6.2 Datenmodellierung

Umsetzung des multidimensionalen Modells

- Interne Verwaltung der Daten durch
  - Relationale Strukturen (Tabellen)
    - Relationales OLAP (ROLAP)
    - Vorteile: Verfügbarkeit, Reife der Systeme
  - Multidimensionale Strukturen (direkte Speicherung)
    - Multidimensionales OLAP (MOLAP)
    - Vorteil: Wegfall der Transformation
- Wichtige Designaspekte
  - Speicherung
  - Anfragebearbeitung

6.2 Datenmodellierung

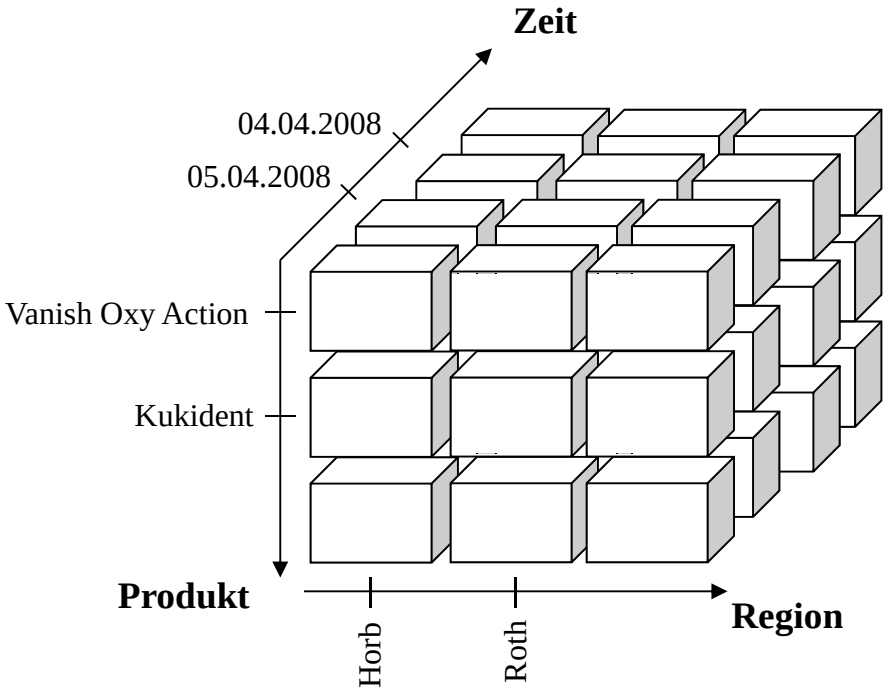
Relationale Umsetzung: Anforderungen

- Vermeidung des Verlusts anwendungsbezogener Semantik aus dem multidimensionalen Modell (z.B. Klassifikationshierarchien)
- Effiziente Übersetzung multidimensionaler Anfragen
- Effiziente Verarbeitung der übersetzten Anfragen
- Einfache Pflege der entstandenen Relationen (z.B. Laden neuer Daten)
- Berücksichtigung der Anfragecharakteristik und des Datenvolumens von Analyseanwendungen

6.2 Datenmodellierung

Relationale Umsetzung: Faktentabelle

- Ausgangspunkt: Umsetzung des Data-Cubes ohne Klassifikationshierarchien
  - Dimensionen und Kennzahlen => Attribute der Relation
  - Zellen => Tupel der Relation



Dimensionen *Kennzahl*

Artikel	Filiale	Tag	Verkäufe
Vanish Ox. A.	Horb	04.04.2008	4
Vanish Ox. A.	Horb	05.04.2008	1
Kukident	Horb	04.04.2008	12
Kukident	Roth	04.04.2008	0
Vanish Ox. A.	Roth	05.04.2008	2
⋮			

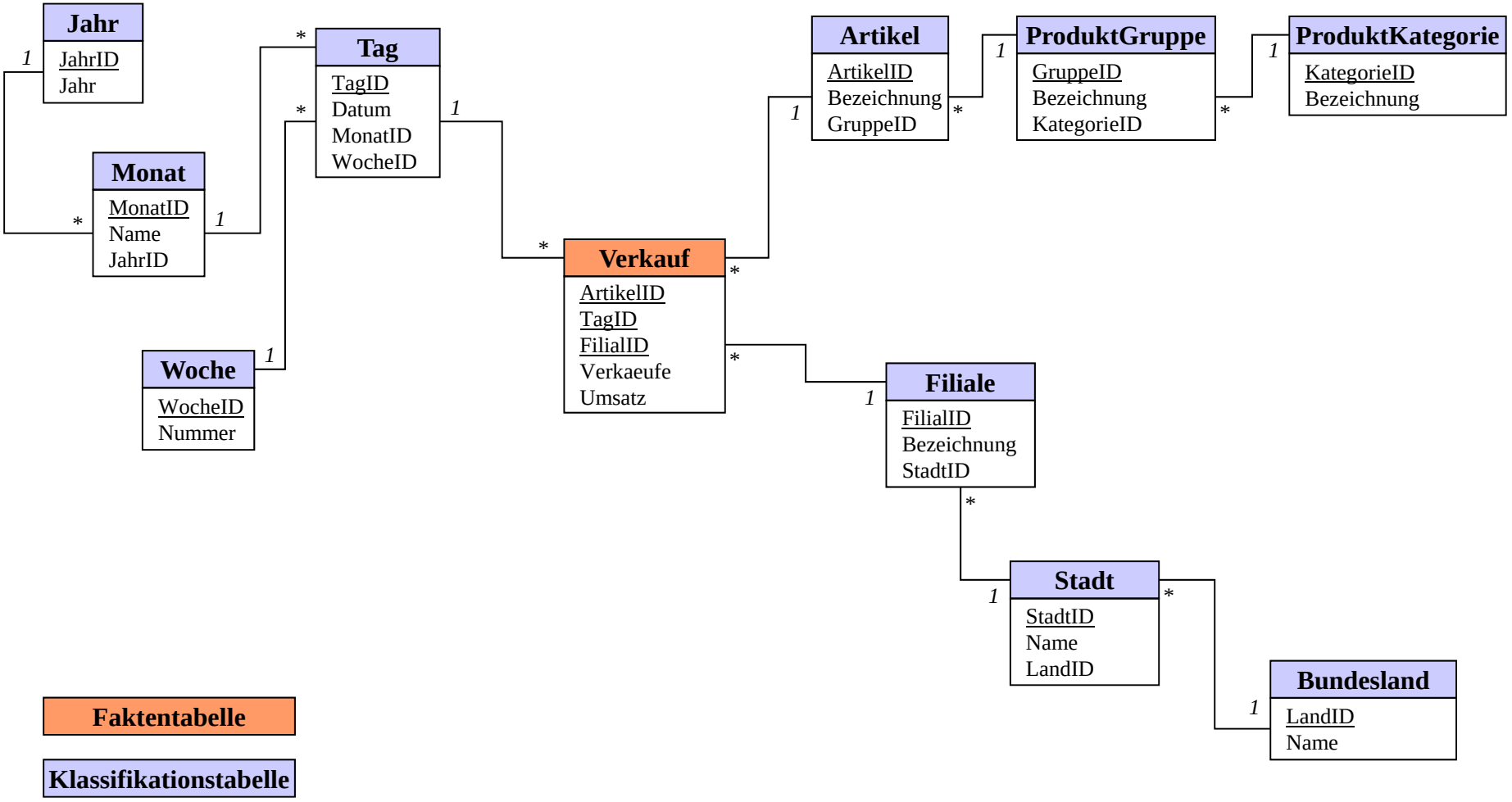
6.2 Datenmodellierung

Relationale Umsetzung: Snowflake Schema

- Abbildung von Klassifikationen?
- Eigene Tabelle für jede Klassifikationsstufe (Artikel, Produktgruppe, …)
- Klassifikationstabellen enthalten
  - ID für entsprechenden Klassifikationsknoten
  - Beschreibende Attribute (Marke, Hersteller, Bezeichnung, …)
  - Fremdschlüssel der direkt übergeordneten Klassifikationsstufe
- Faktentabelle enthält
  - Kenngrößen
  - Fremdschlüssel der jeweils niedrigsten Klassifikationsstufe der einzelnen Dimensionen
  - Fremdschlüssel bilden zusammengesetzte Primärschlüssel der Faktentabelle

6.2 Datenmodellierung

Snowflake Schema: Beispiel



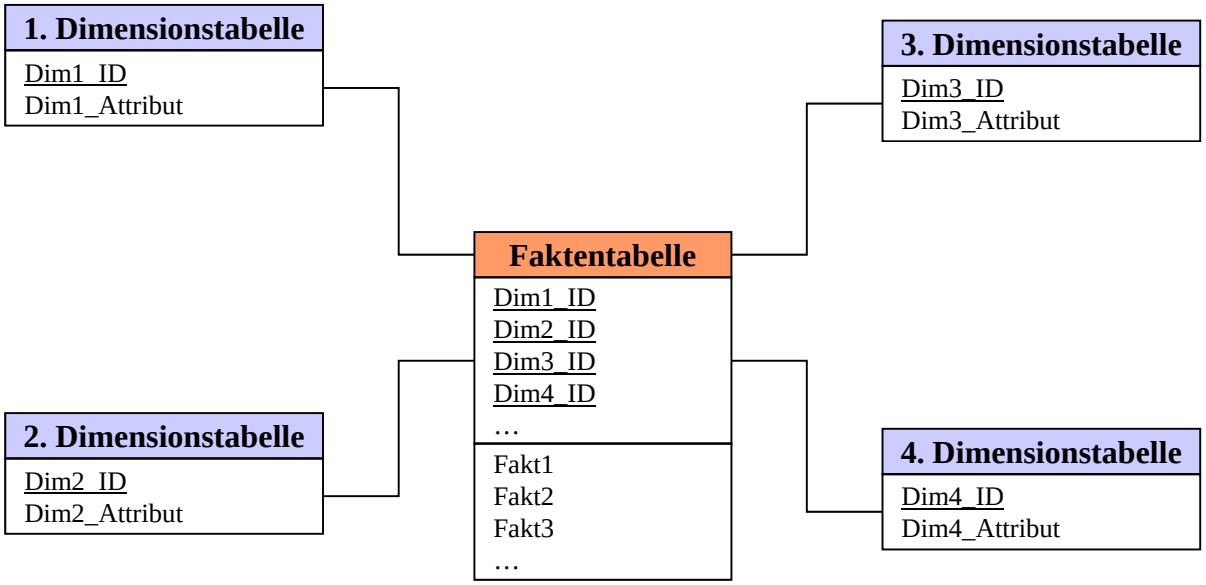
6.2 Datenmodellierung

Relationale Umsetzung: Star Schema

- Snowflake Schema ist normalisiert
  - Keine Update-Anomalien
  - ABER: Zusammenholen von Informationen erfordert Join über mehrere Tabellen
- Star Schema
  - Denormalisierung der zu einer Dimension gehörenden Tabellen
  - Für jede Dimension genau eine ***Dimensionstabelle***
  - Redundanzen in der Dimensionstabelle für schnellere Anfragebearbeitung

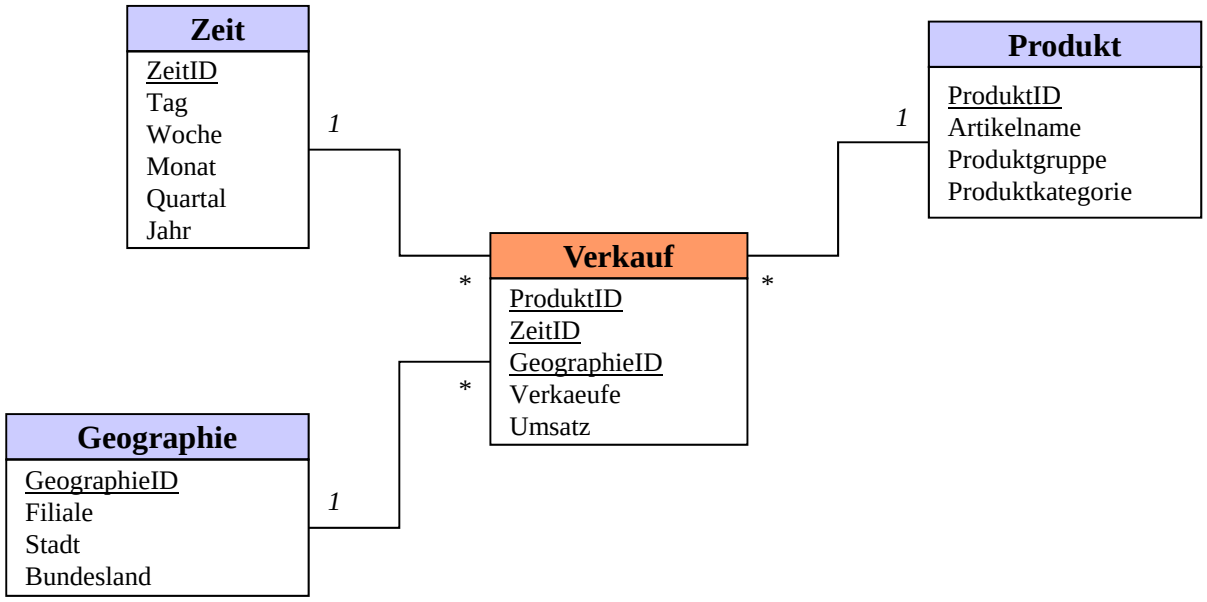
6.2 Datenmodellierung

Star Schema: Visualizierung



6.2 Datenmodellierung

Star Schema: Beispiel



6.2 Datenmodellierung

Relationale Umsetzung: Mischformen

- Idee: Abbildung einzelner Dimensionen anhand von Snowflake oder Star Schema
  - Kriterien
    - Änderungshäufigkeit der Dimensionen

      Reduzierung des Pflegeaufwands => Snowflake
    - Anzahl der Klassifikationsstufen einer Dimension

      Höhere Effizienz durch größere Redundanz => Star
    - ...

Relationale Umsetzung: Begriff

- Galaxie (Multi-Cube, Hyper-Cube)
  - Mehrere Faktentabellen im Star Schema teilweise mit gleichen Dimensionstabellen verknüpft
- Fact Constellation
  - Speicherung vorberechneter Aggregate in Faktentabelle (z.B. Umsatz für Region)

6.2 Datenmodellierung

Relationale Umsetzung: Probleme

- Transformation multidimensionaler Anfragen in relationale Repräsentation nötig
- Einsatz komplexer Anfragewerkzeuge nötig (OLAP-Werkzeuge)
- Semantikverlust
  - Unterscheidung zwischen Kennzahlen und Dimensionen in der Faktentabelle nicht gegeben
  - Unterscheidung zwischen beschreibenden Attributen und Attributen zum Hierarchie-Aufbau in Dimensionstabellen nicht gegeben
- Daher:

  direkte multidimensionale Speicherung besser ???

6.2 Datenmodellierung

Multidimensionale Umsetzung

- Idee:
  - Verwende entsprechende Datenstrukturen für Data-Cube und Dimensionen
  - Speicherung des Data-Cube als Array
  - Ordnung der Dimensionen nötig, damit Zellen des Data-Cube addressiert werden können
- Bemerkung
  - Häufig proprietäre Strukturen (und Systeme)

Multidimensionale Umsetzung (cont.)

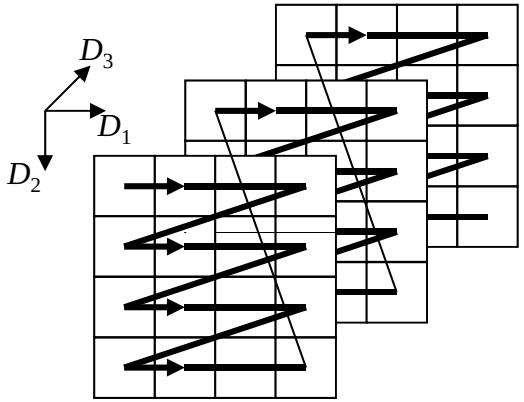
- Datenstruktur für eine Dimension
  - Endliche geordnete Liste von Dimensionswerten (aller Klassifikationsstufen)
  - Dimensionswerte: einfache, atomare Datentypen (String, Integer, Date, …)
- Datenstruktur für Cube
  - Für *d* Dimensionen: *d*-dimensionaler Raum
  - Bei *m* Werten in einer Dimension: Aufteilung des Würfels in *m* parallele Ebenen => endliche gleichgroße Liste von Ebenen je Dimension
  - Zelle eines *d*-dimensionalen Cubes wird eindeutig über *d* Dimensionswerten identifiziert
  - Pro Kennzahl in Zelle ein entsprechendes Array

6.2 Datenmodellierung

Multidimensionale Umsetzung (cont.)

- Speicherung des Data-Cube:
  - Linearisierung des *d*-dimensionalen Arrays in ein 1-dimensionales Array
  - Koordinaten der Würfelzellen (Dimensionen) „entsprechen“ Indizes des Arrays
  - Indexbrechnung für Zelle mit Koordinaten $z = x_1, \ldots, x_d$

$$
\begin{aligned}
Index(z) \quad = \quad & x_1 + (x_2 - 1) \cdot |D_1| \\
+ \quad & (x_3 - 1) \cdot |D_1| \cdot |D_2| \\
& \vdots \\
+ \quad & (x_d - 1) \cdot |D_1| \cdot \ldots \cdot |D_{d-1}|
\end{aligned}
$$



6.2 Datenmodellierung

Multidimensionale Umsetzung (cont.)

- Vorteile
  - Direkte OLAP-Unterstützung
  - Analytische Mächtigkeit
- Grenzen
  - Hohe Zahl an Plattenzugriffen bei ungünstiger Linearisierungsreihenfolge
  - Durch die Ordnung der Dimensionswerte (für Array-Abbildung nötig) keine einfache Änderung an Dimensionen möglich
  - Kein Standard für multidimensionale DBMS
- Oft: Hybride Speicherung HOLAP = MOLAP + ROLAP
  - Relationale Speicherung der Datenbasis
  - Multidimensionale Speicherung für häufig aggregierte Daten (z.B. angefragte (Teil-)Data Cubes)

6 Einführung in Data Warehouses

Übersicht

6.3 Anfragebearbeitung

Motivation

- Typische Anfragen an Data Warehouses beinhalten Aggregationen

  Wieviele Artikel der Produktgruppe Elektrogeräte wurden im Januar 2000 pro Tag in den einzelnen Regionen in Bayern verkauft?

- Charakteristik typischer DW-Anfragen
  - Große Menge vorhandener Fakten
  - Daraus nur ein bestimmter, in den meisten Dimensionen beschränkter Datenbereich angefragt
  - Problem: Aggregation auf großen Datenmengen

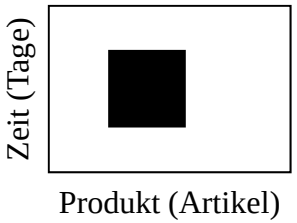
    z.B. 1TB Verkaufsdaten komplett einlesen dauert bei einer Leserate von 200 MB/s: 1000000/200 s = 5000 s

    d.h. ca. 83 min=> inakzeptabel!!!

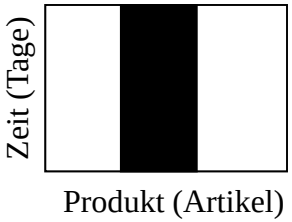
6.3 Anfragebearbeitung

Multidimensionale Anfragen

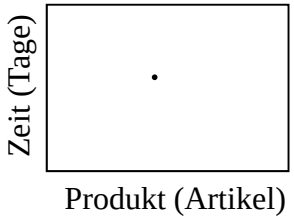
- Bereichsanfrage (range query)



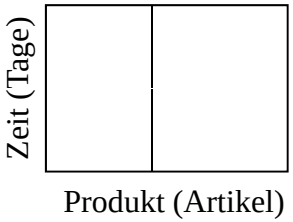
- Partielle Bereichsanfrage (partial range query)



- Punktanfrage (match query)



- Partielle Punktanfrage (partial match query)



- …

6.3 Anfragebearbeitung

Umsetzung

- Grundsätzlich abhängig von der Umsetzung des Schemas
  - Star Schema vs. Snowflake Schema
  - Multidimensionale Speicherung
- Häufige Anfrage-Muster auf relationaler Umsetzung
  - (n+1)-Wege-Join zwischen
    - n Dimensionstabellen
    - Faktentabelle
  - Restriktionen über Dimensionstabellen

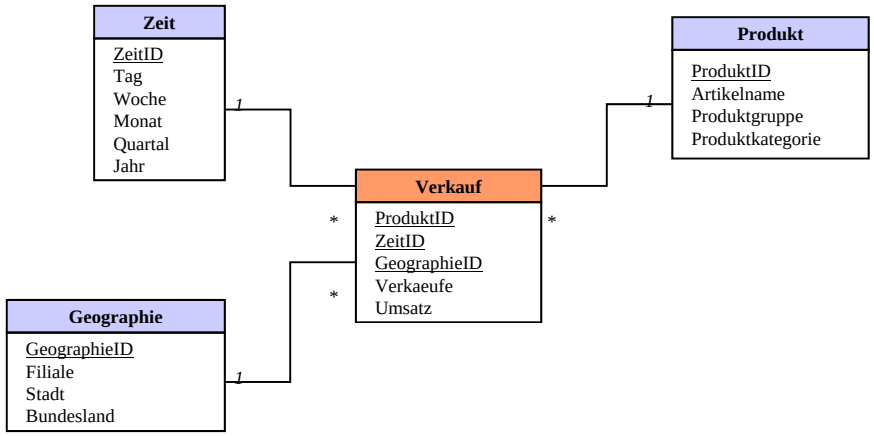
    (z.B. Region = Deutschland, Produktgruppe = Elektrogeräte, Jahr = 2000)

6.3 Anfragebearbeitung

Star Join

- Beispiel

  Wieviele Artikel der Produktgruppe Elektrogeräte wurden im Januar 2000 pro Tag in den einzelnen Regionen in Bayern verkauft?




```
SELECT      Geographie.Bundesland, Zeit.Monat, SUM(Verkaeufe)
FROM              Produkte, Zeit, Geographie, Verkauf
WHERE       Verkauf.ProduktID = Produkt.ProduktID
            AND   Verkauf.ZeitID = Zeit.ZeitID
            AND   Verkauf.GeographieID = Geographie.GeographieID
            AND   Produkt.Produktgruppe = 'Elektrogeraete'
            AND   Geographie.Bundesland = 'Bayern'
            AND   Zeit.Monat = 'Januar 2000'
GROUP BY          Geographie.Region, Zeit.Tag
```

6.3 Anfragebearbeitung

Star Join

- Allgemein:
  - SELECT-Klausel
    - Kenngrößen (evtl. aggregiert)
    - Ergebnisgranularität (der Dimensionen)

      z.B. Zeit.Monat, Geographie.Region
  - FROM-Klausel
    - Faktentabelle
    - Dimensionstabellen
  - WHERE-Klausel
    - Verbundbedingungen
    - Restriktionen in Dimensionen

      z.B. Produkt.Produktgruppe = ‘Elektrogeraete‘, Zeit.Monat= ‘Januar 2000‘, …

6.3 Anfragebearbeitung

Star Join: Optimierung

- Star-Join ist ein typisches Muster für Anfragen in Data Warehouses
- Typische Charakteristik (wegen Star Schema)
  - Sehr große Faktentabelle
  - Relativ kleine, unabhängige Dimensionstabellen
- Heuristiken klassischer relationaler Optimierer schlagen hier meist fehl
  - Optimieren unter der Annahme, dass alle Relationen etwa gleich groß sind

6.3 Anfragebearbeitung

Star Join: Optimierung (cont.)

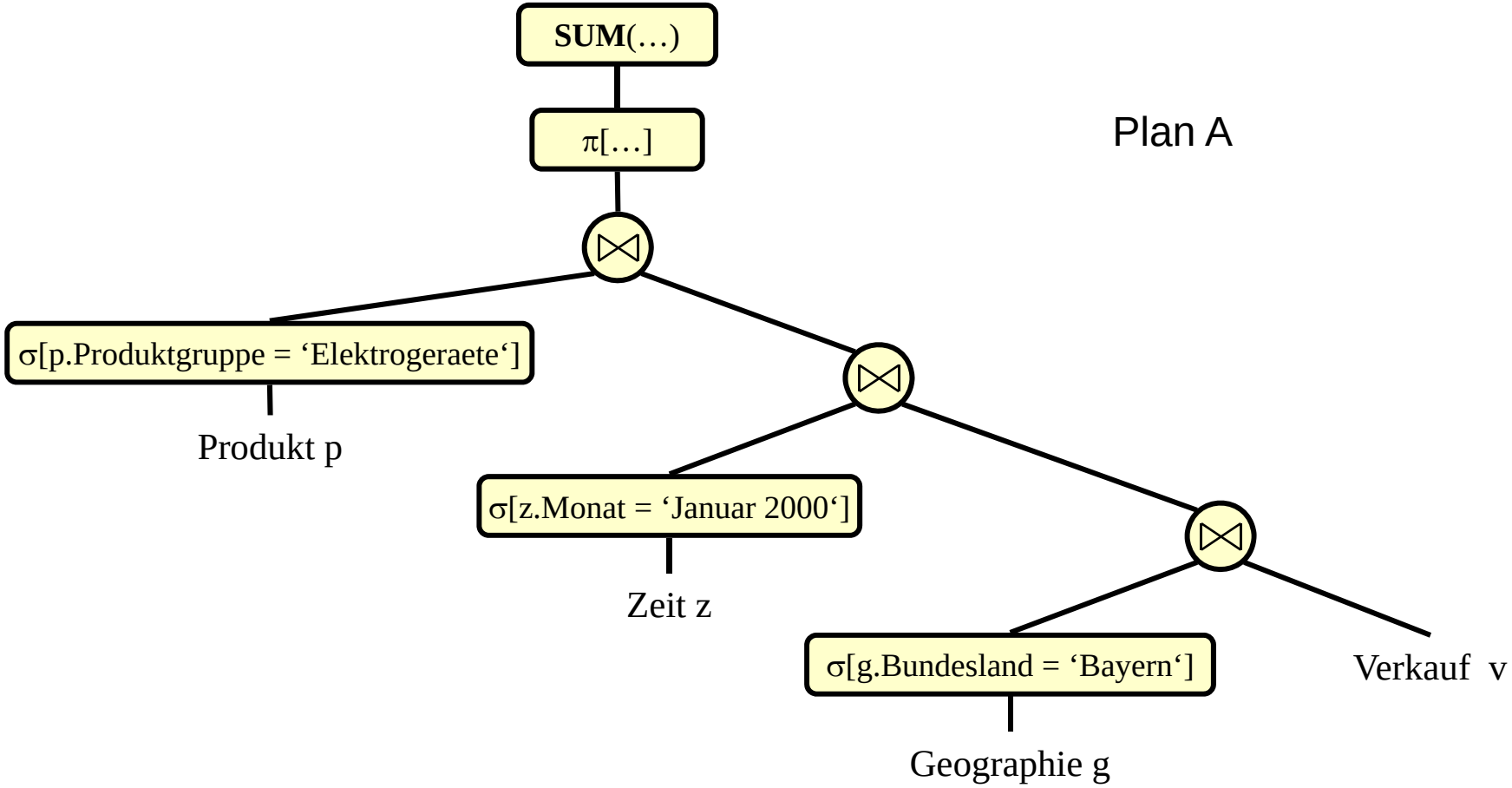
- Auswertungsplan?
  - 4-Wege Join (Join über 4 Tabellen: Verkauf, Produkt, Zeit, Geographie)
  - In relationalen DBS kann typischerweise nur paarweise gejoint werden => Sequenz paarweiser Joins
  - Es gibt 4! = 24 viele mögliche Join-Reihenfolgen (= mögliche Auswertungspläne)
  - Heuristik zur Verringerung der Anzahl der Möglichkeiten:

    Joins zwischen Relationen, die nicht durch Join-Bedingung in Anfrage verknüpft, NICHT betrachten

6.3 Anfragebearbeitung

Star Join: Optimierung (cont.)

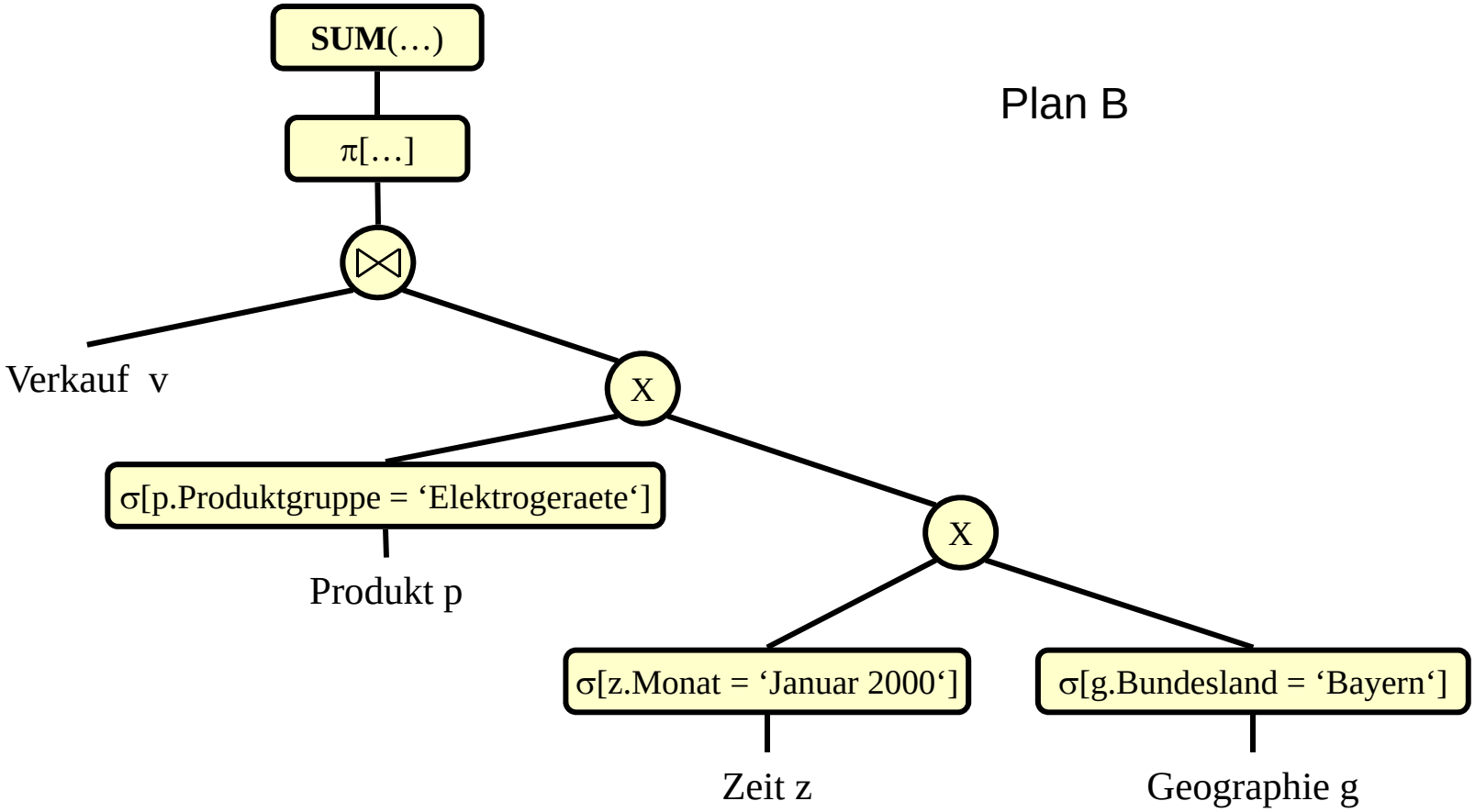
- Optimierter kanonischer Auswertungsplan:



6.3 Anfragebearbeitung

Star Join: Optimierung (cont.)

- Alternativer Auswertungsplan, der üblicherweise nicht betrachtet wird



6.3 Anfragebearbeitung

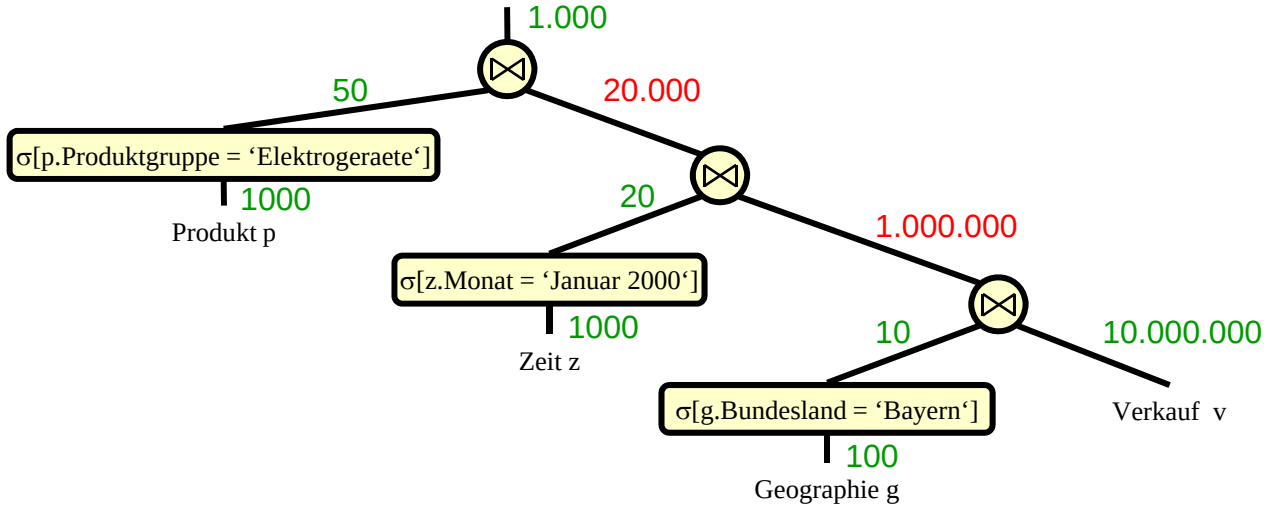
Star Join: Optimierung (cont.)

- Vergleich von Plan A und B
- Szenario:
  - Tabelle Verkauf: 10.000.000 Datensätze
  - 10 Geschäfte in Bayern (von 100)
  - 20 Verkaufstage im Januar 2000 (von 1000 gespeicherten Tagen)
  - 50 Produkte in Produktgruppe „Elektrogeräte“ (von 1000)
  - Gleichverteilung/gleiche Selektivität der einzelnen Ausprägungen

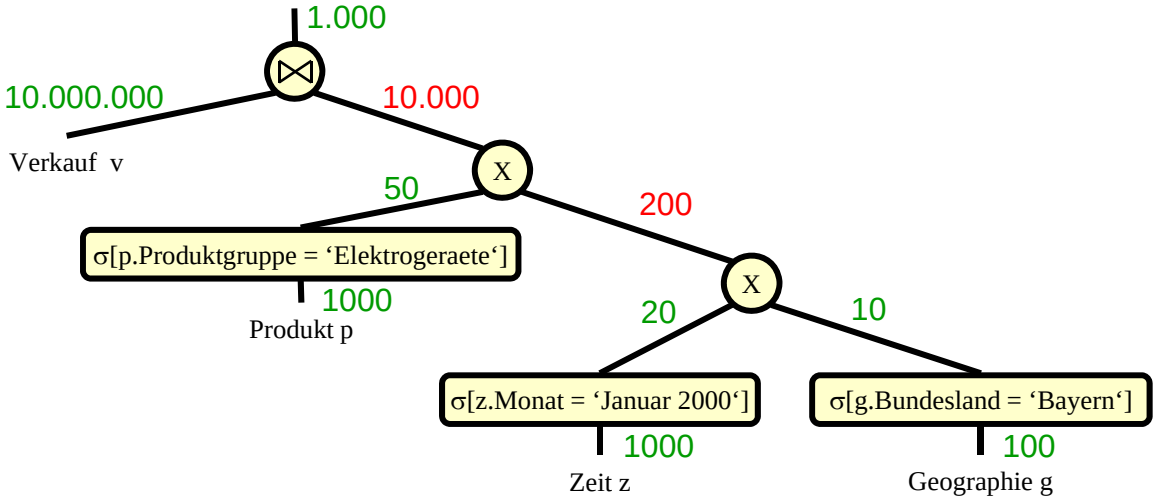
6.3 Anfragebearbeitung

Star Join: Optimierung (cont.)

- Plan A



- Plan B



6.3 Anfragebearbeitung

Roll-UP/Drill-Down

Verdichtungsgrad wird durch **GROUP BY**-Klausel spezifiziert:

- Mehr Attribute in **GROUP BY**
  => weniger starke Verdichtung
  => Drill-Down

- Weniger Attribute in **GROUP BY**
  => stärkere Verdichtung
  => Roll-Up

6.3 Anfragebearbeitung

Weitere Optimierungs-Heuristiken

- Materialisierung von Aggregaten
  - Aggregation ist sehr zeitaufwendig
  - Berechne häufig verwendete Aggregationen einmal und materialisiere deren Ergebnis
- **CUBE**-Operator (z.B in SQL-Server, DB2, Oracle 8i)
  - Aggregationen mit drill-down/roll-up entlang aller Dimensionen, die in **GROUP BY**-Klausel vorkommen
  - Ermöglicht einfachere Formulierung dieser Aggregation
  - Ermöglicht Optimierung dieser Aggregation
    - „normalerweise“ müsste Faktentabelle mehrmals gelesen werden, da Aggregation durch mehrere mit **UNION** verknüpfte Unteranfragen berechnet werden müsste
    - Durch **CUBE**-Operator: nur einmaliges lesen der Faktentabelle