachtelte Schemas:
  - Attribute einer Ebene
  - Attribute geschachtelter Ebenen



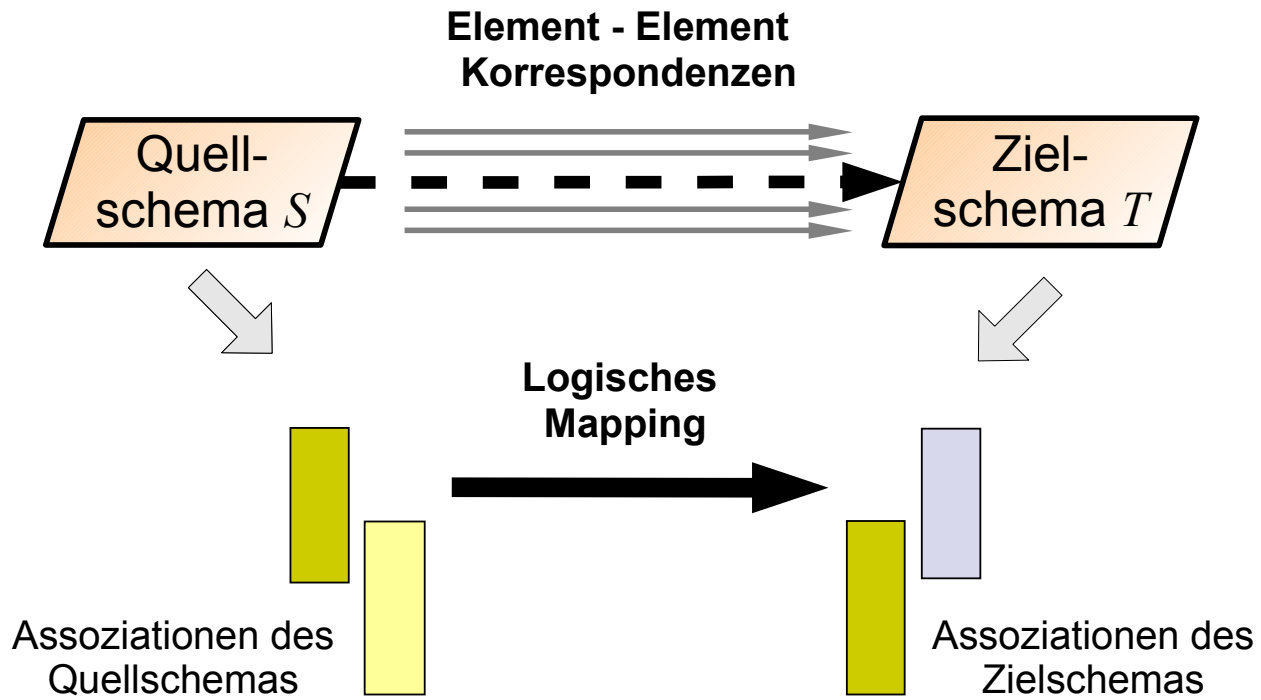
Entdeckung von Assoziationen

- Betrachte nun Schlüssel / Fremdschlüssel (ICs)
- Logische Relation
  - Erweitere jeden primären Pfad durch 'Verfolgen' der ICs (*chase*)

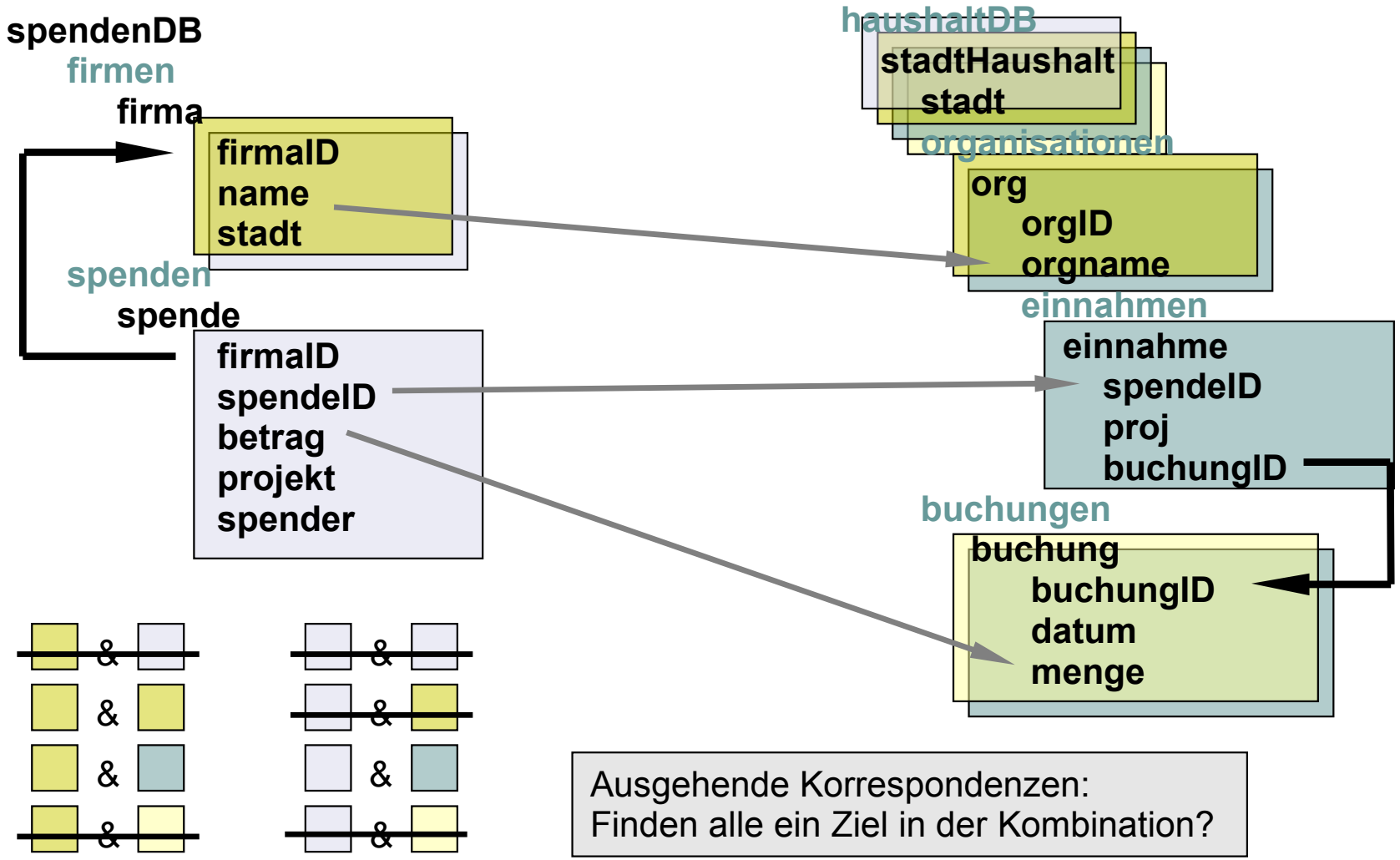


Schritt 2: Entdeckung von logischen Mappings

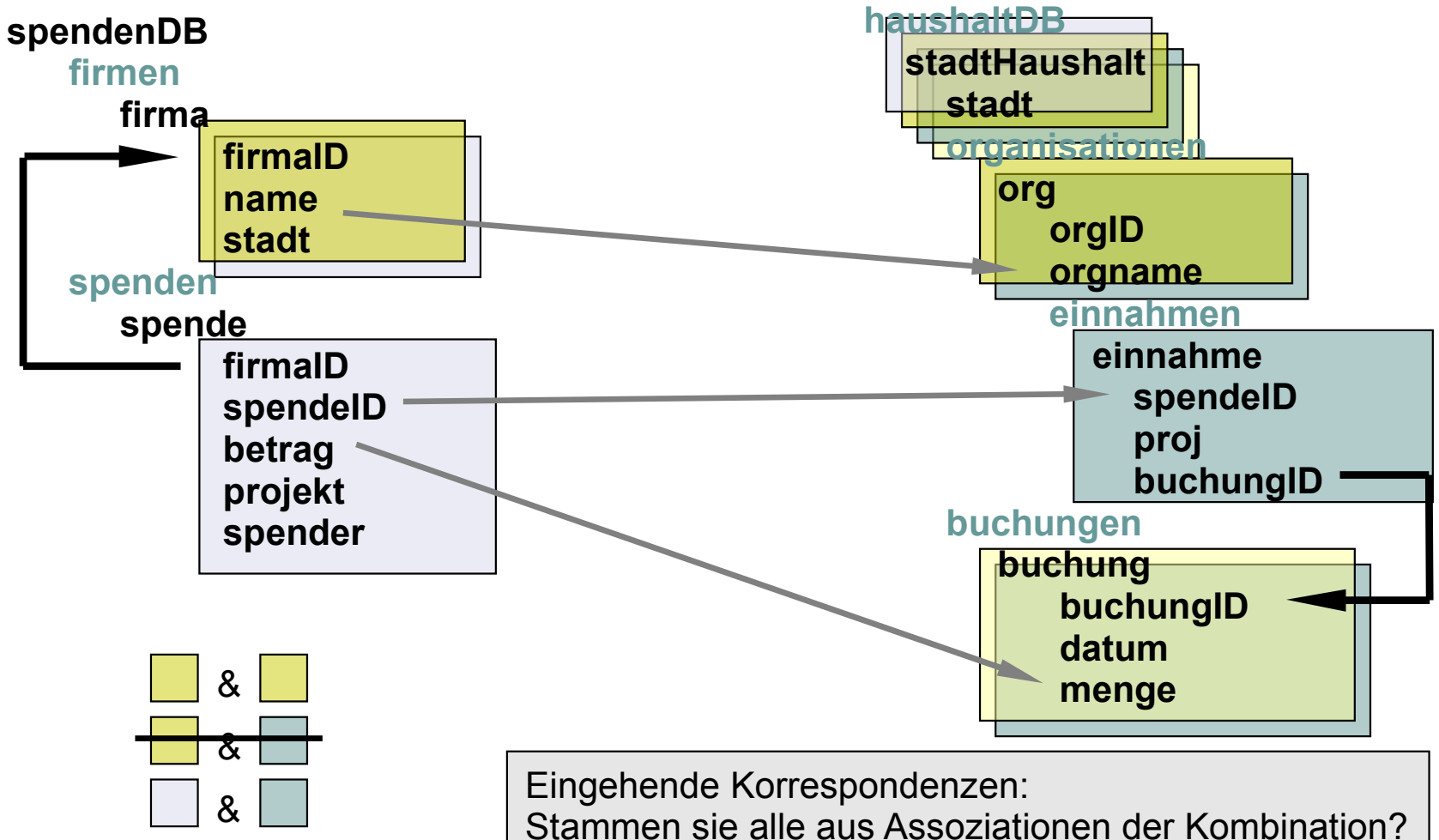
- Entdecke logische Mappings zwischen Quell- und Zielschema
- Betrachte alle Kombinationen aus Assoziationen des Quellschemas und Assoziationen des Zielschemas
- Interessant sind nur Kombinationen, deren Assoziationen durch Korrespondenzen verbunden sind



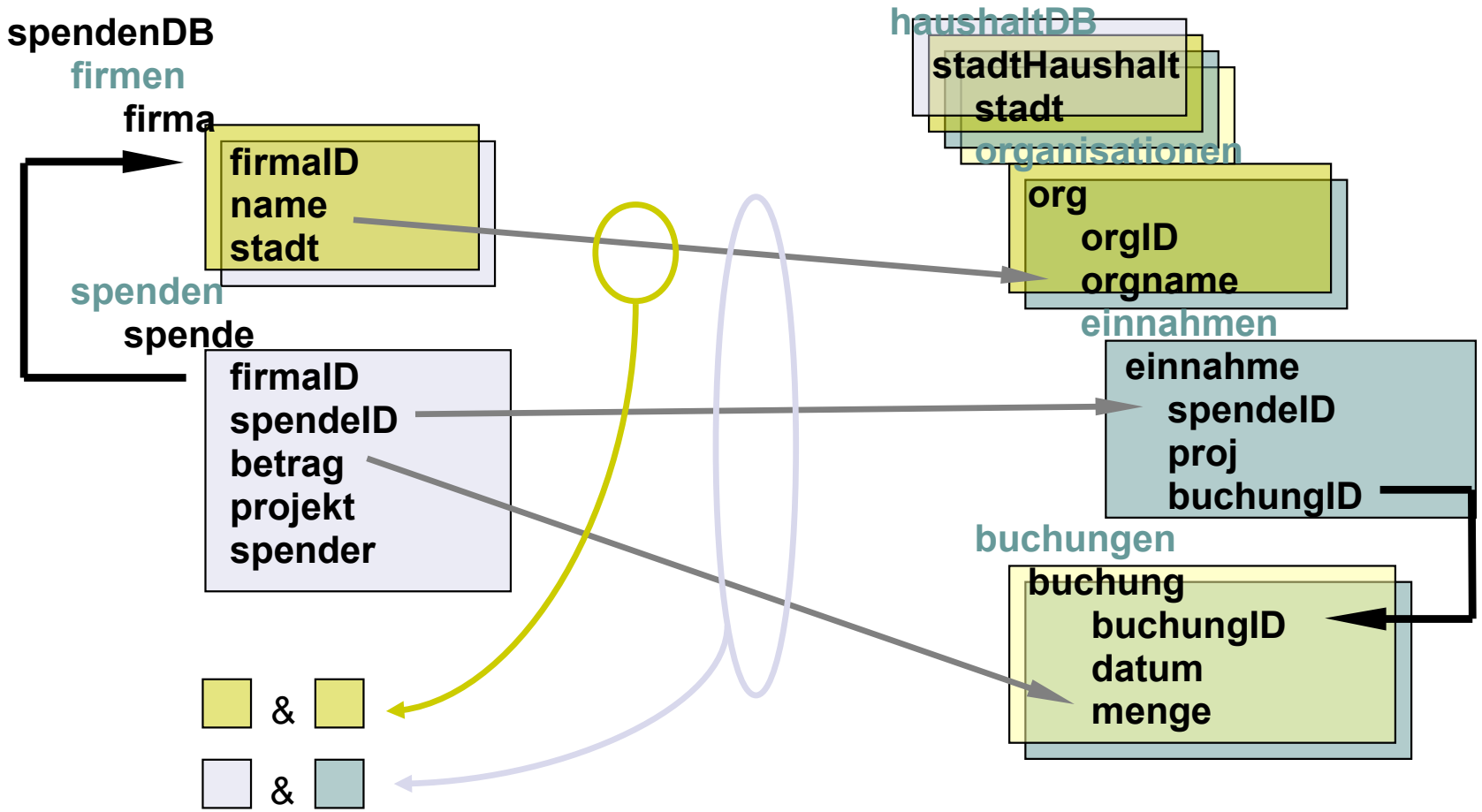
Entdeckung von logischen Mappings



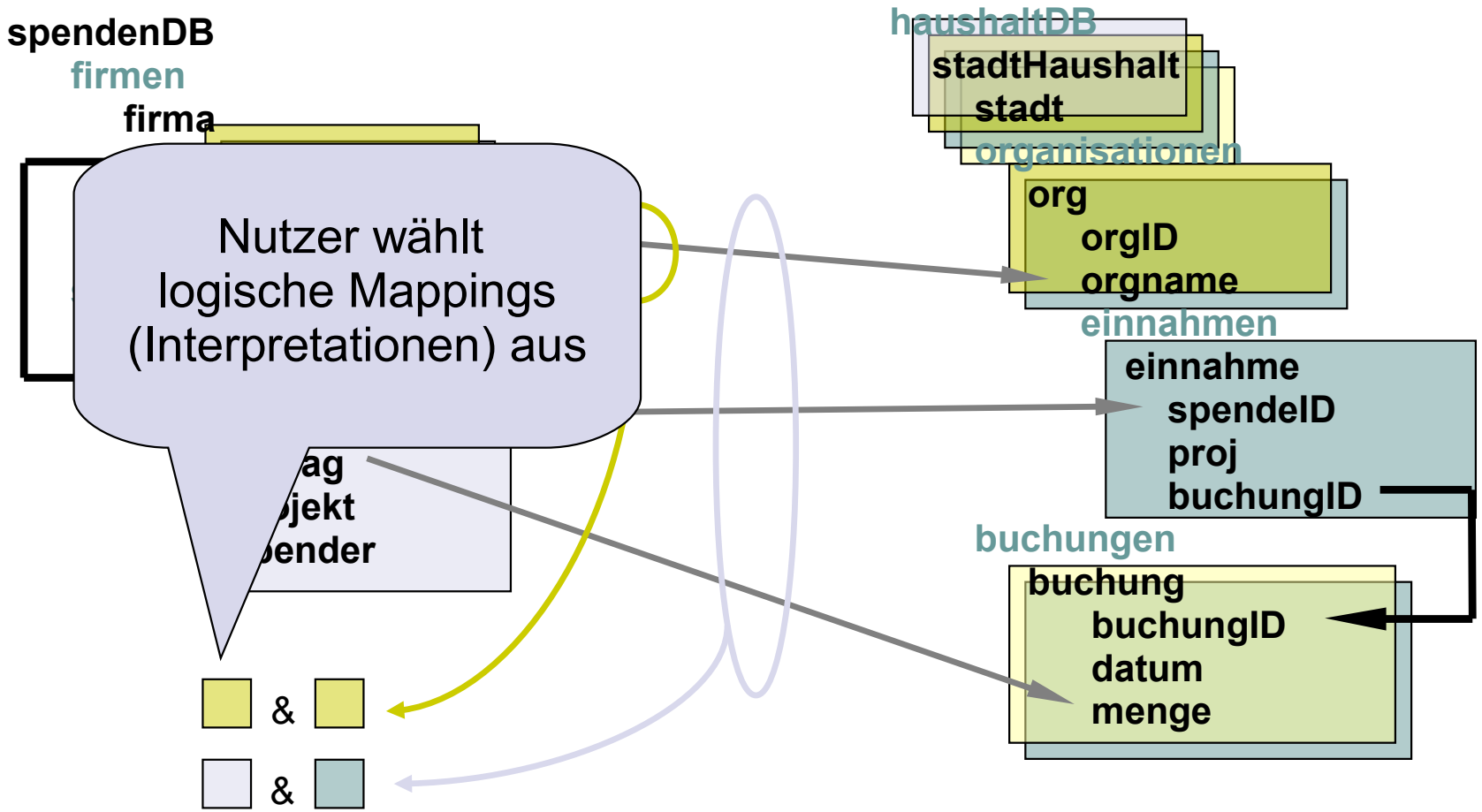
Entdeckung von logischen Mappings



Input des Nutzers



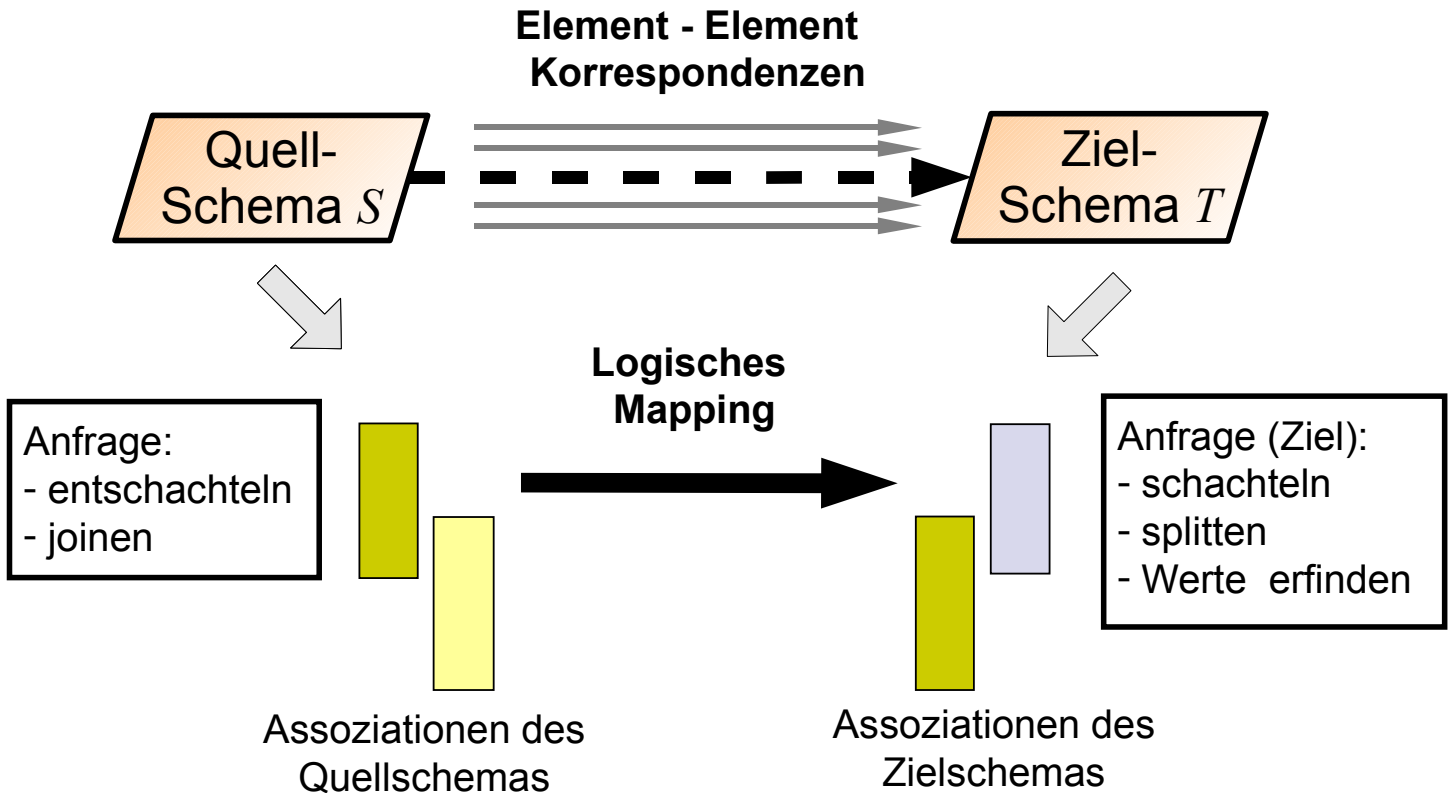
Input des Nutzers



Schritt 3: Erzeugung der Anfragen

Erzeuge für jedes (ausgewählte) logische Mapping eine Anfrage

- Auswahl und verknüpfen der entsprechenden Quelldaten
- Generierung der entsprechenden Zieldaten

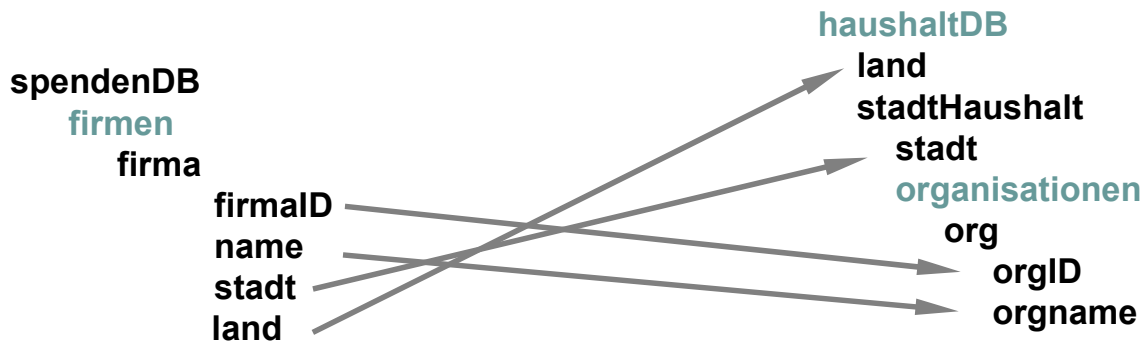


Erzeugung der Anfragen – Probleme

- Erfinden von Werten
  - NULL-Werte nicht immer ausreichend
  - Schlüssel und passende Fremdschlüssel müssen erzeugt werden
- Schachtelung
  - Geschachtelte Strukturen anstatt flacher logischer Relation
  - Gruppierung

Gruppierung

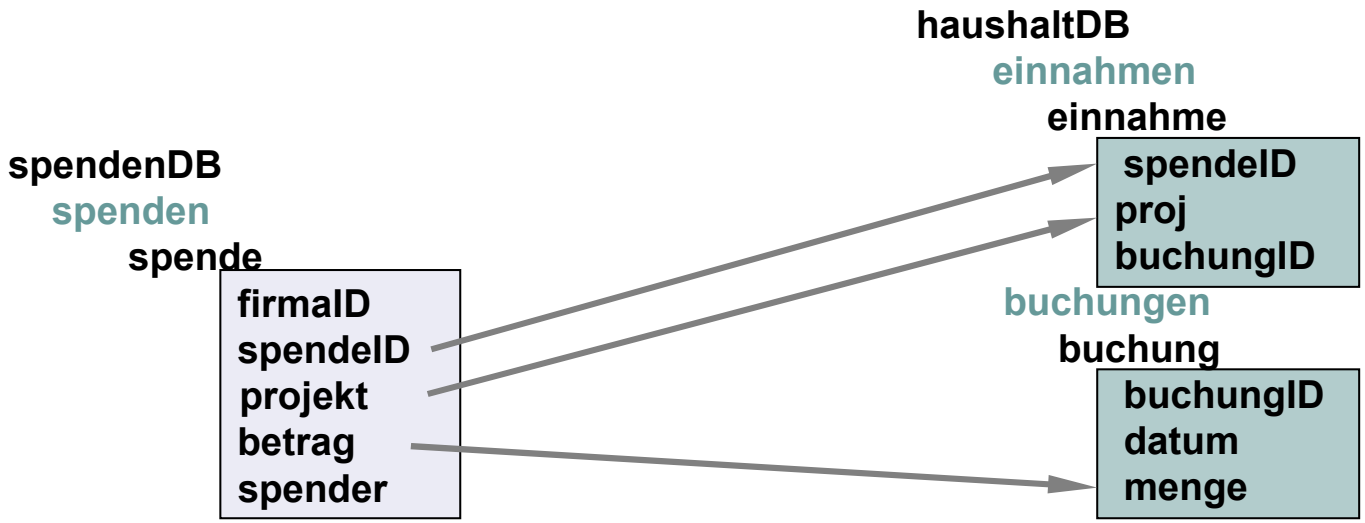
- Alle Attribute erhalten Werte: Aber
  - Assoziationen könnten verloren gehen
  - Neue (falsche) Assoziationen könnten erzeugt werden
- Deshalb: Gruppierung notwendig
- Trick: Virtuelle ID Generierung mittels Skolemfunktion basierend auf *allen* Werten hierarchisch über der aktuellen Relation
  - Jedes Tupel **haushaltDB** erhält ID Sk(**land**, **stadt**)
  - Jedes weitere Tupel aus firmen mit gleichen Werten für **stadt** und **land** errechnet gleiche ID und wird unter gleichem Element geschachtelt



Erzeugung von Anfragen (Beispiel: XQuery)

Erzeugung proprietärer Regeln in Clio (IBM Research)

for x **in** spenden
let $a = x$.spende.spendeID, $b = x$.spende.projekt,
$c = x$.spende.betrag
return <einnahme = <spendeID = a, proj = b, buchungID = Sk(a, b, c)>>
in einnahmen,
<buchung = <buchungID = Sk(a, b, c), datum = **null**, menge = c>>
in buchungen



Literatur

[DHI12] Anhai Doan, Alon Halevy, and Zachary Ives.
Principles of Data Integration.
Morgan Kaufmann, 2012.

[FHP+02] R. Fagin, M. Hernandez, L. Popa, Y. Velegrakis, and R. J. Miller.
Translating web data.
In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, 2002.