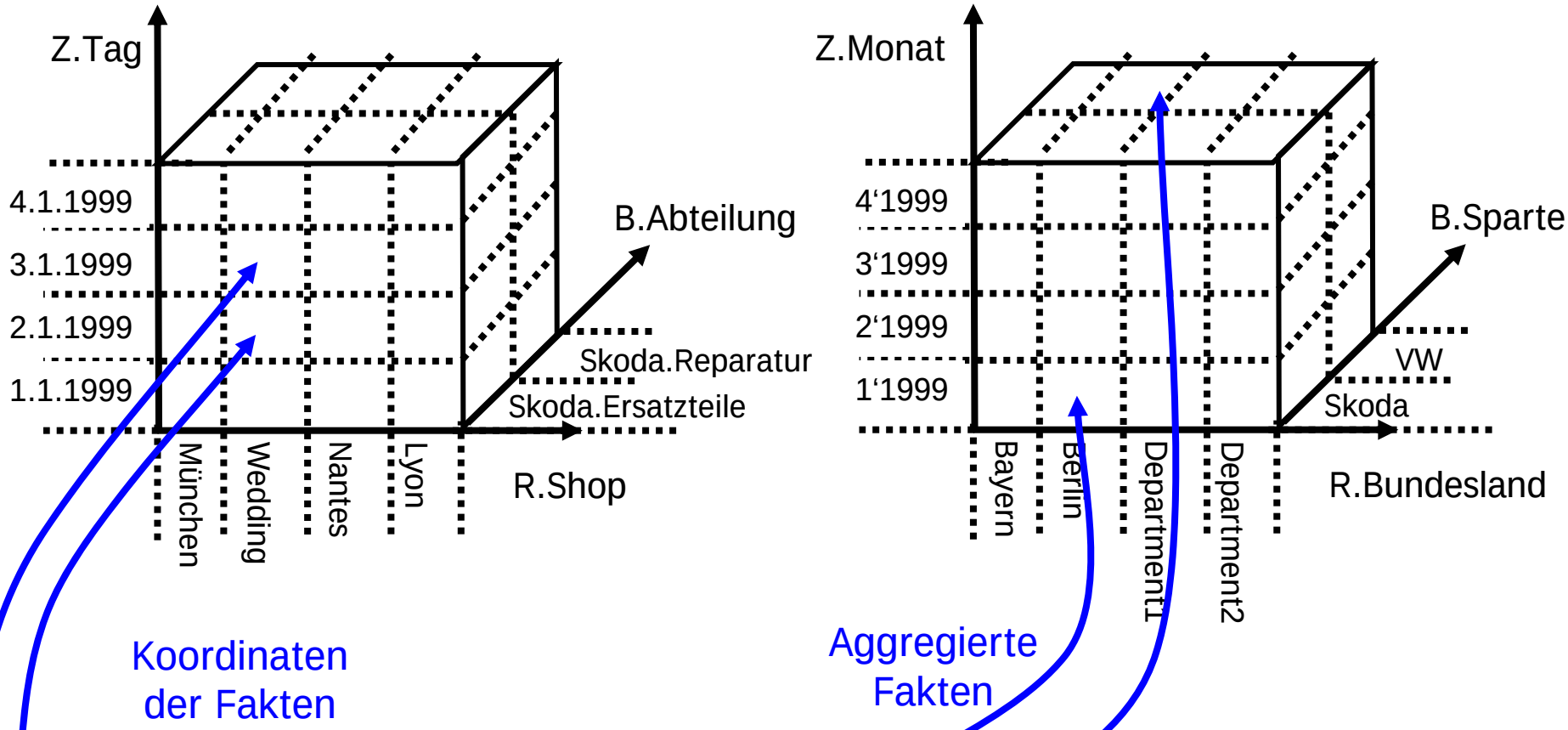
Data Warehousing

Speicherung multidimensionaler Daten

Ulf Leser

Wissensmanagement in der Bioinformatik

Beispiel



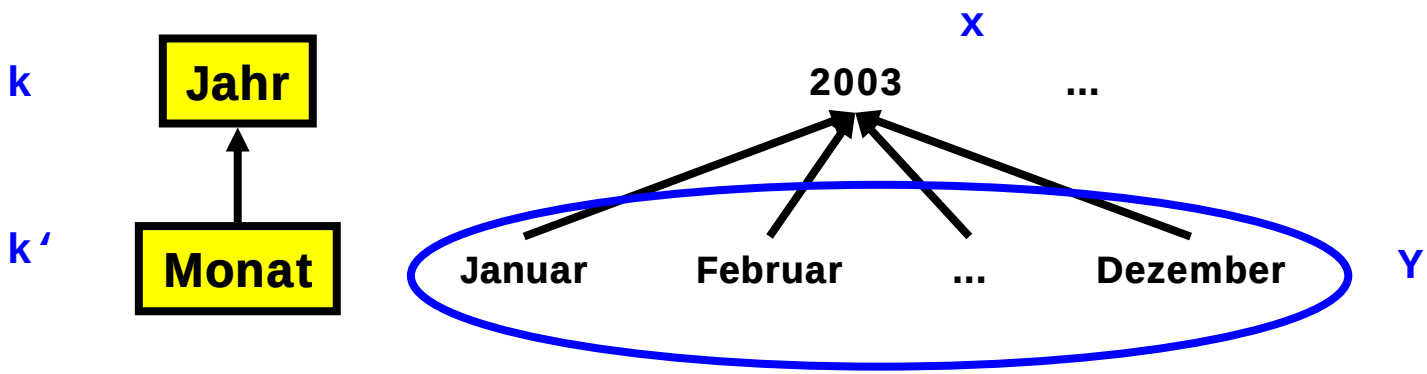
(3.1.1999, Wedding, Ersatzt.) = (...)	(1‘1999, Berlin, Skoda) = (...)
(2.1.1999, Wedding, Ersatzt.) = (...)	(4‘1999, Bayern, VW) = (...)

Würfelinhalt, allgemeiner Fall

- Ein Würfel W=(G,F)
  - Granularität $G=(D_1.k_1,...,D_n.k_n)$
  - Menge Measures $F=\{F_1,...,F_m\}$ mit Aggregatfunktionen $f_1,...,f_m$
- Allgemeiner Fall
  - $W(x_1,...,x_n) = (F_1,...,F_m) =$
    $(F_1(\text{punkte}(x_1) \cap \text{punkte}(x_2) \cap ... \cap \text{punkte}(x_n)),$
    $F_2(\text{punkte}(x_1) \cap \text{punkte}(x_2) \cap ... \cap \text{punkte}(x_n)),$
    ...
    $F_m(\text{punkte}(x_1) \cap \text{punkte}(x_2) \cap ... \cap \text{punkte}(x_n)))$
  - mit Koordinaten $x_i \in \text{knoten}(k_i)$
- Bemerkung
  - Die Schnittmengenbildung berechnet die Menge von Punkten in der Würfelzelle mit Koordinaten $x_1,...,x_n$
  - Jedes Measure wird gesondert aggregiert

Aggregation in Hierarchien

- *Definition.*
  - *Gegeben*
    - *Dimension D, Pfad $P=\{k_0 \rightarrow \ldots \rightarrow k' \rightarrow k \rightarrow \ldots \rightarrow TOP\}$ in D*
    - *$x \in$ knoten(k) ein Klassifikationsknoten der Klassifikationsstufe $k \in P$*
    - *Aggregatfunktion f , Measure F*
  - *Sei Y=kinder(x) die Menge $\{y_1,\ldots,y_n\} \subseteq$ knoten(k') von Klassifikationsknoten von k', von denen x funktional abhängt*
  - *Die Aggregation von Y nach x bzgl. f und F bezeichnet die Berechnung des aggregierten Faktes $F(x)=f(F(y_1),\ldots., F(y_n))$*
  - *Die Aggregation von k' nach k bzgl. f und F bezeichnet die Berechnung von F(x) für alle $x \in$ knoten(k)*
    - Das schreiben wir kurz als F(k)

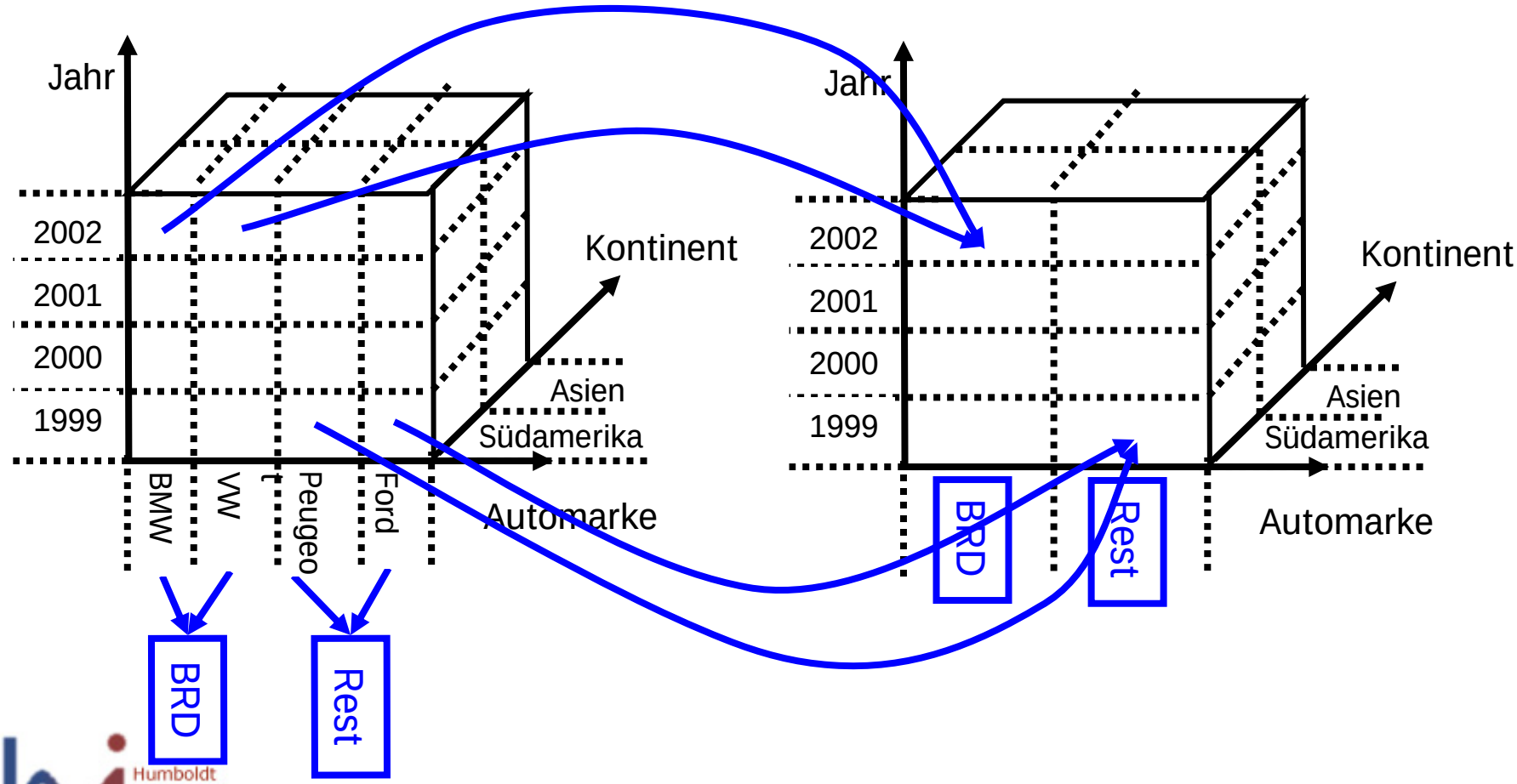


Operationen auf Würfeln

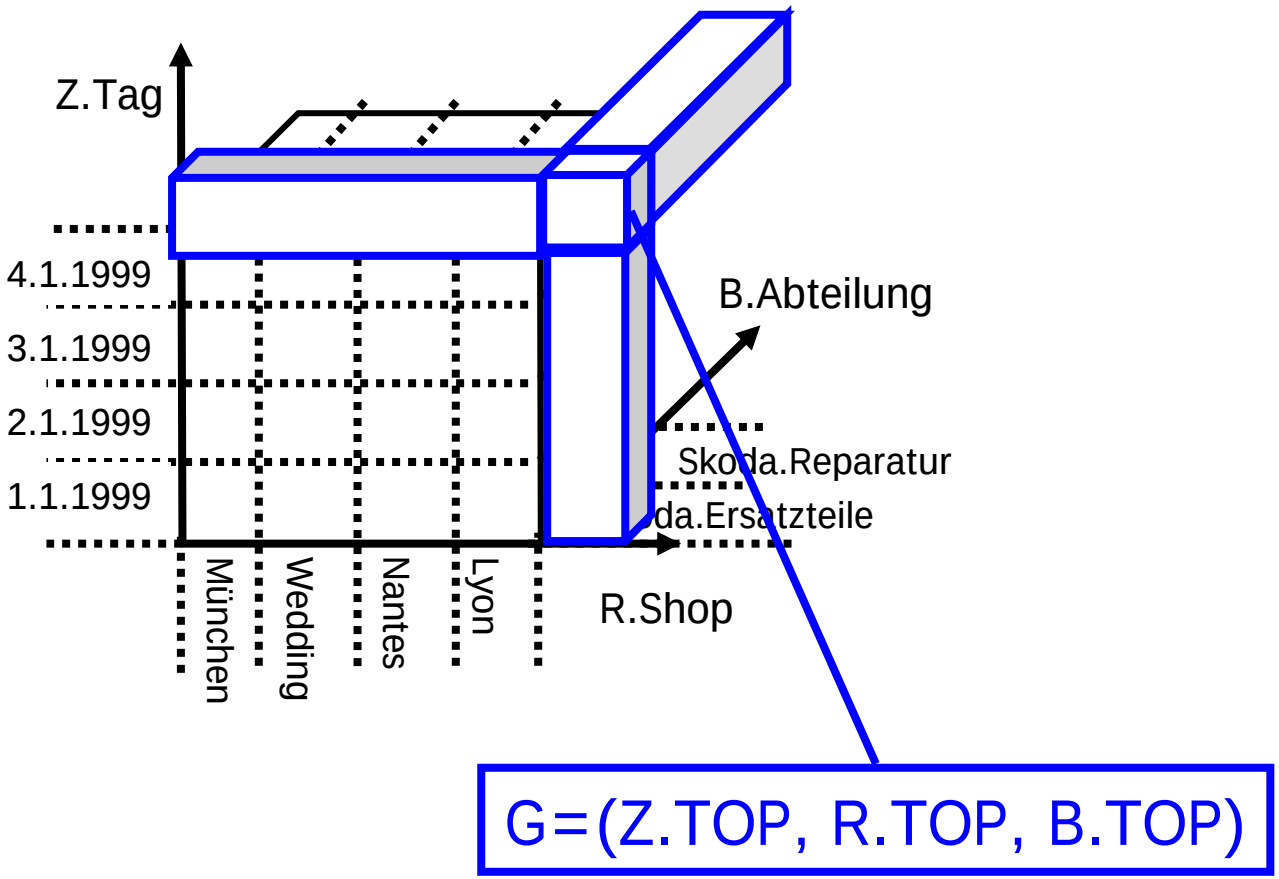
- OLAP Operationen überführen einen Würfel W=(G,F) in einen Würfel W‘=(G‘,F‘)
- Dabei gilt
  - Aggregation: G < G‘
  - Verfeinerung G > G‘
- Eine einfache Operation verändert nur die Klassifikationsstufe einer Dimension in G
  - Komplexe Operationen können auf natürliche Weise aus einfachen Operationen durch Verkettung zusammengesetzt werden
  - Wir betrachten deshalb im folgenden nur einfache Operationen

Aggregation (Roll-Up)

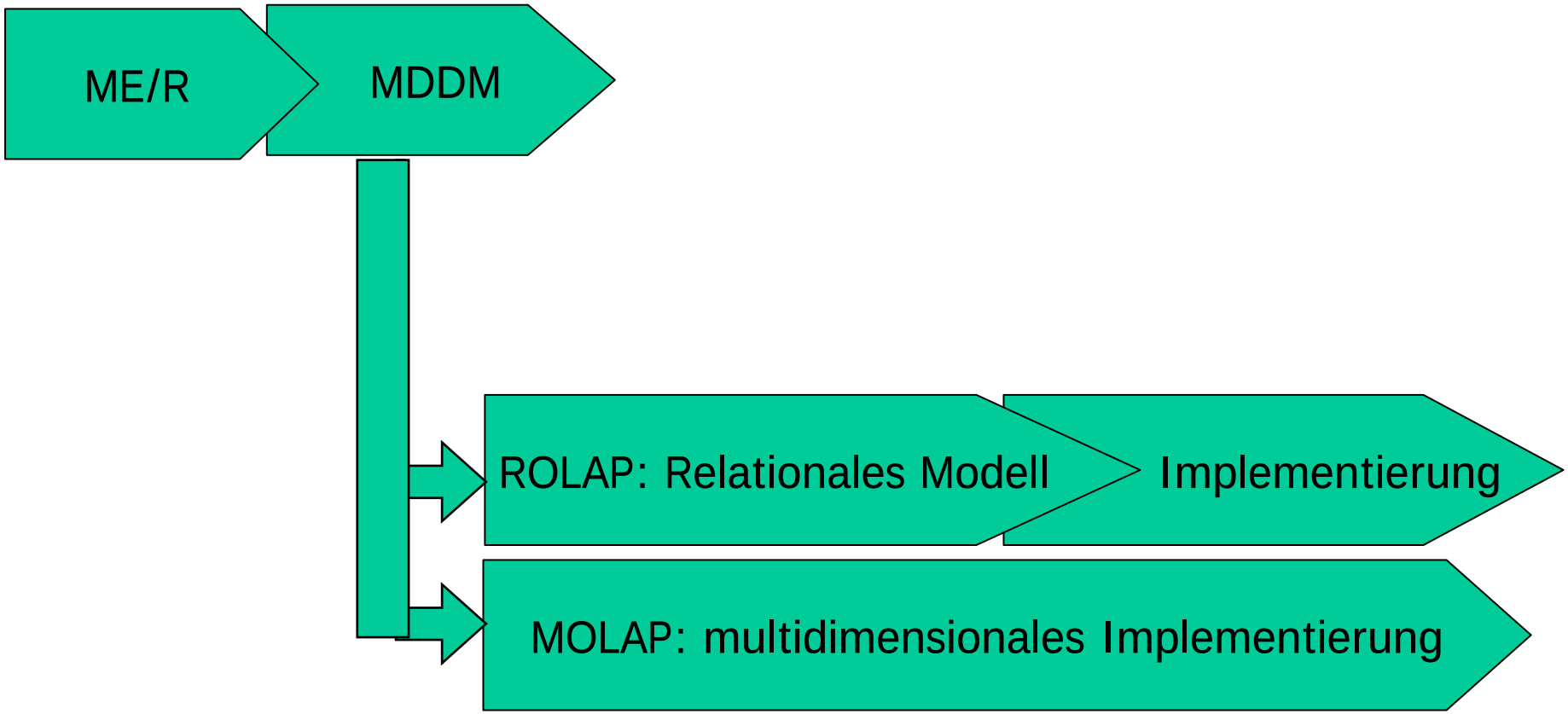
Aggregation entlang eines Klassifikationspfades



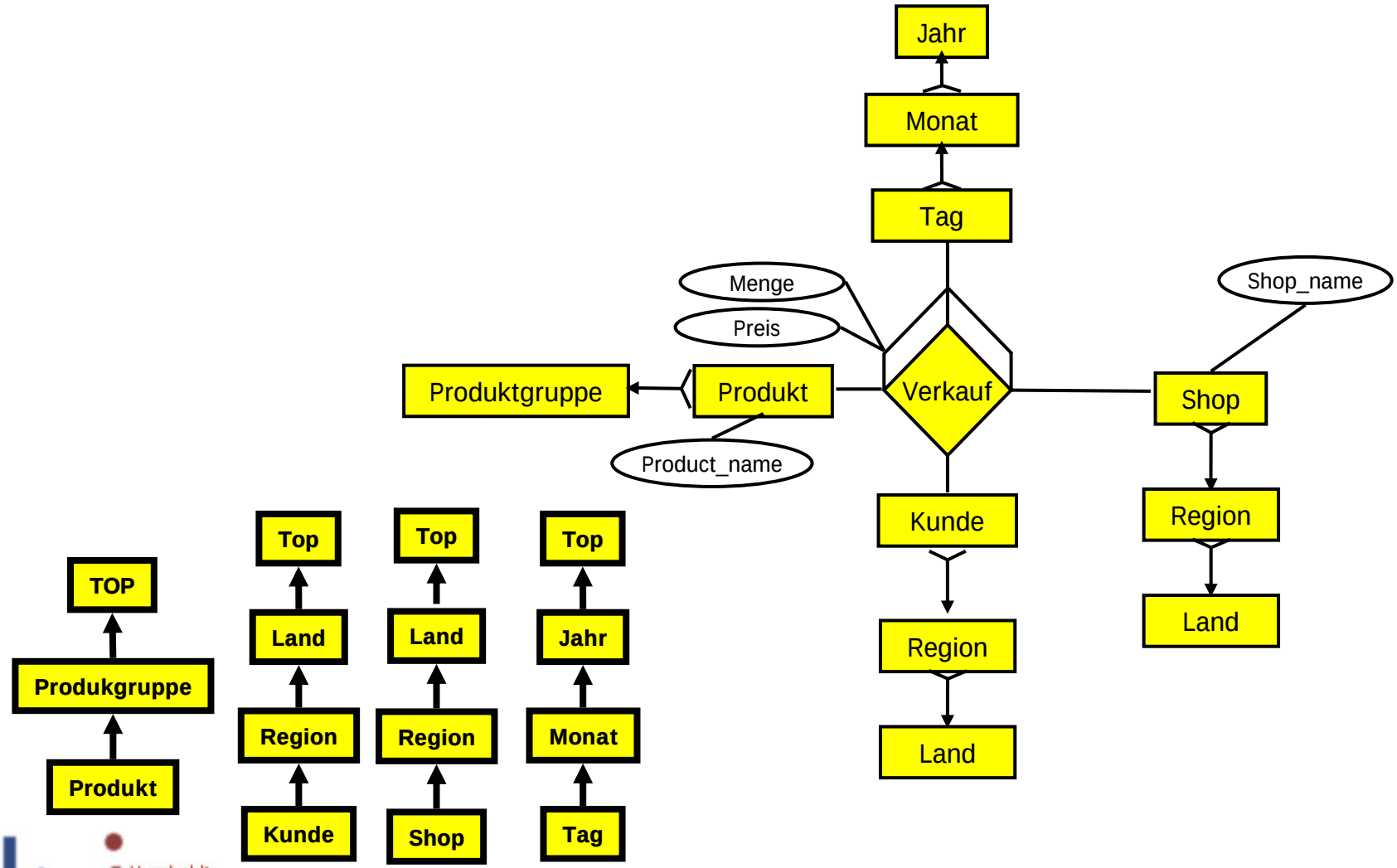
… in mehreren Dimensionen



Gesamtbild


ME/R
MDDM
ROLAP: Relationales Modell
Implementierung
MOLAP: multidimensionales Implementierung

Beispiel


Jahr
Monat
Tag
Menge
Preis
Shop_name
Produktgruppe
Produkt
Verkauf
Shop
Product_name
Kunde
Region
Land
Region
Land
TOP
Produkgruppe
Produkt
Top
Land
Region
Kunde
Top
Land
Region
Shop
Top
Jahr
Monat
Tag

Aggregationen – worauf achten

- Hierarchische Verdichtung entlang eines Pfades

Distributiv	Summe, Count, Max, Min, ...
Algebraisch	AVG (mit G_1=SUM und G_2=CNT) STDDEV, MaxN, ...
Holistisch	MEDIAN, RANK, PERCENTILE Highest Frequency, ...

Aggregationen – worauf achten 2

- Klassifikationsstruktur
  - Überlappungsfreiheit
  - Vollständigkeit
- Verträglichkeit Fakttyp – Aggregatfunktion

	Stock	Flow	Value-per-Unit
MIN/MAX	√	√	√
SUM	Zeit: nein Sonst: √	√	Nie
AVG	√	√	√

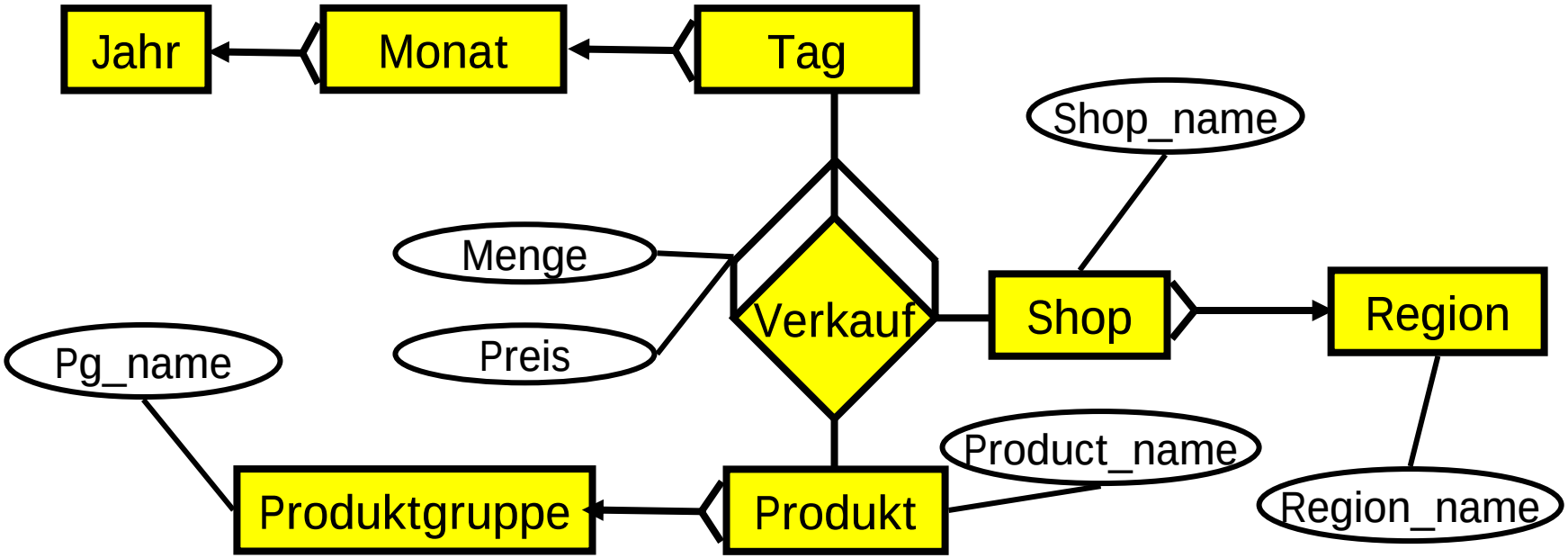
Inhalt dieser Vorlesung

- Speicherung multidimensionale Daten
  - Relationales OLAP (ROLAP) – Verwendung einer relationalen Datenbank
  - Multidimensionales OLAP (MOLAP) – Verwendung spezieller, multidimensionaler Speicherstrukturen

1. ROLAP: MDDM auf RDBMS

- MDDM und relationales Modell nicht vergleichbar: Konzeptionelle versus logische Ebene
- Vollkommen falsch
  - „Relationales Modell ist zweidimensional, MDDM ist beliebig-dimensional“
- Richtig
  - „Relationales Modell kann beliebig-dimensionale Daten repräsentieren (in Tabellenform), benötigt dazu aber einen Transformationsschritt“
- Verschiedene Abbildungen existieren
  - Speicherverbrauch, Redundanz, Performance von INSERT/SELECT/UPDATE, ...

Beispielmodell



Annahmen für Kostenbetrachtungen

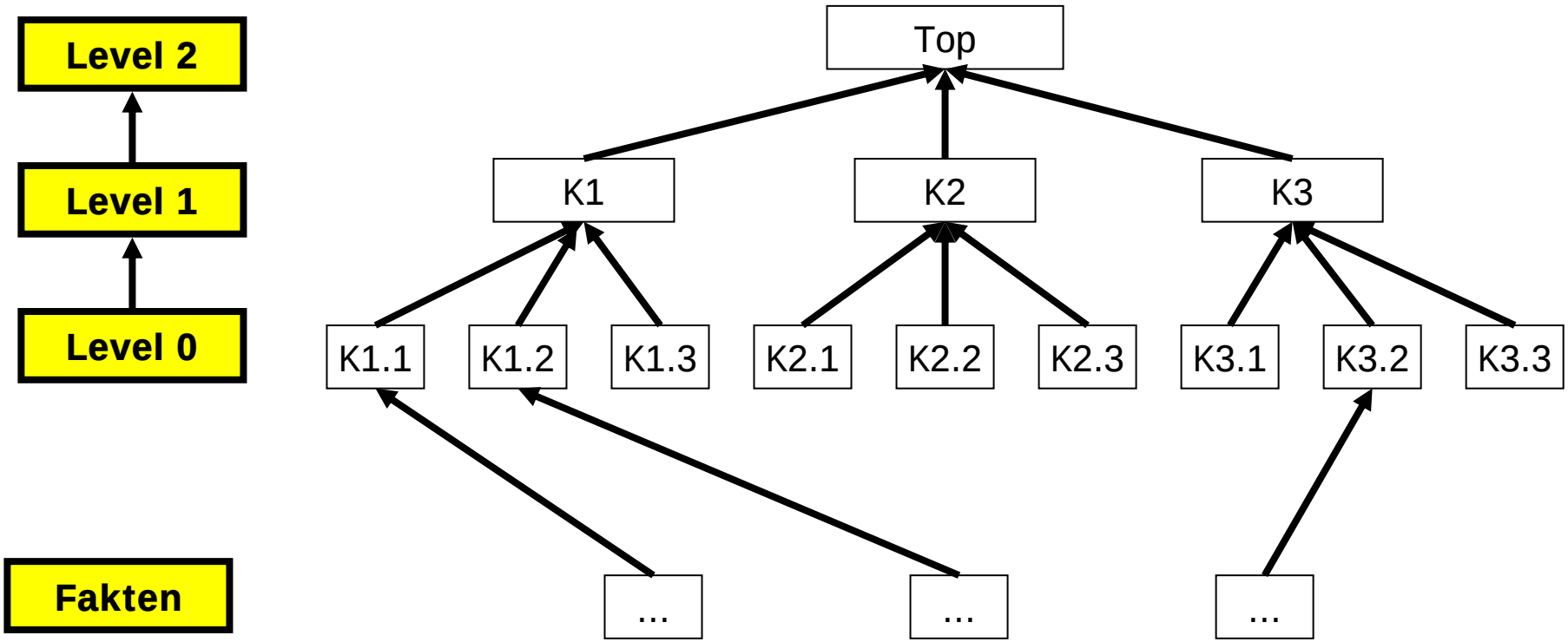
- d Dimensionen, je k Klassifikationsstufen plus Top
- Jeder Klassifikationsknoten hat 3 Kinder
  - Level k: $1=3^0$ Knoten (Top)
  - Level k-1: $3=3^1$ Knoten
  - Level k-2: $9=3^2$ Knoten
  - ...
  - Level 0: Höchste Granularität, 3^k Knoten

➢ n= $\sum_{i=0...k} 3^i$ Knoten pro Dimension

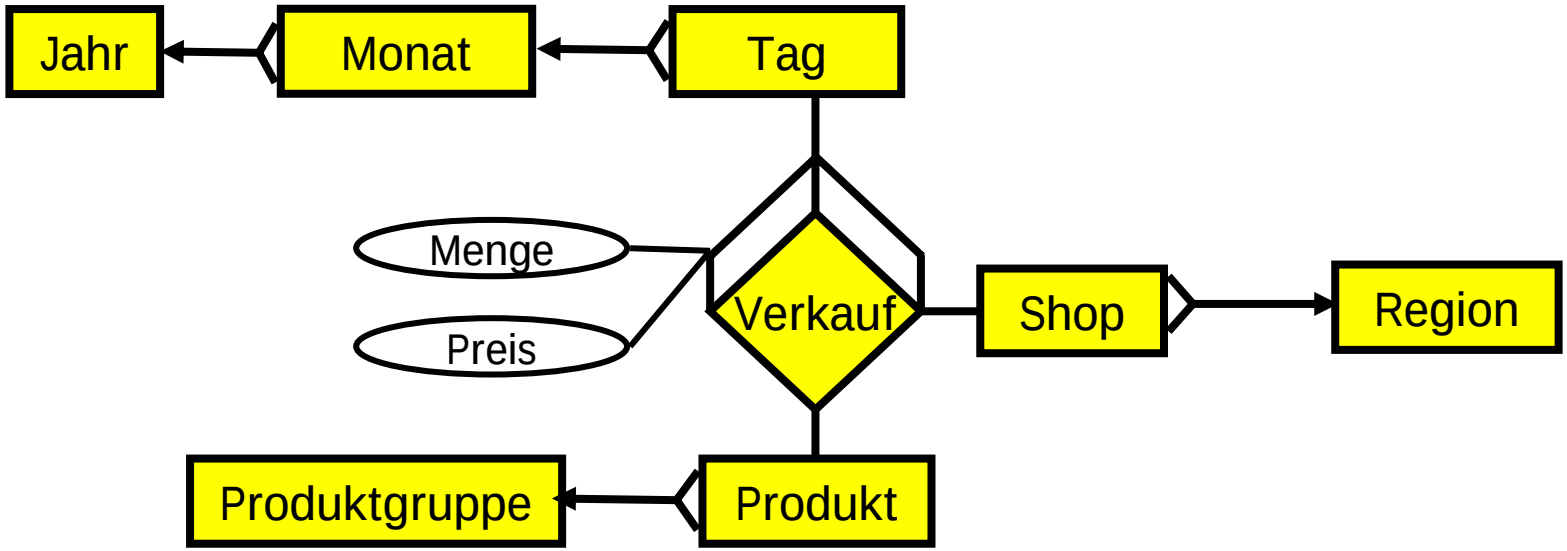
- m Fakten, gleichverteilt in allen Dimensionen
- Attribut: b Bytes
- Knoten haben nur ID
- Es gibt f Measures

Volle Klassifikationshierarchie

Zu jedem Knoten gibt es gleich viele Kinder/Fakten

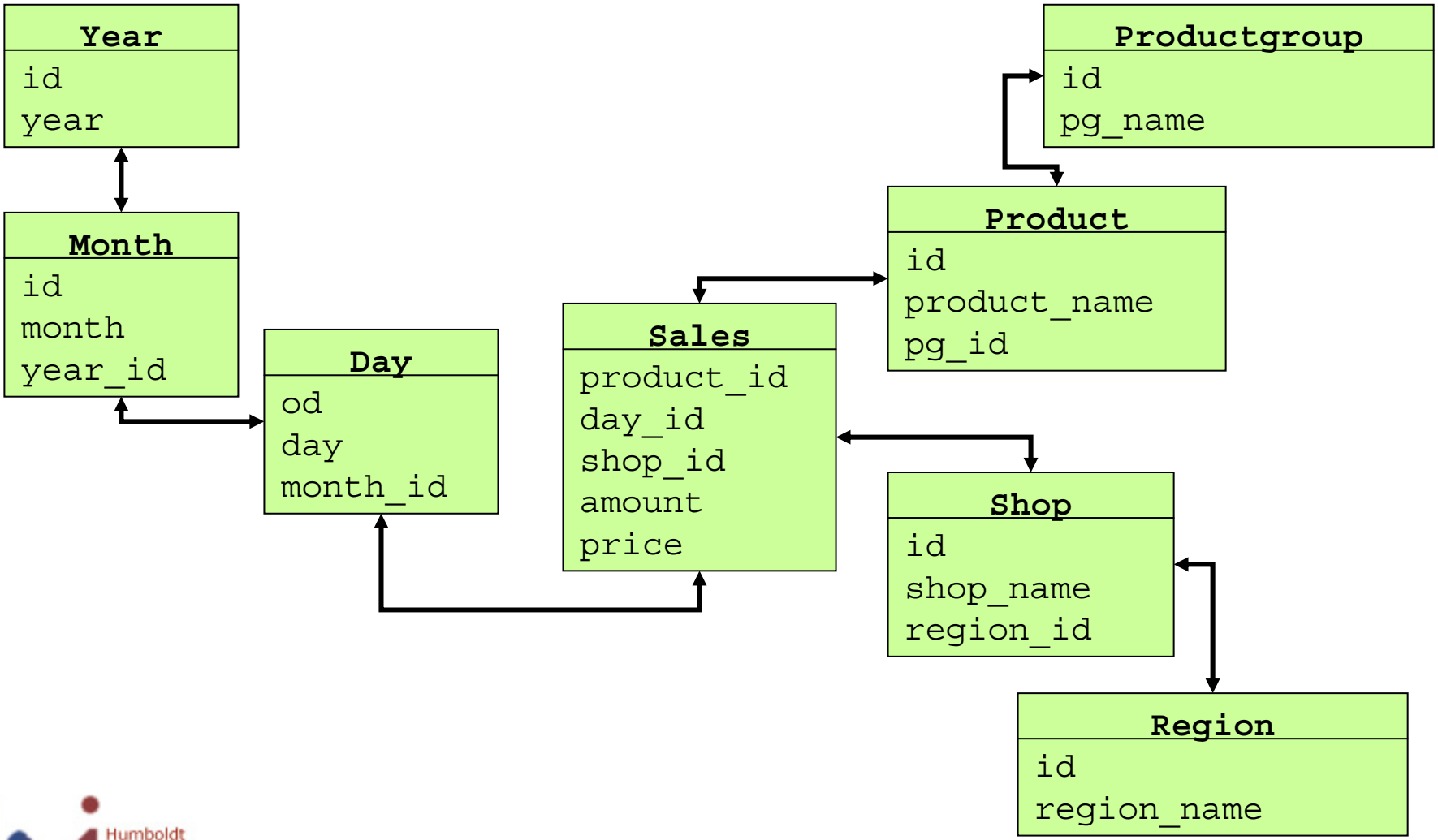


Beispielquery



Alle Verkäufe der Produktgruppe „Bier“ nach Shop und Jahr

Variante 1 - Snowflake



Snowflake Schema

- Faktentabelle
  - Measures plus Foreign Key der kleinsten Klassifikationsstufe jeder Dimension
- Dimensionstabellen
  - Eine Tabelle pro Klassifikationsstufe
  - Attribute: Knotenattribute und ID
  - Ein Tupel pro Klassifikationsknoten
- Eigenschaften
  - Voll normalisiert
  - Speicherplatz:

Ein Tupel pro Klassifikationsknoten jeder Dimension

$$\left(\left(d + f\right) * m + d * n\right) * b$$

Fremdschlüssel je Dimension

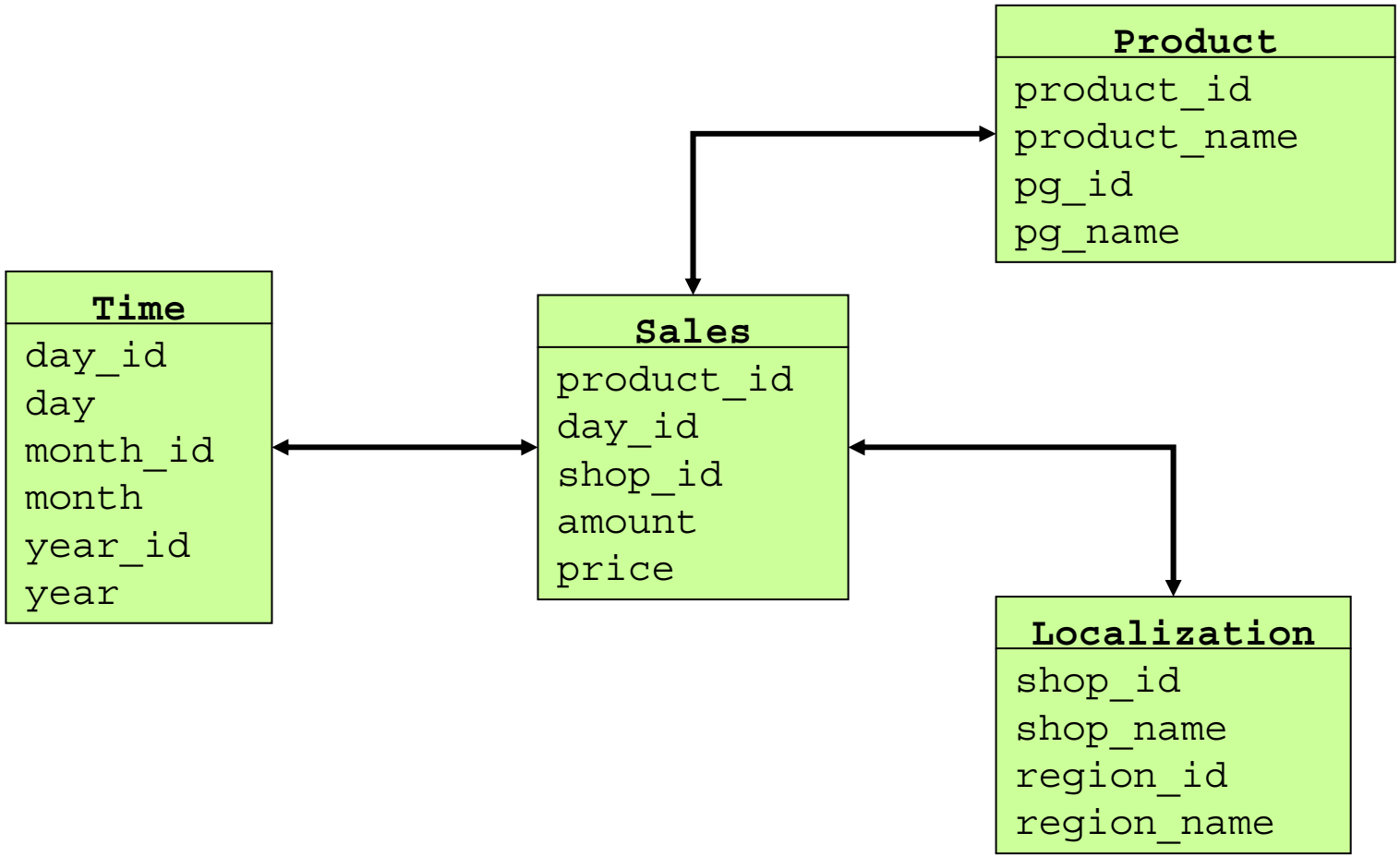
Measures

Snowflake - Query

```
SELECT X.shop_name, y.year, sum(amount*price)
FROM sales S, ... PG, P, D, M, Y, shop X
WHERE  PG.name=„Bier“ AND
       PG.id = P.pg_id AND
       P.id = S.product_id AND
       D.id = S.day_id AND
       M.id = D.month_id AND
       Y.id = M.year_id AND
       X.id = S.shop_id
group by X.shop_name, Y.year
```

- 6 Joins
- 4 Joins, wenn shop_name / year nicht notwendig
- Anzahl Joins steigt in jeder Dimension linear mit Länge der Aggregationspfade in der Anfrage

Variante 2: Star Schema



Star Schema

- Faktentabelle
  - Measures plus Foreign Key der kleinsten Klassifikationsstufe jeder Dimension
- Dimensionstabellen
  - Eine Tabelle pro Dimension
  - Attribute: ID und Knotenattribute aller Klassifikationsstufen
  - Ein Tupel pro Klassifikationsknoten mit Level 0
- Eigenschaften
  - Denormalisierte Dimensionen
  - Speicherplatz

Ein Tupel pro Klassifikationsknoten Level 0

$$\left(\left(d+f\right)*m+d*3^{k}*k\right)*b$$

Ein Attribut pro Klassifikationsstufe

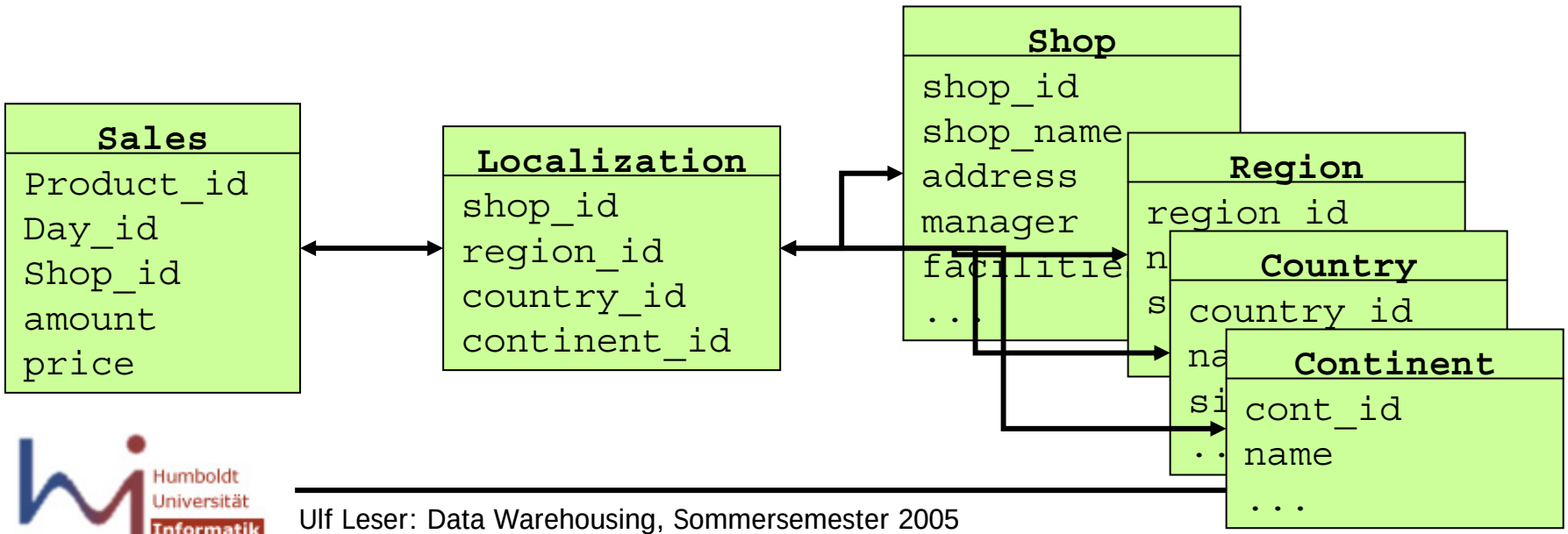
Star - Query

```
SELECT L.shop_name, T.year, sum(amount*price)
FROM sales S, product P, time T, localization L
WHERE  P.pg_name=„Bier“ AND
       P.product_id = S.product_id AND
       T.day_id = S.day_id AND
       L.shop_id = S.shop_id
group by L.shop_name, T.year
```

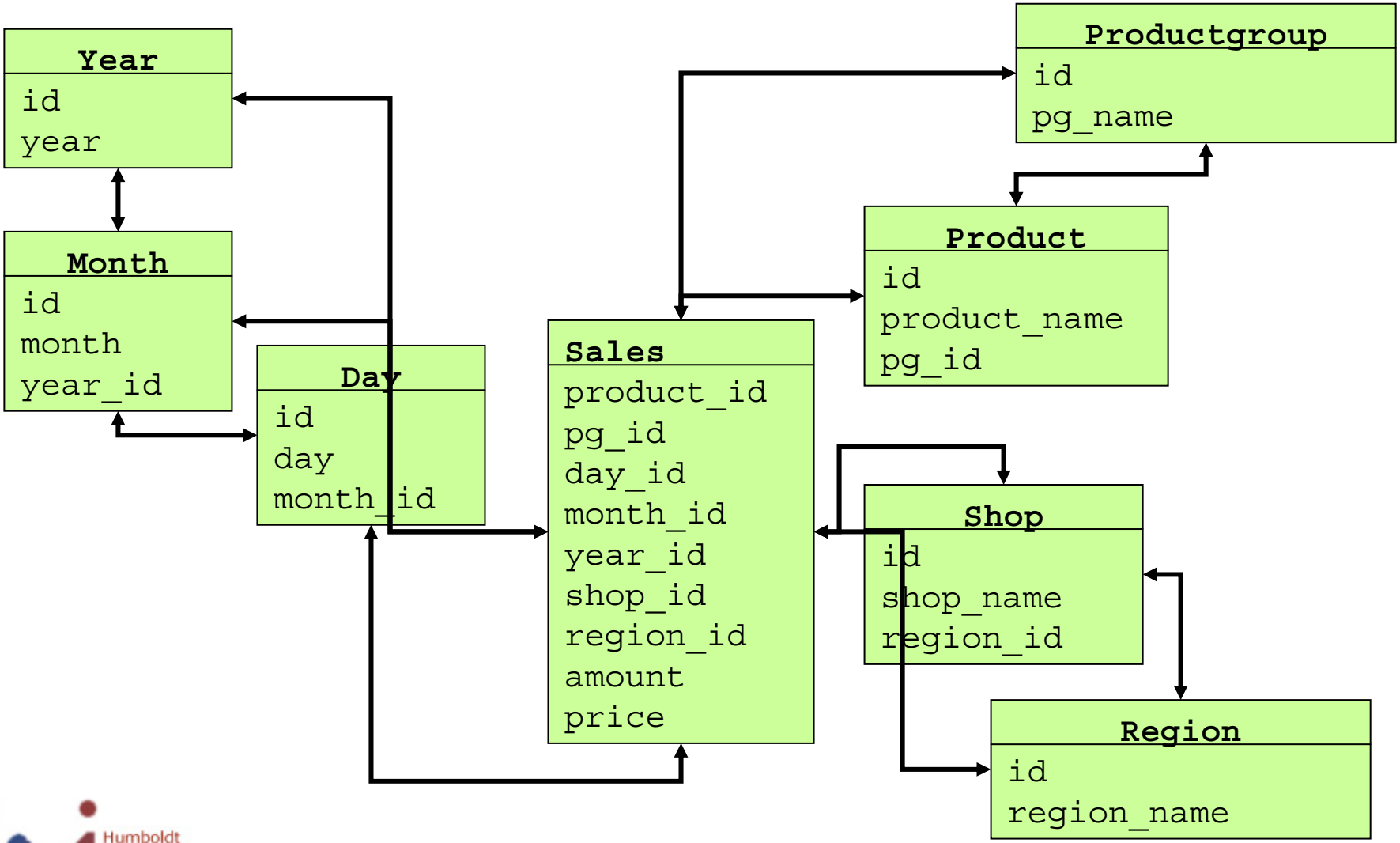
- 3 Joins
- Keine Reduktion wenn Jahrname nicht notwendig
- 2 Joins, wenn Shopname nicht notwendig
- Anzahl Joins
  - Unabhängig von Länge der Aggregationspfade in der Anfrage
  - Steigt linear mit der Anzahl Dimensionen in der Anfrage

Variante: Knotenattribute im Star Schema

- Speicherung in
  - Eigenen Tabellen pro Klassifikationsstufe: Dimensionstabelle repräsentiert Hierarchie und enthält nur Keys
    - Normalisierter Entwurf, speicherplatzeffizient, zusätzliche Joins notwendig
  - Dimensionstabelle
    - Redundante Daten, keine zusätzlichen Joins



Variante 3: Fullfact



Fullfact

- Faktentabelle
  - Measures plus Foreign Key jeder Klassifikationsstufe
- Dimensionstabellen
  - Eine Tabelle pro Klassifikationsstufe
  - Attribute: Knotenattribute und ID
  - Ein Tuple pro Klassifikationsknoten
- Eigenschaften
  - Normalisierte Dimensionen, denormalisierte Fakten
  - Speicherverbrauch

$$\left(\left(d * k + f\right) * m + d * n\right) * b$$

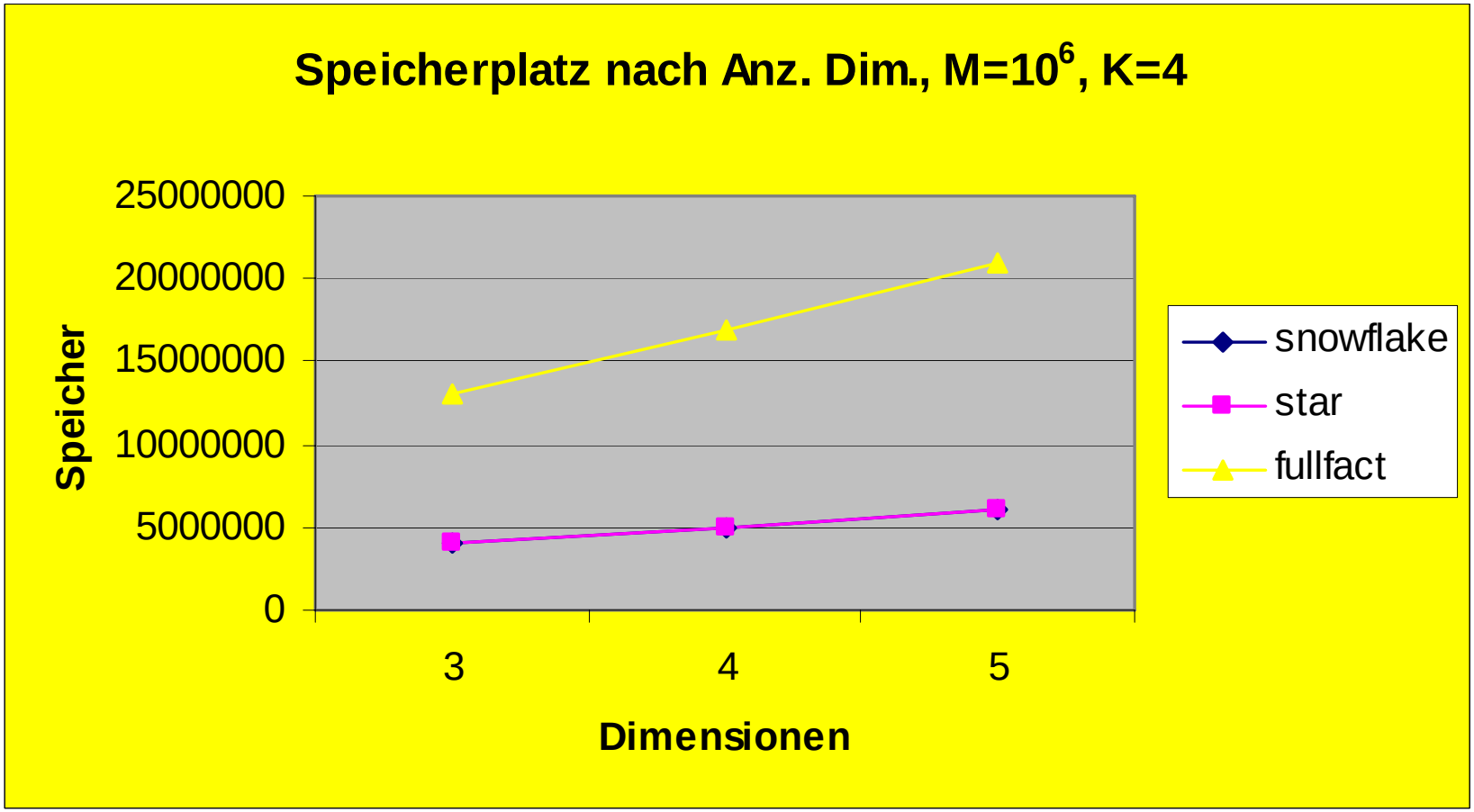
Ein FK pro Klassifikationsstufe

Fullfact Query

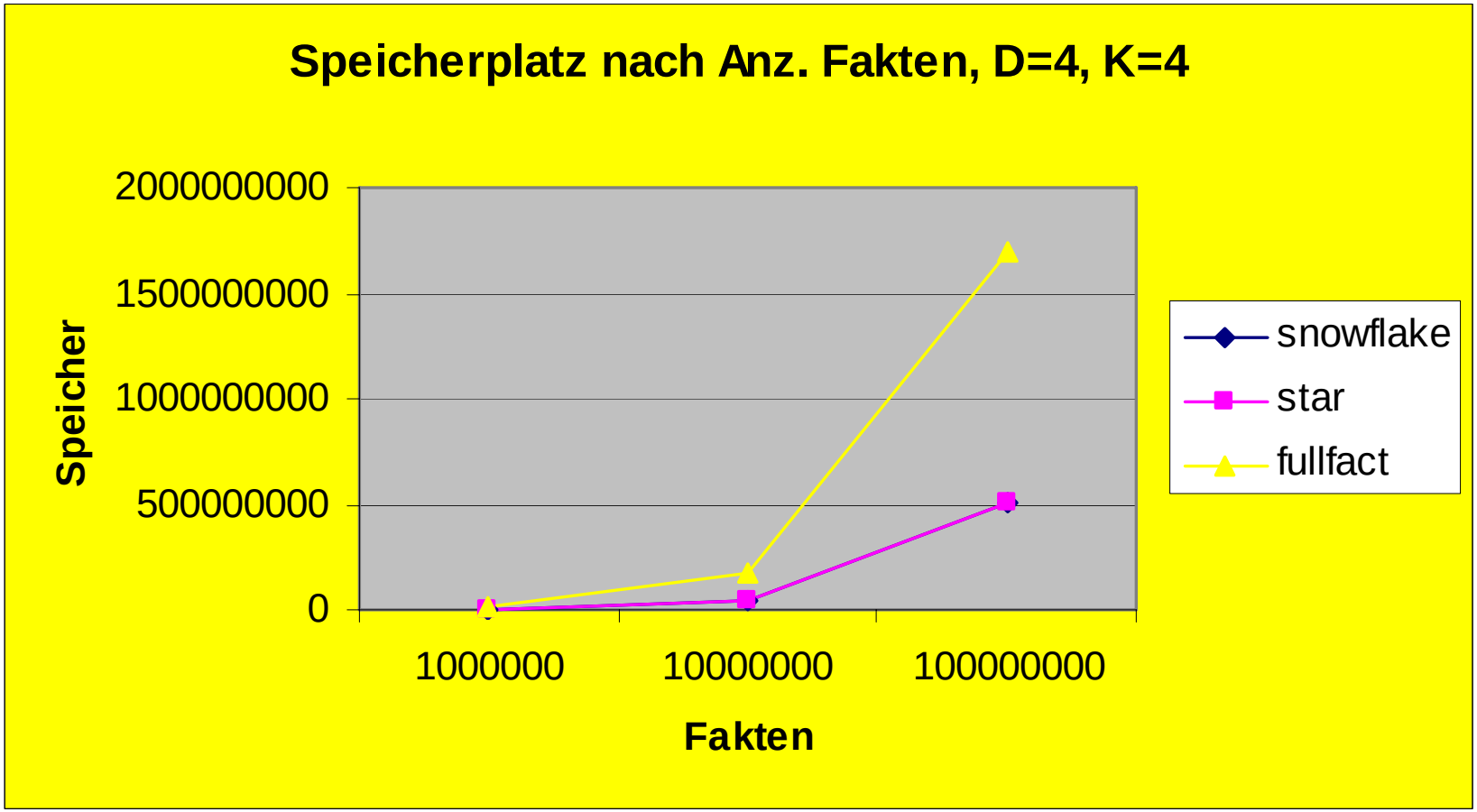
```
SELECT X.shop_name, T.year, sum(amount*price)
FROM sales S, productgroup PG, year Y, shop X
WHERE  PG.pg_name=„Bier“ AND
       PG.id = S.pg_id AND
       Y.id = S.year_id AND
       X.id = S.shop_id
group by X.shop_name, Y.year
```

- 3 Joins
- 1 Join, wenn Shopname / Jahrname nicht notwendig
- Wichtiger Effekt (Vergleich zu Star Schema)
  - Zu joinende Dimensionstabellen sind kleiner in allen Klassifikationsstufen mit Level != 0
- Anzahl Joins
  - Unabhängig von Länge der Aggregationspfade in der Anfrage
  - Steigt linear mit der Anzahl Klassifikationsstufen in der Anfrage

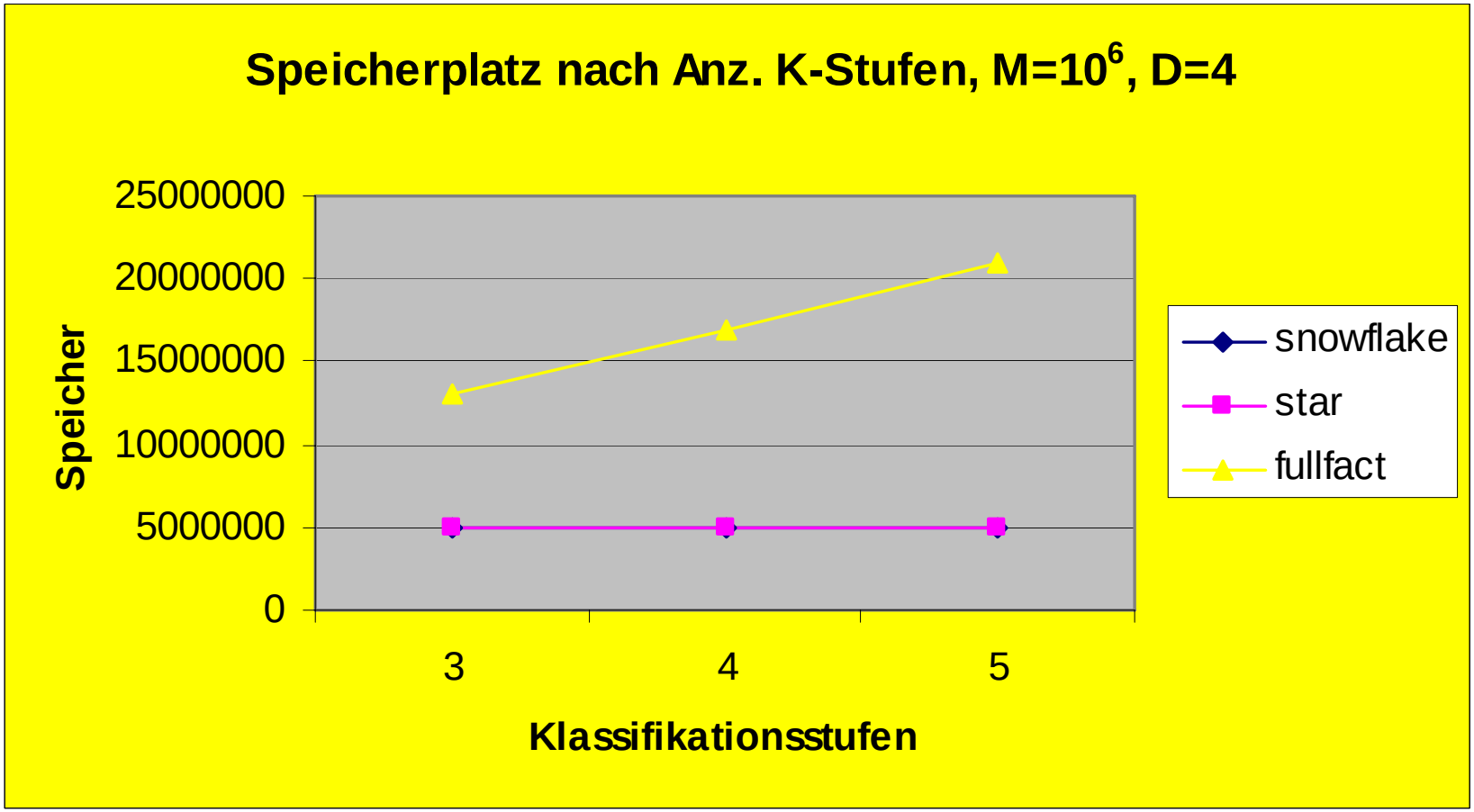
Speicherverbrauch 1



Speicherverbrauch 2



Speicherverbrauch 3



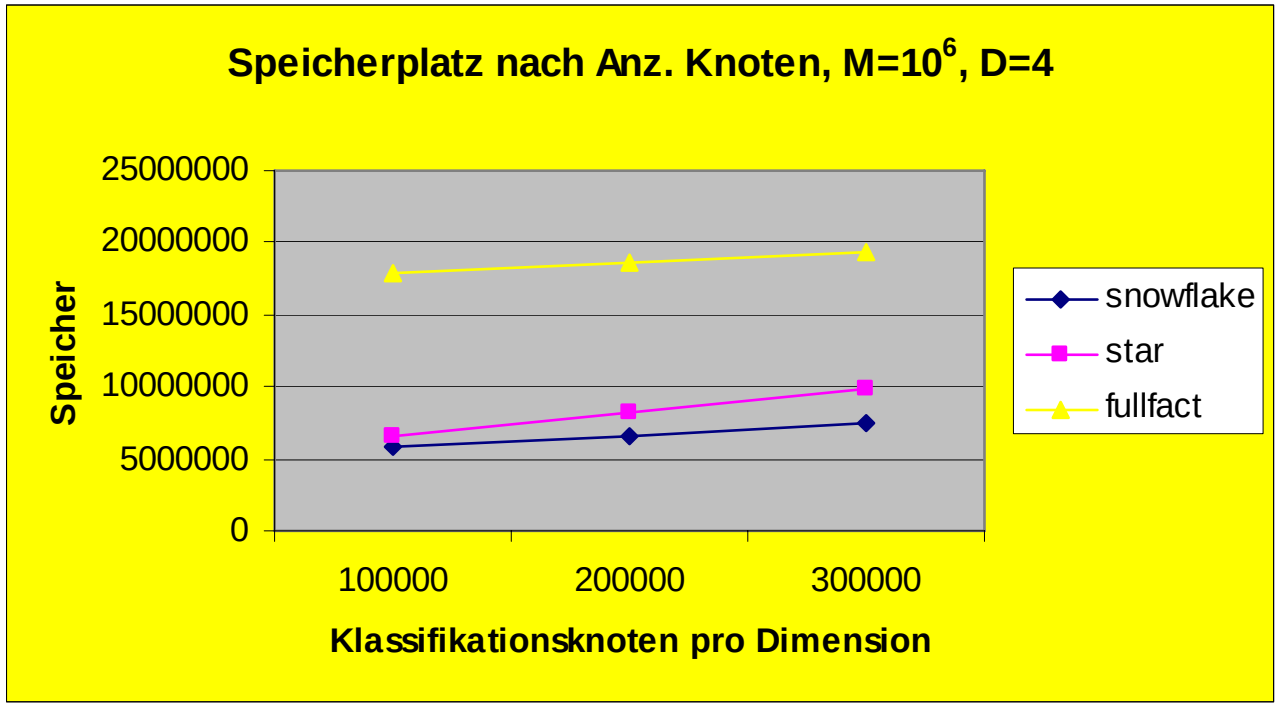
Fazit – Speicher und Query

- Speicherverbrauch Snowflake / Star praktisch identisch
  - Wenn Bedarf für Dimensionen vernachlässigbar
- Fullfact mit deutlich höherem Speicherverbrauch
  - Dafür minimale Anzahl Joins (im Idealfall 0)
- Laufzeitverhalten hängt von mehr Faktoren als dem Schema ab
  - Bereichs- oder Punktanfrage
  - Indexierung
  - Selektivität der Bedingungen
  - Gruppierung und Aggregation
  - ...
- ... aber Joins sind tendenziell teuer

Entartete DWH

Anzahl Klassifikationsknoten nähert sich Anzahl Fakten

- Bsp.: Experimentelle Daten (Knoten = Proben, Werte = Ergebnisse von Tests)

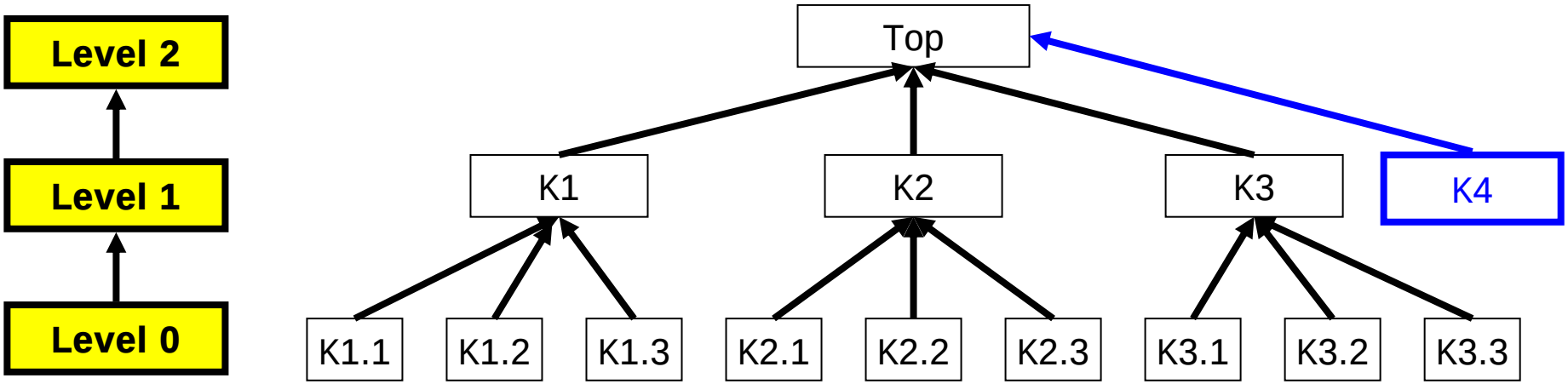


Änderungen in Klassifikationshierarchie

- Auch Dimensionen verändern sich
- Mögliche Operationen
  - Gelöschte / neue Klassifikationsknoten
  - Gelöschte / neue Klassifikationsstufen
  - Neue Dimensionen
- Aufwand für Operationen abhängig von Modellierung
- Im folgenden: 4 Operationen

Änderung: InsN

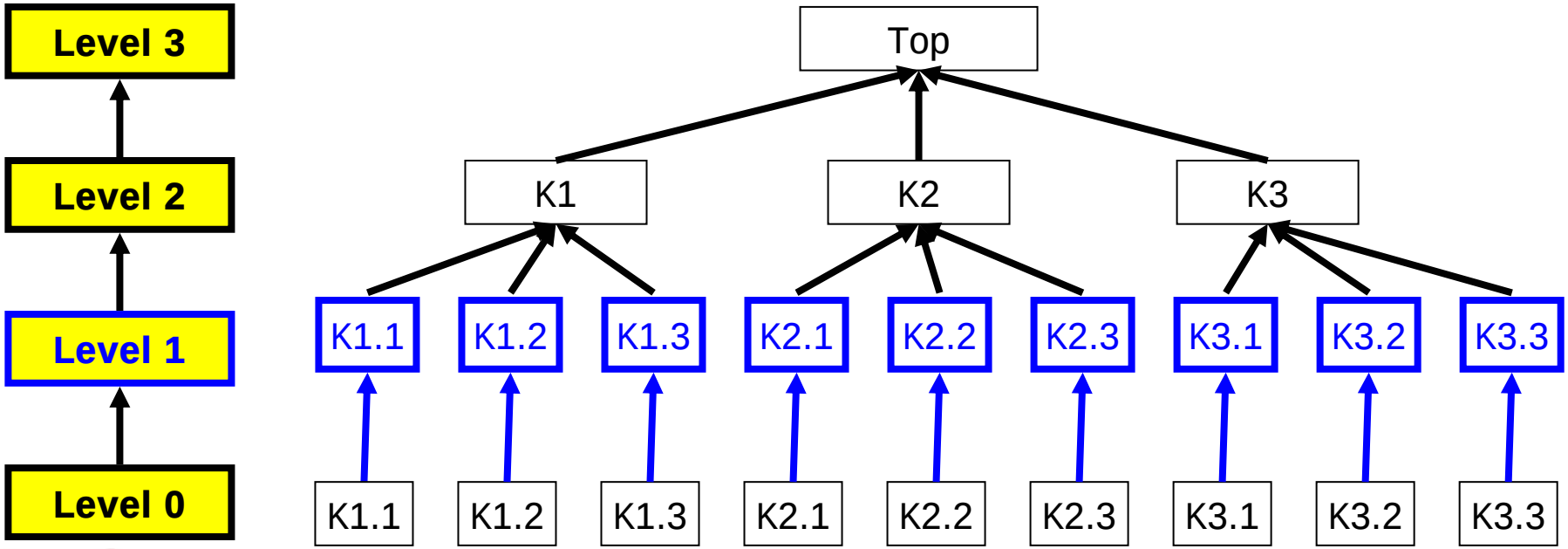
Einfügen eines neuen Klassifikationsknoten in einen Pfad an Position i≠0 ohne Nachkommen
(i=1, k=2)



Änderung: InsS

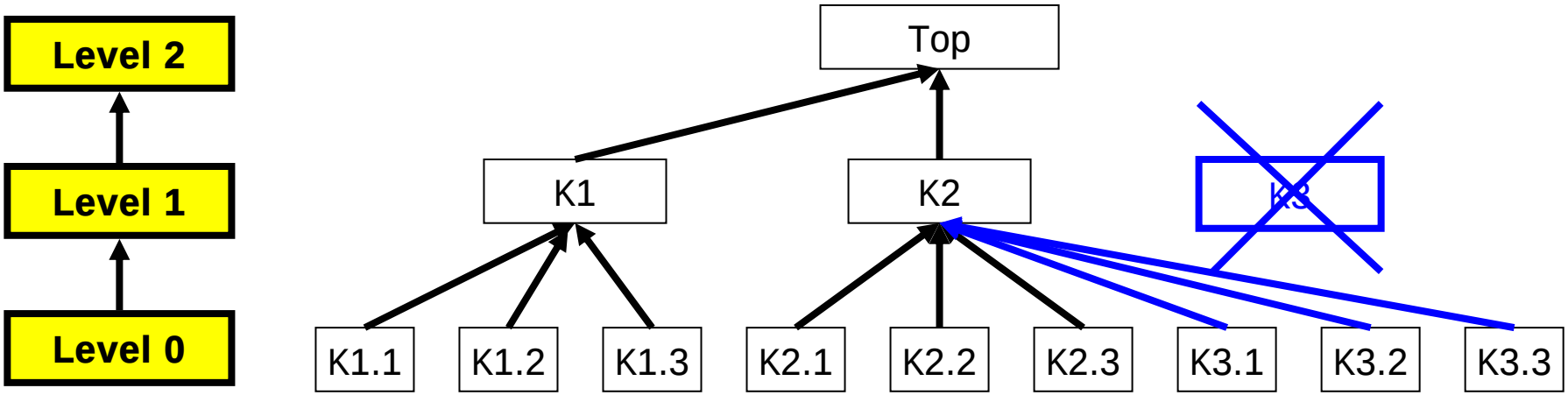
Einfügen einer neuen Klassifikationsstufe in einen Pfad zwischen Position i und i-1 (i>1) mit $3^{(k+1)-i}$ Klassifikationsknoten mit Umhängung Knoten Level i-1

(i=2, k=2 / 3)



Änderung: DelN

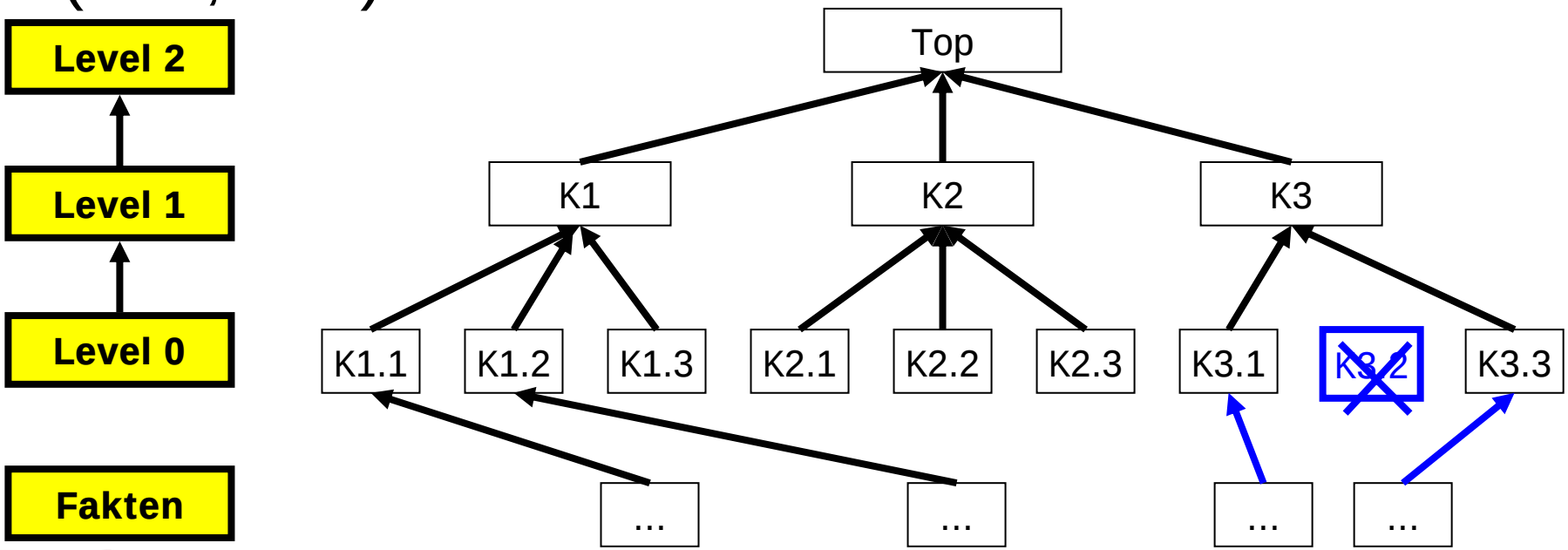
Löschen eines Klassifikationsknoten in einem Pfad an Position i≠0; Umhängen der Nachkommen
(i=1, k=2)



Änderung: $DelN_0$

Löschen eines Klassifikationsknoten in einem Pfad an Position i=0; Umhängen der Fakten auf andere Knoten mit Level 0

(i= 0, k=2)



Änderungskosten

Semantisch fragwürdig: NULL in Knoten mit Level j<i

	Snowflake	Star	Fullfact
InsN	1 Insert	(1 Insert)	1 Insert
InsS	1 neue Tabelle, $3^{(k+1)-i}$ Updates (Umhängung)	1 neues Attribut, 3^k Updates (Wert des Attrib.)	1 neue Tabelle, 1 neues Attribut, m Updates (in Faktentabelle)
DelN	1 Delete, 3 Update	0 Delete, 3 Update	1 Delete, $m/3^{k-i}$ Updates (Level i: 3^{k-i} Knoten)
$DelN_0$	1 Delete, $m/3^k$ Update	1 Delete, $m/3^k$ Update	1 Delete, $m/3^k$ Updates

Fazit

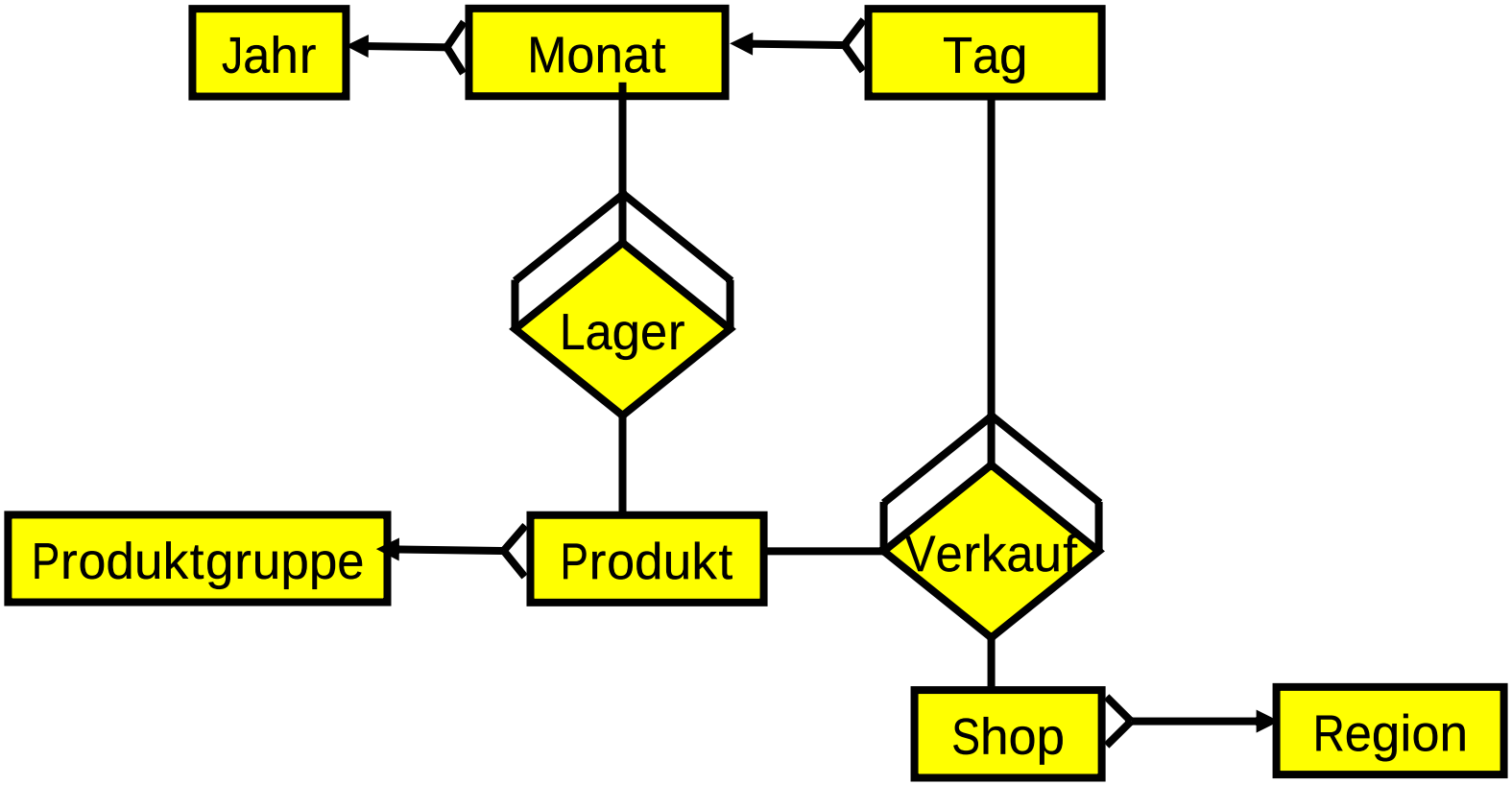
- Insert/Delete in Dimensionen
  - Teuer bei Fullfakt
  - Moderat bei Star / Snowflake
- Insert/Delete von Blatt – Klassifikationsknoten
  - Werden durch FK aus Faktentabelle adressiert
  - Immer teuer (je nach Größe und Werteverteilung innerhalb der Dimension)
- Star Schema „verlangt“ balancierte Hierarchien
  - Sonst NULL Werte in Knoten-IDs und -attributen
  - Probleme beim Aggregieren/Gruppieren über NULLs

Weitere Varianten

- Galaxy schema
- Constellation
- Rekursive Modellierung der Klassifikations-hierarchie

Galaxy Schema

Mehrere Faktentabellen

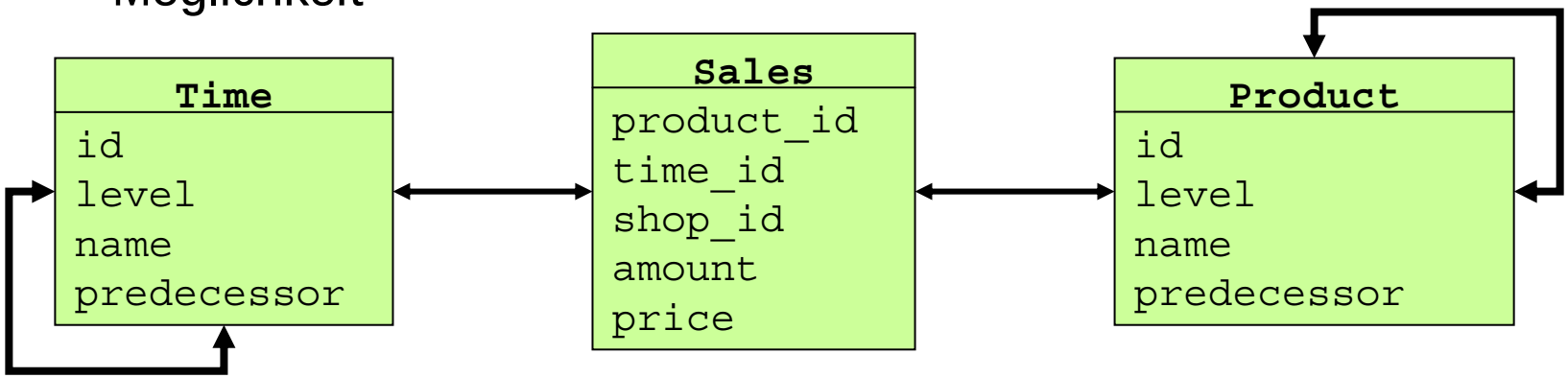


Konstellation Schema

- Modellierung aggregierter Daten
- Zwei Wege
  - Einbeziehung als spezielle Fakten in Faktentabelle
  - Modellierung in eigenen Tabellen
- Konstellation, „Fact Constellation“ Schema = Star Schema mit eigenen Summentabellen
- Siehe „Materialized Views“

Rekursive Modellierung

- Unbeschränkt lange, unbalancierter Klassenhierarchien
  - Beispiel: Stücklisten (Teil von ...)
  - Weder in Star noch Snowflake Schema möglich
  - Erweiterungen immer DDL Operationen
- Möglichkeit



- Probleme
  - Rekursive Anfragen oder viele Self-Joins (wie viele ?)
  - Keine individuellen Attribute pro Stufe
  - Eingeschränkte referenzielle Integrität
    - Sales.time_ID könnte auf Knoten in time mit Level≠0 zeigen

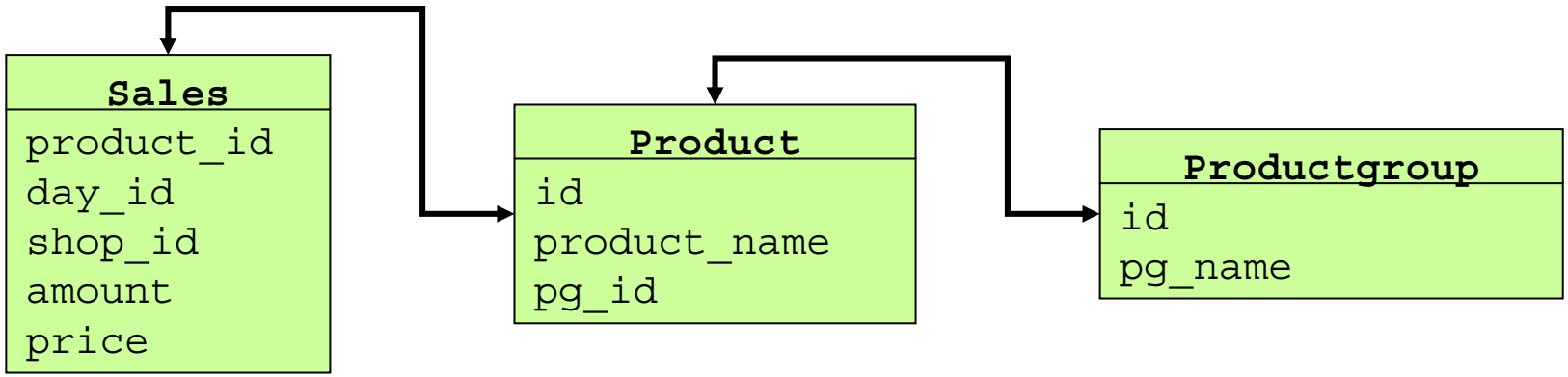
Fazit

- Sehr viele Joins, kein Platzproblem, große, statische Klassifikationshierarchien mit wenigen Stufen
  - ➢ Fullfact Schema
- Viele Änderungen in Klassifikation, viele Klassifikationsstufen
  - ➢ Kein Fullfact Schema
- Viele Klassifikationsstufen, Aggregate über alle Stufen
  - ➢ Eher Star als Snowflake
- Unbeschränkt lange Klassifikationshierarchien
  - ➢ Rekursive Modellierung
- Standard: Star (Snowflake) Schema
  - – Star Schema – Star Join

Einschub: MDDM Konstrukte in Oracle

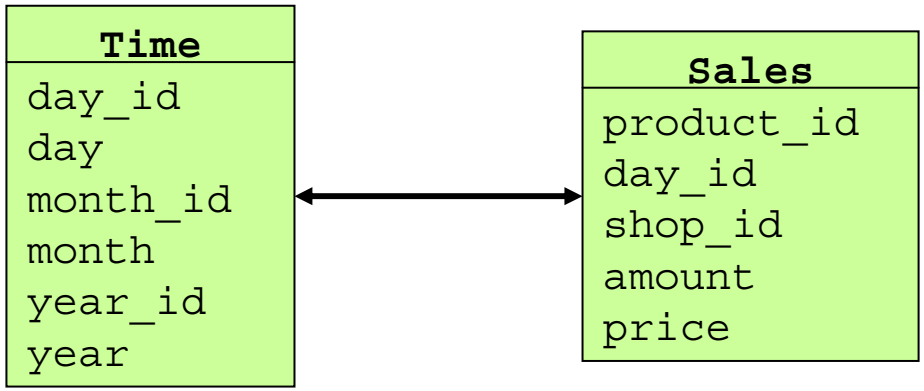
- Oracle unterstützt
  - Dimensionen
  - Klassifikationsstufen
  - Klassifikationspfade
- Definition mit SQL Befehlen
- Setzen auf existierende relationale Schema auf
  - Star Schema oder
  - Snowflake Schema

Beispiel: Snowflake Schema




```
CREATE DIMENSION s_pg
  LEVEL product IS (product.p_id)
  LEVEL productgroup IS (productgroup.pg_id)
HIERARCHY pg_rollup (
  product CHILD OF productgroup
  JOIN KEY (product.pg_id)
    REFERENCES productgroup
);
```

Beispiel: STAR Schema




```
CREATE DIMENSION s_time
  LEVEL day IS (time.day_id)
  LEVEL month IS (time.month_id)
  LEVEL year IS (time.year_id)
HIERARCHY pg_rollup (
  day CHILD OF
  month CHILD OF
  year)
ATTRIBUTE day DETERMINES (day)
ATTRIBUTE month DETERMINES (month)
ATTRIBUTE year DETERMINES (year)
```

Multiple Hierarchien in einer Dimension

Dimension

Klassifikationsstufen

Klassifikationspfad

Klassifikationspfad

```
CREATE DIMENSION times_dim
  LEVEL day IS TIMES.TIME_ID
  LEVEL month IS TIMES.CALENDAR_MONTH_DESC
  LEVEL quarter IS TIMES.CALENDAR_QUARTER_DESC
  LEVEL year IS TIMES.CALENDAR_YEAR
  LEVEL fis_week IS TIMES.WEEK_ENDING_DAY
  LEVEL fis_month IS TIMES.FISCAL_MONTH_DESC
  LEVEL fis_quarter IS TIMES.FISCAL_QUARTER_DESC
  LEVEL fis_year IS TIMES.FISCAL_YEAR
HIERARCHY cal_rollup (
  day CHILD OF
  month CHILD OF
  quarter CHILD OF
  year )
HIERARCHY fis_rollup (
  day CHILD OF
  fis_week CHILD OF
  fis_month CHILD OF
  fis_quarter CHILD OF
  fis_year )
<attribute determination clauses>...
```

Eigenschaften

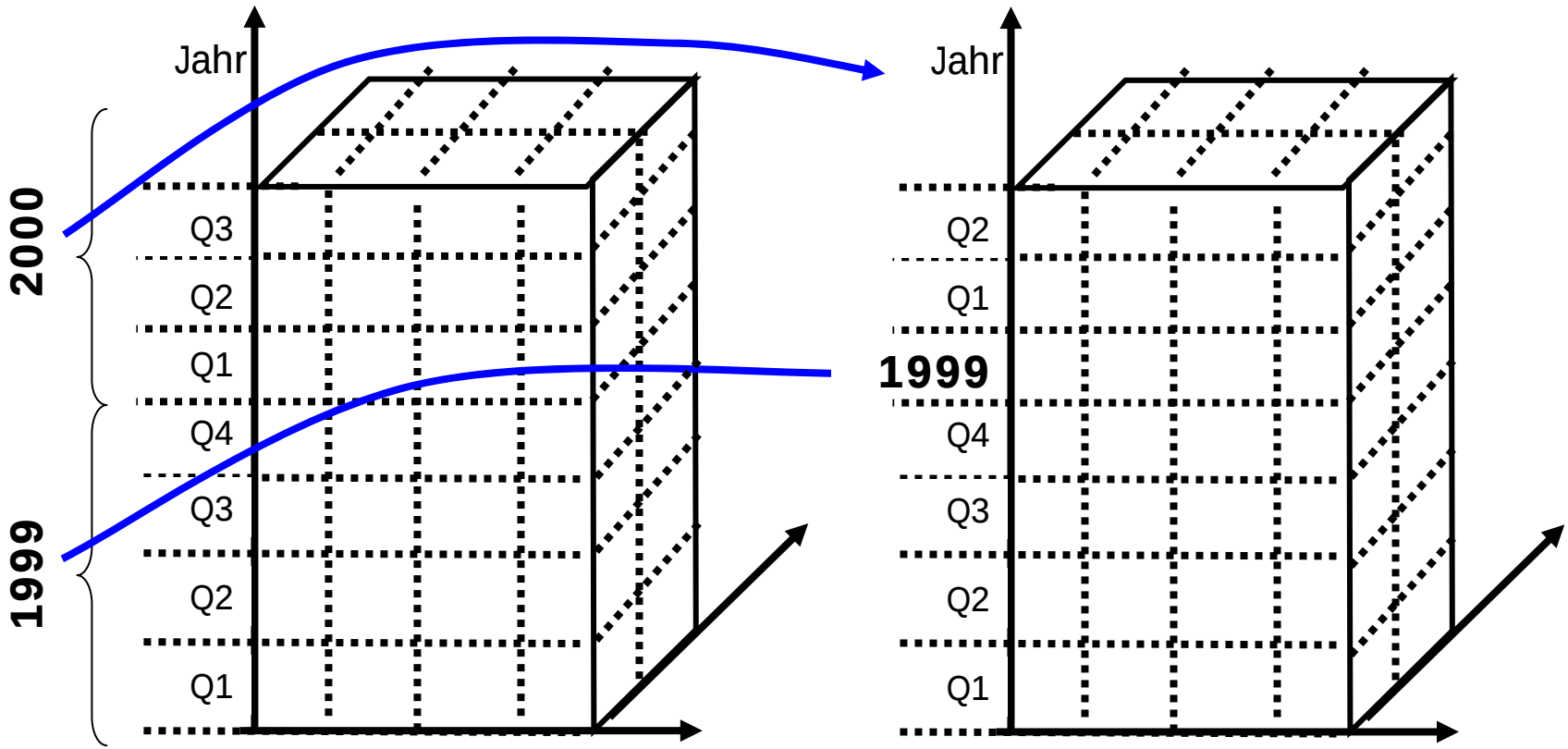
- Erwartet vollständige und überlappungsfreie Hierarchien
  - *„The child_key_columns must be non-null and the parent key must be unique and non-null. You need not define constraints to enforce these conditions, but queries may return incorrect results if these conditions are not true.“*
- Analyse und Anzeige über PL/SQL Packages
  - `DEMO_DIM, DBMS_OLAP`
- `Attribute ... Determines` Clause
  - Spezifiziert funktionale Abhängigkeiten innerhalb einer Tabelle
  - Wird benutzt, um ggf. Präaggregation über abhängige Werte benutzen zu können
- Benutzung
  - Für Query Rewrite (siehe „Materialisierte Sichten“)
  - Für Client-Software / Visualisierung / Navigation

2. MOLAP

- Grundidee
  - Speicherung multidimensionaler Daten ...
  - ... in multidimensionalen Arrays
- Im folgenden nur
  - 1 Würfel, 1 Fakt
  - Keine Betrachtung von Knotenattributen
  - Klassifikationsknoten haben nur diskrete Werte
    - Bei numerischen Knoten: Diskretisierung

Klassifikationshierarchien

Modellierung als eingebettete Knoten der untersten Stufe



Array Adressierung

- Sei $d_i = |D_i|$, d Dimensionen,
- b: Speicherplatz pro Attribut (Key oder Wert)
- Linearisierte Darstellung
  - <1,1,1>, <1,1,2>, ... , <1,1,d_1>, <1,2,1>, ...
- Offset der Zelle <a_1,...a_n>

$$b * \sum_{i=1}^{d} \left((a_i - 1) * \prod_{j=1}^{i-1} d_j \right)$$

Arrayspeicherung - Probleme

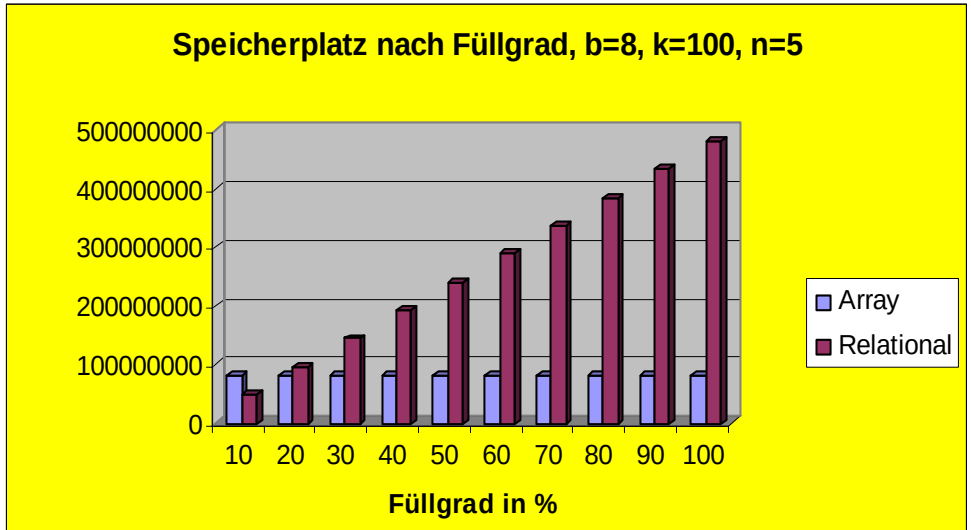
- Naive MOLAP: Unterschiedliche Performanz je nach Dimensionsreihenfolge in Array / Anfrage
  - Daten liegen nach D_n geordnet auf Platte
  - Zugriff auf <2,2,X> - sequentieller Read
  - Zugriff auf <X,1,3> oder <3,X,3> - Zugriff auf viele Blöcke
  - RDBMS mit ähnlichen Problemen
  - Multidimensionale Datenstrukturen notwendig
- Speicherung verschiedener Werte pro Zelle?
  - Mehrere Verkäufe von „CocaCola“ an einem Tag in einem Shop
  - Auflösung durch Listen o.ä.

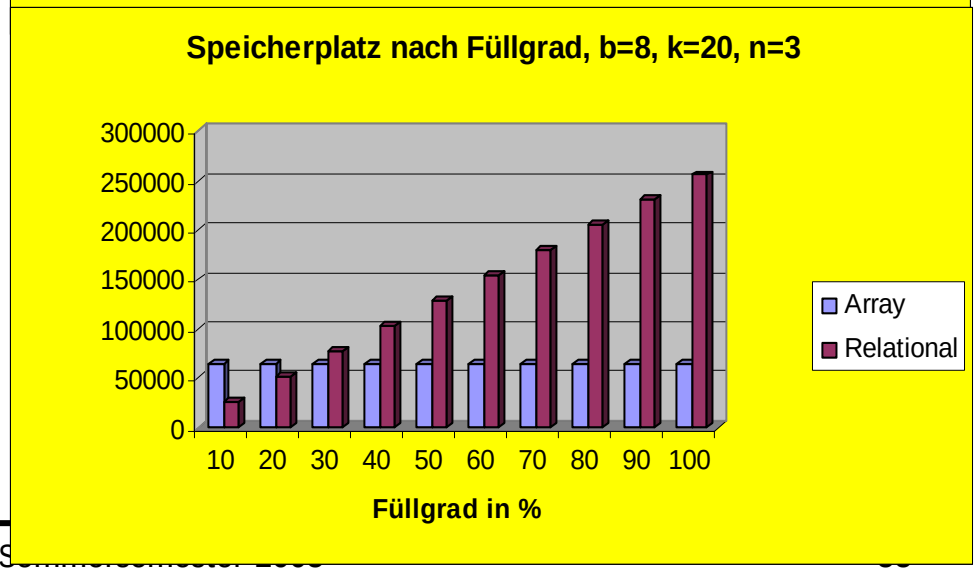
Speicherverbrauch

	Array	**Relational (Star Schema)**
Speicherung Koordinaten	Implizit (Linearisierung)	Explizit (redundant)
Leere Zellen	Belegen Platz	Belegen keinen Platz
Neue Klas. Knoten	Komplette Reorganisation Starkes Wachstum im Speicherverbrauch	Neue Zeile in Dimensionstabelle Kaum Wachstum im Speicherverbrauch
Speicher-verbrauch	$b * \prod_{i=1}^{d} d_i$	$b * m * d$ (plus Measures)

Vergleich Speicherplatz

- Faktoren
  - Füllgrad
  - k: Anz. Knoten
  - n: Anz. Dimensionen
- Schon bei geringen Füllgraden ist Array platzeffizienter
- Aber: Tatsächlicher Füllgrad ist bei vielen Dimensionen bzw. Klassifikations-knoten oft erstaunlich gering
  - Bei weitem nicht alle Kombinationen vorhanden





Sparse Matrices

- Viele Klassifikationsstufen und -knoten
  - dünn besetzte Matrizen
    - geringer Füllungsgrad
    - ineffiziente Speicherplatznutzung
    - überflüssiges I/O (NULLs müssen gelesen werden!)
- Komprimierung
  - Run-Length Encoding (RLE): Schlecht indizierbar
  - 2-Ebenen Speicherung
    - Ebene 1: Array mit dünn besetzten Dimensionen
    - Ebene 2: Arrays mit dicht besetzten Dimensionen
    - Gewinn: Viele Ebene 2 Blöcke leer – weglassen
  - Parallelität zu multidimensionalen Indexstrukturen

Fazit MOLAP

- Verschiedene Verfahren möglich
- Kein Standard vorhanden
- Verwendung in proprietären Produkten
- „Volksmeinung“
  - Schnell für kleine – mittlere Datenmengen
  - Schlechte Skalierbarkeit (> 50GB)
- Typischer Einsatz
  - (Regelmäßige) Snapshots des (ROLAP) Cubes auf Client in MOLAP Datenbank laden
  - Vorberechnung der notwendigen Aggregate
  - Sehr schnelles, intuitives User-Interface
  - Read-Only