Übersicht

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?

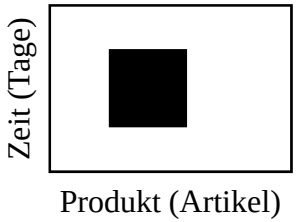
- Charakteristik typischer DW-Anfragen
  - Große Menge vorhandener Fakten
  - Daraus nur ein bestimmter, in den meisten Dimensionen beschränkter Datenbereich angefragt
  - Problem: Aggregation auf großen Datenmengen

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

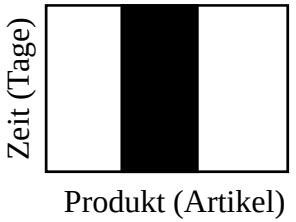
    d.h. ca. 83 min => inakzeptabel!!!

Multidimensionale Anfragen

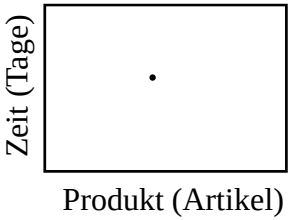
- Bereichsanfrage (range query)



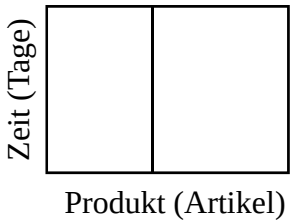
- Partielle Bereichsanfrage (partial range query)



- Punktanfrage (match query)



- Partielle Punktanfrage (partial match query)



- …

Umsetzung

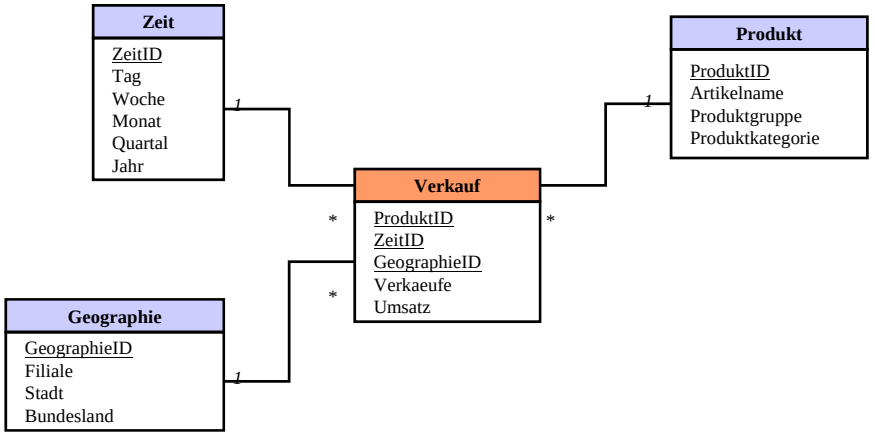
- Grundsätzlich abhängig von der Umsetzung des Schemas
  - Star Schema vs. Snowflake Schema
  - Multidimensionale Speicherung
- Häufige Anfrage-Muster auf relationaler Umsetzung
  - (*n*+1)-Wege-Join zwischen
    - *n* Dimensionstabellen
    - Faktentabelle
  - Restriktionen über Dimensionstabellen

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

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
```

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‘, …

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

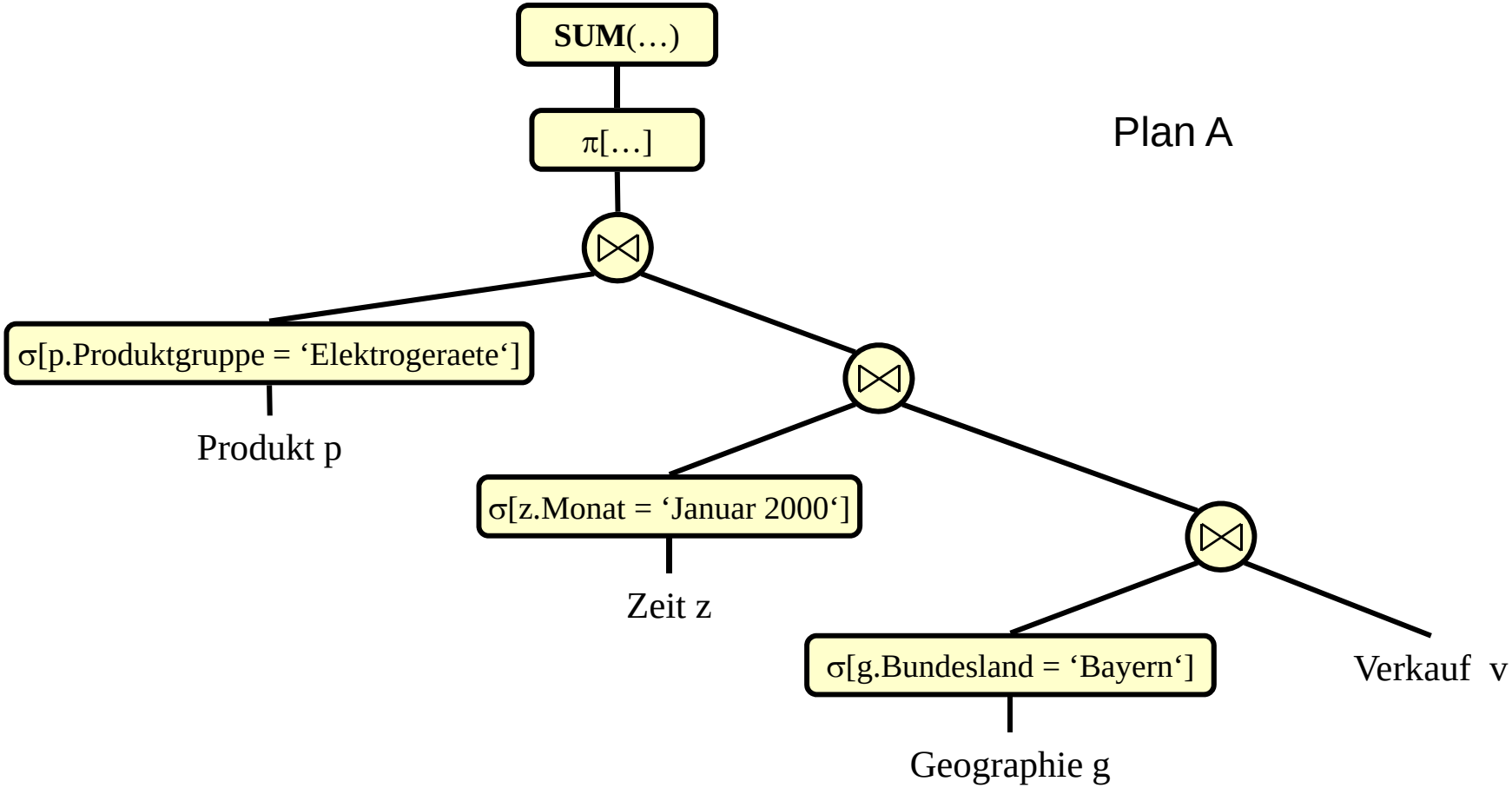
Star Join: Optimierung (cont.)

– 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

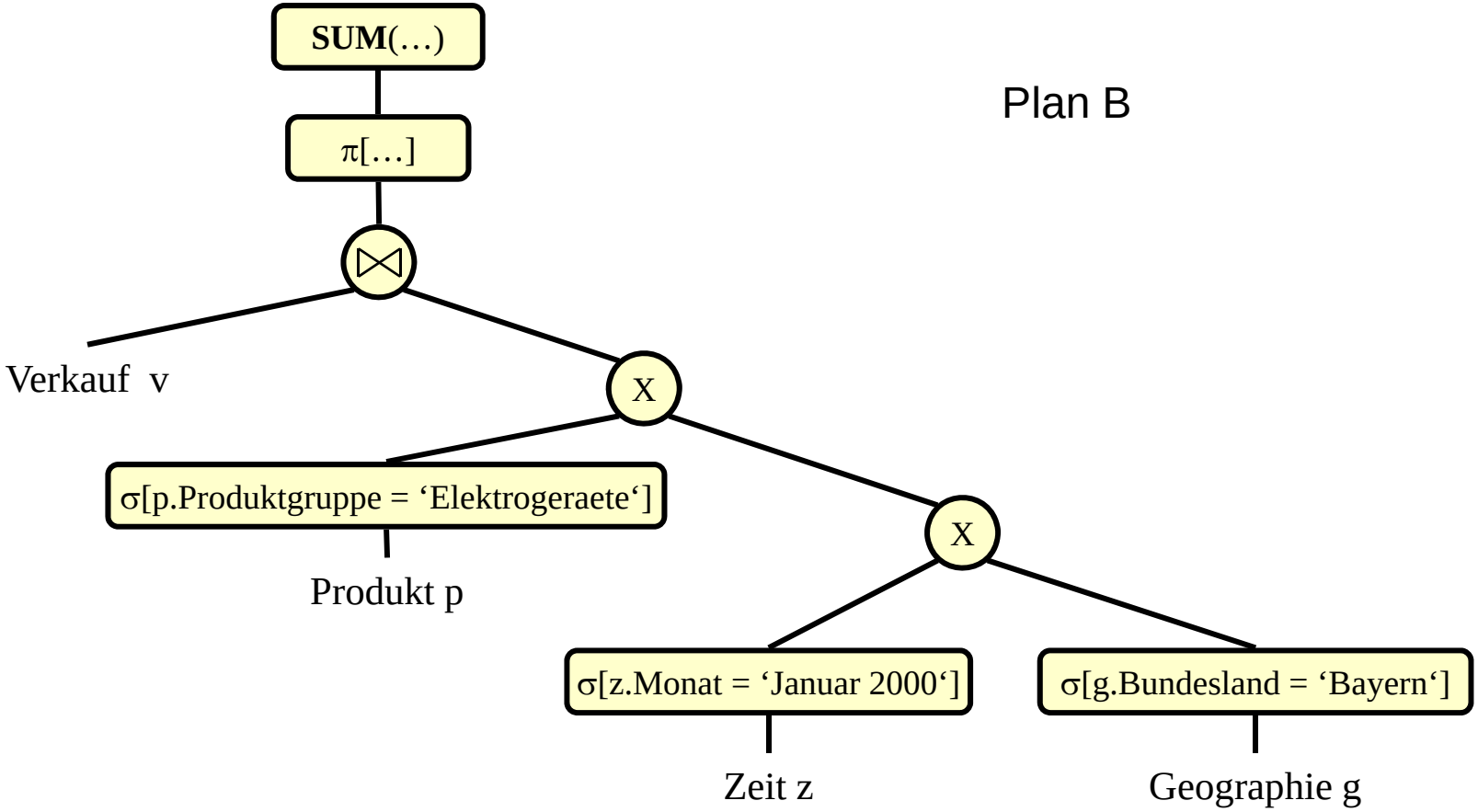
Star Join: Optimierung (cont.)

– Optimierter kanonischer Auswertungsplan:



Star Join: Optimierung (cont.)

– Alternativer Auswertungsplan, der üblicherweise nicht betrachtet wird



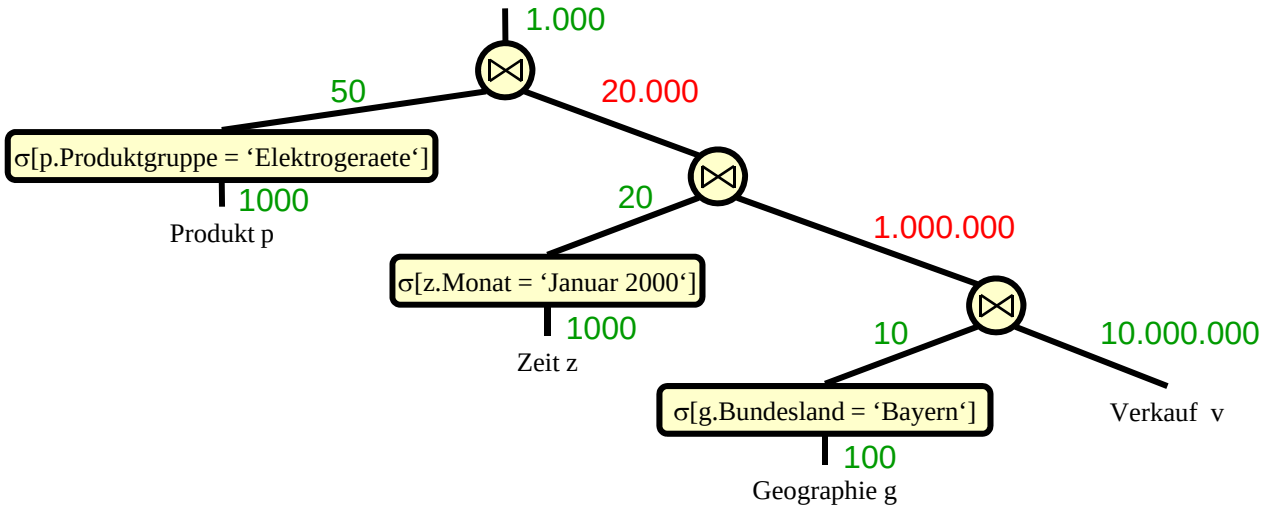
Plan B

Star Join: Optimierung (cont.)

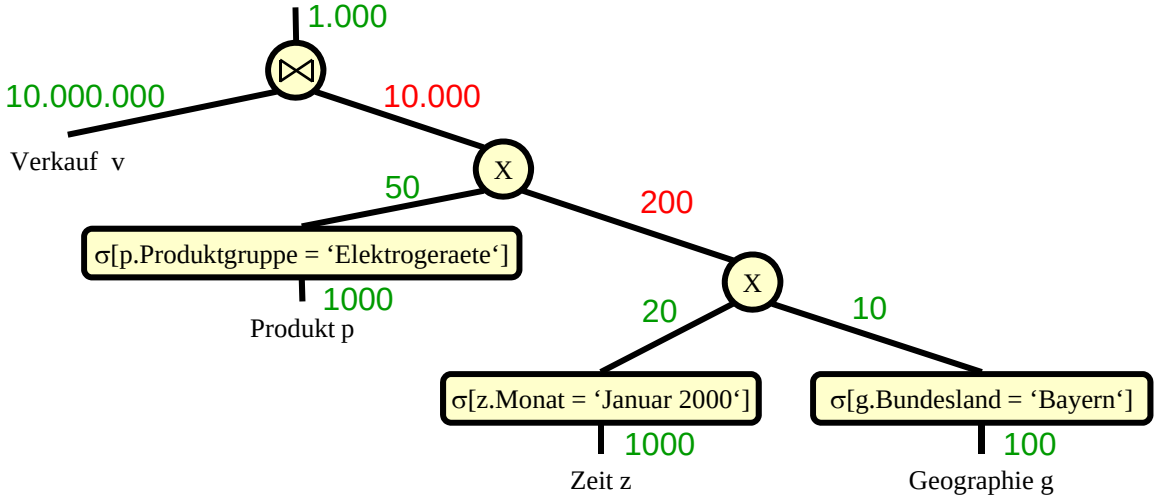
- Vergleich von Plan A und B
- Szenario:
  - Tabelle Verkauf: 10.000.000 Datensätze
  - 10 Geschäfte in Bayern (von 100)
  - 20 Verkaufstage im Januar 2000 (von 1000 gespeicherten Tagen)
  - 50 Produkte in Produktgruppe „Elektrogeräte“ (von 1000)
  - Gleichverteilung/gleiche Selektivität der einzelnen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

Star Join: Optimierung (cont.)

– Plan A



– Plan B



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

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