Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf

Folien teilweise übernommen von Matthias Gimbel

Datenbankentwurf: Schritte

- Entwurfsschritte
  - Anforderungsanalyse
  - konzeptioneller DB-Entwurf
    - ➔ konzeptionelles Schema (unabhängig vom Zieldatendatenmodell)
  - logischer DB-Entwurf
    - ➔ logisches Schema (in konkretem Datenmodell)
  - physischer DB-Entwurf
    - ➔ internes/physisches Schema (in konkretem Datenbanksystem)
- iterativer Entwurfsprozess
- konzeptueller und logischer DB-Entwurf nicht immer leicht zu trennen

Datenbankentwurf: Vergleich

<table>
<tr><th></th><th>Relationaler DB-Entwurf</th><th colspan="2">Multidimensionaler DB-Entwurf</th></tr>
<tr><td>konzeptuelles Schema</td><td>E/R-Modellierung, UML</td><td colspan="2">ME/R, mUML, …</td></tr>
<tr><td rowspan="2">logisches Schema</td><td rowspan="2">Relationen mit Attributen</td><td colspan="2">Datenwürfel mit Summenattributen:
Fakten und Kennzahlen</td></tr>
<tr><td colspan="2">Dimensionshierarchien
mit Kategorienattributen</td></tr>
<tr><td>internes/physisches Schema</td><td>Speicherorganisation: Indexstrukturen, Partitionierung, …</td><td>Relationale Speicherorgani-sation (ROLAP)</td><td>Multidimensionale Speicherorgani-sation (MOLAP)</td></tr>
</table>

Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf
- Konzeptueller Datenbankentwurf -

Datenmodell für Data Warehouses

Woran ist ein Analyst interessiert?

- an **Kennzahlen**
  - Umsatz, Gewinn, Menge, Kilometer
- gruppiert nach **Dimensionen**
  - **Produkt**: Produktkategorie, Produktklasse, Produktgruppe, Produktabteilung
  - **Region**: Filiale, Gemeinde, Landkreis, Bundesland, Staat
  - **Zeit**: Tag, Woche, Monat, Quartal, Jahr
  - **Auftraggeber/Kunde**: Alter, Geschlecht, Beruf, Einkommen, …
  - …
  - ... und allen sinnvollen **Kombinationen**

Datenmodell für Data Warehouses

- Datenmodell muss zwei Arten von Informationen unterscheiden:
  - **Quantifizierende** Informationen (Kennzahlen)
  - **Qualifizierende** Informationen: (Dimensionen)

Quantifizierende Informationen

Quantifizierende Informationen sind oft **numerisch** und bestehen aus:

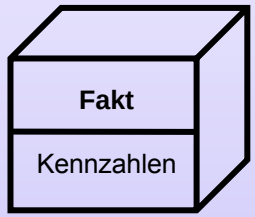
- **Fakten** („Basiskennzahlen“)
  - z. B.: Einnahmen aus einer Dienstleistung
- **Kennzahlen** („abgeleitete Kennzahlen“)
  - Berechnungsvorschrift über existierende Fakten
    - Zählung: `count()`
    - Summierung: `sum()`
    - Mittelwertbildung: `avg()`
    - Minimum: `min()`
    - Maximum: `max()`
    - Varianz
    - Standardabweichung
    - …

Konzeptueller DB-Entwurf

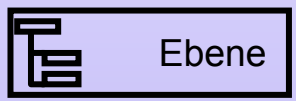
- Erweiterung bestehender Entwurfstechniken an Anforderungen durch Multidimensionalität
  - ME/R
    multidimensionales E/R
  - mUML
    multidimensionales UML

ME/R-Modellierung

- **Erweiterung** des E/R-Modells um **multidimensionale** Konstrukte:
  - Fakt



  - Dimensionsebene

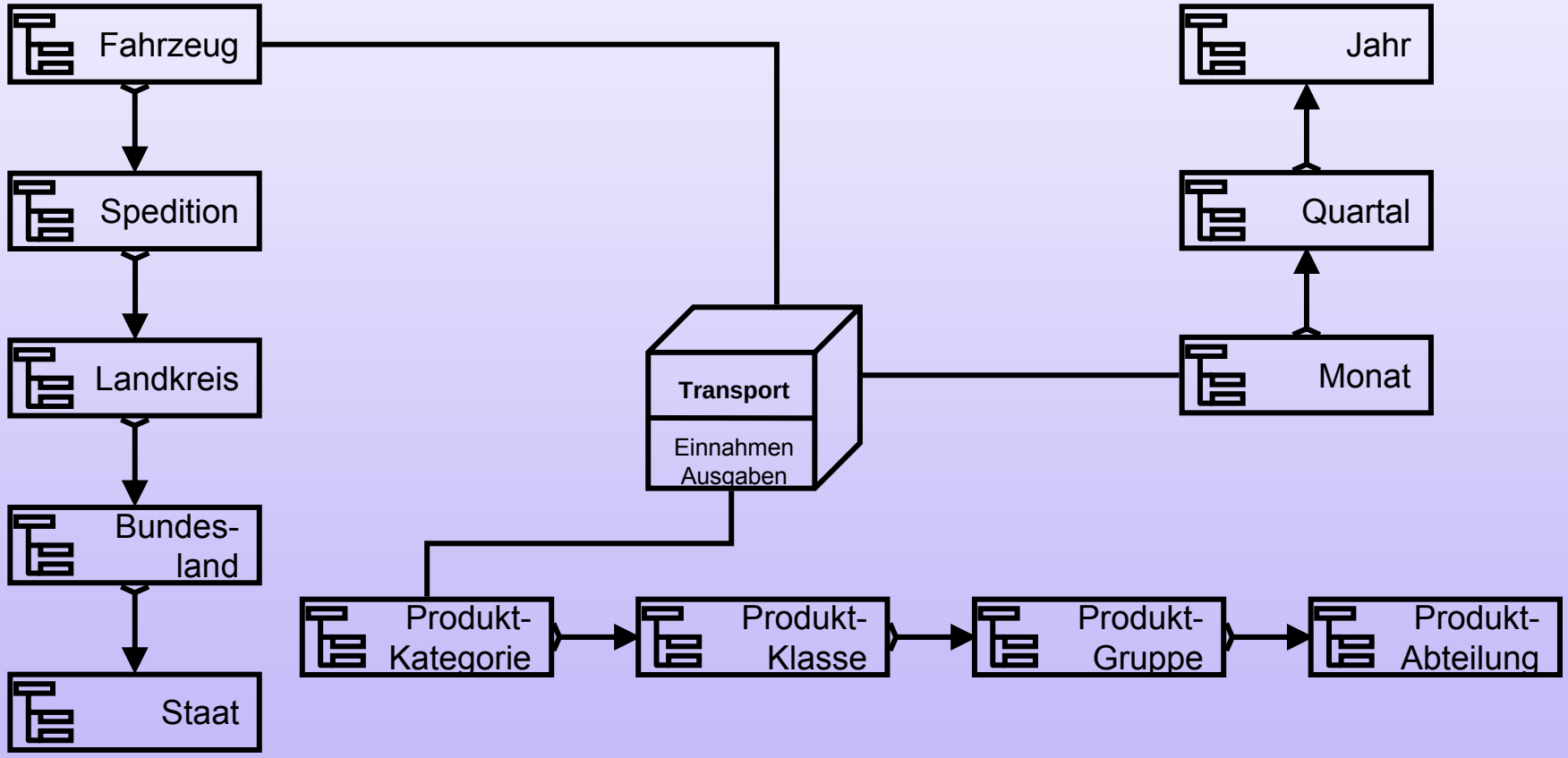


  - Klassifikationsbeziehung

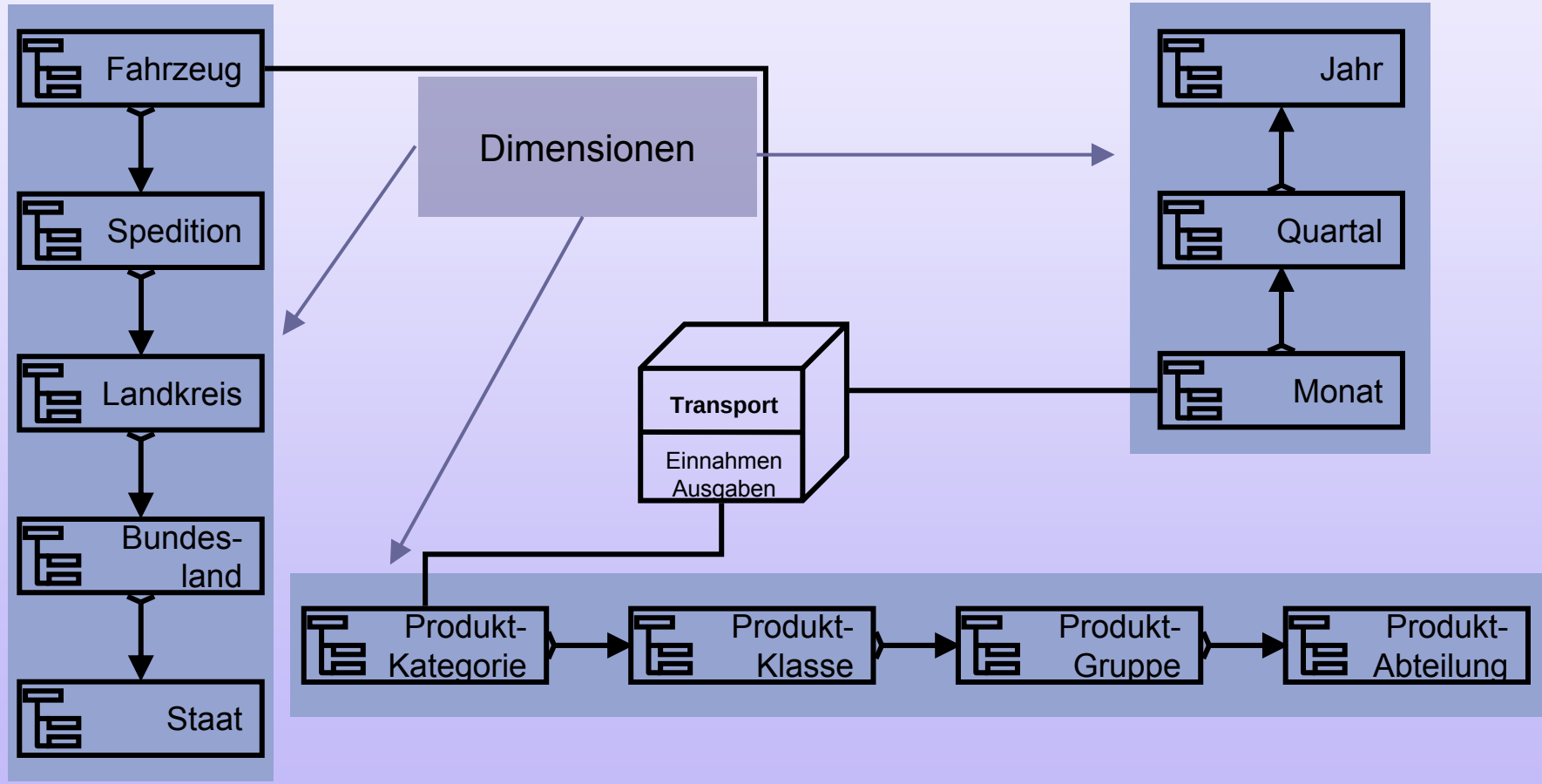
  - Faktbeziehung

Aus E/R-Notation übernommen

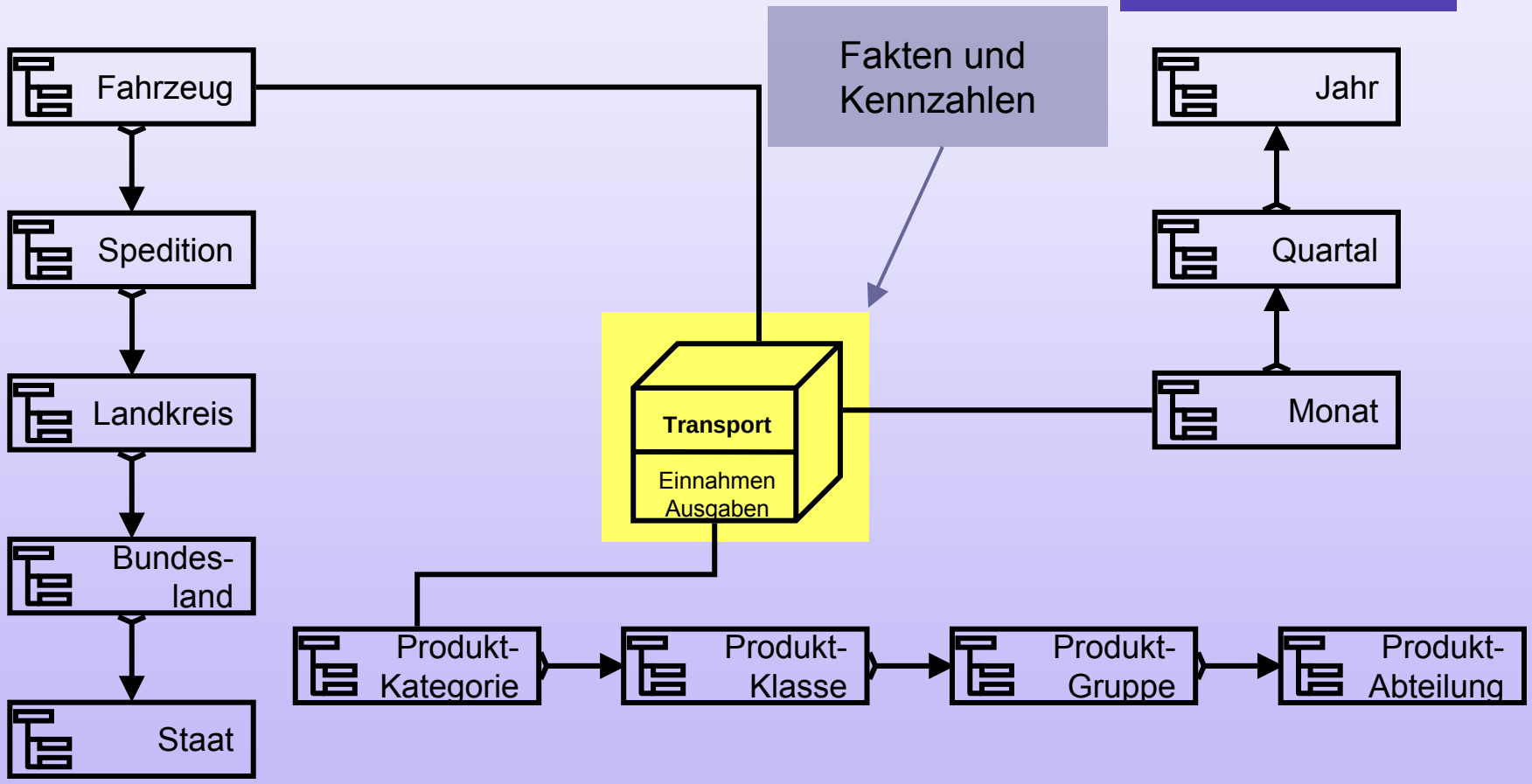
ME/R-Modellierung: Beispiel



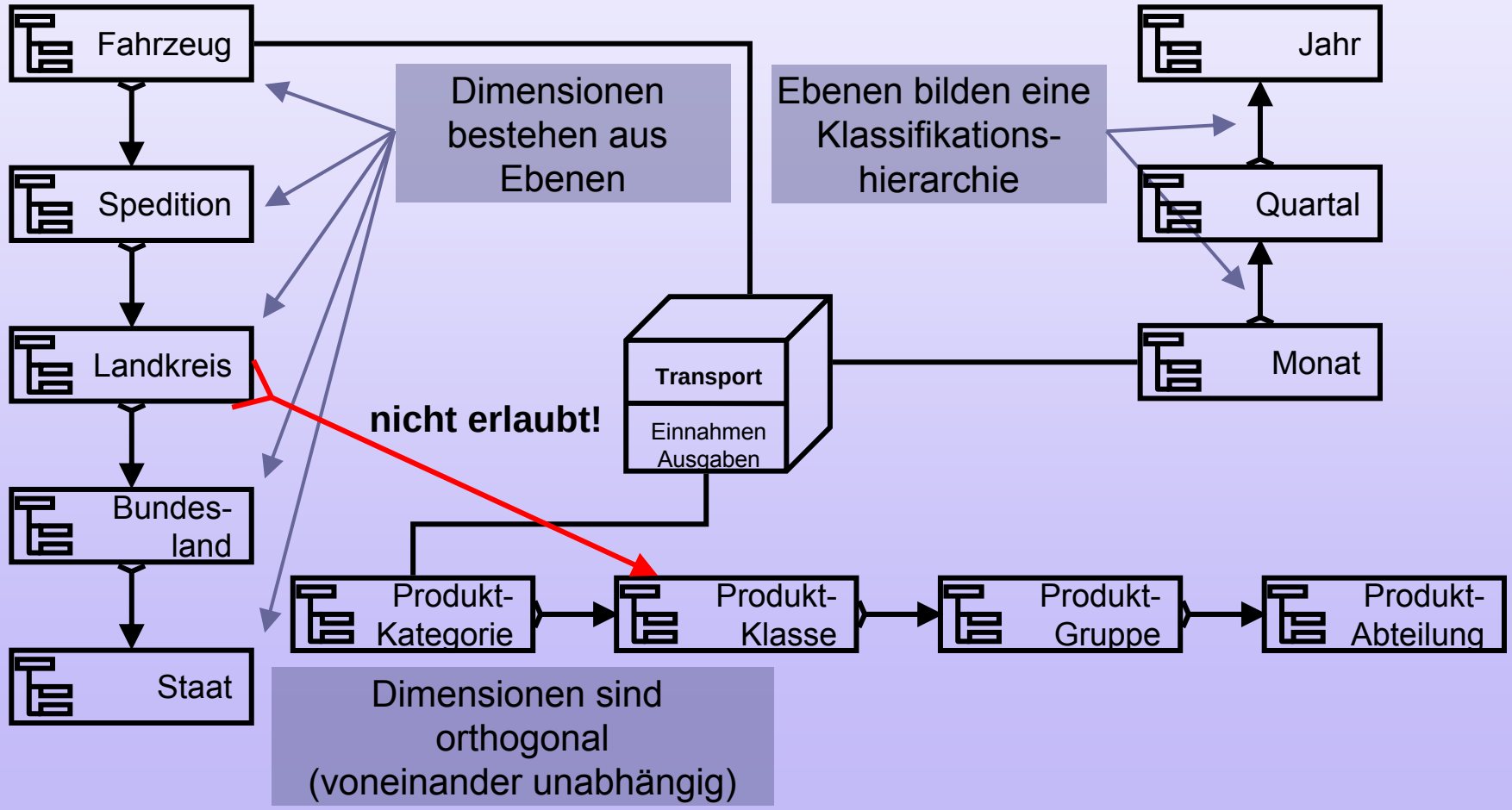
ME/R-Modellierung: Beispiel



ME/R-Modellierung: Beispiel

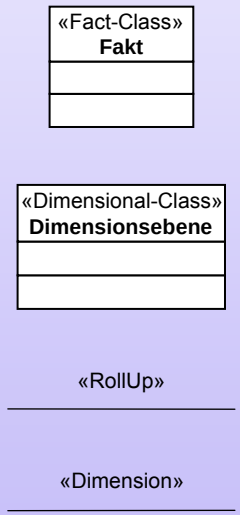


ME/R-Modellierung: Beispiel

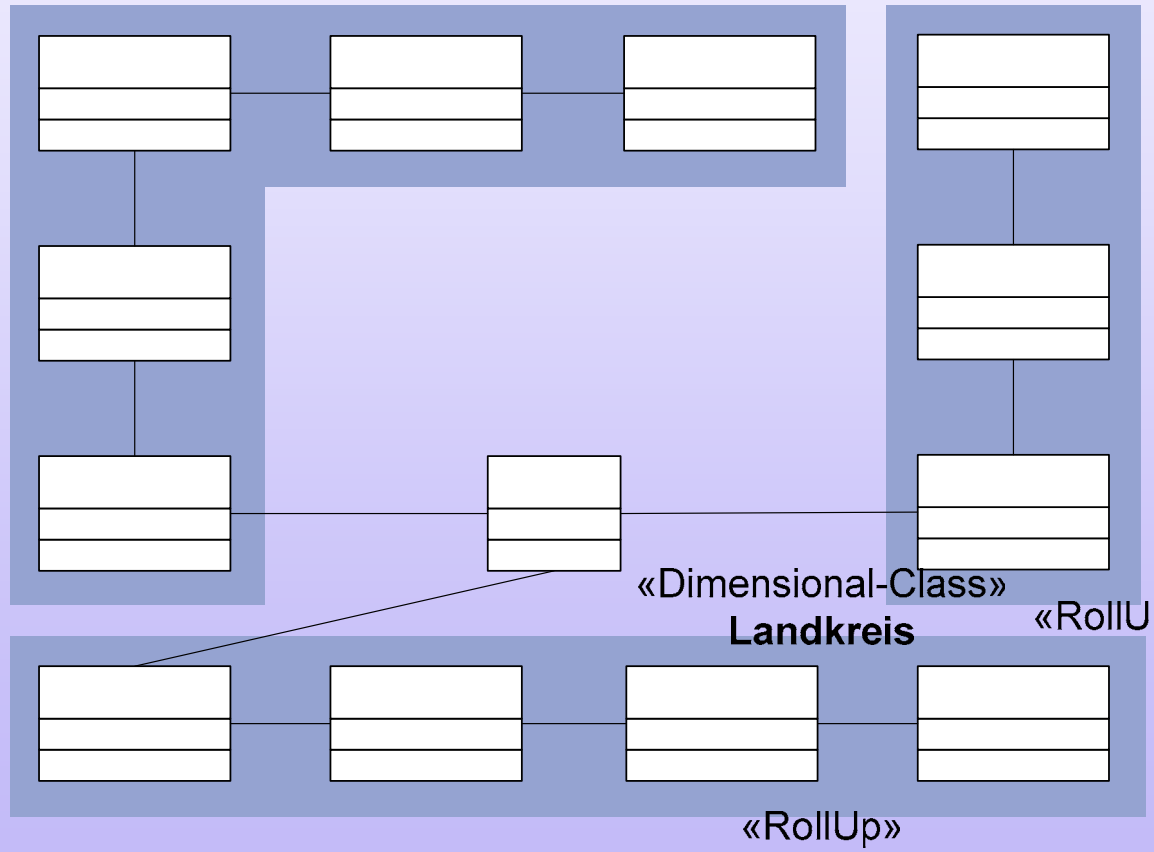


mUML-Modellierung

- **Erweiterung** des UML-Klassendiagramms um **multidimensionale** Konstrukte durch **Steoreotypen**:
  - Fakt
  - Dimensionsebene
  - Klassifikationsbeziehung
  - Faktbeziehung



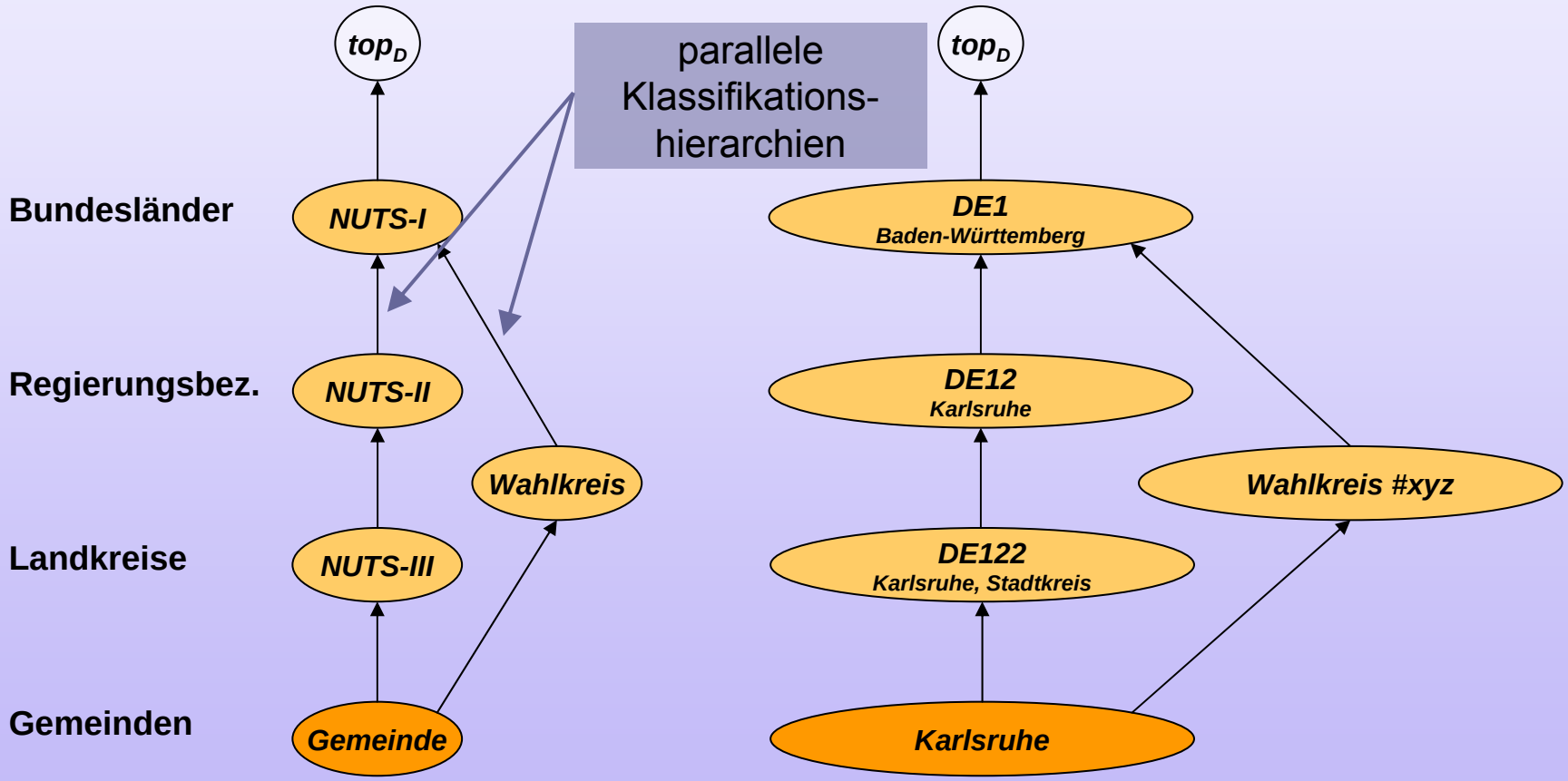
mUML-Modellierung: Beispiel



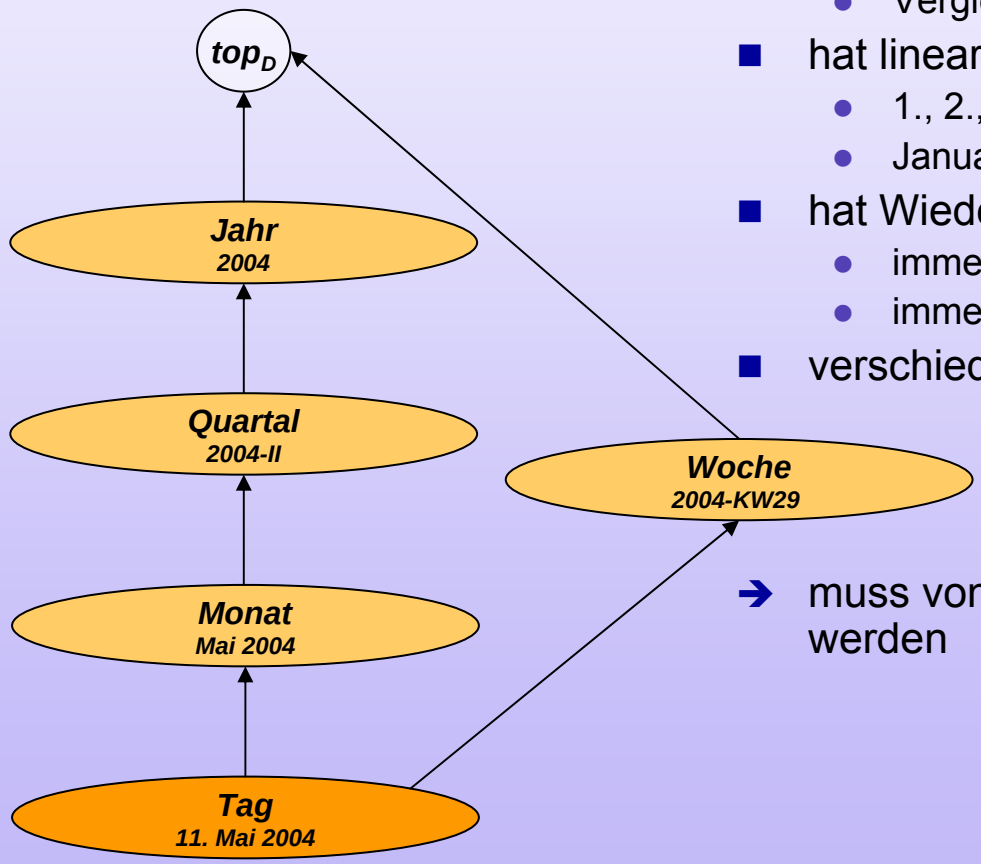
Qualifizierende Informationen

- Dimensionen
  - Dimensionen sind **orthogonal** (voneinander unabhängig)
  - Dimensionen bestehen aus **Dimensionsebenen**
  - Dimensionsebenen bilden eine **Klassifikationshierarchie**

Dimension: Ebenen



Dimension: Zeit



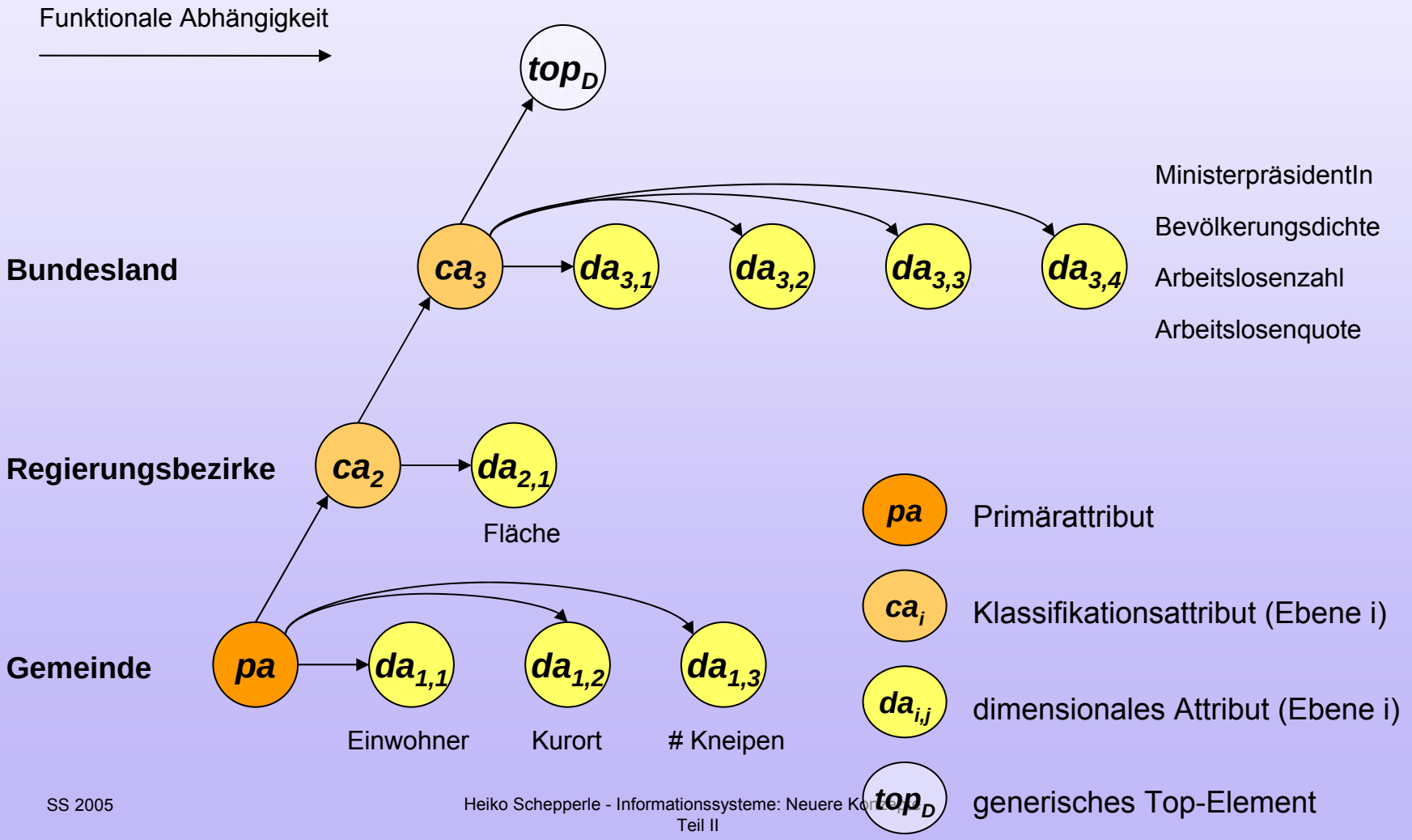
- kommt in fast jeder Anfrage vor:
  - Vergleich Quartal mit Vorjahresquartal
- hat linearen Charakter
  - 1., 2., 3., …
  - Januar, Februar, März, …
- hat Wiederholungscharakter
  - immer montags
  - immer im 4. Quartal
- verschiedene Klassifikationspfade denkbar

➔ muss von jedem Analysesystem unterstützt werden

Funktionale Abhängigkeiten

- Funktionale Abhängigkeit
  = functional dependency (FD)
  - Zwischen zwei Attributen A und B existiert eine funktionale Abhängigkeit (**A→B**) genau dann, wenn **für jedes a aus A genau ein b aus B** existiert.
  - Für Attribute A, B, C:
    - Man schreibt AB→C, wenn für jedes Paar (a,b) mit a aus A und b aus B genau ein c aus C existiert.
    - Man schreibt A→BC, wenn A→B und A→C gilt.

Dimension: Kategorieattribute



Informationssysteme: Neuere Konzepte – Teil II

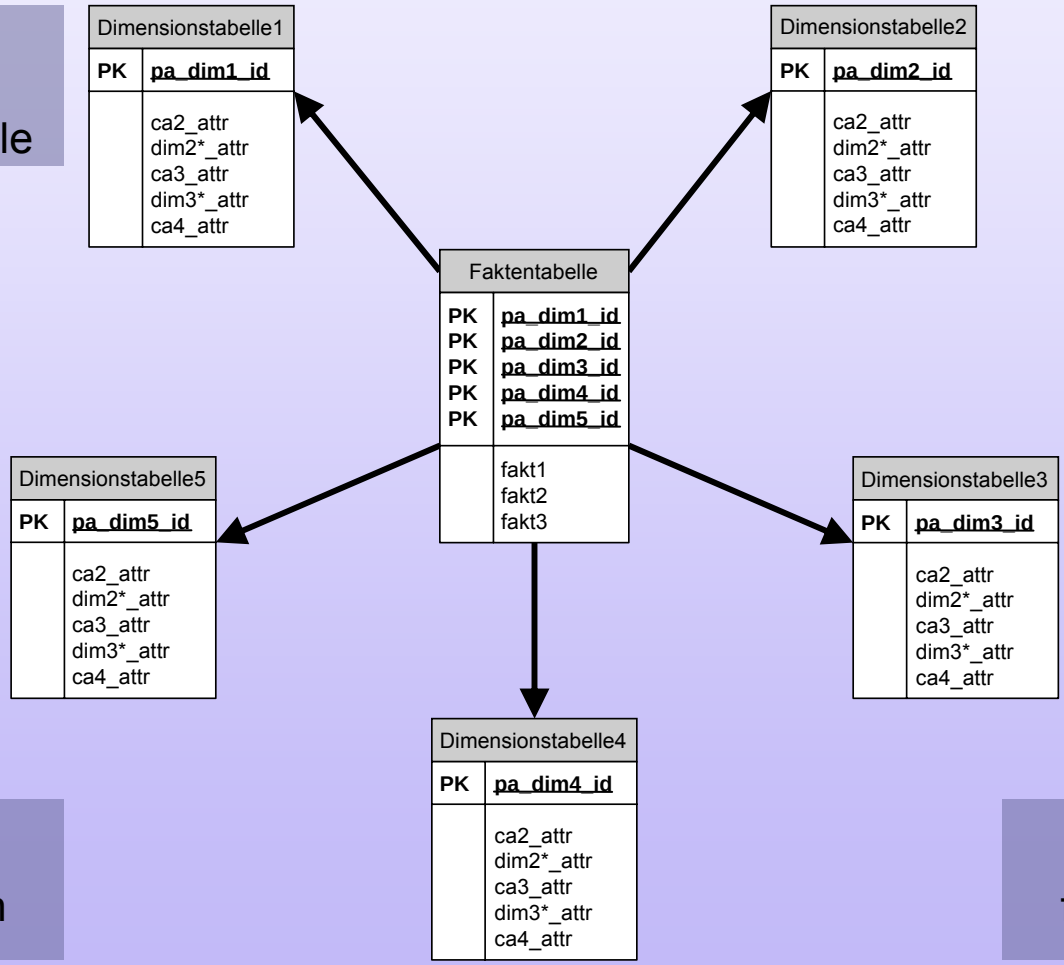
Kapitel 2: Data-Warehouse-Entwurf
- Logischer Datenbankentwurf -

Relationale Abbildung

- Wie lassen sich Kennzahlen und Dimensionen gut in einem relationalen Schema abbilden?
  - möglichst gut verständlich
  - auch relationales Schema sollte Zusammenhänge (z. B. Fakt- und Klassifikationsbeziehungen) abbilden
  - möglichst **performante Anfragebearbeitung**
  - möglichst redundanzfrei

Star-Schema

Dimensionen **sternförmig** um Faktentabelle



1 Tabelle pro Dimension

1 Tabelle für alle Fakten

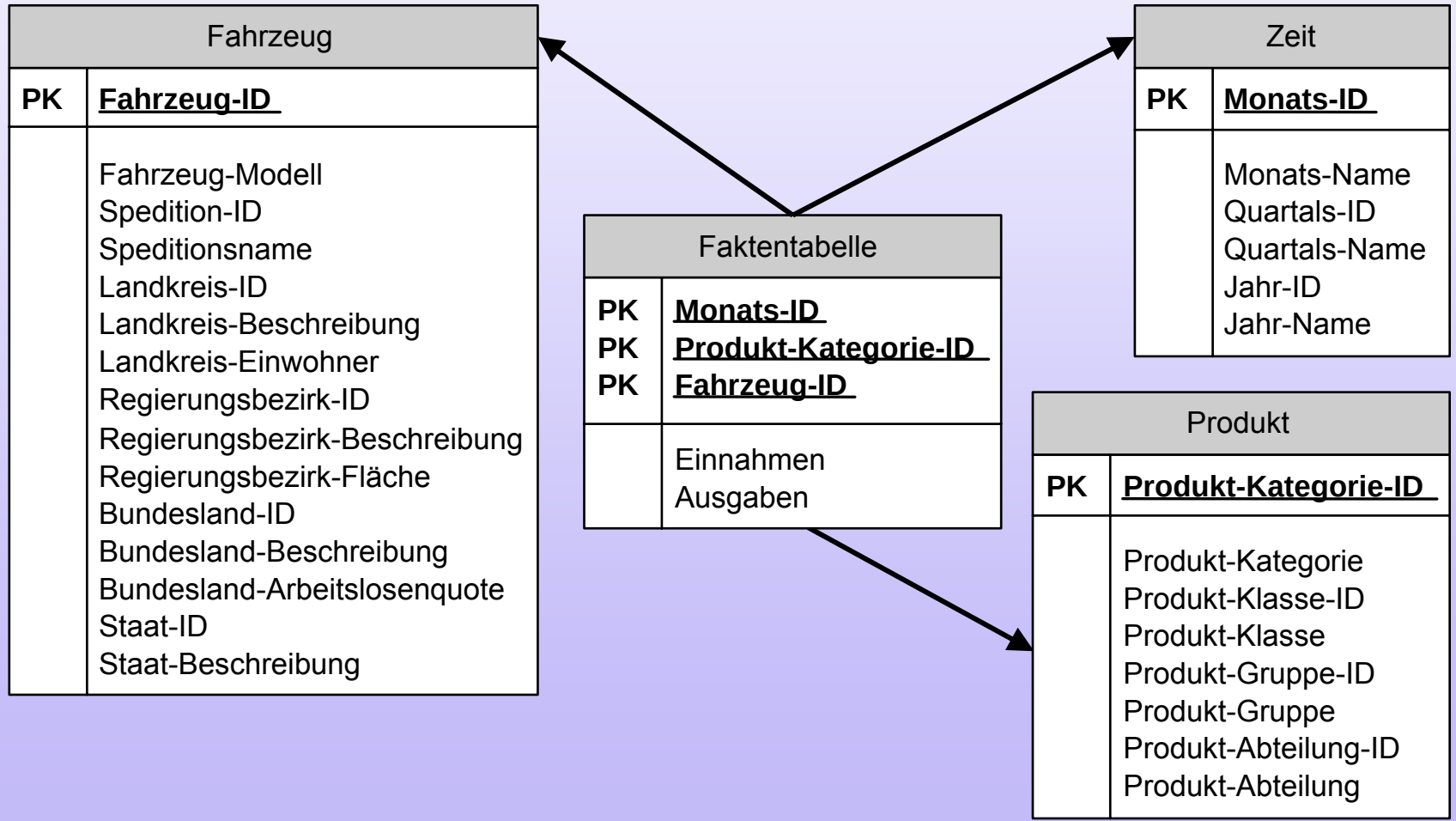
Faktentabelle

- Faktentabelle = fact table
- enthält Analysegegenstand, eigentliche Geschäftsdaten
- Tupel besteht aus
  - Zeigern auf Dimensionstabellen (Fremdschlüssel), die den Kontext des Tupels eindeutig bestimmen
  - den eigentlichen Kennzahlen
- Schlüssel der Faktentabelle =Gesamtheit der Dimensionszeiger
- Faktentabelle kann viele Millionen Tupel enthalten

Dimensionstabelle

- Dimensionstabelle = dimensional table
- pro Dimension existiert eine Dimensionstabelle
- Dimensionstabelle enthält
  - einen eindeutigen Schlüssel (Primärattribut), z.B. Produktnummer, Datums-ID, ...
  - weitere (beschreibende) Attribute der Dimension (Klassifikations- und dimensionale Attribute)
- Dimensionstabelle deutlich kleiner als Faktentabelle

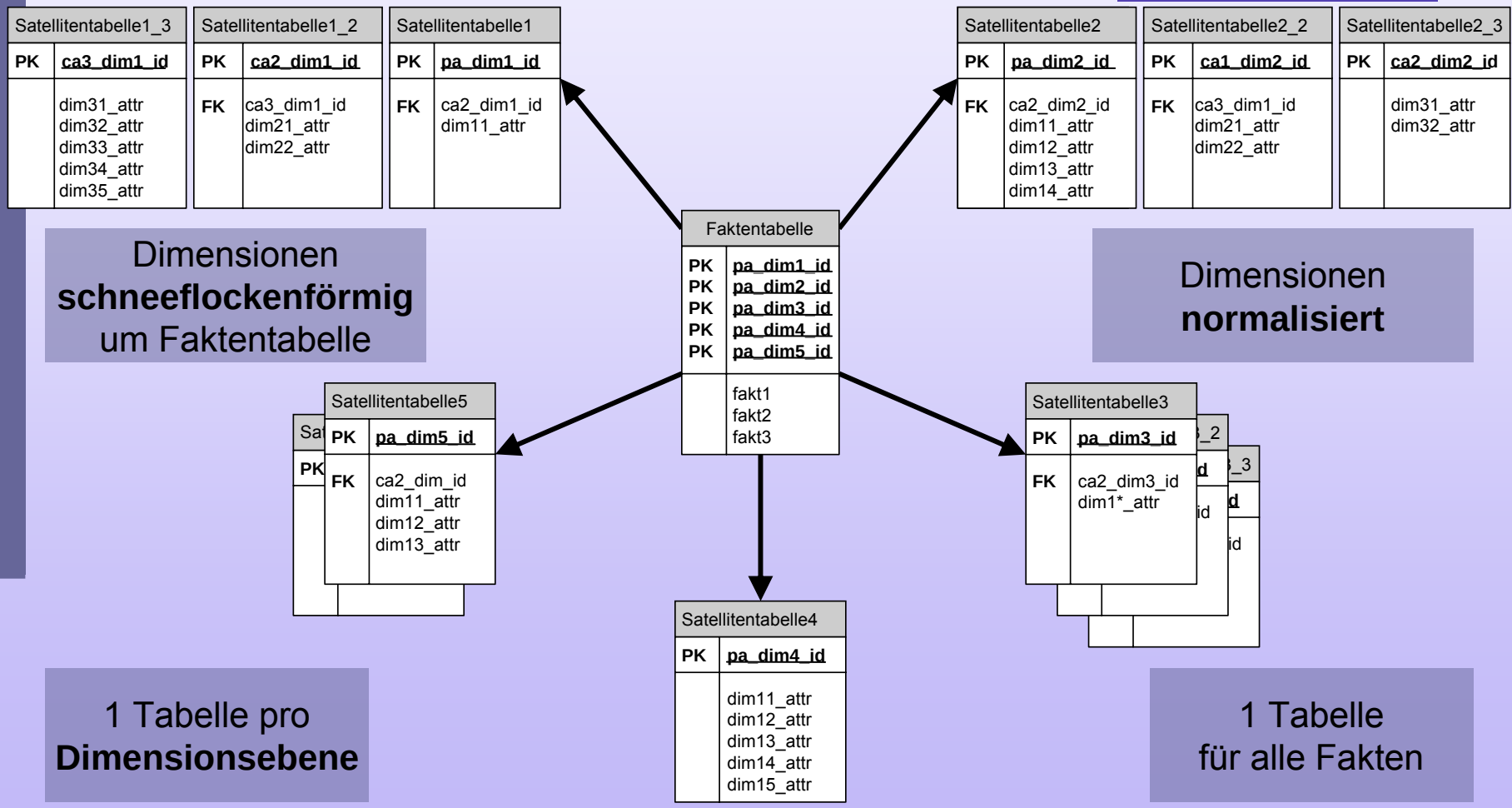
Star-Schema: Beispiel



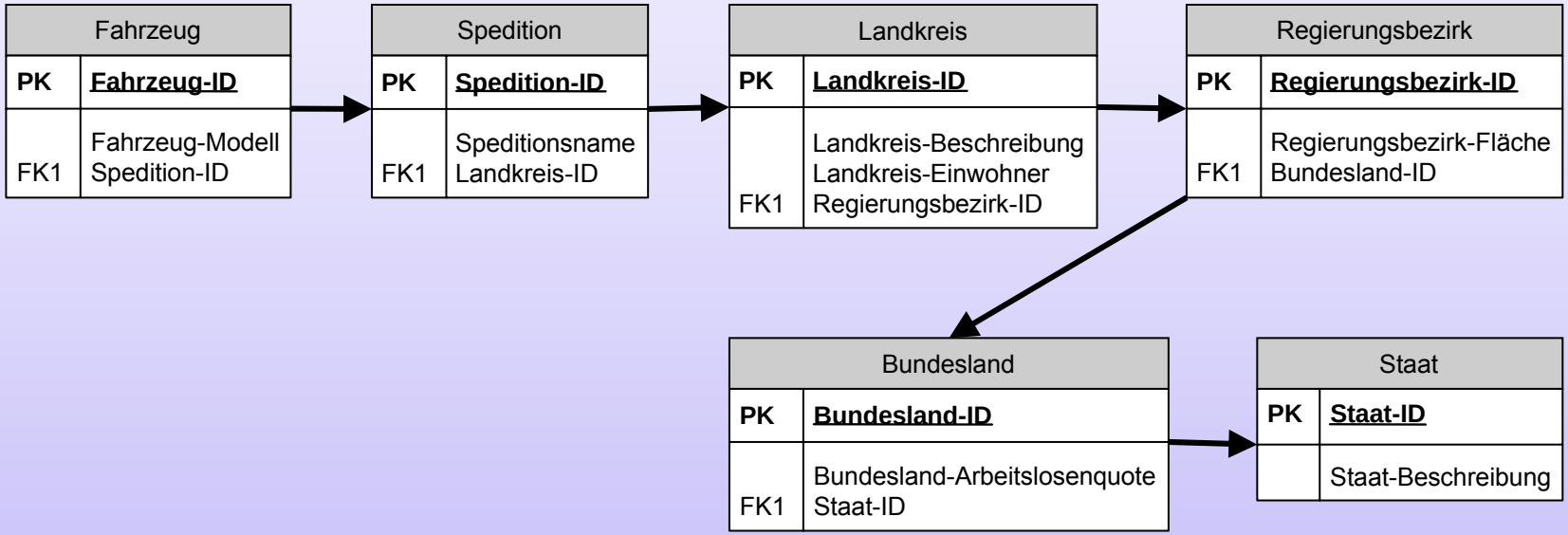
Star-Schema

- Eigenschaften
  - sehr übersichtlich
  - einfach erweiterbar
  - recht performante Anfragebearbeitung
- Probleme
  - **denormalisierte Dimensionstabellen** eventuell immer noch groß und redundant
  - **Dimensionsklassifikationen nicht** explizit **modelliert**

Snowflake-Schema



Snowflake-Dimension: Beispiel



Snowflake-Schema

- Warum muss jetzt alles ID heißen?
  - Eindeutigkeit der Referenzkette in der Klassifkationshierarchie:
    - Eine Monats-ID aus der Faktentabelle gehört zu einem bestimmten Quartal in einem bestimmten Jahr
    - „2. Quartal“ reicht dafür nicht, weil das Jahr nicht mehr herausgefunden werden kann

➔gegebenfalls sind zu den IDs in den Satellitentabellen noch Beschreibungen zu ergänzen (Speditionsname, Landkreis-Beschreibung, …)

Snowflake-Schema

- Probleme
  - weniger übersichtlich
  - weniger performante Anfragebearbeitung
- Eigenschaften
  - **normalisierte Dimensionstabellen**
  - Dimensionsklassifikationen explizit modelliert

Star- vs. Snowflake-Schema

- Welche Variante ist besser?
  - ➔ Welches Schema geeignet ist, hängt vom Anwendungsprofil ab!
  - ➔ Dennoch wird sehr häufig ein Star-Schema oder ein leicht modifiziertes Star-Schema verwendet.

Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf
- Physischer Datenbankentwurf -

Speicherorganisation

- ROLAP
  relationale Speicherorganisation
- MOLAP
  multidimensionale Speicherorganisation
- HOLAP
  hybride Speicherorganisation

ROLAP

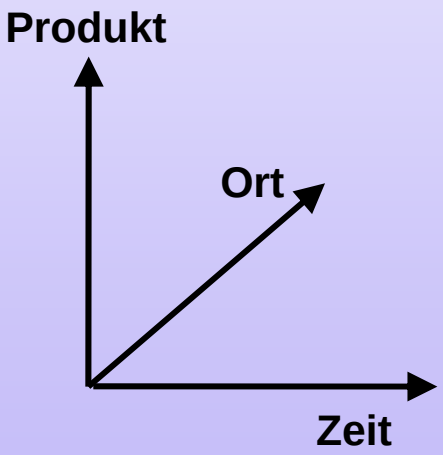
- ROLAP = Relational OLAP
- relationale Speicherung des multidimensionalen Datenmodells
- Speicherung der aggregierten Daten in Relationen
- geringere Performanz
- „unbegrenzt“ skalierbar

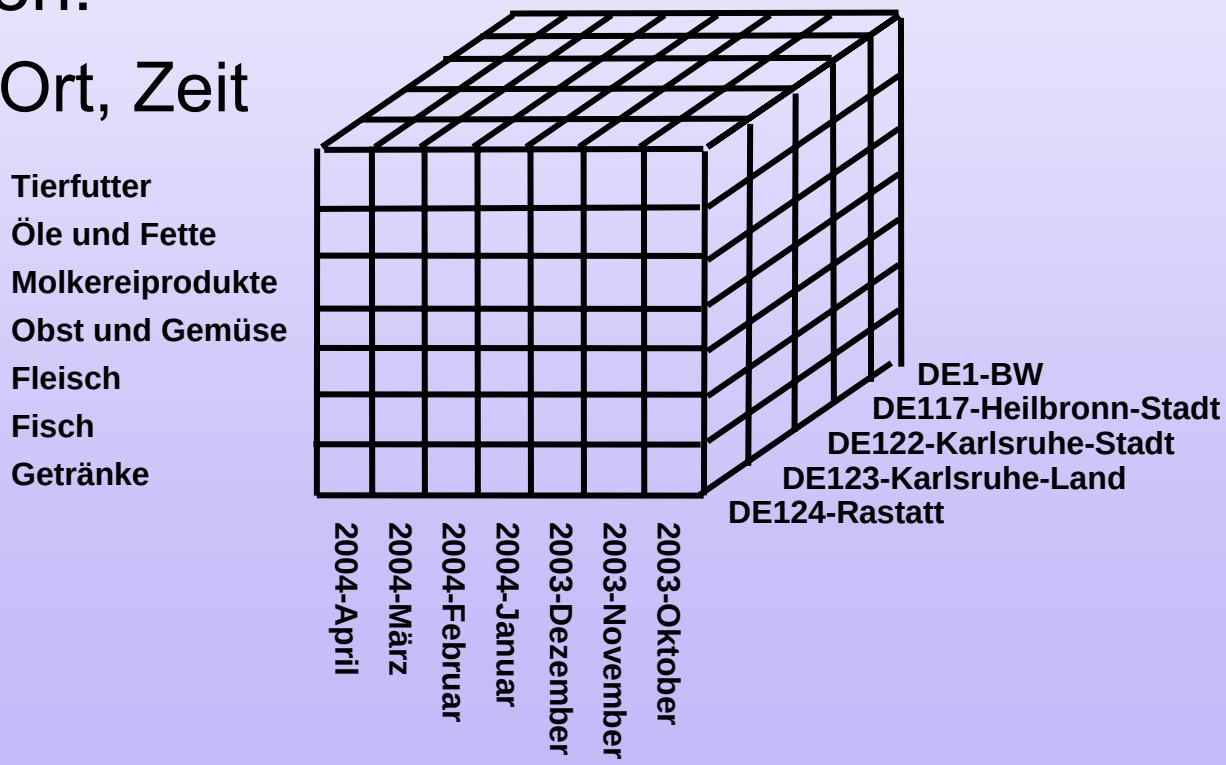
MOLAP

- MOLAP = Multidimensional OLAP
- multidimensionale Speicherung des multidimensionalen Datenmodells
- Speicherung der aggregierten Daten in speziellen multidimensionalen Datenstrukturen
- hohe Performanz
- begrenzte Skalierbarkeit

Multidimensionale Speicherung

- Würfel („Hypercube“)
- Dimensionen:
  - Produkt, Ort, Zeit





Multidimensionale Speicherung

- Array-Speicherung multidimensionaler Daten

➔Linearisierung

- Dimensionen D_1, D_2, D_3, …, D_m
- Punkt $(x_1,x_2,x_3,\ldots,x_n)$

$$z = 1 + \sum_{i=1}^{n} ((x_i - 1) \prod_{j=1}^{i-1} |D_j|)$$

- Beispiel: n=3, $|D_1|$=3, $|D_2|$=4, $|D_3|$=2; x=(2,3,1)

$$z = 1 + (x_1 - 1) + (x_2 - 1)|D_1| + (x_3 - 1)|D_1||D_2|$$

$$z = 1 + (2 - 1) + (3 - 1) * 3 + (1 - 1) * 3 * 4$$

Multidimensionale Speicherung

- Probleme
  - Reihenfolge der Dimensionen spielt eine Rolle
  - Nachbarschaftsbeziehung geht verloren
  - bei dünn besetztem Würfel hoher Speicherbedarf
    - dicht besetzt (dense) bei hohem Füllgrad
    - dünn besetzt (sparse) bei niedrigem Füllgrad
    - Füllgrad = Anzahl der nicht leeren Zellen / Anzahl aller Zellen
    - Eine Zelle mit dem Wert „0“ ist nicht leer!

ROLAP vs. MOLAP

	ROLAP	MOLAP
	relationales OLAP	multidimensionales OLAP
Speicherung	relationale Speicherstruktur (Tabellen, Tupel)	spezielle multidimensionale Speicherstruktur (Arrays)
Speicherzugriff	über Schlüssel, über Indizes, …	über Berechnungs-vorschrift
Performanz	mittel	hoch
Skalierbarkeit	„unbegrenzt“	begrenzt

HOLAP

- Hybrides OLAP
- Kombination von ROLAP und MOLAP
- Problem
  - MOLAP gut für dicht besetzte Würfel (leere Zellen belegen unnötig Speicher)
  - ROLAP gut für dünn besetzte Würfel (jedes Tupel speichert Schlüsselattribute)
  - ➔ Zwei-Ebenen-Datenstruktur

Zwei-Ebenen-Datenstruktur

- Unterscheidung in dünn- (sparse) und dichtbesetzte (dense) Dimensionen
- speichere Relation für dünn besetzte Dimensionen mit Zeiger auf einen Block
- Block ist Würfel aus dicht besetzten Dimensionen
- falls alle Dimensionen dünn besetzt sind und relational gespeichert werden:
  - ➔ Block besteht aus einer Zelle
  - ➔ äquivalent zu ROLAP
- falls alle Dimensionen dicht besetzt sind und in einem Block gespeichert werden:
  - ➔ Relation ist leer bzw. enthält Zeiger auf genau einen Block
  - ➔ äquivalent zu MOLAP

Weitere Elemente

- Es existieren weitere Elemente des physische Datenbankentwurfs
  - Indexstrukturen
  - Materialisierung
  - Partitionierung

Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf
- Indexstrukturen -

Indexstrukturen

- Wie werden die gewünschten Tupel einer Tabelle gefunden?
  - lineare Suche?
  - ➔linearer Aufwand
  - Binärsuche?
  - ➔logarithmischer Such-Aufwand, aber teure Sortierung nötig
  - geht‘s besser?
  - ➔durch Einsatz von Indexstrukturen(=Indizes)

Indexstrukturen

- Aufgabe einer Indexstruktur
  - effizienter, wertbasierter Zugriff auf
    - einzelne Tupel
      - EMQ=Exact Match Query (assoziativer/wertbasierter Zugriff)
    - mehrere Tupel über Wertebereichen
      - RQ = Range Query (Assoziativer Zugriff und sequentielles Lesen)
      - Beispiel: WHERE Jahrgang = 1998

Indexstrukturen

- Ziel:
  - Minimierung des zu übertragenden Datenvolumens (Seiten)
    - Festplattenzugriff hat den größten Einfluss auf Antwortzeit
    - auf Festplatte erfolgt Zugriff in Blöcken
    - Indexstrukturen arbeiten auf Seiten (4kB – 8kB) (hier kann 1:1-Zuordnung zu Blöcken angenommen werden)
  - Minimierung der wahlfreien Zugriffe
    - wahlfreier Zugriff
      Zugriff auf Blöcke unabhängig von der Reihenfolge der Speicherung
    - sequentieller Zugriff
      Zugriff auf Blöcke in der Reihenfolge der Speicherung

Varianten ...

- Lineare Strukturen
  - Listen
    - einfach-verkettet
    - doppelt-verkettet
  - Arrays
- Hashtabellen
- Bäume
  - B-Baum
  - B*-Baum

... im Vergleich

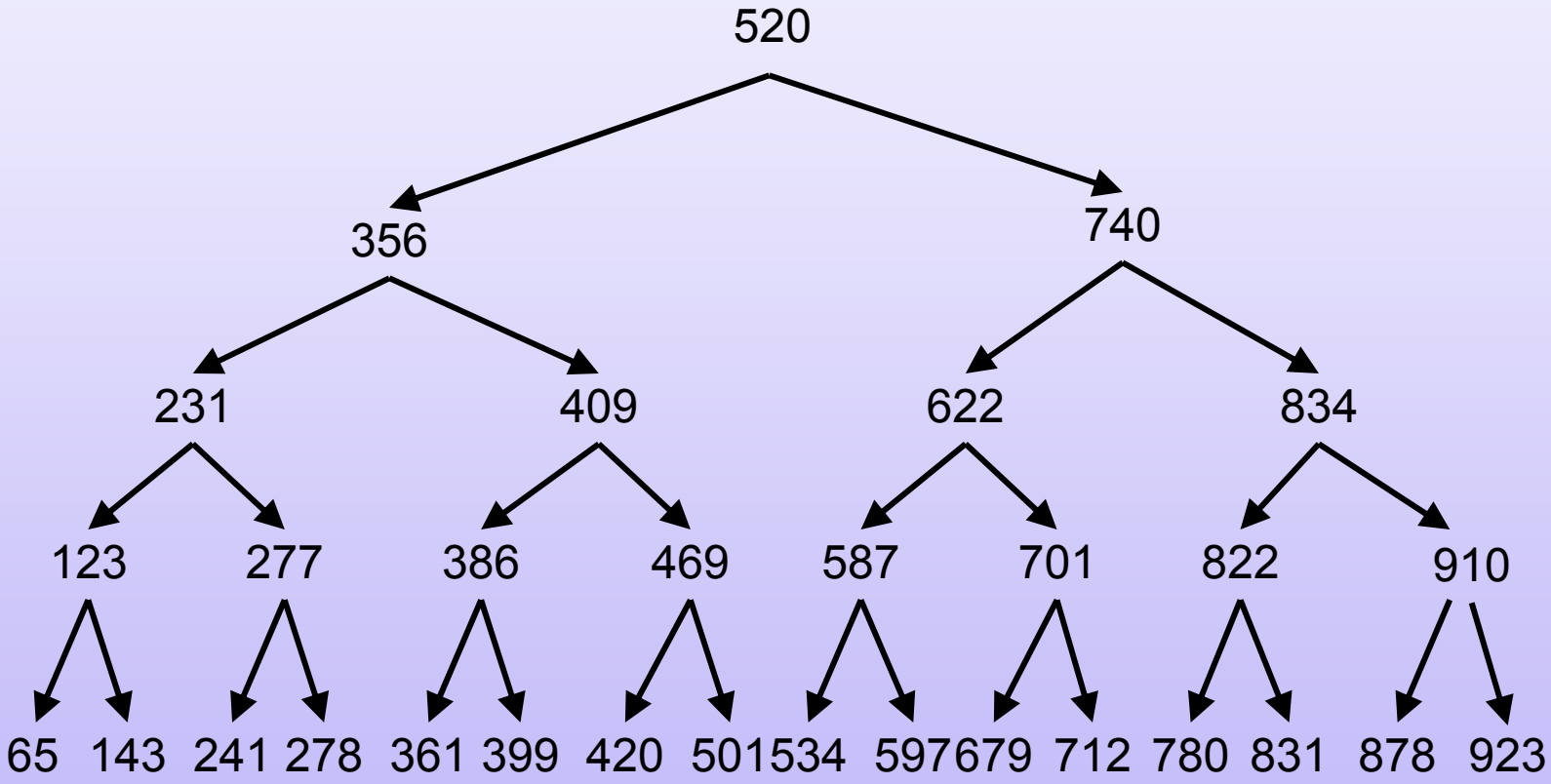
	Einfügen Löschen	assoziativer Zugriff	Index-zugriff	sequentielles Lesen
Listen	++	--	-	+
Array	--	+	++	++
Hashtable	+	++	-	--
Baum	+	+	-	+(+)

- Einfügen/Löschen sowie sequentielles Lesen benötigt meist zusätzlich noch assoziativen Zugriff zum Auffinden der korrekten Position

➔ bei Listen problematisch

➔ meist Bäume als Zugriffsstruktur

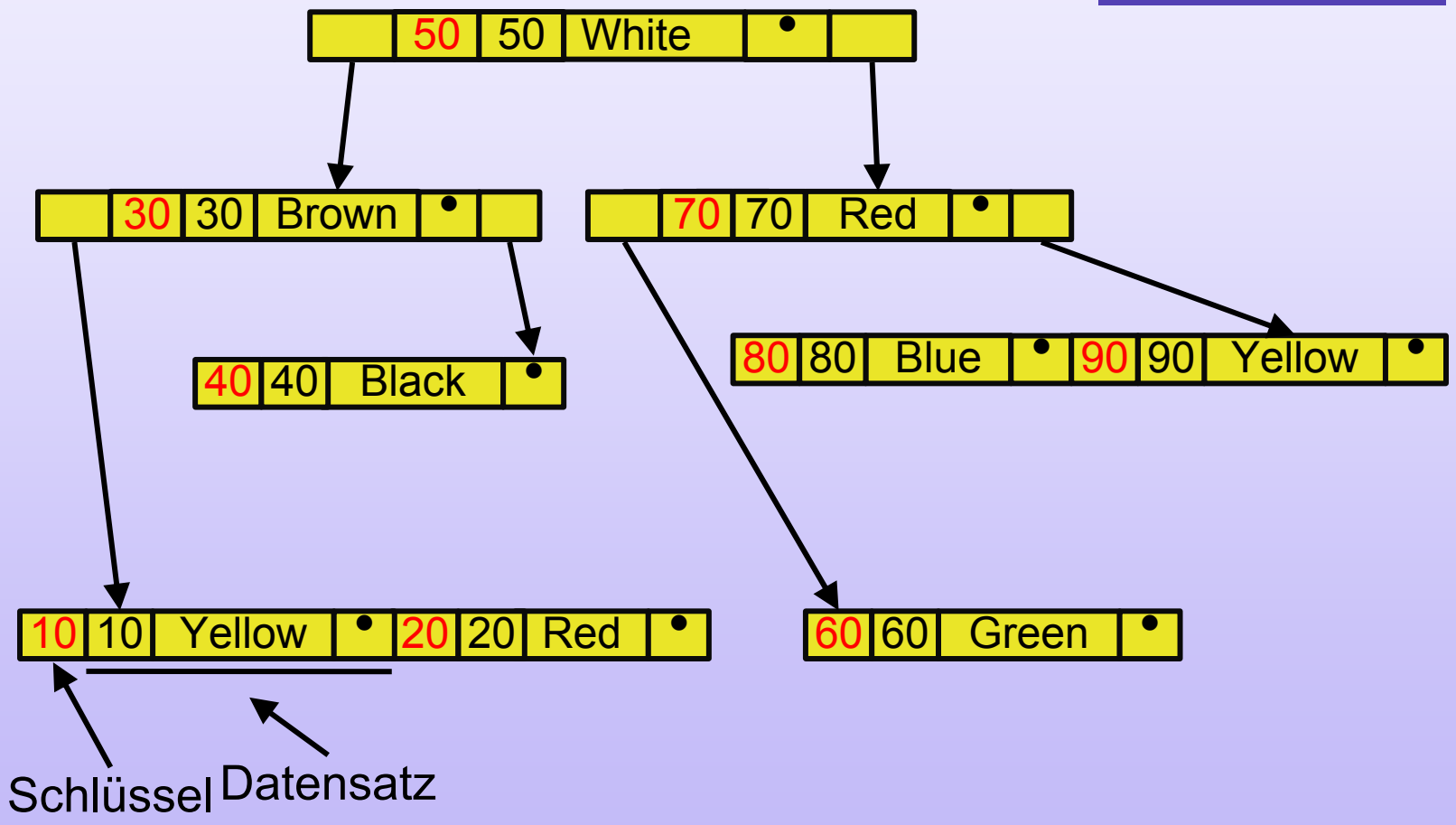
Binärbaum


520
356
740
231
409
622
834
123
277
386
469
587
701
822
910
65
143
241
278
361
399
420
501
534
597
679
712
780
831
878
923

Binärbaum: Eigenschaften

- Probleme:
  - Sehr viele Navigationen bei Direktzugriff
  - Aufteilung auf Seiten unklar
  - Balancierung?
- Idee: Mehrwegbaum:
  - Anzahl der Elemente pro Knoten > 1
  - Knotenelemente in Sortierreihenfolge (wegen sequentieller Verarbeitung)
  - garantierter Füllgrad (Splitten und Vereinigen von Knoten)

B-Baum


50 50 White
30 30 Brown
70 70 Red
40 40 Black
80 80 Blue
90 90 Yellow
10 10 Yellow
20 20 Red
60 60 Green
Schlüssel
Datensatz

B-Baum

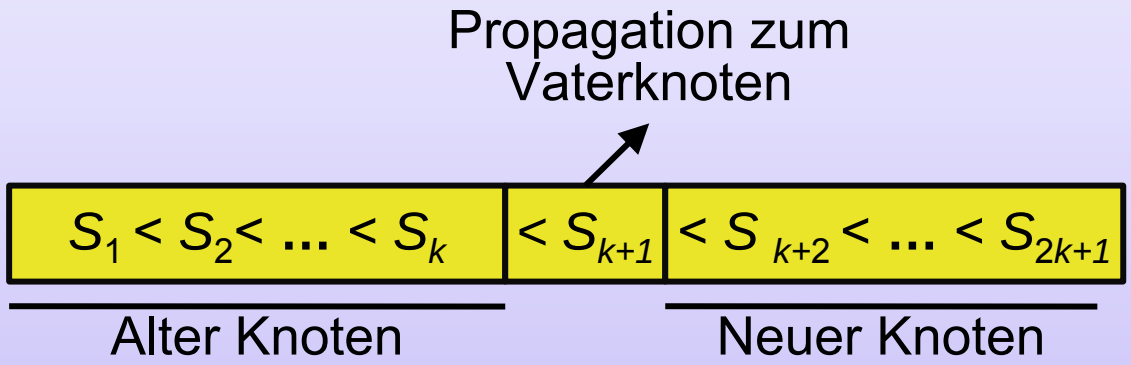
- Jeder innere Knoten hat n+1 Söhne mit $k \leq n < 2k$, Ausnahme: Wurzel ($1 \leq n < 2k$)
- Jeder Knoten ist mit n sortiert angeordneten Schlüsseln belegt
- Seien $S_1,...,S_n$ die Schlüssel, so gilt:
  - Z_0 weist auf den Unterbaum mit Schlüsseln < S_1
  - Z_j (für j=1...n-1) weisen auf Unterbäume, deren Schlüssel echt zwischen S_j und S_{j+1} liegen.
  - Z_n weist auf den Unterbaum, dessen Schlüssel größer als S_n sind.
- Jeder Weg von der Wurzel zu einem Blatt hat die gleiche Länge (= Höhe des Baumes)

B-Baum: Einfügen

- Suche nach dem Schlüssel des neuen Datensatzes (notfalls bis zum Blatt).
- Falls gefunden: Einfügen des neuen Datensatzes in den entsprechenden Knoten des B-Baums (bei nicht eindeutigem Schlüssel).
- Falls nicht gefunden: Einfügen des neuen Datensatzes in das bei der Suche erreichte Blatt.
- Bei Überlauf eines Knotens wird der Knoten gespalten, d.h. die Einträge des Knotens werden auf zwei Knoten verteilt.

B-Baum: Überlauf/Unterlauf

- Der Datensatz, der in der Mitte des übergelaufenen Knotens liegt, wird zum Vaterknoten propagiert.

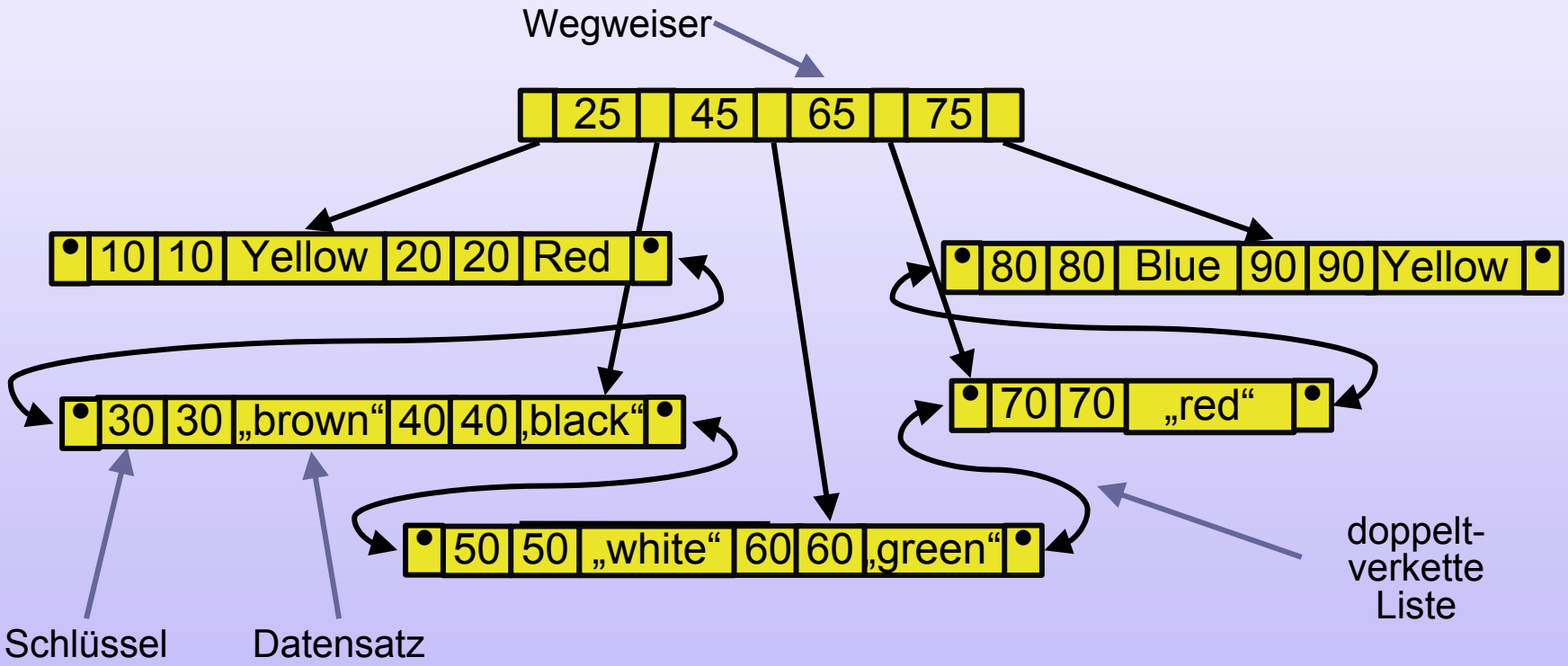


- Bei Unterlauf (n<k) werden zuerst Elemente aus Nachbarknoten verschoben
- Falls dort ebenfalls Unterlauf ➔ Knoten wird mit Nachbarknoten verschmolzen

B-Baum: Eigenschaften

- Probleme
  - Höhe = Anzahl Zugriffe = $\log_s(n)$
  - Verzweigungsgrad s = Knotengröße / (Schlüssel- + Datensatzgröße)
  - s und damit Baumhöhe abhängig von Datensatzlänge
  - immer noch Ineffizienz beim sequentiellen Lesen
- Lösung
  - “hohler Baum”, Daten nur an den Blattknoten
  - Verkettung der Blattknoten

B*-Baum



- Regeln zum Über-/Unterlauf wie beim B-Baum

Indexarten

- Primärindex
  Index über eindeutiges Attribut (ID=Schlüsselattribut) der Datenkollektion
- Sekundärindex
  Index über nicht eindeutiges Attribut (ID≠Schlüsselattribut) der Datenkollektion
- Clusterindex
  Datenkollektion ist physisch in der Reihenfolge der Indexwerte angeordnet

➔ geclusterte Primärindizes

➔ geclusterte Sekundärindizes

Was es sonst noch gibt

- Mehrdimensionale Datenstrukturen
  - Anwendungsfelder:
    - Geographische Datenbanken (GIS)
    - CAD-Datenbanken (EDM/PDM)
    - Data Warehouses
  - Beispiele:
    - UB-Baum: space filling curve + B-Baum
    - Bitmap-Index
    - R*-Baum: Bounding Rectangle in inneren Knoten
    - k-d-B-Baum: achsenparallele Splits

Indizes im Data Warehouse

- Welche Indexstrukturen unterstützen Data-Warehouse-Anfragen besonders gut?
- Problem
  - Filterung auf der Faktentabelle
- Anforderung
  - Effiziente Verknüpfung von Einschränkungen der Faktentabelle auf disjunkten Attributmengen

Indizes im Data Warehouse

- Variante 1
  - 1 B-Baum auf einer Dimension
  - Restliche Einschränkungen werden zu normalen Selektionsbedingungen
  - ➔ Viel zu viele Tupel gelesen!
- Variante 2
  - 1 B-Baum für jede Dimension
  - für jede Dimension IDs der selektierten Tupel bestimmen
  - Schnittmenge bilden
  - hoher Wartungsaufwand (pro Insert bis zu 50 Indizes anzupassen)
- Variante 3
  - Multidimensionale Indizes

Multidimensionale Indizes

- Immer Kompromiss zwischen
  - Füllgrad
  - Überlappungsfreiheit
  - Balancierung des Baumes
  - Clusterung benachbarter Daten

Bitmap-Index

- Bitmap-Index besteht aus einer **Menge von Bitvektoren**
- Struktur
  - 1 Bitmap-Index pro Dimension
  - 1 Bitvektor pro Attributwert
  - Länge des Bitvektors = #Tupel der Faktentabelle
  - Eintrag 1 in Position i eines Bitvektors ➔ Tupel i hat entsprechenden Wert
  - Eintrag 0 in Position i eines Bitvektors ➔ Tupel i hat nicht entsprechenden Wert
- Verwendung
  - zuerst innerhalb der Dimension ODER-Verknüpfung aller passenden Bitvektoren
  - dann UND-Verknüpfung der resultierenden Bitvektoren verschiedener Dimensionen

Bitmap-Index

Nr.		Monats-ID	Produkt-ID	Fahrzeug-ID	Einnahmen	Ausgaben
1		2004-Jan	5	1	1235	879
2		2004-Jan	7	3	5321	6345
3		2004-Feb	3	2	543	367
4		2004-Mar	4	1	235	198
5		2004-Apr	3	3	5432	5399

- alle Lebensmittel-Transporte (Produkt-ID=3)
  - 00101 (Tupel Nr. 3 und Nr. 5)
- alle Lebensmittel-Transporte (Produkt-ID=3) von Fahrzeug 3
  - 00101 UND 01001 = 00001 (nur Tupel Nr. 5)
- alle Kleidungs- (Produkt-ID=5) und Schuh-Transporte (Produkt-ID=7) im Januar 2004
  - (10000 ODER 01000) UND 11000 = 11000 UND 11000 = 11000 (Tupel Nr. 1 und Nr. 2)

Produkt-ID		1	2	3	4	5	...
1		0	0	0	0	0	
2		0	0	0	0	0	
3		0	0	1	0	1	
4		0	0	0	1	0	
5		1	0	0	0	0	
6		0	0	0	0	0	
7		0	1	0	0	0	

Monats-ID		1	2	3	4	5	...
2004-Jan		1	1	0	0	0	
2004-Feb		0	0	1	0	0	
2004-Mar		0	0	0	1	0	
2004-Apr		0	0	0	0	1	

Fahrzeug-ID		1	2	3	4	5	...
1		1	0	0	1	0	
2		0	0	1	0	0	
3		0	1	0	0	1	

Bitmap-Index

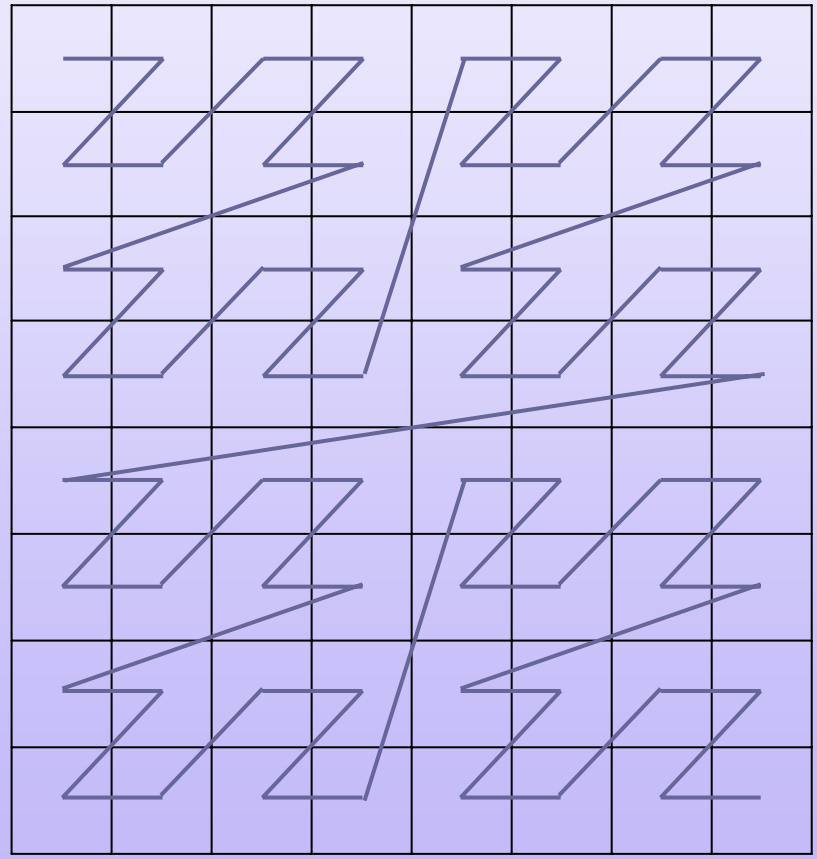
- Problem
  - Bitmap-Index braucht Platz
    - #Anzahl benötigter Bytes = Kardinalität (Anzahl der verschiedenen Werte) der Domäne * Anzahl Tupel der Faktentabelle / 8
    - Unterstützung für Range Queries nicht optimal
- Verbesserungsansätze
  - Bereichsbasierte Kodierung (range-based encoding)
  - Komprimierung von Bitvektoren

UB-Baum

- Linearisierung des mehrdimensionalen Raums durch „space filling curve“, anschließend Verwendung eines eindimensionalen Index (B*-Baum)
- Nachbarschaftsbeziehung bleibt besser erhalten als einfache Linearisierung (wie bei MOLAP beschrieben)
- Beispiel Z-Kurve

UB-Baum

- Reihenfolge der Dimensionen:
  erst →, dann ↓
- Z-Wert:
  Position in der Z-Kurve
- Z-Region:
  Anfangs- und Endpunkt als Paar von Z-Werten



Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf
- Materialisierung -

Materialisierung

- Faktentabelle durchsuchen und Berechnung von Aggregationen kostet Zeit
- Idee
  - häufig gebrauchte aggregierte Zwischenergebnisse materialisieren (physisch ablegen, speichern)
- Systematisches Vorgehen
  - Dimensionshierarchien aufstellen
  - Häufigkeit der Zugriffe auf die einzelnen Aggregationsstufen protokollieren
  - Modell aufstellen, welche Ergebnisse von welchem Zwischenergebnis profitieren
  - kostenbasierte Entscheidung treffen (Speicherplatz ↔ Lesezugriff ↔ Schreibzugriff)

Materialisierung

- Kostenmodell
  - Speicherplatzkosten
  - Update-Kosten
    - also Kosten für Anpassung materialisierter Zwischenergebnisse bei Datenänderung
    - wobei Update auch von Materialisierung profitieren kann
- Nichttriviales Problem: Query Rewriting
  - also Umschreiben von Anfragen, so dass bestehende Materialisierungen genutzt werden können

Materialisierung: Beispiel

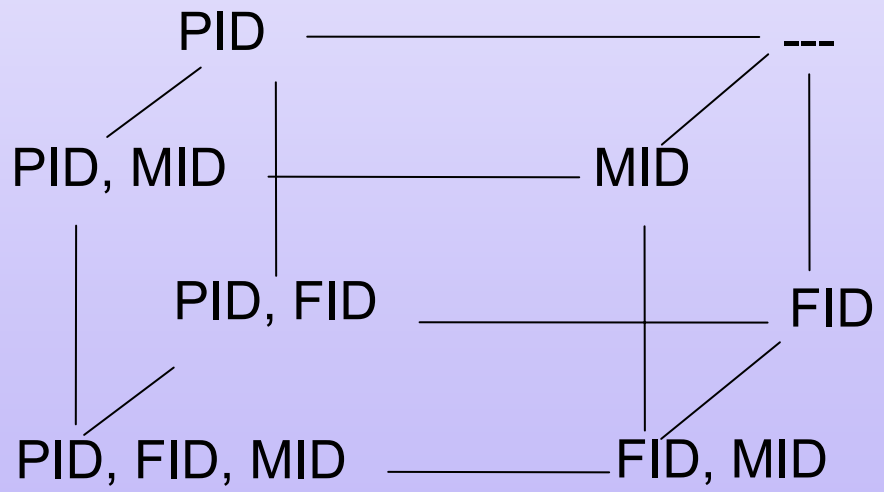
Monats-ID	Produkt-ID	Fahrzeug-ID	Einnahmen	Ausgaben
2004-Jan	5	1	1235	879
2004-Jan	7	3	5321	6345
2004-Feb	3	2	543	367
2004-Mar	4	1	235	198
2004-Apr	3	3	5432	5399
2004-Jan	2	2	745	4536
2004-Feb	2	3	346	636
2004-Jan	4	1	6246	3677
2004-Apr	1	2	326	436
2004-Apr	2	2	6436	7858
2004-May	4	3	8658	6356
2004-Jun	5	1	568	456
2004-May	4	3	5868	3167
2004-Jun	5	2	8762	6788

Quartals-ID	Produkt-ID	Einnahmen
2004-I	2	1091
2004-I	3	543
2004-I	4	6481
2004-I	5	1235
2004-I	7	5321
2004-II	1	326
2004-II	2	6436
2004-II	3	5432
2004-II	4	14526
2004-II	5	9330

- Wie sieht eine Anfrage nach den Einnahmen bei Lebensmittel-Transporten (Produkt-ID=3) pro Quartal mit und ohne Einsatz der aggregierten Tabelle aus?
- Star- oder Snowflake-Schema?

Materialisierung: Beispiel

- Aggregationsdimensionen:
  - Produkt (PID), Fahrzeug (FID), Monat (MID)



Materialisierung

- Erweiterung um mehrstufige Aggregationshierarchien (wie z.B. Monat → Quartal im Beispiel) und gegebenenfalls Dimensionseinschränkungen führen zu explosionsartigem Wachstum der Möglichkeiten!

Informationssysteme: Neuere Konzepte – Teil II

Kapitel 2: Data-Warehouse-Entwurf
- Data-Warehouse-Werkzeuge -

Data-Warehouse-Werkzeuge

- **ETL-Werkzeuge**
  - Werkzeuge zur **Unterstützung des Aufbaus und der Wartung** eines Data Warehouse
  - Beispiele:
    - DataJunction
    - DataStage (Ascential)
    - …
- **OLAP-Werkzeuge**
  - Werkzeuge zur **multidimensionalen Datenanalyse**
  - Beispiele:
    - Hyperion Essbase
    - BusinessObjects
    - Cognos
    - MicroStrategy
    - …

Vergleich DW-Werkzeuge

Funktionsumfang der Business-Intelligence-Werkzeuge

	Daten-integration	Datenbank-management 1)	Online-analyse 2)	Webapplikations-aufbau
Arcplan				●
Ascential	●			
Brio				●
Business Objects	+			●
Cognos	●		●	●
Crystal Decisions			●	●
Hyperion	◗		●	+
IBM	◗	●		
Informatica	●			+
Microsoft	◗	●	●	
Microstrategy			●	
MIK			●	
MIS			●	◗
NCR Teradata		●		
Oracle	◗	●	●	
SAP		◗	●	+
SAS	◗	●	●	◗

1) Relationales Datenbank-Management-System (RDBMS)
2) Online Analytical Processing (OLAP), Datenbank oder explizites Werkzeug

Literatur

- A. Bauer, H. Günzel: ***Data Warehouse Systeme***. Architektur, Entwicklung, Anwendung. dpunkt.verlag, 2001.
- W. Lehner: ***Datenbanktechnologie für Data-Warehouse-Systeme***. dpunkt.verlag, 2003.

Anfragebearbeitung: SQL

- Beispiel SQL

  SELECT <Felder>

  FROM <Tabellen>

  WHERE <Einschränkungen>

  GROUP BY <Gruppierungsfelder>

  HAVING <Gruppeneinschränkungen>

- Oder

  UPDATE <Tabelle>

  SET <Feld> = <Wert>

  WHERE <Einschränkungen>

Anfragebearbeitung: SQL

- Zum Aufwärmen:
  - SELECT Name, Vname, AVG(Note) AS Schnitt
  - FROM Klausuren, Schüler
  - WHERE Jahrgang = 1998
  - AND Schüler.ID = Klausuren.SID
  - GROUPBY Name, Vname
  - HAVING Schnitt > 4.0
- Was macht diese Anfrage?