Vorlesungs-Übersicht

1) Einführung und Definitionen
2) Architektur eines Data-Warehouse-Systems
3) Das multidimensionale Datenmodell
4) ETL: Extraktion, Transformation, Laden
5) Anfrageverarbeitung und -optimierung
6) Indexstrukturen für das multidimensionale Datenmodell
7) Materialisierte Views
8) Metadaten
9) OLAP, Data Mining, Process Mining
10) Zusammenfassung und Ausblick

Vorlesung Data-Warehouse-Systeme im Wintersemester 2008/09

Kapitel 7

- Materialisierte Views -

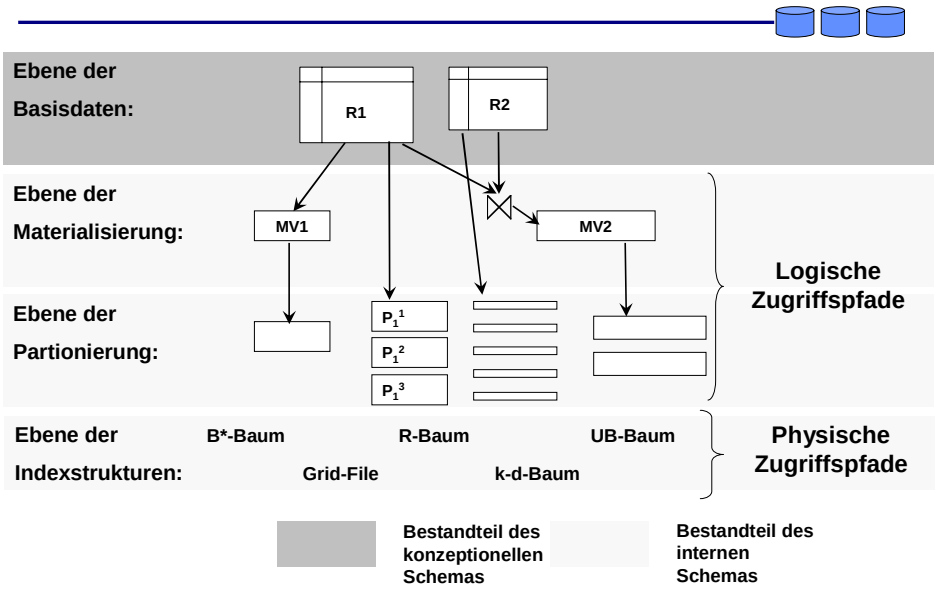
Kapitel 7: Überblick

7.1 Motivation und Einordnung: Optimierungsmöglichkeiten

DWH-Anfragen zum Teil sehr komplex → Optimierung

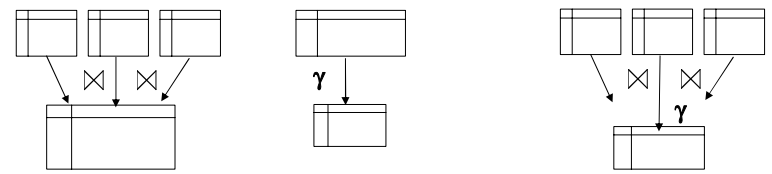
- ❒ (logische) Anfrageoptimierung → Kapitel 5
- ❒ Optimierung von Star-Queries → Kapitel 5
- ❒ Partitionierung → Kapitel 5
- ❒ Optimierung komplexer Gruppierungsfunktionen → Kapitel 5
- ❒ Optimierung durch Einsatz von Indexstrukturen → Kapitel 6
- ❒ Materialisierte Views → Kapitel 7

7.1 Motivation und Einordnung (1)



7.1 Motivation und Einordnung (2)

- Erhaltung der Redundanzfreiheit: Partitionierung
  - Aufteilung eines (logischen) Datenbankobjekts in einzelne Partitionen
  - Und zwar abhängig vom konkreten Anwendungsszenario
- Explizite Verletzung der Redundanzfreiheit: Materialisierung
  - Erkennung der Bestandteile häufiger Datenbankanfragen
  - Einmalige Vorberechnung und Abspeichern des Ergebnisses
  - Verwendung der Materialisierungen anstelle der Originaldaten
  - Verschiedene Realisierungsmöglichkeiten
    - Materialisierung von Verbundoperationen (*materialized join view*)
    - Materialisierung von Aggregrationsoperationen (*materialized aggregate view*)
    - Materialisierung von Verbund- und Aggregationsoperationen (*materialized aggregate-join view*)



a) materialized join view *b) materialized aggregate view* *c) materialized aggregate-join view*

Kapitel 7: Überblick

7.2 Partitionierung (1)

- Eine Partitionierung von Datenbeständen bietet unter Umständen enorme Optimierungsmöglichkeiten:
  - Verwaltung sehr großer Relationen:
    - Im Data-Warehouse-Betrieb werden (ständig) neue Daten eingefügt und müssen (über einen gewissen Zeitraum) historisch verfügbar sein.
    - Klassische Einfüge- und Löschoperationen auf großen Tabellen häufig ineffizient → Hinzunahme und Wegnahme von Partitionen dagegen effizient realisierbar (über DDL-Anweisungen)
  - Überspringen von Partitionen bei Anfrageauswertung:
    - Restriktionen können gegen Partionierungskriterien ausgewertet werden → Reduktion der auszuwertenden Partitionen
  - Ausnutzung paralleler Datenbank- und Systemarchitekturen:
    - Systemtechnische Optimierungen
    - Ermöglicht spezielle Joinoperationen (z.B. parallel hash join)

7.2 Partitionierung (2)

- Zentrale Frage: welche Partitionierungsstrategie?
- Wir haben in Kapitel 5 bereits die horizontale oder Bereichspartitionierung kennen gelernt:
- Zum Beispiel Partitionierung der Faktentabelle nach ProduktID (Annahme: ProduktID $\in$ [100; 10000]:
  - Verkauf = Verkauf1 $\cup$ Verkauf2 $\cup$ Verkauf3 mit
  - Verkauf1:= $\sigma_{100 \leq \text{ProduktID} \leq 2000}$ (Verkauf)
  - Verkauf2:= $\sigma_{2001 \leq \text{ProduktID} \leq 5000}$ (Verkauf)
  - Verkauf3:= $\sigma_{5001 \leq \text{ProduktID} \leq 10000}$ (Verkauf)
- In Oracle-SQL-Dialekt:

```
CREATE TABLE Verkauf (…,
PARTITION BY RANGE(ProduktID)
(PARTITION ProduktID2000 VALUES LESS THAN `2000`),
(PARTITION ProduktID5000 VALUES LESS THAN `5000`),
(PARTITION ProduktID10000 VALUES LESS THAN `10000`));
```


7.2 Partitionierung (3)

- Hash-Partitionierung:
  - Verteilung basiert hier auf (linearer) Hash-Funktion (und nicht auf inhaltlichen Kriterien!)
  - Häufig: *h(x) := x mod p* mit p entspricht Anzahl der Partitionen
  - In Oracle-SQL-Dialekt:

```
CREATE TABLE Verkauf (…,
PARTITION BY HASH(ProduktID) PARTITIONS 3);
```

  - Verkauf wird entlang ProduktID auf nicht explizit benannte Partitionen verteilt
- Kombinierte Bereichs- und Hash-Partitionierung:
  - Erster Schritt: Bereichspartitionierung
  - Zweiter Schritt: Verfeinerung durch Hash-Partitionierung (*sub partitions*)

```
CREATE TABLE Verkauf (…,
PARTITION BY RANGE(ProduktID)
   SUBPARTITION BY HASH (ZeitID)…
```

7.2 Partitionen – Operationen

- Hinzufügen:
  - Erweitern einer Tabelle um einen logischen Speicherbereich

```
ALTER TABLE VERKAUF
ADD PARTITION ProdID12000 VALUES LESS THAN `12000`,
```

- Löschen:
  - Vergleichbar mit DROP TABLE Anweisung

```
ALTER TABLE Verkauf
DROP PARTITION ProdID2000;
```

- Austausch:
  - Partition wird gegen Relation mit identischer Struktur ausgetauscht
  - Beispiel: in Relation VerkaufNeu werden Verkaufsfakten für neue Produkte mit ProduktID zwischen 10000 und 12000 eingefügt und aufbereitet
  - Dann Einspielen in Verkauf durch Austausch der vorher angelegten (und meistens leeren) Partition ProdID12000

```
ALTER TABLE Verkauf
EXCHANGE PARTITION ProdID12000 WITH VerkaufNeu
```


Kapitel 7: Überblick

7.3 Verwendung materialisierter Views – Grundlagen

- Eigenschaften von Anfragen in einem DWH:
  - Vielzahl von Anfragen bezieht sich auf nahezu gleiche Menge von Relationen
  - Anfragen ähneln sich oft, meist Aggregationsanfragen
  - überwiegend lesend auf weitgehend stabilem Datenbestand
- Problem:
  - Online Auswertungen auf großer Rohdatenbasis können unter Umständen sehr lange dauern (> 30 min).
- Daraus ergibt sich die Idee der Materialisierung von (aggregierten) Datenbanksichten (Views):
  - Vorberechnungen von Zwischenergebnissen werden einmal erstellt und mehrfach wieder verwendet → Anfrageausführungszeit sinkt deutlich.
  - Bsp.: Vorberechnung der wichtigsten Aggregationsstufen wie Monat, Quartal, Jahr
  - Nachteil der Redundanz:
    - erhöhter Platzverbrauch
    - Konsistenzprobleme

7.3 Verwendung materialisierter Views – Problembereiche

- Verwendung materialisierter Views:
  - Welche Views sind zur Anfragebearbeitung zu bilden?
  - Existenz materialisierter Sichten darf Anfrageformulierung nicht beeinflussen → Gewährleistung einer transparenten Nutzung durch das System
  - Ableitbarkeit durch kostenbasierte Konstruktion einer Ersetzung
- Auswahl materialisierter Views:
  - Abstimmung zwischen Reduktion der Anfragelaufzeit und zusätzlichem Speicherplatz nötig (Optimum nicht immer bestimmbar).
  - statische Views = materialisierte Views
  - dynamische Views: werden neu berechnet und bis zur Verdrängung im Cache gehalten.
- Wartung (Aktualisierung) materialisierter Views:
  - Änderungen im Datenbestand erfordern Neuberechnung der materialisierten Sicht oder Update-Propagation.
  - effiziente Synchronisation der betroffenen Views durch inkrementelle Aktualisierung
  - Aktualisierungszeitpunkt?
  - eventuell Konsistenzprobleme zwischen berechneten Daten und materialisierten Views!

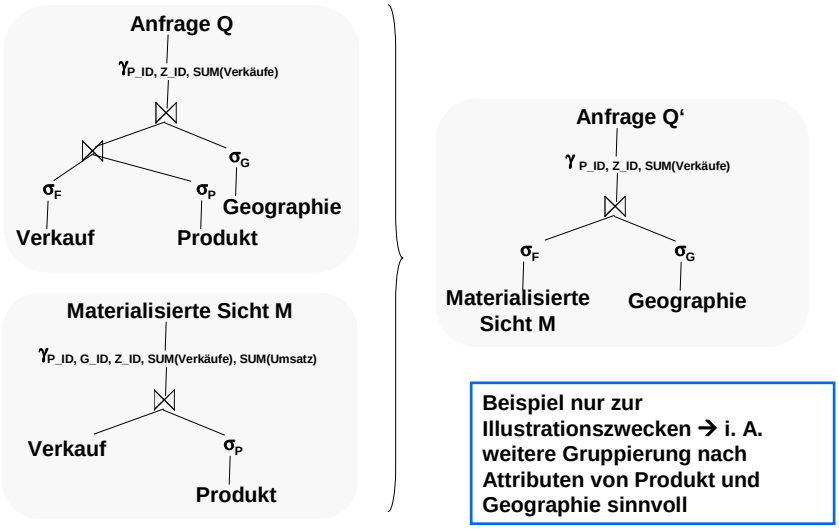
7.3 Verwendung materialisierter Views (1)

Ziel:

- Konstruktion einer *Ersetzung* der Anfrage unter Verwendung von Views (→ *query rewriting*), damit transparente Nutzung der materialisierten Views gewährleistet ist.
- Kostenbasierte Berechnung, ob Query schneller direkt oder unter Verwendung materialisierter Views ausgewertet werden kann.
- Wenn die View nicht genau einem Teilgraph des Operatorbaums entspricht, sind unter Umständen Kompensationsoperationen nötig (Was ist dann billiger / schneller?).
- Wie schon häufiger erkannt: Vermeidung des Zugriffs auf die sehr große Faktentabelle (z.B. „Verkauf“) ist günstig.

7.3 Verwendung materialisierter Views (2)

Unterstützung von Monoblock-Anfragen:



Beispiel nur zur Illustrationszwecken → i. A. weitere Gruppierung nach Attributen von Produkt und Geographie sinnvoll

7.3 Verwendung materialisierter Views (3)

- FRAGE: Wann ist eine solche Query-Restrukturierung zulässig?
- Definition: Eine Anfrage Q‘ ist eine *gültige Ersetzung* der Anfrage Q unter Verwendung der Materialisierung M, wenn Q und Q‘ das gleiche Multimengenergebnis liefern.
- Im Allgemeinen NP-vollständig, es existieren jedoch effiziente Lösungsalgorithmen für alle relevanten Spezialfälle.
- Im DWH-Bereich interessant:
  - Aggregatanfragen in Form folgender Monoblock-Anfragen (→ star query).

```
SELECT <Gruppierungsattribute>, <AGG(Kenngröße)>
FROM <Faktentabelle>, <Dimensionstabellen>
WHERE <Joinbedingungen> AND <Restriktionsbedingungen>
GROUP BY <Gruppierungsattribute>;
```


7.3 Verwendung materialisierter Views (4)

- allgemeine Restrukturierungstechnik: Verallgemeinerte Projektionen (*generalized projections*) [GuHQ95]
- Ziel: bei Restrukturierung soll von Umformungen / Operatorbaumtransformationen der „klassischen“ relationalen Algebra Gebrauch gemacht werden können → Gruppierungsopertor „kritisch“, da nicht in „klassischer“ relationaler Algebra enthalten.
- Idee: duplikateliminierende Projektion ist äquivalent zu einer Aggregationsoperation über die selbe Attributmenge

```
SELECT Preis, Produktgruppe, Region
FROM R
GROUP BY Preis, Produktgruppe, Region
```

- kann äquivalent ersetzt werden durch

```
SELECT DISTINCT[1] Preis, Produktgruppe, Region
FROM R
```

  Projektion der relationalen Algebra muss nur noch für die Behandlung von Aggregationsfunktionen erweitert werden:

$\pi_{A1,\ldots,An,agg(S)}(R) \equiv$

```
SELECT A1, …, An, agg(S)
FROM R
GROUP BY A1, …, An
```

[1] Distinct-Anweisung führt zur Duplikateliminierung.

7.3 Verwendung materialisierter Views (5)

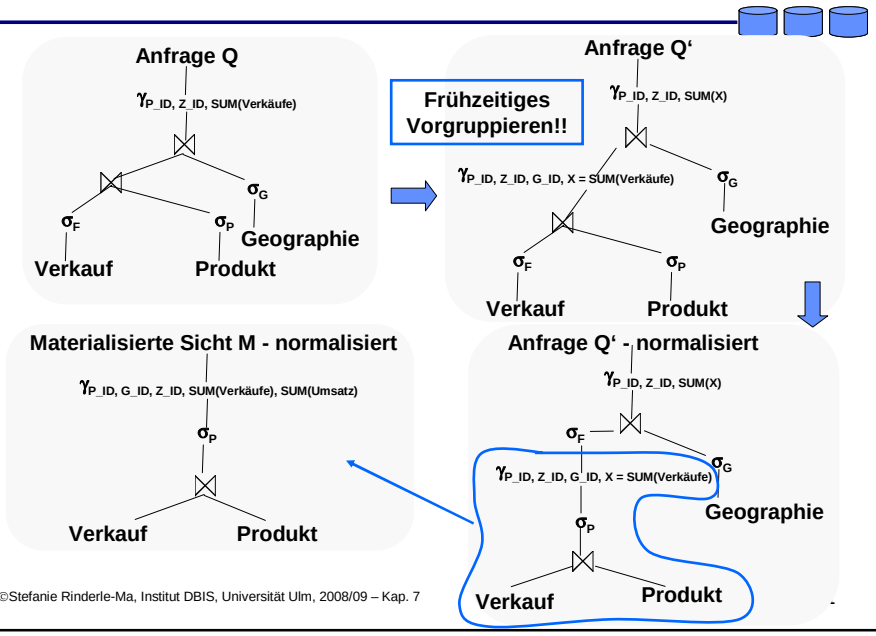
- ferner: Überführung aller Anfrageausdrücke (einschließlich der materialisierten Sichten) in eine Normalform (Definition siehe unten)
- Ziel: paarweiser Vergleich der Anfragegraphen bis eine (maximale) gültige Ersetzung gefunden ist
- Eine Anfrage befindet sich in Normalform, wenn der zugehörige Operatorbaum das folgende Muster aufweist: $\sigma_h \pi \sigma_l \bowtie$ mit:
  - Wurzel des Ausführungsgraphen: Selektion σ_h
  - gefolgt von einer erweiterten Projektion π und einer weiteren Selektion σ_l
  - anschließend folgen alle Verbundoperationen $\bowtie$
  - wobei die Selektionsprädikate h und l in konjunktiver Normalform vorliegen
  - Anwendung von Umformungsregeln für Operatorbäume analog der Optimierung in der relationalen Algebra, etwa *push, pull, split, coalesce* (→ Kapitel 5):
    - *pull* / *push*: Selektionen / Projektionen werden noch oben gezogen bzw. nach unten gedrückt
    - *split* / *coalesce*: verallgemeinerte Projektion wird gesplittet bzw. mehrere verallgemeinerte Projektionen werden zu einer Projektion zusammengefasst
    - Beachte: Aufgrund Duplikatsensitivität bei COUNT und SUM müssen unter Umständen explizite Zählvariablen eingeführt werden (→ Kapitel 5)

7.3 Verwendung materialisierter Views (6)

Zurück zu unserem Ausgangsproblem: Ableitbarkeit einer Anfrage

- Voraussetzungen für Ableitbarkeit [GuHQ95]:
- Unter den folgenden Bedingungen kann dann aus einer ursprünglichen Anfrage Q bezüglich einer vorhandenen materialisierten View M eine äquivalente Anfrage Q' gebildet werden:
  - die Selektionsbedingungen in M dürfen nicht restriktiver als in Q sein
  - die Gruppierungsattribute von π(Q) müssen eine Teilmenge der Gruppierungsattribute von π(M) sein
  - die Aggregationsfunktionen von π(Q) müssen aus den Aggregationsfunktionen von π(M) ableitbar sein,
  - additive Aggregatsfunktion: SUM in Anfrage → SUM in M
  - semi-additiv berechenbare Aggregatsfunktion: AVG in Anfrage → SUM und COUNT in M
  - zusätzliche Selektionsbedingungen in Q müssen auf die materialisierte Sicht M anwendbar sein.

7.3 Verwendung materialisierter Views (7)

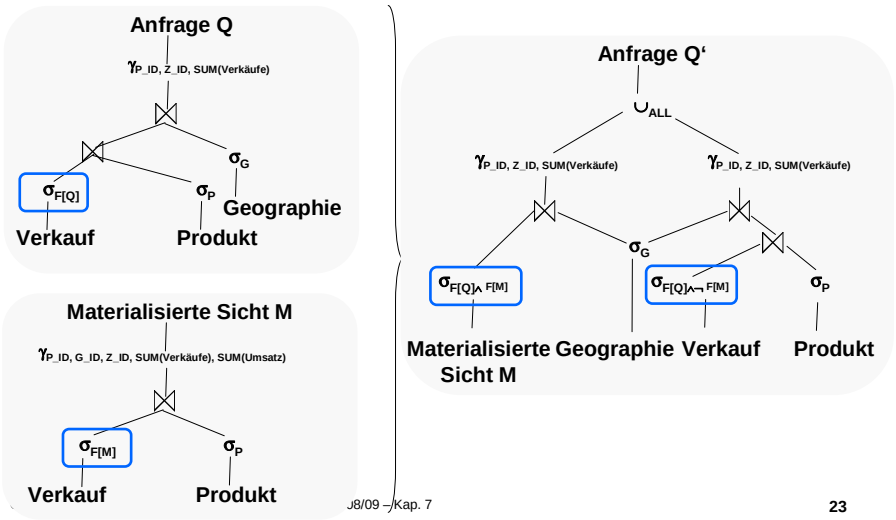


7.3 Verwendung materialisierter Views (8)

- Problem: Bisherige Beschränkung der Verwendung materialisierter Sichten (unter Ausnützung der verallgemeinerten Projektionen) durch die Forderung, dass die Anfrage restriktiver sein muss als die materialisierte Sicht
- Lösung (Unterstützung von Multiblockanfragen):
  - Durch geeignete Aufteilung der Selektionsbedingung einer Anfrage ist es möglich, eine materialisierte Sicht für einen Teil der Anfrage zu verwenden.
  - Zur Berechnung der nicht durch die Sicht abgedeckten Teile wird auf die Detaildaten zugegriffen [SDJL96]:
  - Aufteilung der Anfrage Q hinsichtlich der Selektionskriterien in zwei Teilanfragen Q_a und Q_b derart dass: $\sigma(Q_a) = (\sigma(Q) \wedge \sigma(M))$ und $\sigma(Q_b) = (\sigma(Q) \wedge \neg \sigma(M))$
  - Auf der Teilanfrage Q_a kann dann ein Restrukturierungsalgorithmus zur Verwendung einer Monoblockanfrage ausgeführt werden
  - Die Ergebnisse der Teilanfragen werden nun über die Duplikat erhaltende Vereinigungsoperation „UNION ALL“ zum Endergebnis zusammengefasst
  - Voraussetzungen:
    - Erfüllbarkeit von $\sigma(Q_a)$
    - durch die materialisierte Sicht nicht abgedeckte Selektionen müssen sich auf Attribute der Restanfrage oder auf Attribute der Selektionsklausel der Sicht beziehen
  - durch iterative Anwendung des Verfahrens kann, unabhängig von der Reihenfolge der Zerlegung, eine äquivalente restrukturierte Anfrage generiert werden

7.3 Verwendung materialisierter Views (9)

Unterstützung von Multiblock-Anfragen:



7.3 Verwendung materialisierter Views (10)

- Technik für Multiblock-Anfragen erlaubt die Verwendung von materialisierten Views, deren Restriktionsbedingungen stärker sind als die der gestellten Anfrage.
- Dies erfordert zusätzlichen Anfrageteil, in dem die in der materialisierten Sicht fehlenden Fakten berücksichtigt werden!
- Bei ungünstigen Restriktionsbedingungen kann der durch diesen zusätzlichen Anfrageteil entstehende Aufwand insgesamt auch größer werden als bei Nichtverwendung der materialisierten Sicht (→ Antwortzeit!)

Kapitel 7: Überblick

7.4 Auswertungskontext für Aggregatanfragen (1)

- Problem der Viewauswahl:
  - Menge von Anfragen $Q = \{Q_1, ..., Q_n\}$ mit Anfragehäufigkeiten $\{f_1, ..., f_n\}$ und Speicherplatz der Größe S
  - Die optimale Menge materialisierter Sichten $V = \{ V_1, ..., V_k \}$ ist gesucht, bei der:
    - Die Kosten der Anfrage und Aktualisierung minimal sind
    - Die Speicherbegrenzung S nicht überschritten wird
- führt zu Überlegungen bezüglich:
  - der Gruppierungskombinationen
  - der Einschränkung der Sicht
  - statischer oder dynamischen Auswahl

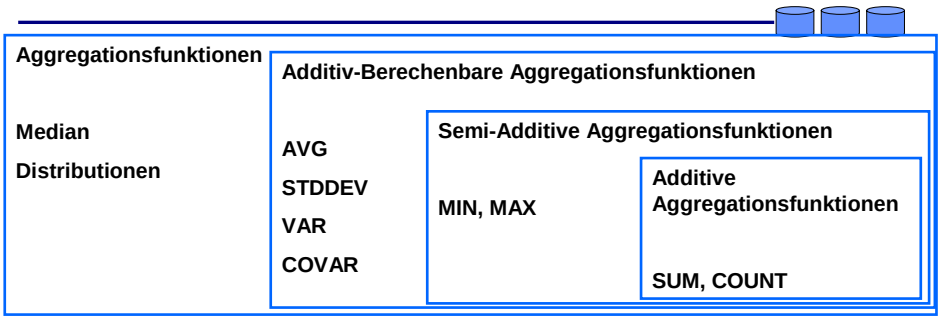
7.4 Auswertungskontext für Aggregatanfragen (2)

- Für die Auswahl von materialisierten Sichten benötigen wir den jeweiligen Auswertekontext für Anfragen mit Aggregationsfunktionen.
- Einführung eines so genannten Aggregationsgitters, welches Ableitungsbeziehungen zwischen Aggregationen in Form eines azyklischen Graphen darstellt (vergleiche Auswertung des CUBE-Operators in Kapitel 5).
- Grund: Jeder Knoten im Aggregationsgitter entspricht einer Möglichkeit, eine materialisierte Sicht zu bilden!
- Mittels Aggregationsgitters erfolgt eine Berücksichtigung der Menge der Gruppierungsattribute, da diese eine Partitionierung des Datenbestands festlegen.
- Frage: Welche Aggregate können aus welchen anderen Aggregaten abgeleitet werden?

7.4 Auswertungskontext für Aggregatanfragen (3)

- Voraussetzung für Wiederverwendung von Vorberechnungen (Ableitbarkeit) ist die Additivität der jeweiligen Aggregationsfunktion.
- Klassifikation von Aggregationsfunktionen [Lehn98]:
  - Additive Aggregationsfunktionen: Aggregationsfunktion F() ist additiv, wenn sich bzgl. der Kennzahlen des Datenwürfels eine kommutative Gruppe bildet, also (1) $F(x_1 \cup x_2) = F(\{F(x_1), F(x_2)\})$ gilt und (2) inverses Element $F^{-1}()$ existiert.
    - Beispiel Summation: $SUM(\{1,2\} \cup \{3,4\}) = SUM(\{SUM(\{1,2\}), SUM(\{3,4\})\}) = 7$ und $SUM(\{1,2\}) = SUM(\{1,2,3,4\}) – SUM(\{3,4\}) = 3$
  - Semi-Additive Aggregationsfunktion: Es gilt nur (1) $F(x_1 \cup x_2) = F(\{F(x_1), F(x_2)\})$, d.h. es existiert kein inverses Element.
    - Beispiel Minimum: $MIN(\{1,2\} \cup \{3,4\}) = MIN(\{MIN(\{1,2\}), MIN(\{3,4\})\}) = 1$
  - Additiv-Berechenbare Aggregationsfunktionen: Bedingung (1) gilt nur in abgeschwächter Form, d.h. (1‘) $\exists$ G() über additive oder semi-additive Aggregationsfunktionen $F_1(), \ldots, F_n()$, so dass $F(x) = G(F_1(x), \ldots, F_n())$
    - Beispiel Durchschnitt: AVG(x) = SUM(x)/COUNT(X)
  - Weitere Aggregationsfunktionen (z.B. Median)

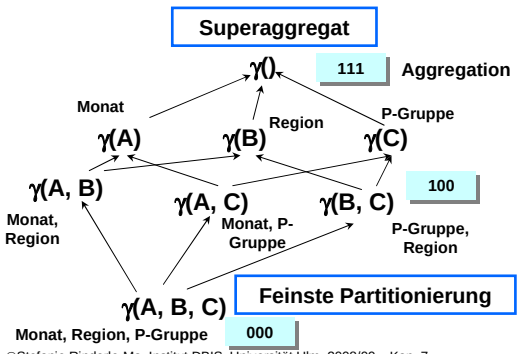
7.4 Auswertungskontext für Aggregatanfragen (4)

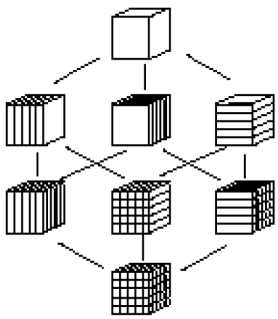


- „Tauglichkeit“ für Ableitungen gegeben bei:
  - Additiven Aggregationsfunktionen→ uneingeschränkt
  - Semi-additiven Aggregationsfunktionen → Ableitung von Anfrageergebnisse, jedoch Probleme bei Aktualisierungen; hier nur für Einfügungen anwendbar, durch Fehlen des inversen Elements keine Löschungen inkrementell wartbar
- Immer beachten: Kompatibilität der Gruppierungsattribute!

7.4 Auswertungskontext für Aggregatanfragen (5)

- Aggregationsgitter für funktional unabhängige Gruppierungsattribute (vergleiche Kapitel 5):
  - wird von einer Menge von funktional unabhängigen Gruppierungsattributen aufgespannt (hier 3 Gruppierungsattribute A, B, C, z.B. Monat, Region, Produktgruppe)





7.4 Auswertungskontext für Aggregatanfragen (6)

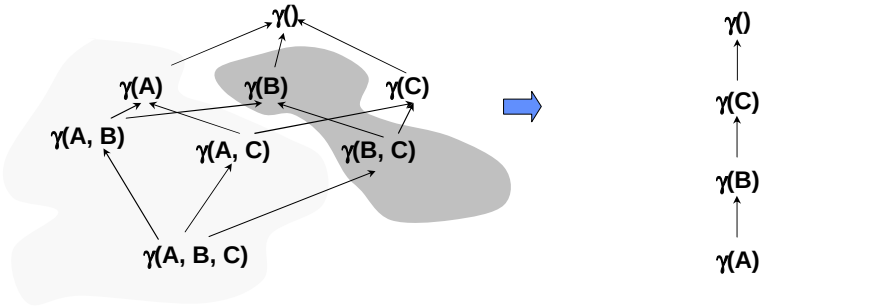
- Die Partitionierung des Datenbestandes wird durch die Menge der Gruppierungsattribute festgelegt und ist bei der Auswahl von materialisierten Sichten ebenfalls zu berücksichtigen.
- Das Aggregationsgitter zeigt an, welche Kombinationen von Gruppierungsattributen entweder direkt oder indirekt von anderen Kombinationen ableitbar sind.
- Das oberste Element wird als Superaggregat bezeichnet und entspricht einer Aggregation über alle eingehenden Einzelwerte, d.h. keiner Gruppierung.
- Die Pfeile geben an, welche Aggregationen bzw. Gruppierungen aus welchen anderen berechnet werden können, z.B. Gruppierung nach B kann aus der Gruppierungen nach (A, B), (B, C), oder (A, B, C) berechnet werden.
- Die Zahl der Knoten im Aggregationsgitter wächst exponentiell mit der Zahl der Gruppierungsattribute (Dimensionen), d.h. bei n Gruppierungsattributen ergeben sich 2^n Knoten im Aggregrationsgitter (also mögliche Gruppierungen)

7.4 Auswertungskontext für Aggregatanfragen (7)

- Bereits bekannt: Jeder Knoten im Aggregationsgitter entspricht einer Möglichkeit, eine materialisierte Sicht zu bilden.
- Aufgrund der exponentiellen Anzahl an potentiell materialisierbaren Sichten muss eine Auswahl getroffen werden (aus Speicherplatz- sowie Aufwandsgründen).
- Beispiel: für 16 Gruppierungsattribute ergeben sich 65534 Knoten im Aggregationsgitter → vollständige Materialisierung ist nicht mehr möglich!
- Ausweg: Die funktionalen Abhängigkeiten zwischen Gruppierungsattributen (bestehen z.B. zwischen den verschiedenen Ebenen einer Klassifikationshierarchie) erlauben eine Reduktion des Aggregationsgitters wie folgt:
  - wenn A → B gilt, repräsentieren die Knoten (A) und (A, B) die selbe Gruppierung (bzgl. Feinheit der Partitionierung)
  - Beispiel: Artikel → Produktgruppe ⇒ Gruppierung nach Artikel γ(Artikel) ist äquivalent zu γ(Artikel, Produktgruppe)

7.4 Auswertungskontext für Aggregatanfragen (8)

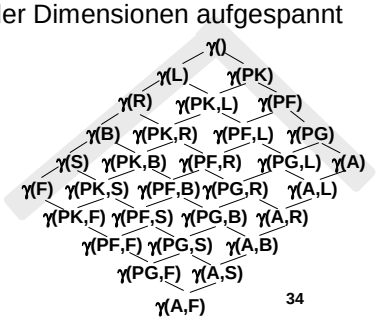
Gegeben: Funktionale Abhängigkeiten: A → B → C



- Alle identisch schattierten Kombinationen weisen die gleiche Partitionierung auf dem Datenbestand auf.
- Die „Feinheit“ der Gruppierung wird durch die Attribute A, B und C bestimmt.
- Unter Beachtung der funktionalen Abhängigkeiten reduziert sich das vollständige Aggregationsgitter somit von ursprünglich 8 auf 4 Knoten.
- Eine Erweiterung der Reduktion auf mehrere Dimensionen ist möglich, indem alle Kombinationen eliminiert werden, die abhängige Klassifikationsstufen beinhalten.

7.4 Auswertungskontext für Aggregatanfragen (9)

- Für DWH interessant: Kombination aus Dimensionen mit jeweiligen Klassifikationsstufen, die funktional unabhängig sind, z.B. Dimensionen Geographie und Produkt im Verkaufswürfel.
- Die Aggregationsgitter für die einzelnen Dimensionen sind jeweils durch den Teilgraphen repräsentiert, in dem keine Klassifikationsstufen der anderen Dimension in den Knoten auftreten (obere Kanten).
- Durch die Kombination der Klassifikationsschemata wird das Kreuzprodukt der Klassifikationsstufen beider Dimensionen aufgespannt
- Die Ableitbarkeitsbeziehung, die durch das Aggregationsgitter nach der Reduktion ausgedrückt wird, stellt genau die Relation „feiner als“ auf den Granularitäten dar.
- Reduktion von $2^{(4+5)} = 512$ auf Knoten (4+1)*(5+1) = 30 Knoten



7.4 Auswertungskontext für Aggregatanfragen (10)

- Zur Abschätzung des Aufwandes einer Materialisierung für eine bestimmte Gruppierungskombination ist die Ausprägung eines Aggregationsgitters bzw. einzelner Knoten von Interesse.
- Einflussfaktoren bzgl. des speicherplatzmäßigen Mehraufwandes:
  - Form des Aggregationsgitters durch Berücksichtigung funktionaler Abhängigkeiten innerhalb der Gruppierungsattribute,
  - Verdichtungsgrad, d.h. Anzahl unterschiedlicher Ausprägungen geteilt durch die Anzahl aller vorhandenen Ausprägungen für jedes Gruppierungsattribut.
  - Dünnbesetztheit des Datenbestandes (*sparsity factor*): Dieser prozentuale Faktor gibt an, welcher Anteil an möglichen Ausprägungen von Attributkombinationen tatsächlich existiert.
- Annahme: statistische Gleichverteilung des zugrunde liegenden Datenbestandes [SDNR96] → Abschätzung des Mehraufwandes durch folgende Funktion: $f(n) = n*(1 – (1- 1/n)^k) = n – n*(1 – 1/n)^k$
  - Wobei n der Anzahl potentieller Aggregate und k dem tatsächlichen Detail-Datenvolumen entsprechen.

Kapitel 7: Überblick

7.5 Statische Auswahl materialisierter Sichten (1)

- Problem: Auswahl einer möglichst optimalen Menge an zu materialisierenden Sichten.
- Vorgehen:
  - Statische Auswahl der Menge zu einem bestimmten Zeitpunkt durch einen Datenbankadministrator oder durch einen Algorithmus
  - Keine Veränderung der Menge materialisierter Sichten bis zur nächsten Aktualisierung des Data Warehouses
  - Keine Berücksichtigung des aktuellen Anfrageverhaltens bei der Auswahl (evtl. Berücksichtigung historischer Daten)
- Ein Auswahlverfahren basiert auf Aussagen über Verhältnis von Mehraufwand an Speicherplatz und Reduzierung von Anfragelaufzeiten.
- Deshalb im Folgenden Ausführungen zu Aufwand und Nutzen materialisierter Sichten mit Aggregationen.
- Im Allgemeinen sind diese Aussagen von vielen Faktoren (die sehr unterschiedlich sein können) abhängig, z.B. Indexstrukturen → deshalb Einführung einer monotonen Kostenfunktion als „gemeinsamer“ Nenner

7.5 Statische Auswahl materialisierter Sichten (2)

- Einführung eines linearen Kostenmodells
- Definition (*Kostenfunktion):* Eine Funktion $c_q(n)$ heißt *Kostenfunktion*, wenn sie die Kosten zur Berechnung der Anfrage q aus dem zu n korrespondierenden Gitterpunkt im Aggregationsgitter liefert, falls q aus der zu n gehörenden Aggregationskombination berechenbar ist. Andernfalls liefert $c_q(n) = \infty$.
- Definition (*Monotonie einer Kostenfunktion):* Eine Anfragekostenfunktion $c_q(n)$ ist monoton, wenn für zwei beliebige Knoten $n_i, n_j \in N$ aus einem Aggregationsgitter gilt:
  $|n_i| < |n_j| \Rightarrow c_q(n_i) < c_q(n_j)$,
  wobei $|n_k|$ = Anzahl Tupel im Gitterpunkt n_k

7.5 Statische Auswahl materialisierter Sichten (3)

Kosten einer Materialisierungskonfiguration:

- Aufwand zur Beantwortung von Anfrage q auf Menge M = $\{n_1, \ldots, n_k\}$ von Gitterpunkten in einem Aggregationsgitter: $c_q(M) = min_{ni \in M} (c_q(n_i))$
- Gesamtanfragekosten für eine Menge von Anfragen Q = $\{q_1, \ldots, q_m\}$, welche mit Häufigkeit $f(q_i)$ gestellt werden basierend auf M: $C(Q,M) = \Sigma_{qi \in Q} f(q_i) * c_{qi}(M)$
- Gesamtaktualisierungskosten basierend auf lokalen Aktualisierungskosten $u(n_i)$ und Aktualisierungsrate $h(n_i)$: $U(M) = \Sigma_{ni \in M} h(n_i) * u(n_i)$
- Gesamtkosten einer Gruppierungskombination zur Auswertung einer Menge von Anfragen: $C_{Gesamt}(Q, M) = C(Q, M) + U(M)$
- Nutzwert eines Aggregrationsgitterknotens:
  - M: Menge der bereits materialisierten Knoten, Q: Menge von Anfragen. Dann ergibt sich der Nutzwert eines zusätzlich materialisierten Gitterpunktes n durch:

$$B_Q(n, M) = \begin{cases} C_{Gesamt}(Q, M) - C_{Gesamt}(Q, M \cup \{n\}) \text{ falls } C_{Gesamt}(Q, M \cup \{n\}) < C_{Gesamt}(Q, M) \\ 0 \text{ sonst} \end{cases}$$

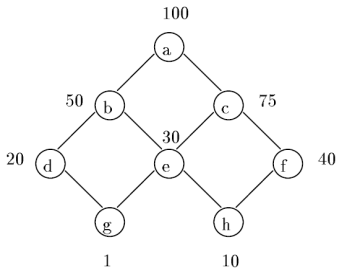
7.5 Statische Auswahl materialisierter Sichten (4)

Auswahlverfahren [HaRU96]: wegen der NP-Vollständigkeit wird nach Greedy-Prinzip gearbeitet

- vorgegeben: Aggregationsgitter und maximaler Speicherplatz S
- Selektion einer Menge von zu materialisierenden Knoten mit möglichst großen Nutzen für das Gesamtsystem
- zur Abschätzung des Aufwandes wird ein Verfahren zur Beurteilung des zusätzlich benötigten Mehraufwandes an Speicher für die Materialisierung des Knotens genutzt
- in der Initialisierungsphase Übernahme des am feinsten partitionierten Gitterpunktes um die Auswertbarkeit aller Anfragen sicher zu stellen
- Bestimmung des Gitterpunktes mit dem größten Nutzen für das Gesamtsystem bezüglich der aktuellen Materialisierungskonfiguration M
- Hinzufügen des Gitterpunktes zur Lösungsmenge und Addition des geschätzten Speichermehraufwandes zum bisherigen Aufwand
- Wiederholung des Vorgehens bis der maximale Speichermehraufwand erreicht wird

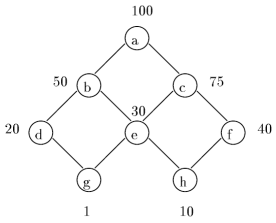
7.5 Statische Auswahl materialisierter Sichten (5)

- Gegeben sei folgendes (abstraktes) Aggregationsgitter (Beispiel in Anlehnung an [HaRU96].
- Jeder Knoten ist mit seiner Kardinalität versehen (Basis: 100 Datenwerte, feinste Partitionierung bei Knoten a)
- Bei Vollauswertung ergeben sich 326 Datensätze, gegenüber 100 ursprünglichen → 226% Speichermehraufwand



7.5 Statische Auswahl materialisierter Sichten (6)

- Beschränkung: S = 110% von 100 → 110 Tupel zusätzlich in Views
- Erster Durchlauf: Knoten b wird ausgewählt, da sich für 5 Knoten (b,d,e,g,h) eine Verbesserung von 50 Tupeln ergibt (S=50%)
- Zweiter Durchlauf: für f selbst ergibt sich Verbesserung von 60 und für h eine Verbesserung von 10 (aus 40 anstatt 50 für b) (S= 90%)
- Dritter Durchlauf: Bei Materialisierung von d ergibt sich für d und g eine Verbesserung von jeweils 30 (S=110%)
- Verfahren bricht ab, Speichermehraufwand erreicht.



	1. It	2. It	3. It	S
b	50*5 = 250			50
c	25*5=125	25*2=50	25*1=25	
d	80*2‘=160	30*2=60	30*2=60	20
e	70*3=210	20*3=60	20+20+10 =50	
f	60*2=120	60+10=70		40
g	99*1=99	49*1=49	49*1=49	
h	90*1=90	40*1=40	30*1=30	

7.5 Statische Auswahl materialisierter Sichten (7)

Bewertung des Verfahrens [HaRU96]

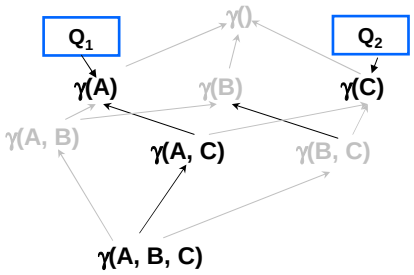
- Die Komplexität des Auswahlalgorithmus ist $O(n^3)$ mit n = Anzahl Gitterelemente.
- Untere Schranke der Güte der Lösung gegenüber der optimalen Lösung beträgt 63%.
- Eine explizite Festlegung einer Anfragemenge ist nicht erlaubt.
- Annahme: Referenzierung jedes Knoten mit der gleichen Wahrscheinlichkeit
- Zentrale Einschränkung des Algorithmus: geringe Skalierbarkeit wegen $O(n^3)$

7.5 Statische Auswahl materialisierter Sichten (8)

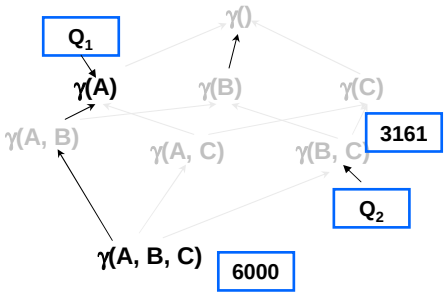
- Deshalb: Optimierung des Verfahrens [BaPT97]
- Ziel: Reduzierung des Aggregationsgitters vor Anwendung des Auswahlverfahrens [HaRU96]
- Anwendung folgender Heuristiken:
  - Reduktion redundanter Gitterknoten aufgrund funktionaler Abhängigkeiten
  - Iterative Konstruktion eines Teilgitters:
    - Für eine vorgegebene Menge von Anfragen werden paarweise alle größten gemeinsamen Vorgänger ermittelt und dem Teilgitter hinzugefügt
    - Eliminierung aller Knoten welche ausprägungsseitig keine Verdichtung um einen vorgegebenen Faktor versprechen
- Die durch die Heuristiken reduzierten Gitter bilden dann den Ausgangspunkt für das vorgestellte Auswahlverfahren.

7.5 Statische Auswahl materialisierter Sichten (9)

- In a) werden (A) und (C) von Q1 und Q2 referenziert → finde größten gemeinsamen Vorgänger (A, C) und nehme ihn in das Gitter auf
- In b) wird bei einer Reduktionsrate von 50% Knoten (B, C) eliminiert.



a) Reduktion durch Referenzierung



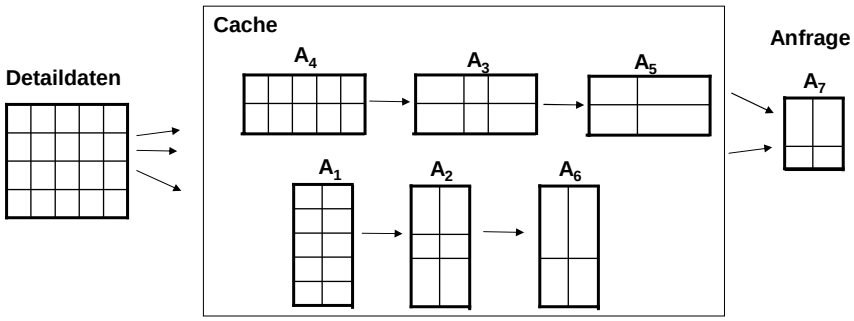
b) Reduktion durch vorgegebenen Faktor

7.5 Dynamische Auswahl materialisierter Sichten (1)

- Die Anwendbarkeit statischer Auswahlverfahren wird durch den Ad-hoc-Charakter von OLAP-Anfragen (können häufig nur schwer vorhergesagt werden) und die hohe Änderungshäufigkeit von Anfragemustern eingeschränkt → Anfragedynamik sollte berücksichtigt werden
- Deshalb: dynamische Verwaltung von materialisierten Sichten durch Speichern von Anfrageergebnissen
- Anfrageergebnisse werden häufig bei nachfolgenden Anfragen (im Rahmen der selben OLAP-Sitzung) verwendet.
- Semantisches Caching: Pufferung von Anfrageergebnissen, wobei entschieden werden muss, ob ein (später) angefordertes Objekt effizient von einem Pufferobjekt ableitbar ist oder nicht.
- Verdrängungsstrategien: Auch für das dynamische Hinzufügen von Speicherseiten gelten Beschränkungen hinsichtlich des Speichermehraufwands.
- Einflussfaktoren: Zeit des letzten Zugriffs, Referenzierungshäufigkeit, Größe, Kosten für Neuberechnung etc.

7.5 Dynamische Auswahl materialisierter Sichten (2)

- Ausgehend von Detaildaten sukzessive Berechnung der Anfragen A_1 bis A_7 (beachte Anfragereihenfolge!)
- Dann muss zunächst A_1 aus den Detaildaten berechnet werden.
- A_2 kann von A_1 abgeleitet werden.
- A_3 muss aus Detaildaten berechnet werden.
- Auch A_4 muss aus Detaildaten berechnet werden, da niedrigere Aggregationsstufe als A3
- A_5 und A_6 können aus Cache beantwortet werden.



Kapitel 7: Überblick

7.6 Aktualisierung materialisierter Sichten (1)

Aktualisierungsalternativen:

- Rematerialisierung:
  - Löschung und Neuberechnung der materialisierten Sicht
  - Nachteil: ineffizient, falls nur geringe Änderung des Datenbestandes
- Inkrementelle Aktualisierung (wird im Weiteren betrachtet):
  - Änderungen der Sicht basierend auf *Änderungen* der der Sicht zugrunde liegenden Basisrelationen
  - Nach Modifikation wie Tupeleinfügung, -löschung, -änderung der Basisrelationen wird V' aus V nach folgender Formel berechnet:

    $V' = V \uplus \Delta V = (V - \Delta^{-}(V)) \cup \Delta^{+}(V)$
  - wobei:
    - V'= neuer Zustand
    - V= alter Zustand
    - ΔV = Änderungen der Sicht V bestehend aus $\Delta^{-}(V)$ = zu löschenden Tupel und $\Delta^{+}(V)$ = hinzuzufügenden Tupel
    - $\uplus$ = additive Vereinigung, zwecks Duplikaterhaltung

7.6 Aktualisierung materialisierter Sichten (2)

Klassifikation von Aktualisierungsalgorithmen:

- Probleme:
  - Teurer Zugriff auf Basisrelation
  - Konsistenz (bei verzögerten Updates)
- Zeitpunkt der Aktualisierung:
  - Sofortige Aktualisierung: synchrone Transaktion für abgeleitete Daten (→ immer konsistente Sichten mit den Basisrelationen, aber teuer)
  - Verzögerte Aktualisierung: erst beim ersten Lesezugriff (→ Verlagerung der Kosten, Wartezeit)
  - Snapshot-Aktualisierung: Wenn Zugriff auf veraltete Daten tolerierbar ist (→ snapshot vergleiche auch Kapitel 4)

7.6 Aktualisierung materialisierter Sichten (3)

Wie kann der Zugriff auf die Basisrelationen realisiert werden?

- Die meisten Algorithmen verwenden zur Berechnung von ΔV
  - Definition der Sicht
  - Änderungen der Basisrelation
  - Basisrelation selbst
- negative Eigenschaft:
  - Performanzverluste, da Zugriff auf Ausprägung der Basisrelationen
- Behebungsalternativen:
  - Reduktion der Zugriffe auf Basisrelationen
- Hilfsmittel zur Erreichung (Zusatzinformationen: „je mehr desto besser“):
  - Schemainformationen
  - Counting-Algorithmus
  - Hilfs-Sichten
- Aktualisierung ohne jeglichen Basisrelationenzugriff :
  - Z.B. synchrone Aufsammlung der Änderungen in Delta-Relationen
  - Verwendung der Log-Dateien, etc.
- Außerdem wichtig: Sperrzeiten für Aktualisierungen minimieren → *propagate&refresh*-Verfahren: Aktualisierungen werden vorberechnet (so wie möglich) und dann nur noch in die Sichten eingebracht.

7.6 Aktualisierung materialisierter Sichten (5)

Konsistenz in Data-Warehouse-Systemen:

- Anwenderdefinierte Aktualitätsanforderungen:
  - Anforderung bezüglich der Aktualität von materialisierten Sichten in Bezug auf die zugrunde liegenden Basisrelationen
  - Abstandsmaße zur Anforderung:
    - zeitliche: z.B. 1 Tag, 1 Woche usw.
    - wertemäßige: z.B. 10 Einheiten, 2% Abweichungstoleranz
    - versionsbezogene: z.B. Versionsabstand von Basisrelation zu materialisierter Sicht = 2
- Anfragekonsistenz (Sichtenkonsistenz):
  - Konsistenz zwischen einer materialisierten Sicht und den Objekten, aus der sie abgeleitet wird (z.B. aus Basisrelationen oder materialisierte Sichten)
  - Konsequenz: Alle an einer Anfrage beteiligten Objekte müssen auf dem gleichen Aktualitätsniveau gehalten werden.
- Sitzungskonsistenz (Multi-Sichtenkonsistenz):
  - Gleiches Aktualitätsniveau aller Anfragen innerhalb einer Sitzung
  - Problematisch: bei Folge von Drill-Down oder Roll-Up Operationen (d.h. gesamtes DWH mit einer Lesesperre versehen)

7.6 Aktualisierung materialisierter Sichten (6)

Anforderungskatalog:

- Nebenläufige Aktualisierung: Während der Aktualisierung sollte weiterhin lesender Zugriff möglich sein.
- Unterstützung individueller Aktualisierungsstrategien
  - In großen DW: Aktualisierung aufwendiger Prozess.
  - Falls feinere Aktualisierungsgranulate gebildet werden können: Individuelle Aktualisierungszeitpunkte je nach Anforderungen.
- Konsistenzgarantien
  - Minimal: Mindestens innerhalb einer Anfrage müssen alle Daten dem selben Aktualisierungsgrad entsprechen
  - Maximal: Aktualisierungskonsistenz über die ganze Sitzung.

7.6 Aktualisierung materialisierter Sichten (7)

Aktualisierungskonzepte

- Aktualisierung im Ein-Versionenfall:
  - Aktualisierung während bestimmter Zeitfenster (z.B. über Nacht oder am Wochenende)
  - alle materialisierten Sichten werden innerhalb einer Transaktion aktualisiert
  - in kommerziellen Produkten oft verwendet
  - Negative Eigenschaften:
    - keine nebenläufige Aktualisierung
    - wenn Fehler auftritt, reicht Zeitfenster eventuell nicht aus
  - Positiv aus Sicht des Produktherstellers: wesentlich geringere Komplexität der Konsistenzsicherung
  - Spezialfall: Viewgroup-Konzept

7.6 Aktualisierung materialisierter Sichten (7)

Aktualisierungskonzepte

- Aktualisierung im Mehr-Versionenfall:
  - Zum Beispiel der 2VNL-Ansatz (*Two-Version No-Locking*) [QuWi97]:
  - Drei Versionen eines Datenobjektes werden unterschieden, davon existieren maximal zwei gleichzeitig
    - Vorgängerversion
    - aktuelle Version
    - zukünftige Version
  - eine der beiden Versionen kann zum Lesen verwendet werden
  - Aktualisierende (schreibende) Transaktionen erfolgen auf der zukünftigen Version, lesende jedoch immer auf der alten Version
  - Positiv:
    - Lese-/Schreibtransaktionen behindern sich nicht
    - Nebenläufigkeit
  - Negativ:
    - Bei langen Sitzungen (langen Transaktionen) reicht das 2VNL-Verfahren oft nicht aus, da die verwendete Version irgendwann gelöscht wird.
    - → n-VNL-Konzept.

Kapitel 7: Überblick

7.7 Zusammenfassung und weitere Möglichkeiten

- Aufgrund des großen Datenvolumens in DWH ist die „Auswahl“ von Teildatenmengen ein probates Mittel.
- Das Datenvolumen kann mittels Partitionierung in kleinere „Portionen“ zerlegt werden.
- Oder man bildet Sichten auf die Basisdaten.
- Wichtige Punkte hierbei sind:
  - Wie können Sichten geschickt aufeinander aufbauen bzw. voneinander abgeleitet werden, um ständige Neuberechnung auf der einen Seite und unnötiges Vorhalten von Daten auf der anderen Seite zu vermeiden?
  - Welche Sichten sind relevant?
  - Wie kann man materialisierte Sichten warten bzw. aktualisieren?
- Weitere Möglichkeiten:
  - Explizite relationalen Repräsentation von materialisierten Summendaten
    - In zusätzlichen Summentabellen
    - Eingebettet in Faktentabellen

Referenzen

[GuHQ95] A. Gupta, V. Harinarayan, D. Quass: Aggregate Process in Data Warehousing Environments, In Proc. Very Large Databases, S. 358 – 369 (1995)

[SDJL96] D. Srivastava, S. Da, H. Jagadish, A. Levy, Answering Queries with Aggregation Using Views. In Proc. Very Large Databases, S. 318-329 (1996)

[RaRU96] Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman: Implementing Data Cubes Efficiently. SIGMOD Conference 1996: 205-216

[BaPT97] E. Baralis, S. Paraboschi, E. Teniente: Materialized View Selection in a Multidimensional Database. In Proc. Very Large Databases, S. 156-167 (1997)

[QuWi97] On-Line Warehouse View Maintenance. Int‘l SIGMOD Conference, S. 393-404 (1997)