Dynamische Informationsintegration in Data Grids

DBAG-Workshop
Schloss Dagstuhl
30. Juni bis 2. Juli 2005

Jürgen Göres
AG Heterogene Informationssysteme
Technische Universität Kaiserslautern
goeres@informatik.uni-kl.de

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

FACHBEREICH INFORMATIK

HiS

Gliederung

- Dynamische Informationsintegration
- PALADIN
- Integrationsmuster
- Fazit und Ausblick

Data Grids

- Grid-Gedanke: Virtualisierung von IT-Ressourcen
  - analog zum *Utility Grid* "Rechenleistung aus der Steckdose"
  - Ziel: das Grid als leistungsfähiger virtueller Superrechner
- wachsende Bedeutung von Daten im Grid-Umfeld
  - anfangs:
    - Scavenging Grids innerhalb einer Organisation
    - Ressourcen: CPU-Zeit (+Hauptspeicher)
    - beschränkt auf existierende Ressourcen
    - Datentransport als reiner Infrastruktur-Dienst, dateibasiert
  - später:
    - zusätzlich persistenter Speicherplatz als Ressource
    - noch immer dateibasiert, Performance- und Replikationsaspekte
    - *Data Grids* – oder eher *Storage Grids*?
  - heute:
    - zusätzlich existierende (semi-)strukturierte Daten als Ressource
    - *Data(Base) Grids*

Data Grids - Anwendungsszenarien

- Zentrale Frage: Motivation der Datenanbieter?
  - Altruismus (vgl. Open-Source-Gedanke, Wikipedia)
  - Veröffentlichungspflicht
  - Eigeninteresse (z.B. Produktinformationen)
  - Geschäftsmodelle (z.B. Gendaten, Börsenkurse)
  - Begriff der ***Virtuellen Organisation*** (Foster et. al.):
    - kurz- und mittelfristige Kooperationen
    - Zusammenlegen der IT-Ressourcen der Teilnehmer
    - gemeinsame Daten als wesentliches Mittel für Kooperation
- Problem:
  - Quellen sind hochgradig heterogen
    - technische, semantische und logische Heterogenität
  - Gewinnbringende Nutzung der Daten erst durch Bereitstellung einer integrierten Sicht

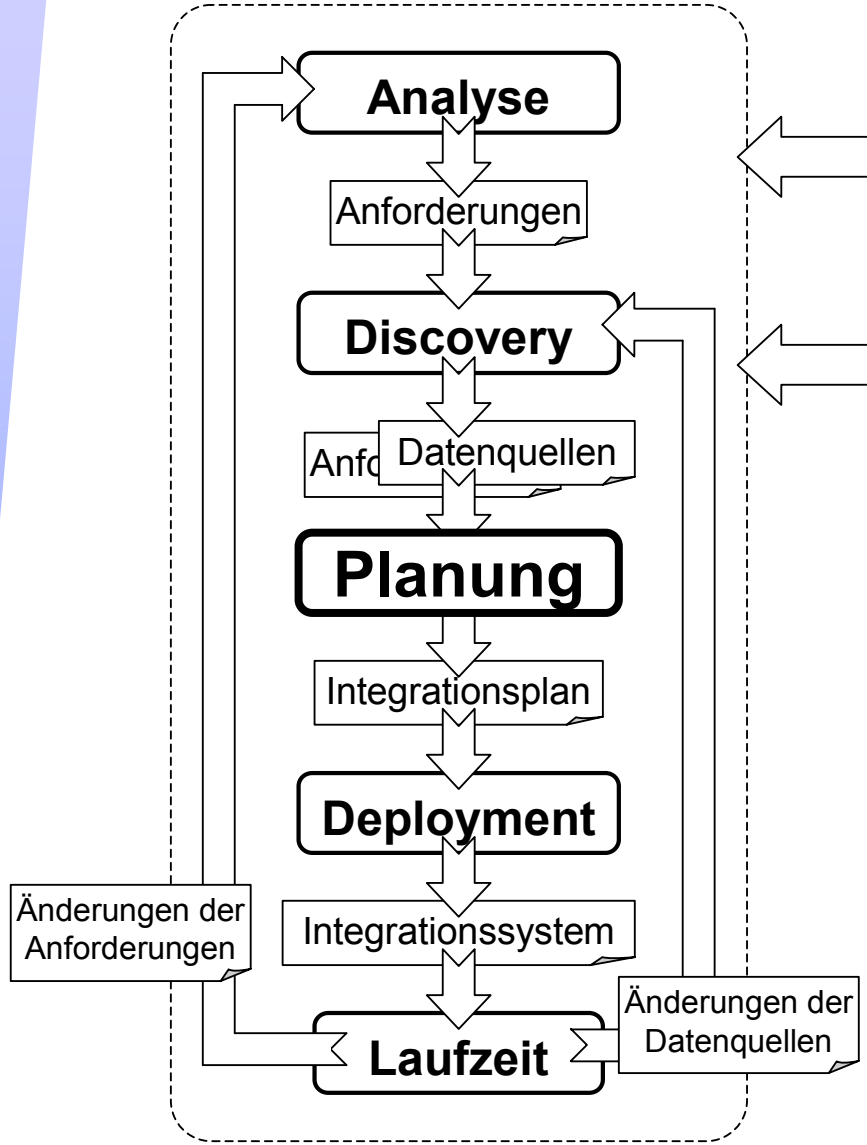
  ➔ Informationsintegration

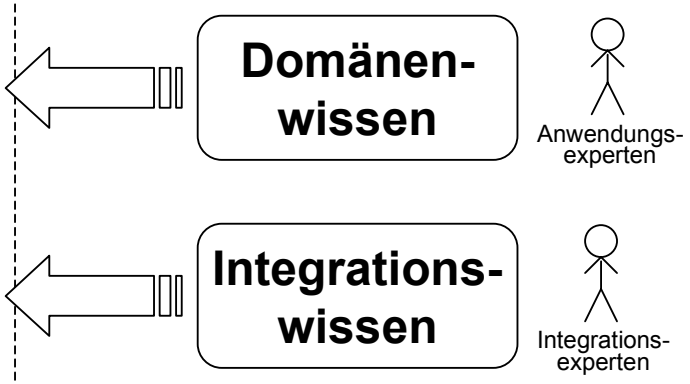
Informationsintegration im Data Grid

- bisher: Definition einheitlicher Schnittstellen
  - DAIS-Arbeitsgruppe des Global Grid Forum (GGF)
  - basierend auf Web-Service-Technologie
  - vier abstrakte Schnittstellen für Grid Data Services:
    - Data Access
    - Data Description
    - Data Management
    - Data Factory
  - konkrete Spezifikationen u.a. für
    - relationalen Datenbanken (WS-DAIR)
    - XML-Datenbanken (WS-DAIX)

  ➔ Überwindung technischer Heterogenität
- semantische und logische Heterogenität?
- Problem: Umgang mit der hohen Dynamik im Grid
  - autonome Datenquellen
  - schnell wechselnde Benutzeranforderungen

Integrationsprozess





➔ langsam und kostspielig

- adäquat in klassischen Szenarien
- in dynamischen Umgebungen nicht einsetzbar

➔ Reduktion des Bedarfs an menschlicher Expertise

unser Schwerpunkt:

- Planungsphase
- Integrationswissen

Gliederung

- Dynamische Informationsintegration
- PALADIN
- Integrationsmuster
- Fazit und Ausblick

PALADIN-Projekt – Zielsetzung

- PALADIN: Pattern-based Approach to Large Scale Dynamic Information Integration
- Ausblenden technischer Heterogenität
  - Standardisierung im Gange
  - Funktionalität im Kern fixiert
- Konzepte zur Überwindung semantischer Heterogenität
  - Schema Matching
  - Domänenschemas
- Schwerpunkt auf logischer Heterogenität:
  - Metamodell für den einheitlichen Umgang mit Daten und Metadaten beliebiger Datenmodelle
  - Maschinenverständliche Repräsentation von Integrationswissen durch Integrationsmuster

PALADIN-Metamodell

- Ziel: Vereinheitlichte Handhabung von Metadaten und Daten beliebiger Datenmodelle
- Common Warehouse Metamodel
  - Stapel von Metaebenen
    - Meta-Metamodell (M3)
    - Metamodelle (M2)
    - Modelle (M1)
    - Daten (M0)
  - Probleme:
    - sehr komplex
    - teilweise veraltet
    - ungeeignete Darstellung der Datenebene

➔ Entwurf eines CWM-basierten Metamodells

  - graphbasierte Repräsentation von Metadaten und Daten
  - Darstellung von Korrespondenzen

Gliederung

- Dynamische Informationsintegration
- PALADIN
- Integrationsmuster
- Fazit und Ausblick

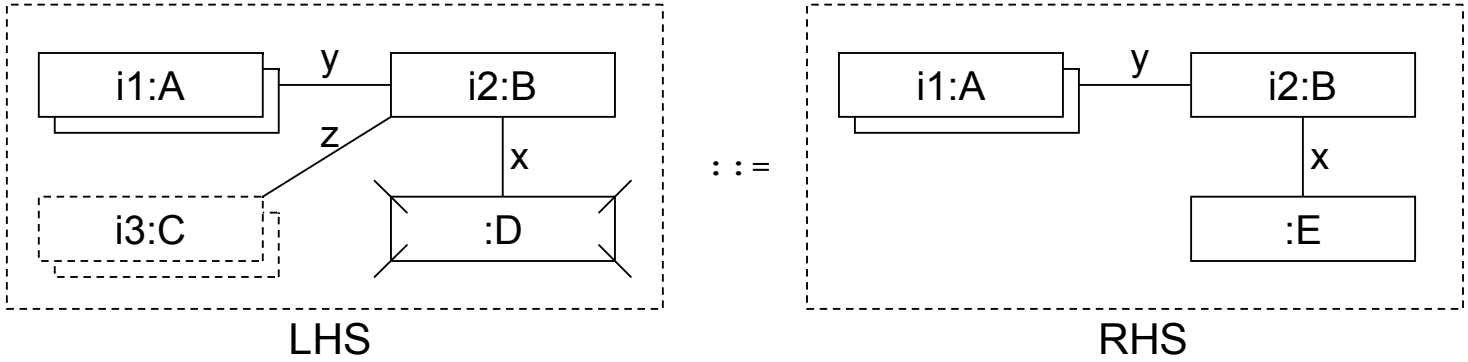
Integrationsmuster

- Wissen zur Überwindung logischer Heterogenität
- bisher: Integrationswissen in Form von Algorithmen
- jetzt: Integrationsmuster
  - deklarativ
  - Beschreiben atomare oder komplexe
    - Problemstellung
    - Lösungsansatz
- einheitliche Darstellung der Muster?

  ➔ PALADIN-Metamodell (PMM)
- Beschreibung als Graphtransformation
  - Unterstützung aller mit PMM beschriebenen Metamodelle
  - große Ausdrucksmächtigkeit:
    - Umformungen zwischen Datenmodellen
    - Schematische Transformationen

Integrationsmuster (II)

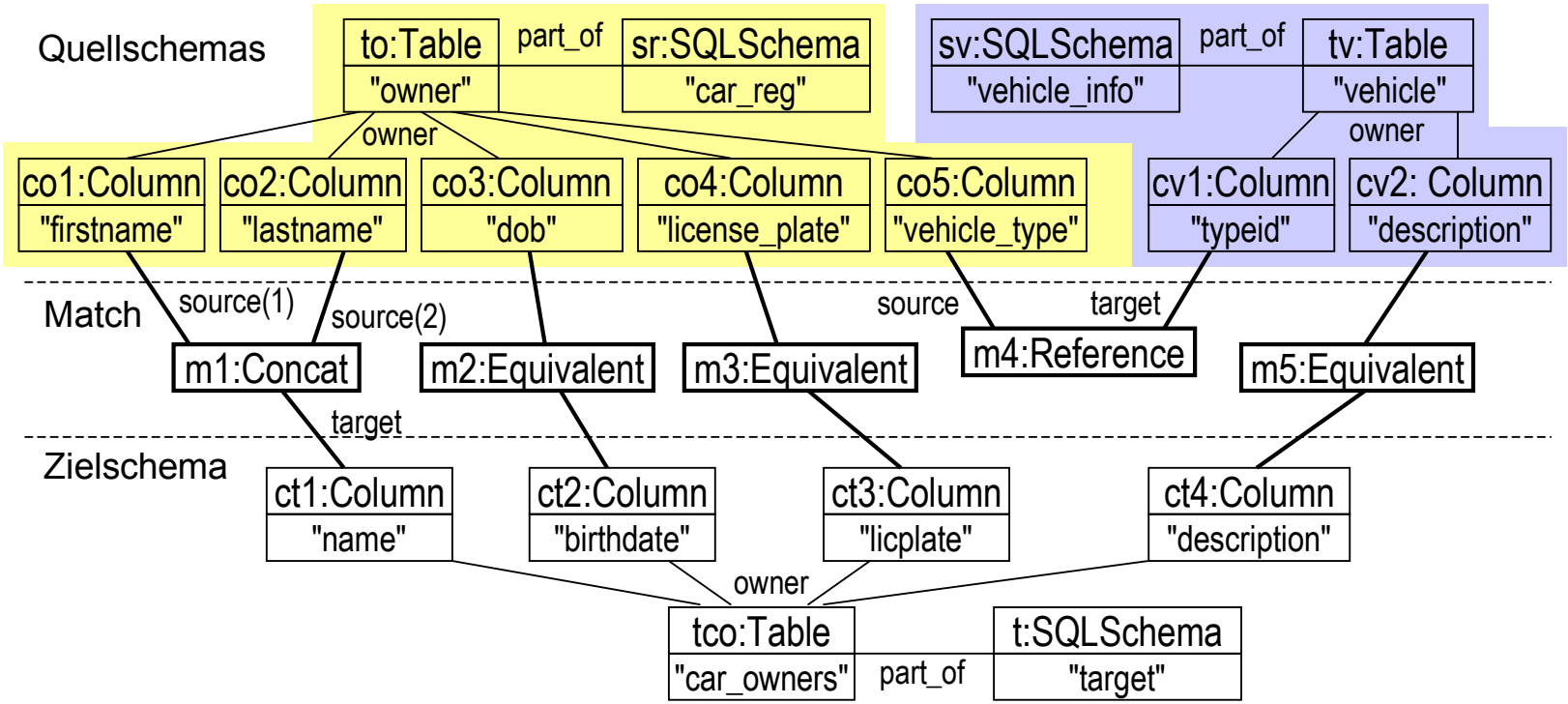
- semi-grafische Notation



- Integrationsmuster beschrieben durch
  - Transformation des Schemas (Ebene M1)
  - Transformation der Daten (Ebene M0)
- Anwendung der Muster
  - Quellschemas ≡ Axiome
  - Zielschema ≡ Hypothese
  - Muster ≡ Regeln

  ➔ Ableitung ≡ Integrationsplan

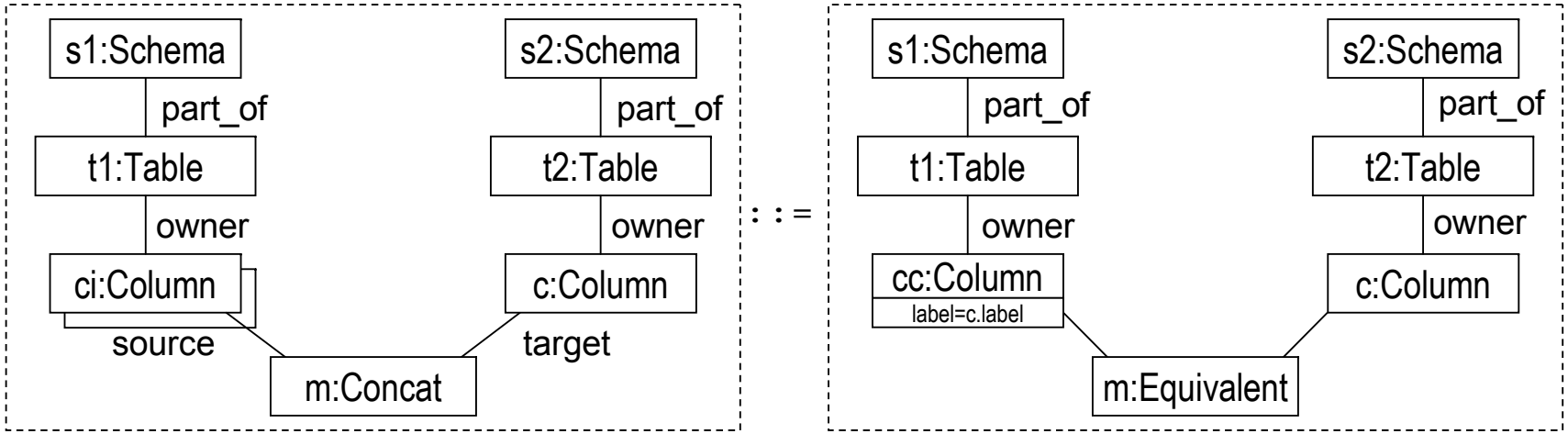
Beispiel - Ausgangssituation



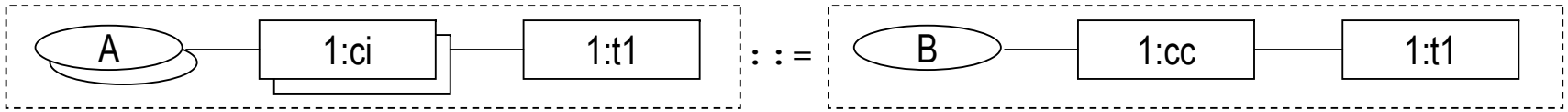
- Probleme
  - Normalisierungsgrad
  - Aufteilung von Anwendungskonzepten auf Schemaelemente
  - Bezeichner
  - überflüssige Spalten

Beispiel - Integrationsmuster

```
Pattern concatenate().m1 {
    condition p1.type=p2.type
```

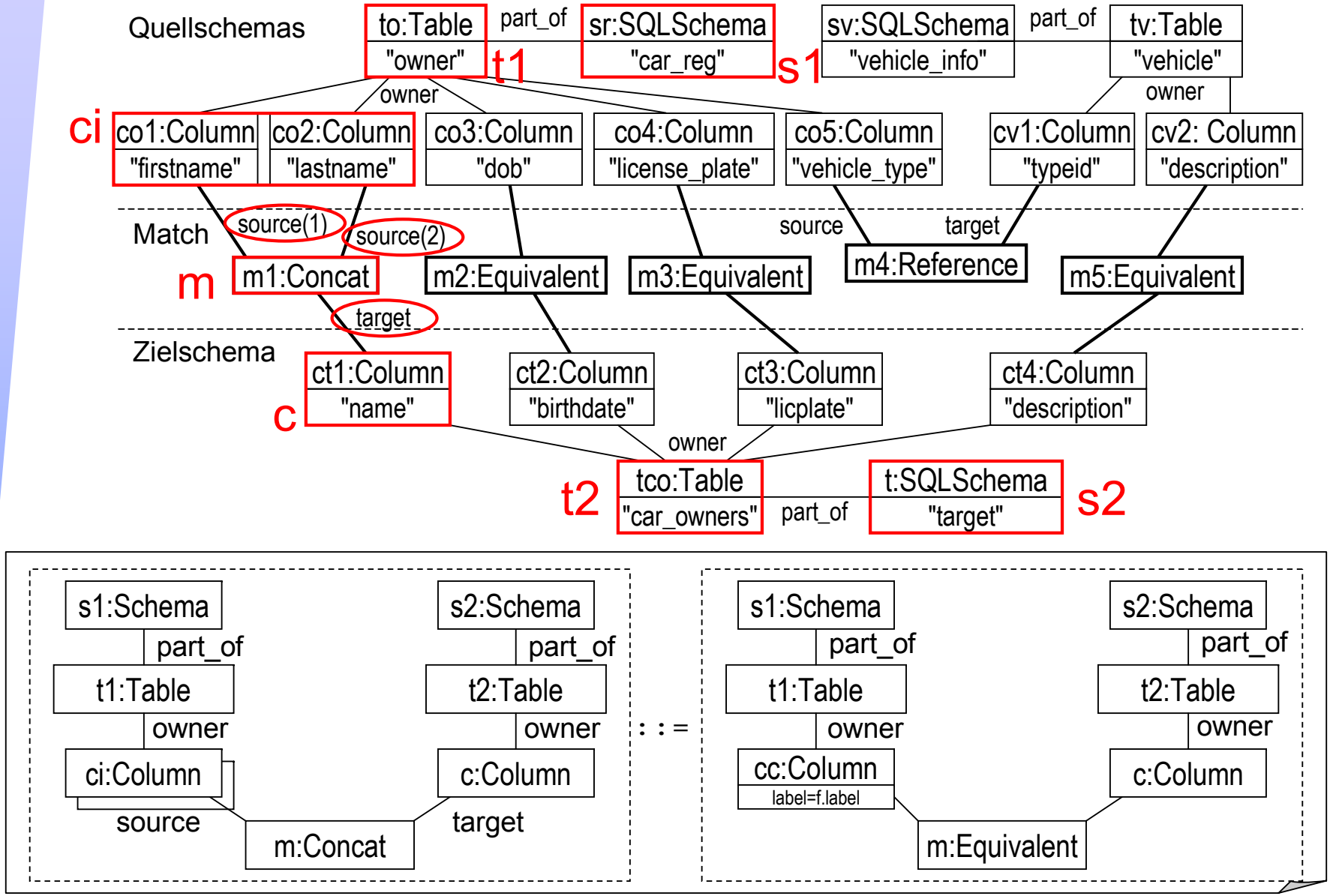



```
}
Pattern concatenate().m0 {
```

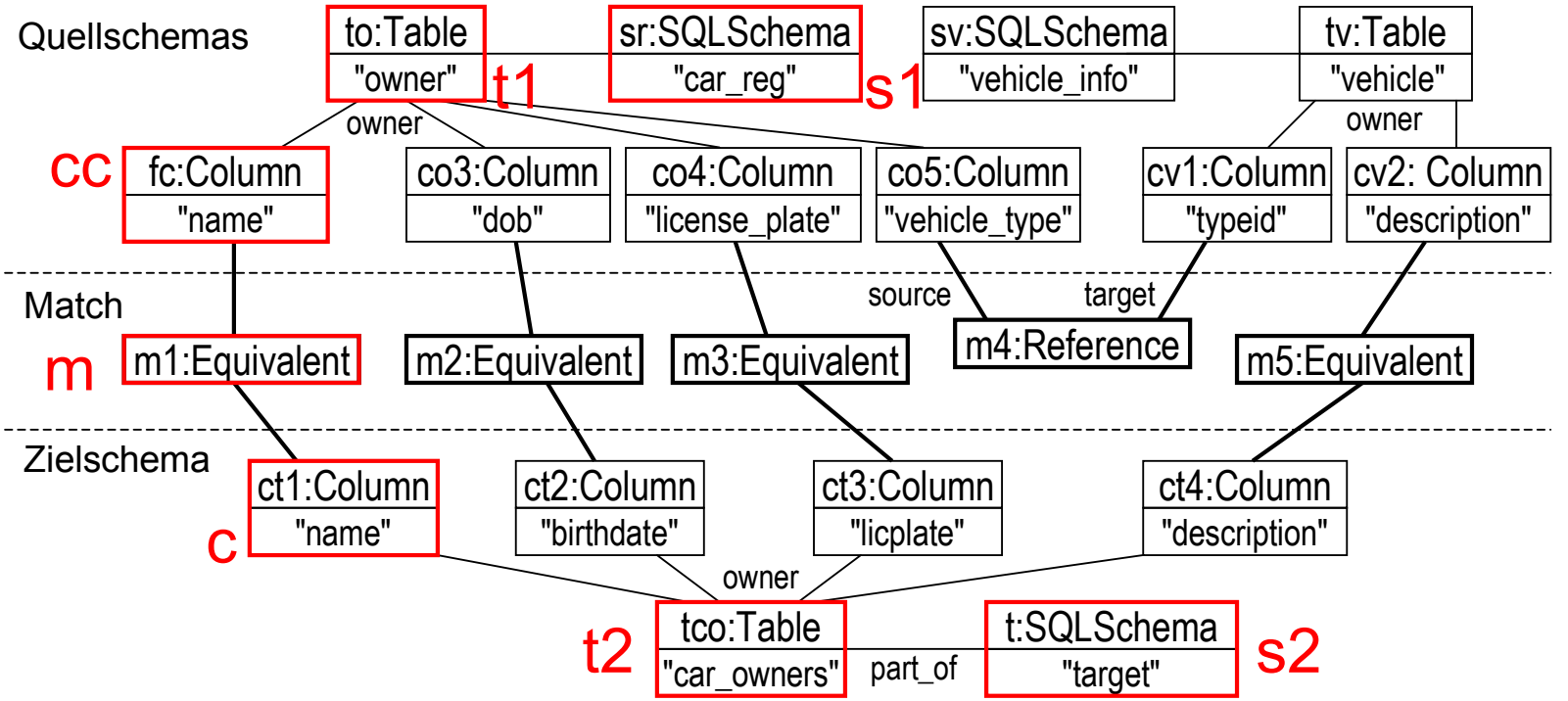


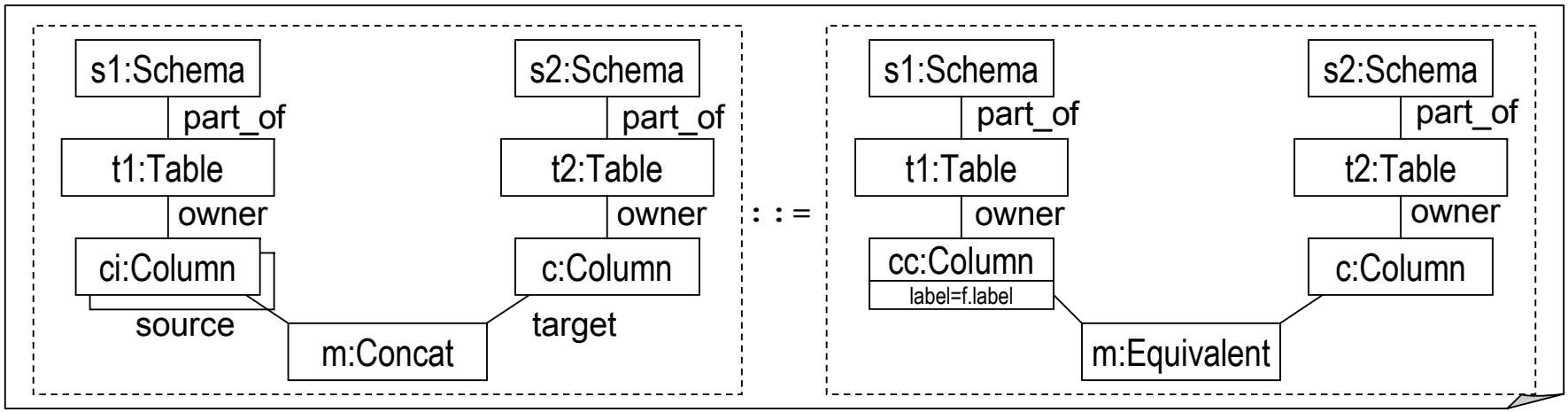

```
    foreach $a in A {
        let B = B + " " + A;
    }
}
```

Beispiel - Musteranwendung

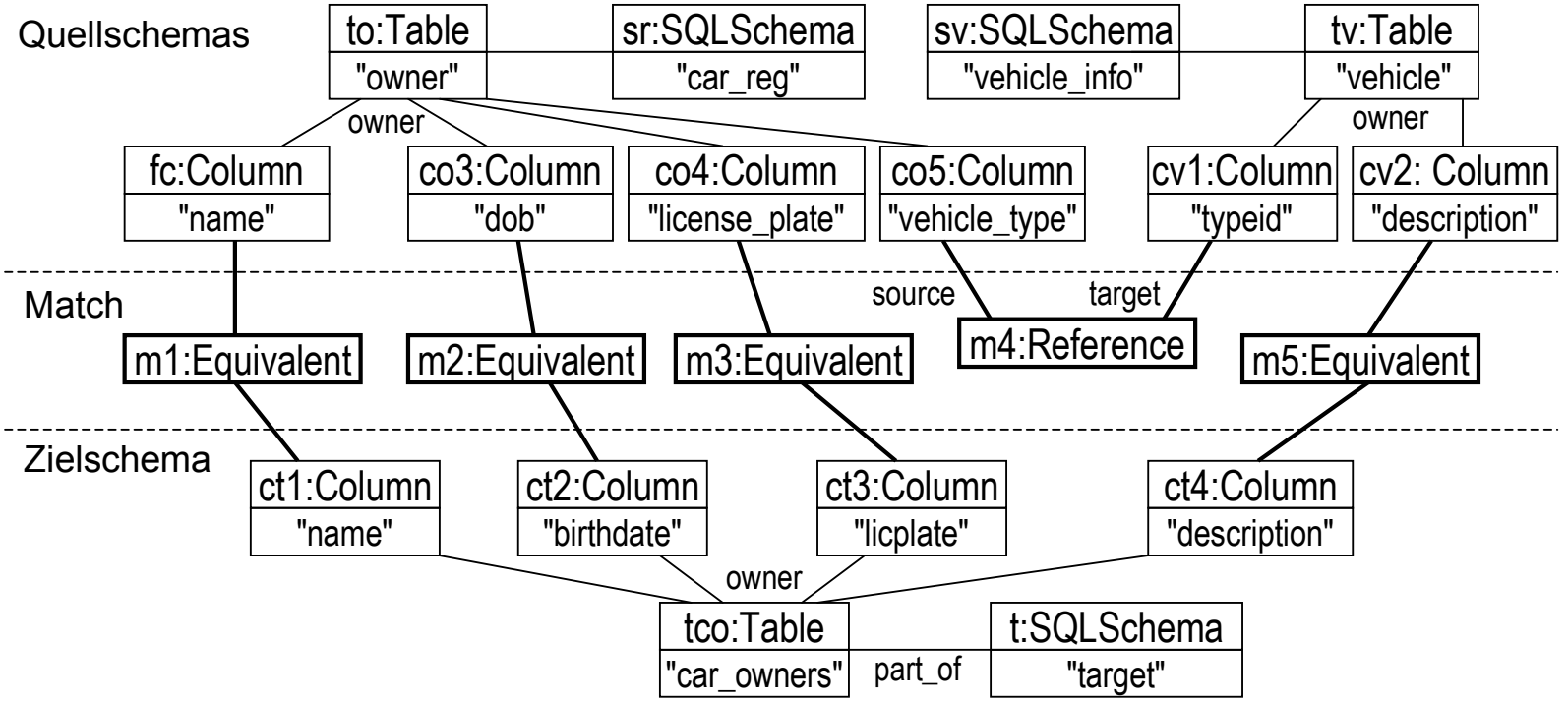


Beispiel - Musteranwendung





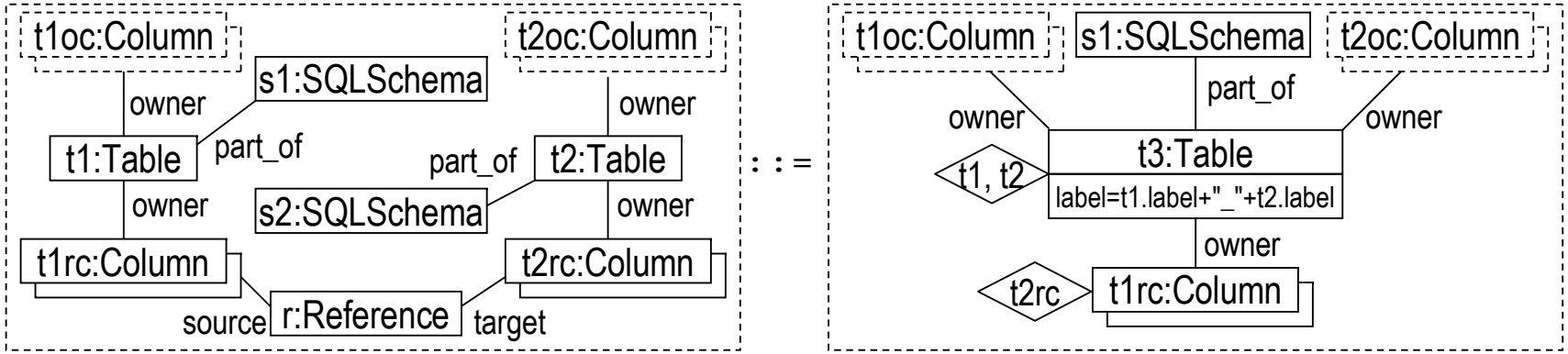
Beispiel - Zwischenergebnis



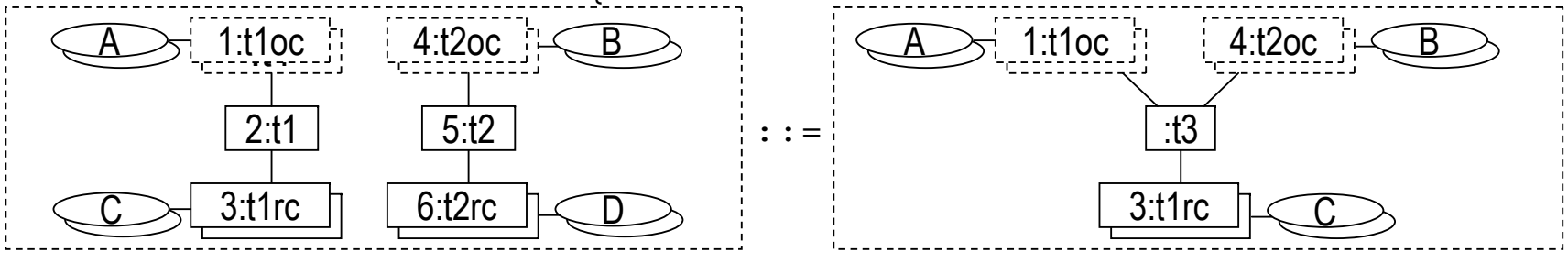
- Probleme
  - Normalisierungsgrad
  - Aufteilung von Anwendungskonzepten auf Schemaelemente ✓
  - Bezeichner
  - überflüssige Spalten

Beispiel - Integrationsmuster 2

```
Pattern denormalise().m1 {
```

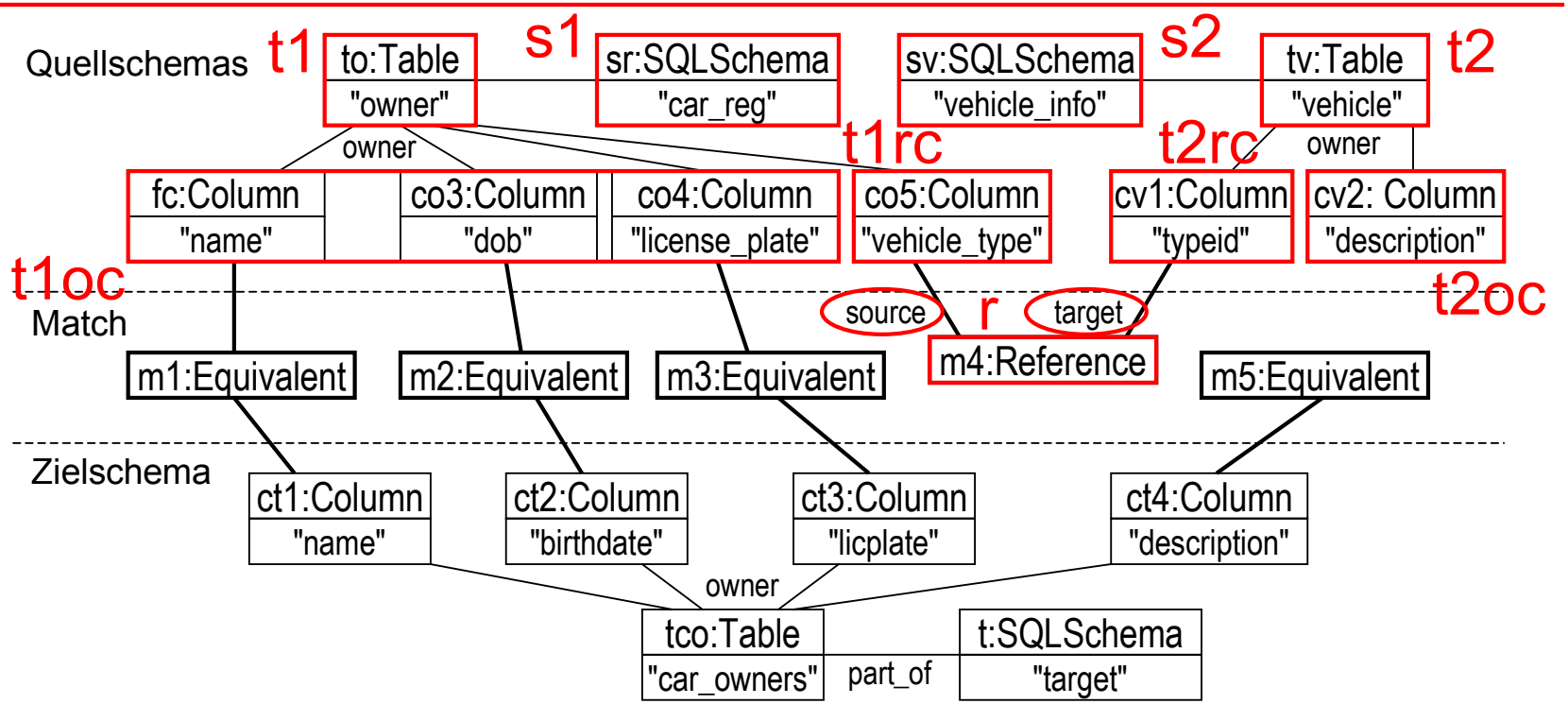


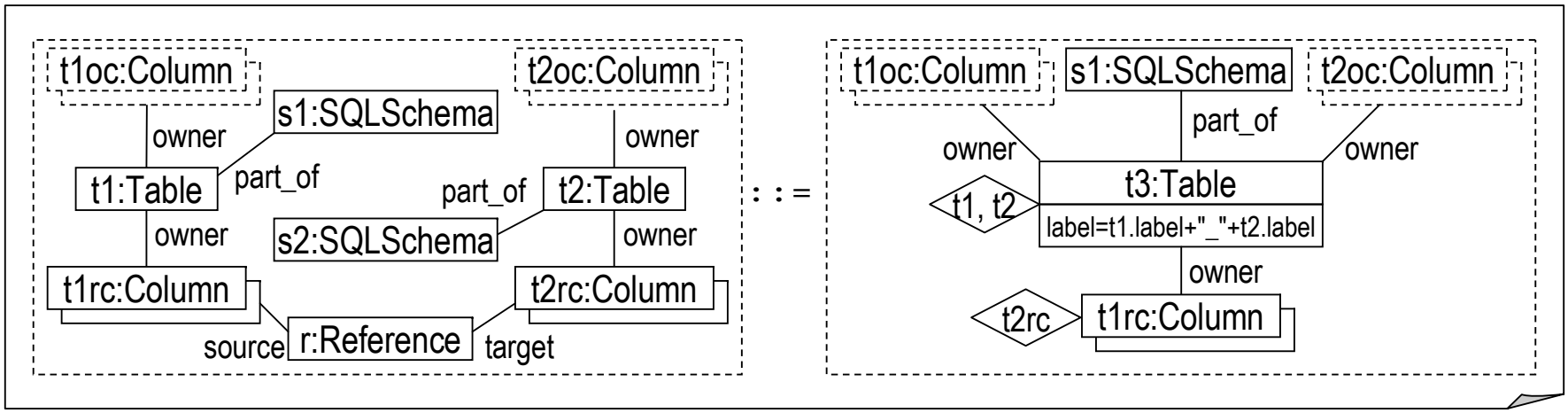

```
}
Pattern denormalise().m0 {
```



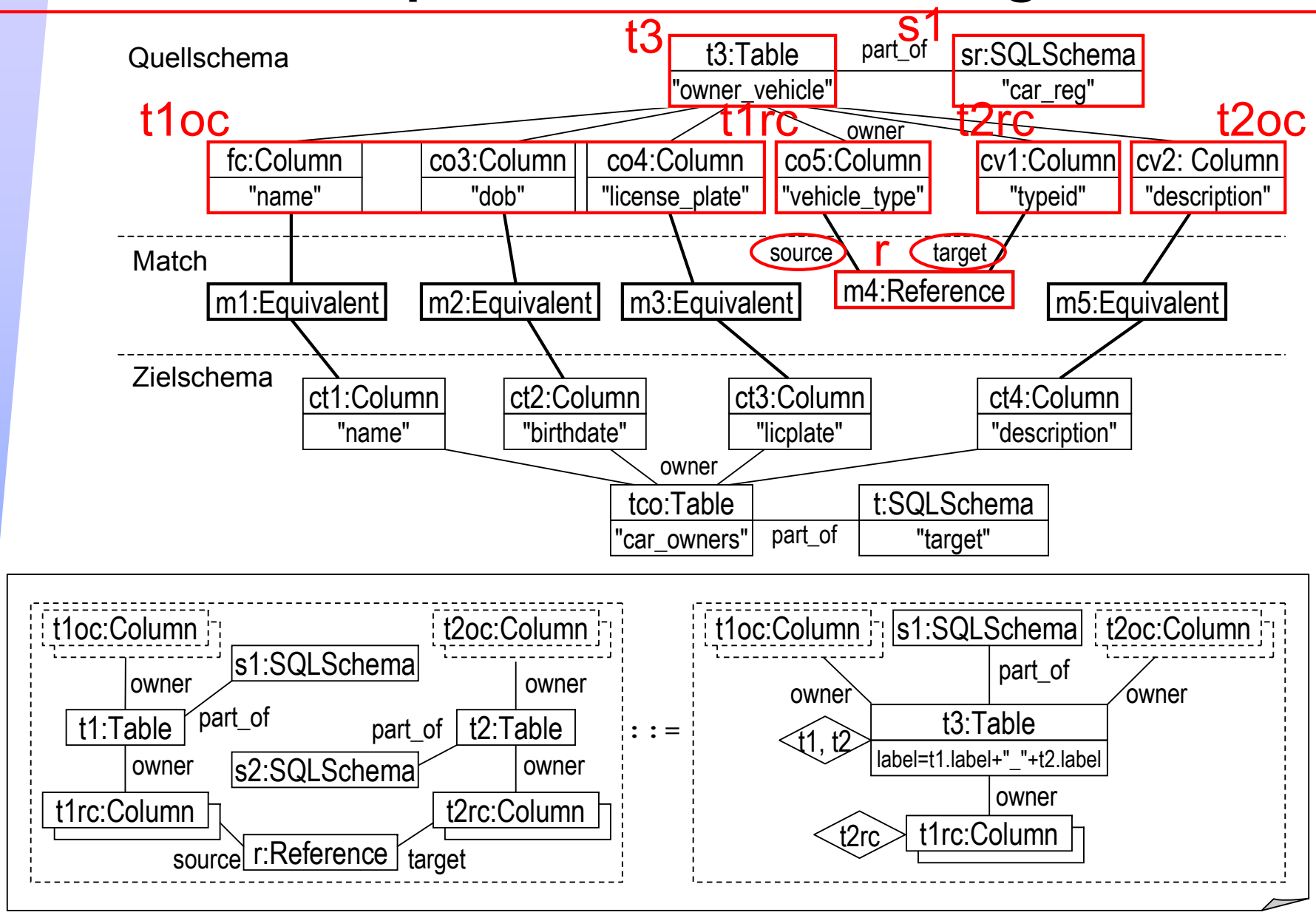

```
 Condition for each c,d in C:,D: c==d
}
```

Beispiel - Musteranwendung

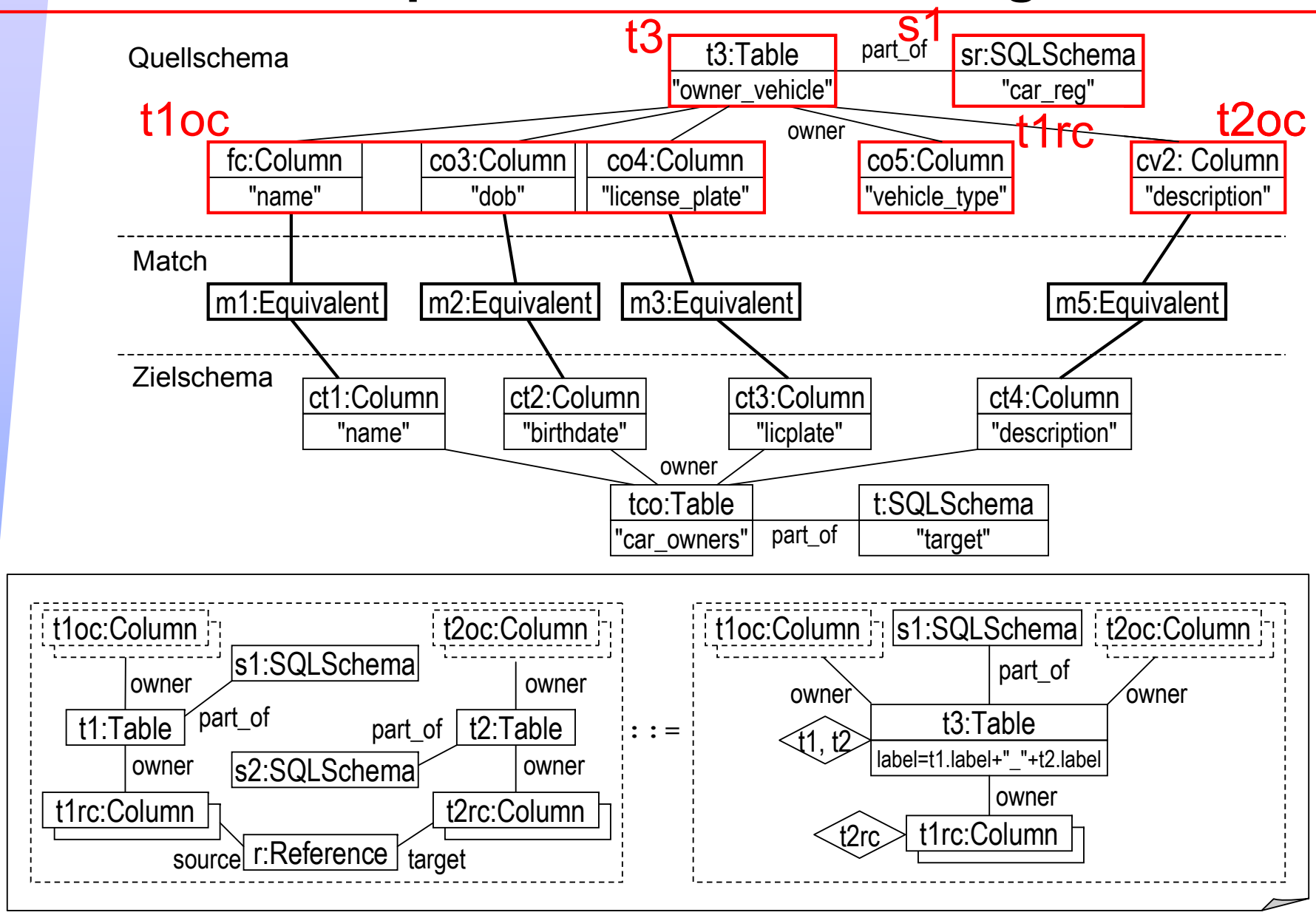




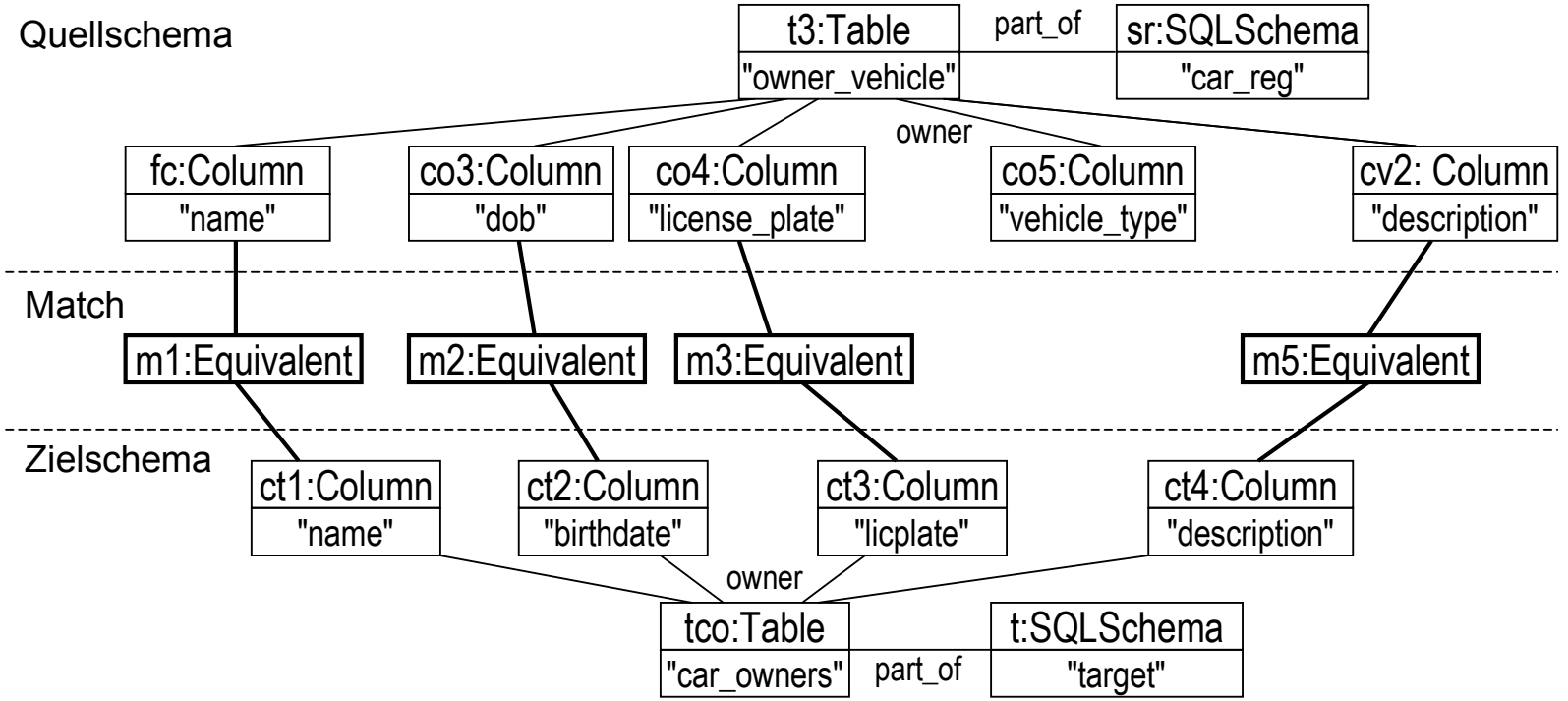
Beispiel - Musteranwendung



Beispiel - Musteranwendung



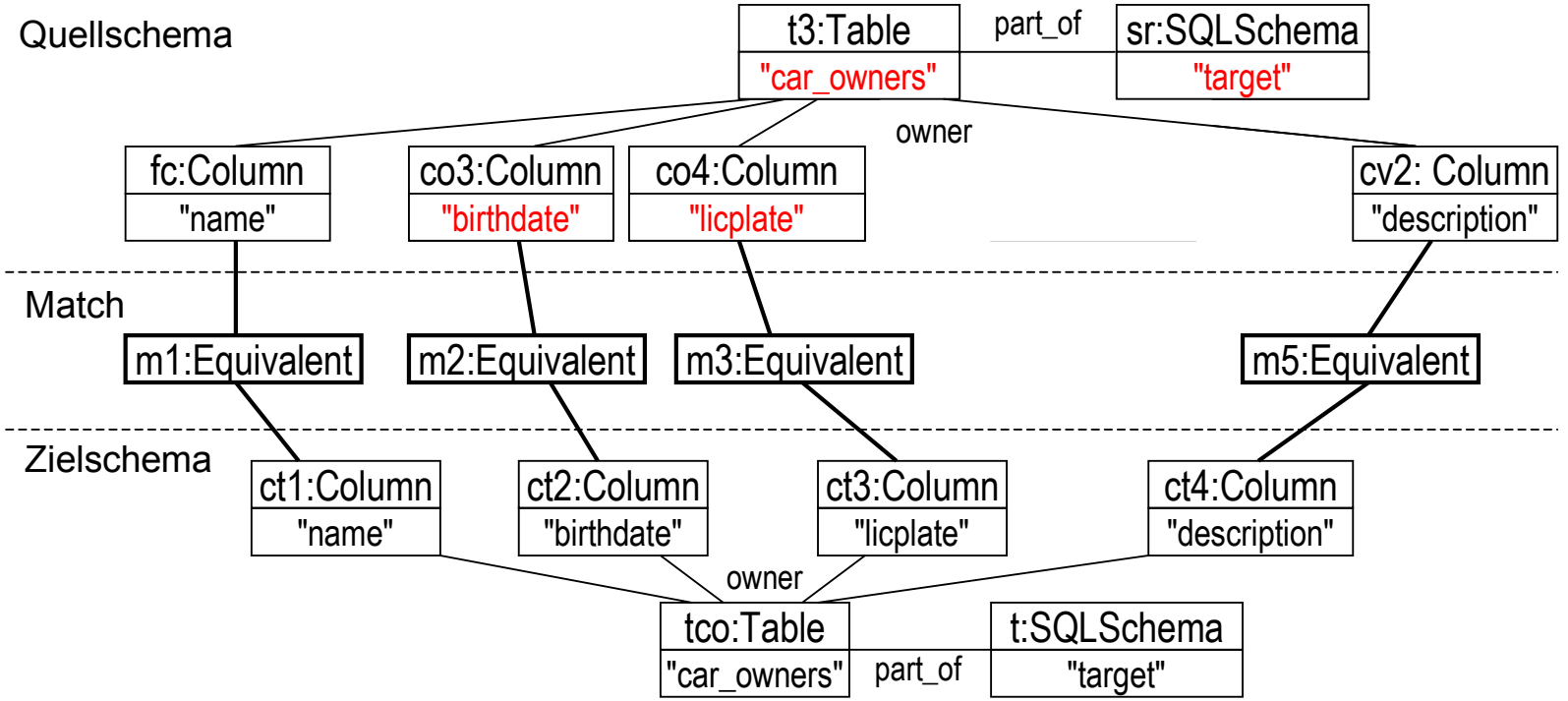
Beispiel - Zwischenergebnis



- Probleme
  - Normalisierungsgrad ✓
  - Aufteilung von Anwendungskonzepten auf Schemaelemente ✓
  - Bezeichner
  - überflüssige Spalten

  } "trivial"

Beispiel - Endergebnis

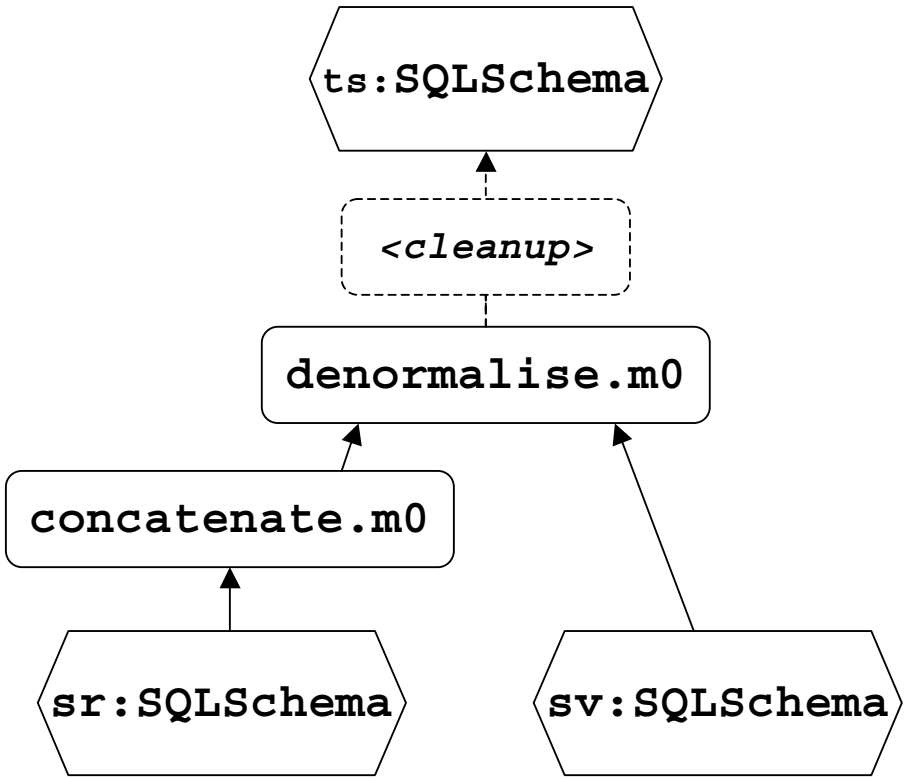


- Probleme
  - Normalisierungsgrad ✓
  - Aufteilung von Anwendungskonzepten auf Schemaelemente ✓
  - Bezeichner
  - überflüssige Spalten

  } "trivial" ✓

Resultierender Integrationsplan

- Ableitung des Zielschemas ergibt abstrakten Integrationsplan
- M0-Bestandteile der Muster bilden Operatoren



Deployment (I)

- Abbildung auf konkreten Integrationsplan
- z.B. als relationale Sichtdefinition

```
CREATE VIEW target.car_owners
  (name, birthdate, licplate, description)
AS
SELECT firstname CONCAT ' ' CONCAT lastname,
       dob, license_plate, description
FROM   car_reg.owner, vehicle_info.vehicle
WHERE  vehicle_type=typeid
```

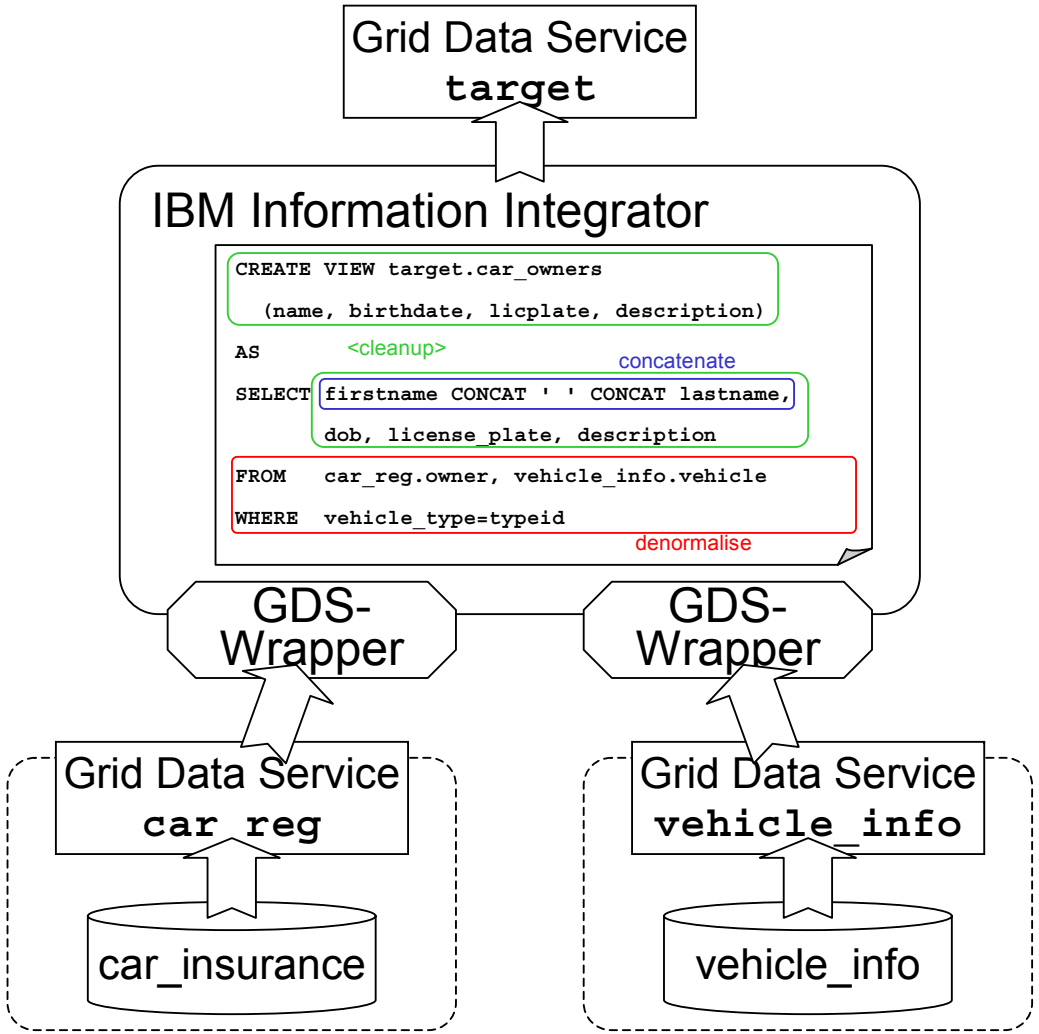
<cleanup>

concatenate

denormalise

Deployment (II)

- Einsatz in einem relationalen Integrationswerkzeug



Gliederung

- Dynamische Informationsintegration
- PALADIN
- Integrationsmuster
- Fazit und Ausblick

Zusammenfassung

- Dynamik des Grid-Umfelds als Herausforderung für Informationsintegration
  - Beschleunigung des Integrationsprozesses
  - Haupthemmnis: menschliches Wissen erforderlich
- PALADIN-Metamodell
  - Darstellung von Metadaten und Daten beliebiger Datenmodelle
  - Darstellung von komplexen inhaltlichen Korrespondenzen
- Erfassung von Wissen durch Integrationsmuster
  - Repräsentieren abstrakte Operatoren
  - Ableiten von abstrakten Integrationsplänen
  - losgelöst von der späteren Zielplattform
  - flexibles Deployment

Ausblick

- Deployment
  - mögliche Laufzeitumgebungen u.a.
    - Choreographie von Web Services (z.B. BPEL, OGSA-DAI Perform)
    - Integrationssysteme
    - ETL-Werkzeuge
  - Verteilungsaspekte von Operatoren
  - Abbildungsregeln als Muster
  - graphorientierte Operatorbeschreibung als Fallback
- Hochsprachen zur Musterspezifikation
  - vereinfachte Definition
  - Kompilation in graphorientierte Darstellung
- Schema Matching mit Mustern
  - Einfügen von Matches ist eine Graphtransformation
  - Problem: Darstellung komplexer Vorverarbeitungsschritte