Seminar: Data Warehousing im Verkehrsbereich

Sichtenmaterialisierung — Anwendungen und Probleme

Juan C. Fries <jcf@jcf.de>

Agenda

1. Einleitung
2. Anwendungen
3. Probleme
4. WHIPS
5. Zusammenfassung

Agenda

1. Einleitung
   i. Motivation
   ii. Begriffserklärung
2. Anwendungen
3. Probleme
4. WHIPS
5. Zusammenfassung

Einleitung

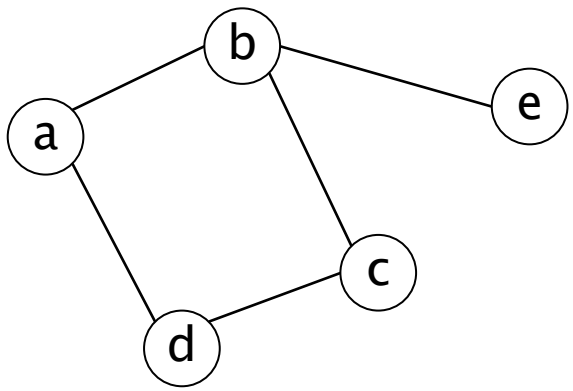
- Sicht: abgeleitete Tabelle aus (mehreren) Relationen
- Data Warehouse = lokale Kopien von Sichten
- Sichtenmaterialisierung
  - Kopieren und Speichern der Tupel einer Sicht
  - Warum? – schneller Datenzugriff, effizienter ...
- Veränderungen in den Sichten können Auswirkungen auf die materialisierten Sichten haben
- Anomalien und Inkonsistenzen zwischen den Basisrelationen und dem Data Warehouse
- View maintenance (Sichtenpflege)
  - Aktuellhalten der materialisierten Sicht
  - inkrementell: Berechnung nur der Änderungen in einer Sicht

Agenda

1. Einleitung
2. Probleme
   i. Beispiel
   ii. View Maintenance Problem
      a. Klassifikation
      b. Verwendung vollständiger Informationen
      c. Verwendung lückenhafter Informationen
   iii. Offene Probleme
3. Anwendungen
4. WHIPS
5. Zusammenfassung

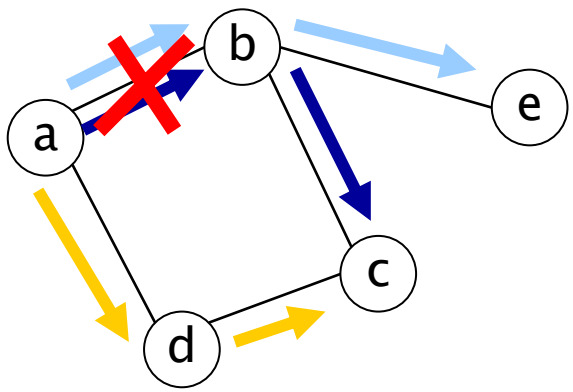
Beispiel

- Basisrelation: $\texttt{link}(S,D)$
- Sicht: $\text{hop}(X,Y) = \Pi_{X,Y}(\texttt{link}(X,V) \bowtie_{V=W} \texttt{link}(W,Y))$
- $\texttt{link} = \{(a,b), (b,c), (b,e), (a,d), (d,c)\}$



Beispiel

- Basisrelation: link(S,D)
- Sicht: hop(X,Y) =
  $\Pi_{X,Y}(\texttt{link}(X,V) \bowtie_{V=W} \texttt{link}(W,Y))$
- link = {(a,b), (b,c), (b,e), (a,d), (d,c)}

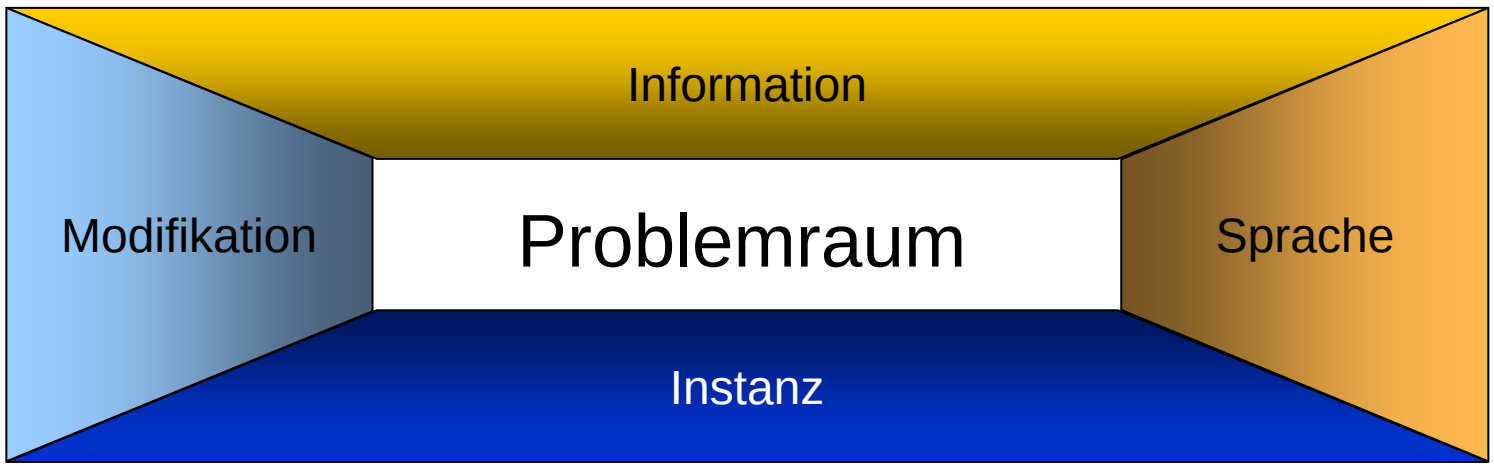


{(a,c), (a,c), (a,e)} =

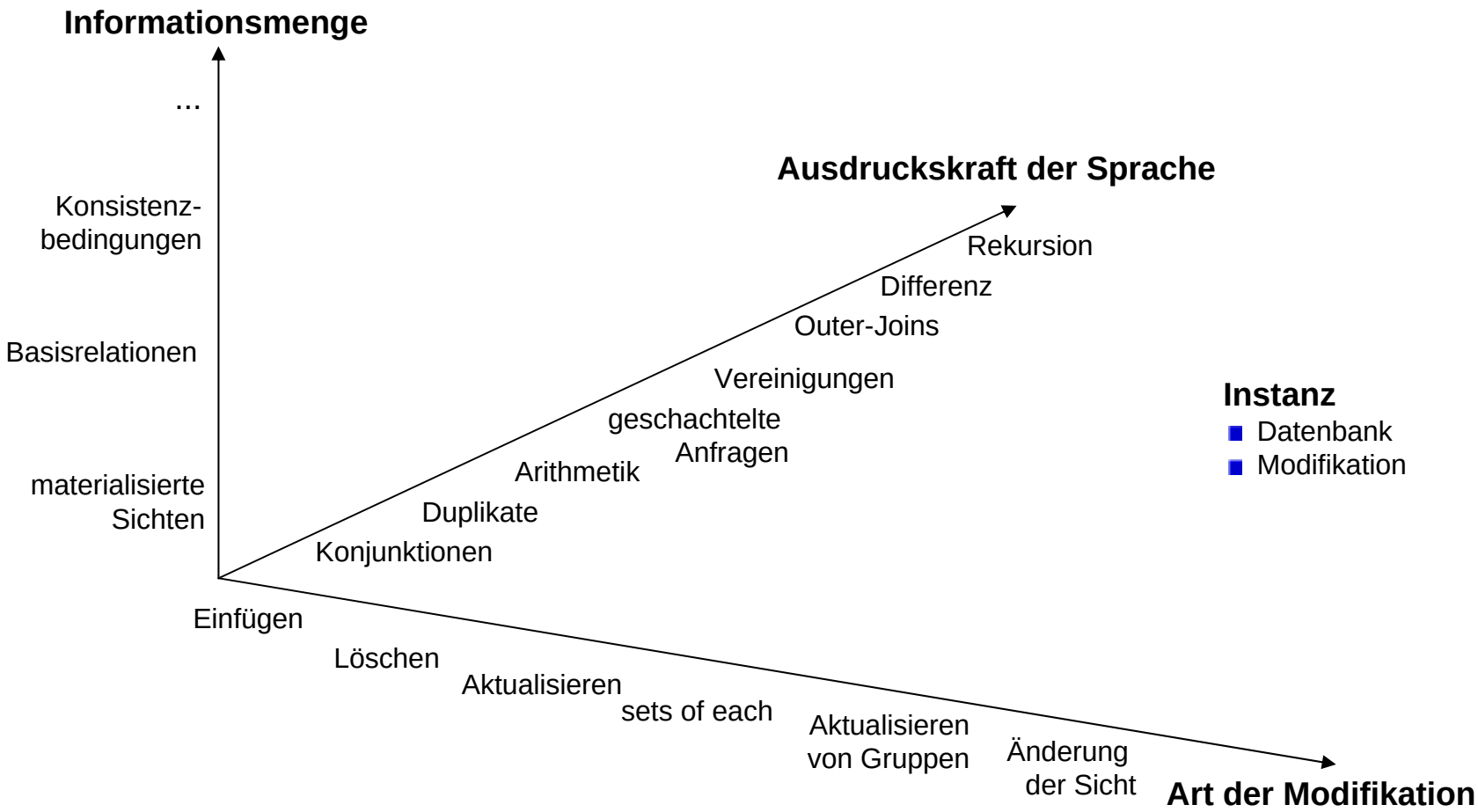
hop = {(a,c), (a,e)} ?

View Maintenance Problem

- Was ist effizienter neu zu berechnen?
  - komplette Sicht
  - nur Veränderungen
- Updates atomar oder Löschen+Einfügen?
- Klassifikation anhand der vier Dimensionen



Problemraum in 3D



Verwendung vollständiger Informationen

- Rekursive Sichten
  - DRed Algorithmus: Drei-Schritt-Verfahren mit Überschätzung
  - The Propagation/Filtration (PF) Algorithm: schleifenweises DRed
  - The Kuchenhoff Algorithm: irrelevante Duplikate möglich
  - The Urpi-Olive Algorithm: Existenzquantoren
- Nicht-rekursive Sichten
  - counting Algorithm: zählt die Ableitungen jedes Tupels
  - Algebraic Differencing: relationale Ausdrücke ohne Redundanz
  - The Ceri-Widom Algorithm: funktionsbestimmte Relationen
  - Alle Algorithmen der rekursiven Sichten
- Outer-Join Sichten
  - äußere Verbindung je der einen Relation mit der Änderungsmenge der anderen Relation
  - Zur Sichtenpflege müssen nur die Sichtmaterialisierung und alle Basisrelationen herangezogen werden.

Verwendung unvollständiger Informationen

- No-Information
  - Query independent of update bzw. irrelevant update problem:
  - Sichtenrelevanz der Änderung
- Self-Maintenance
  - Sicht kann bei jeder Modifikation der Basisrelation sich selbst pflegen
- Partial-reference
  - materialisierte Sicht und Teile der Basisrelation vorhanden
  - Chronik: Sicht ist ohne geänderte Relation verfügbar
  - Change-reference maintainable: nur Sicht und die geänderte Relation sind verfügbar

Offene Probleme

- Problemraum nicht komplett abgedeckt
- Verfahren mit unvollständiger Information
  - funktionale Abhänigkeiten, mehrfache Sichtenmaterialisierungen, horizontale und vertikale Fragmente der Basisrelationen
  - Erweiterung der Sprache um Aggregation, Negation und äußere Verbindungen (Outer-Joins)
  - Erkennen instandzuhaltender Unterklassen unabhängig von der Instanz
- Information Identification (II) Problem
  - Welche Informationen werden für eine effiziente Sichtenpflege (view maintenance) benötigt?

Weitere Probleme

- *vor* oder *nach* commit?
- Teil einer Transaktion?
- *vor* oder *nach* Update der Relation?
- automatisch oder benutzergesteuert?
- Kosten-orientierte Entscheidungen?
- Komplexität unerforscht

Agenda

1. Einleitung
2. Probleme
3. Anwendungen
4. WHIPS
5. Zusammenfassung

Anwendungen

- Data Warehouse
- Chronicle Systems
- Datenvisualisierung
- Mobile Systeme
- Integritätsprüfung
- Anfragenoptimierung

Anwendungen

- Data Warehouse
- Chronicle S
- Datenvisua
- Mobile Sys
- Integritätsprüfung
- Anfragenoptimierung

- Sammlung von Informationen ohne jede Datenbank in das Data Warehouse zu kopieren
- Beantwortung ohne Zugriff auf die Quell-Datenbank (evtl. beschränkt oder sehr teuer)

A

- Reihen von Tupeln in zeitlicher Ordnung
- Größe jenseits jeder Datenbank-Kapazität
- Beantwortung der Anfragen ohne auf das Chronicle System zugreifen zu müssen.
- z. B. Kontostände

- Data Warehous
- Chronicle Systems
- Datenvisualisierung
- Mobile Systeme
- Integritätsprüfung
- Anfragenoptimierung

Anwendungen

- Data Warehouse
- Chronicle Systems
- Datenvisualisierung
- Mo[illegible]Systeme

- zeigen Sichten über Daten einer Datenbank
- häufige Änderungen an den Sichtendefinitionen
- dynamische Reaktion auf interaktive Anfragen

Anwendungen

- Data Warehouse
- Chronicle Systems
- Datenvisualisierung
- Mobile Systeme
- Integritätsprüfung
- Anfragenoptimierung

- eingeschränkte Bandbreite
- hochgradig mobile Teilsysteme
- Anfragen in Abhängigkeit des Ortes
- minimale Datenübertragung
- nur Veränderungen neuberechnen

Anwendungen

- statische Konsistenzbedingungen als eine Menge von Sichten
- nicht-leere Sicht Hinweis auf eine verletzte Bedingung
- Inkrementelle Überprüfung der Konsistenzbedingungen

- Mobile S[illegible]eme
- Integritätsprüfung
- Anfragenoptimierung

Anwendungen

- Data Warehouse
- Chronicle Systems
- Datenvisualisierung
- Mobile Systeme
- Integritätsprüfung
- Anfragenoptimierung

- Optimierung beliebiger Anfragen
- Schnellzugriff allgemein

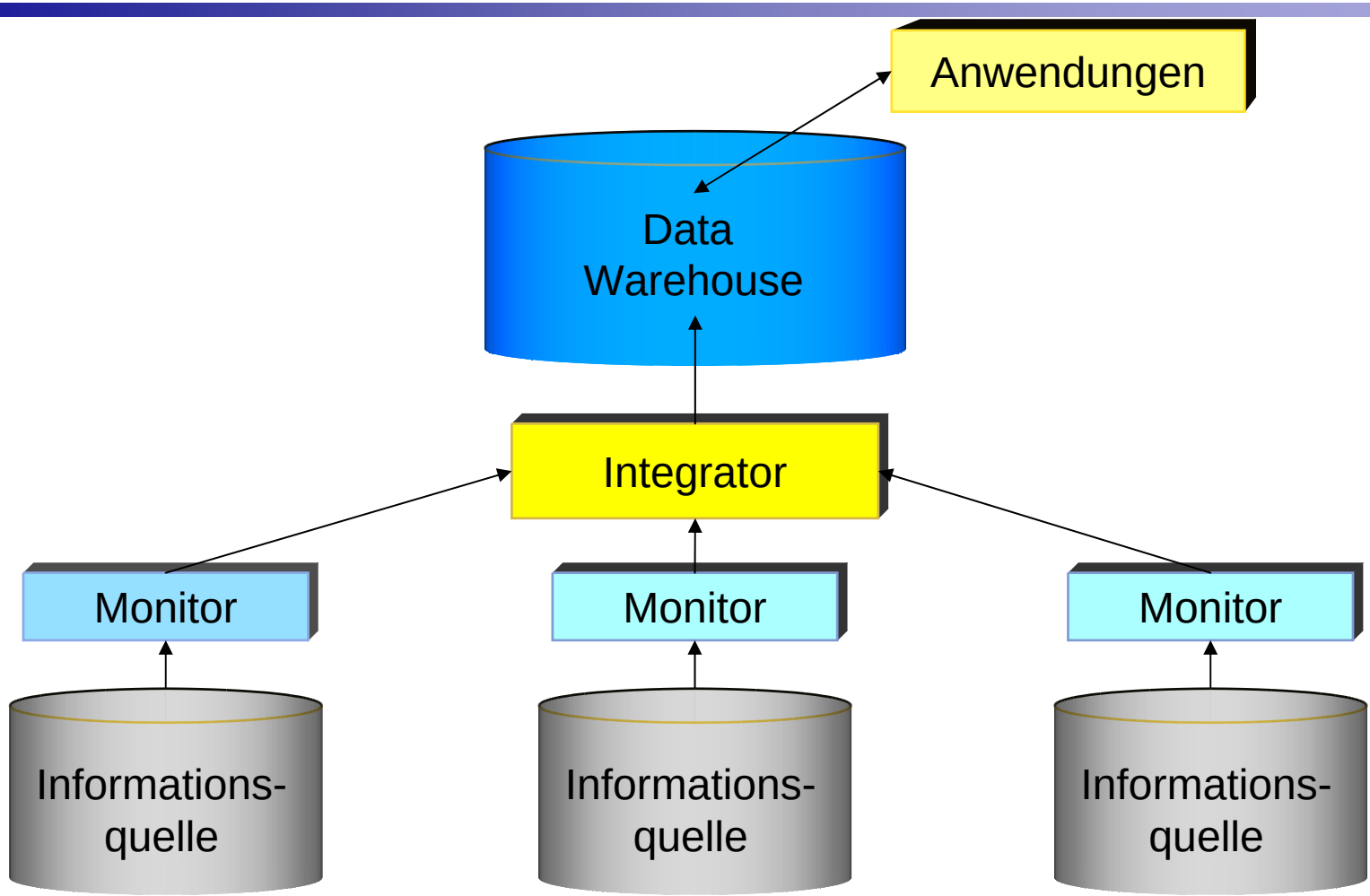
Agenda

1. Einleitung
2. Probleme
3. Anwendungen
4. WHIPS
   i. WHIPS und View Maintenance
   ii. Architektur
   iii. Warehouse Update-Anomalien
5. Zusammenfassung

- WareHouse Information Project at Stanford
- dem View Maintainance Problem ähnlich, aber:
  - Heterogene und
  - autonome Informationsquellen

➔ Überwachung problematisch

WHIPS-Architektur

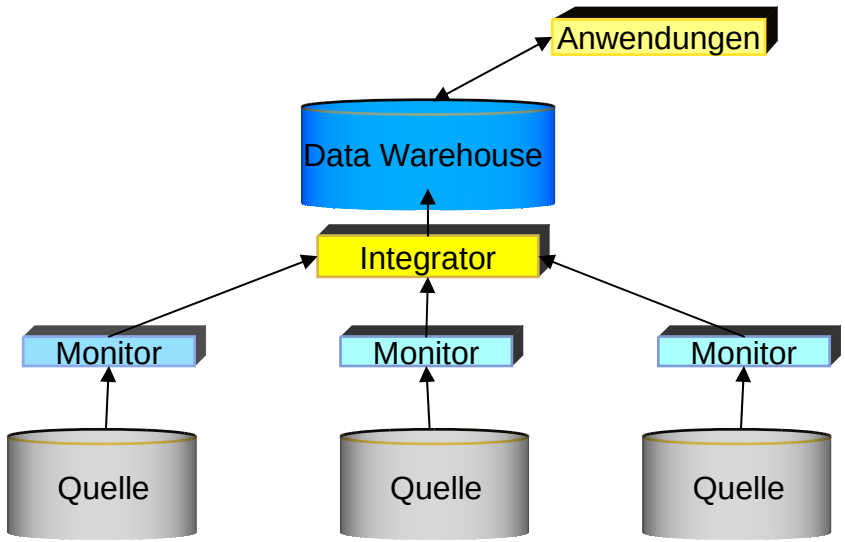


Monitore

- Aufgaben
  - Erkennen von Veränderungen in den Informationsquellen
  - Propagieren dieser Veränderungen
- Lösungen
  - beobachte "update logs"
  - sonst (z. B. legacy systems):
    - Anpassen der Applikationen, so dass bei Änderung Benachrichtigung erfolgt; oder
    - Vergleiche Dumps ➔ snapshot differential problem

Integrator

- Empfang der (Änderungs-) Mitteilungen
- regelbasierende Zusammenführung der Änderungen in das Data Warehouse
- high-level Sprache zur Beschreibung der Integrierung sehr nützlich



Warehouse Update Anomalies

- Informationsquellen und Sichten sind unabhängig und entkoppelt
- Integrator könnte zum Aktualisieren weitere Daten benötigen, die zwischenzeitlich von einem anderen Update verändert wurden ➔ warehouse update anomaly.
- Lösungsansätze
  - Ereignisgesteuerte oder periodische Neukalkulation
  - lokales Speichern aller relevanter Daten
  - Eager Compensating Algorithm: Effekte nebenläufiger Updates kompensierende Anfragen (wenig Overhead)

Agenda

1. Einleitung
2. Probleme
3. Anwendungen
4. WHIPS
5. Zusammenfassung

Zusammenfassung

- nicht immer befriedigende Lösungen
  ➔ offene Probleme
- WHIPS nur *eine* Anwendung
- Fortschritte in der Konsistenzwahrung
- Data-Warehouse- und mobiles System in einem (→ OVID)
- nahezu unbegrenzte Entfaltungsmöglichkeiten

Fragen?