Verwendung materialisierter Views zur Anfrageoptimierung in Data Warehouses

Thomas Nösinger
Lehrstuhl Datenbanken und Informationssysteme
Universität Rostock
thomas.noesinger@uni-rostock.de

1 Einleitung

Ein *Data Warehouse* ist nach [IWIP02] eine themenorientierte, integrierte, persistente und zeitvariante Sammlung von Daten zum Zweck der Analyse und Entscheidungsunterstützung.

In einem Data Warehouse werden *multidimensionale Daten* gespeichert. Im Mittelpunkt eines Data Warehouses stehen *Kennzahlen*, das sind verdichtete Messgrößen, die betriebswirtschaftliche Zusammenhänge beschreiben, wie zum Beispiel: Umsatz, Gewinn und Kosten. Die Kennzahlen im Data Warehouse werden durch verschiedene *Dimensionen* wie Zeit, Ort, Produktart, Verkäufer, usw. beschrieben. Dabei können die unterschiedlichen Dimensionen Klassifikationshierarchien bilden; zwischen diesen bestehen funktionale Abhängigkeiten. Gespeichert werden können solche Daten in einem sogenannten *Star-Schema* (siehe Abbildung 1). Im Zentrum steht dabei die Faktenrelation, beschrieben werden die darin vorkommenden Daten durch die Dimensionsrelationen.

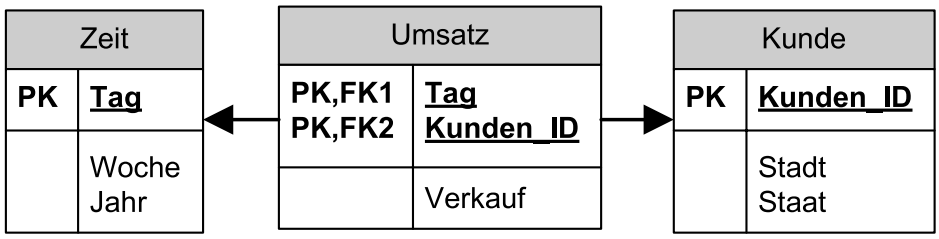

Abbildung 1: Speicherung von Data Warehouse-Daten im Star-Schema

Typische Anfragen an ein Data Warehouse sind *Aggregatfunktionen*, die auf sehr viele gespeicherte Datensätze zugreifen. Aus diesem Grund ist die Auswertung von Data Warehouse-Anfragen zeitaufwendig. Werden bestimmte Aggregationsoperationen mehrfach benötigt, so kann man *materialisierte Views* einsetzen, die Zwischenergebnisse für die Anfragen enthalten. In den materialisierten Views werden voraggregierte Werte gespeichert, dadurch enthalten die Views wesentlich weniger Tupel als die Originalrelationen.

2 Materialisierte Views

Eine materialisierte View ist ein relational gespeicherter Datenwürfel, der das Ergebnis einer multidimensionalen Anfrage an ein Data Warehouse ist. Dabei wird der Datenwürfel des Data Warehouses mit Hilfe des Starschemas relational verwaltet und mittels SQL angefragt. Eine solche Anfrage Q kann in Bezug auf Abbildung 1 wie folgt aussehen:

```
CREATE TABLE AST1 AS
 SELECT Zeit.Tag, Kunde.Stadt,
        SUM(Umsatz.Verkauf) as Verkauf
 FROM Zeit, Kunde, Umsatz
 WHERE Zeit.Tag = Umsatz.Tag AND
       Kunde.Kunden_ID = Umsatz.Kunden_ID
 GROUP BY Zeit.Tag, Kunde.Stadt
```

Das Beispiel illustriert eine Anfrage, die die täglichen Verkäufe in allen Städten aggregiert. Das Ergebnis wird als AST1 (Automatic Summary Table) *dauerhaft als Relation gespeichert* und steht somit zur Optimierung von Aggregatanfragen zur Verfügung. Das heißt, dass falls eine Anfrage Q an das Data Warehouse gestellt wird, die die *täglichen Verkäufe* in den unterschiedlichen Städten selektiert, könnte die View AST1 zur Beantwortung verwendet werden. Es ergibt sich somit die restrukturierte Anfrage Q':

```
SELECT AST1.Tag, AST1.Stadt, AST1.Verkauf
 FROM AST1
 GROUP BY AST1.Tag, AST1.Stadt
```

Unter Ausnutzung von den Klassifikationsstufen und deren funktionalen Abhängigkeiten können ebenfalls Anfragen optimiert werden, deren Hierarchiestufen höher sind als im Vergleich zu AST1. Das heißt falls eine Anfrage P die *jährlichen Verkäufe* selektiert, dann sind die dafür notwendigen Daten in AST1 enthalten und müssten nur entsprechend nachaggregiert werden:

```
SELECT Zeit.Jahr, AST1.Stadt, SUM(AST1.Verkauf)
 FROM AST1, Zeit
 WHERE AST1.Tag = Zeit.Tag
 GROUP BY Zeit.Jahr, AST1.Stadt
```

Visualisiert ergeben die Hierarchiestufen einen Graphen oder Hypercube, mit dem analysiert werden kann, welche Views potentiell zur Beantwortung von Q geeignet sind (siehe Abbildung 2). Dabei werden den unterschiedlichen Hierarchiestufen zur verbesserten Übersichtlichkeit Kombinationen von Zahlen zugeordnet. Diese Kombinationen geben somit Aufschluss darüber, welche Klassifikationsstufen enthalten sind; AST1 besitzt die Kombination (Tag,Stadt) bzw. (0,1).

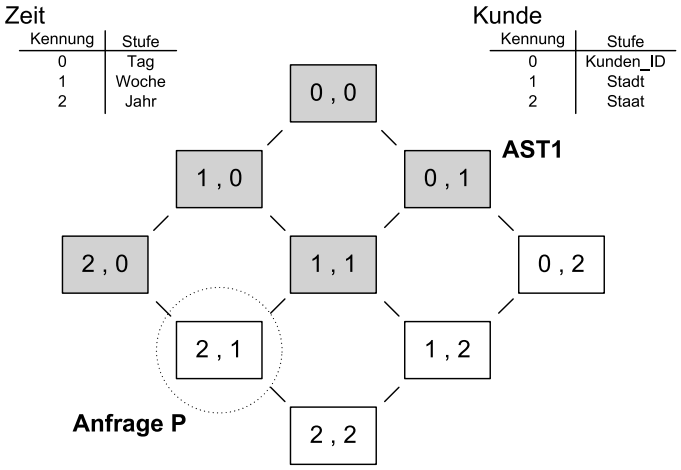

Abbildung 2: Hypercube zur Verwendbarkeit von materialisierten Sichten

Eine Anfrage P mit der Kombination (2,1) könnte unter Verwendung einer View beantwortet werden, falls die Kombination der View ein Vorgänger ist. Die Vorgänger von P sind in Abbildung 2 grau hinterlegt.

3 Anfrageoptimierung mittels materialisierter Views

Damit materialisierte Views in Anfragen verwendet werden können, muss die Information, wie diese Views gebildet wurden, in formaler Weise vorliegen. Dazu werden *Metadaten* verwendet, die für jede View beschreiben, welche

- Klassifikationsstufen,
- Kennzahlen,
- Aggregatfunktionen,
- zusätzliche Relationen,
- Restriktionen und
- Gruppierungsattribute

enthalten sind. Anhand dieser Informationen wird entschieden, ob zum Beispiel alle notwendigen Kennzahlen vorliegen, die Restriktionen einer View die der Anfrage inkludieren oder auch ob die Aggregatfunktionen übereinstimmen. Relationen, welche die Metadaten speichern, wurden dazu entwickelt; einen Überblick über die Relationen gibt Abbildung 3.

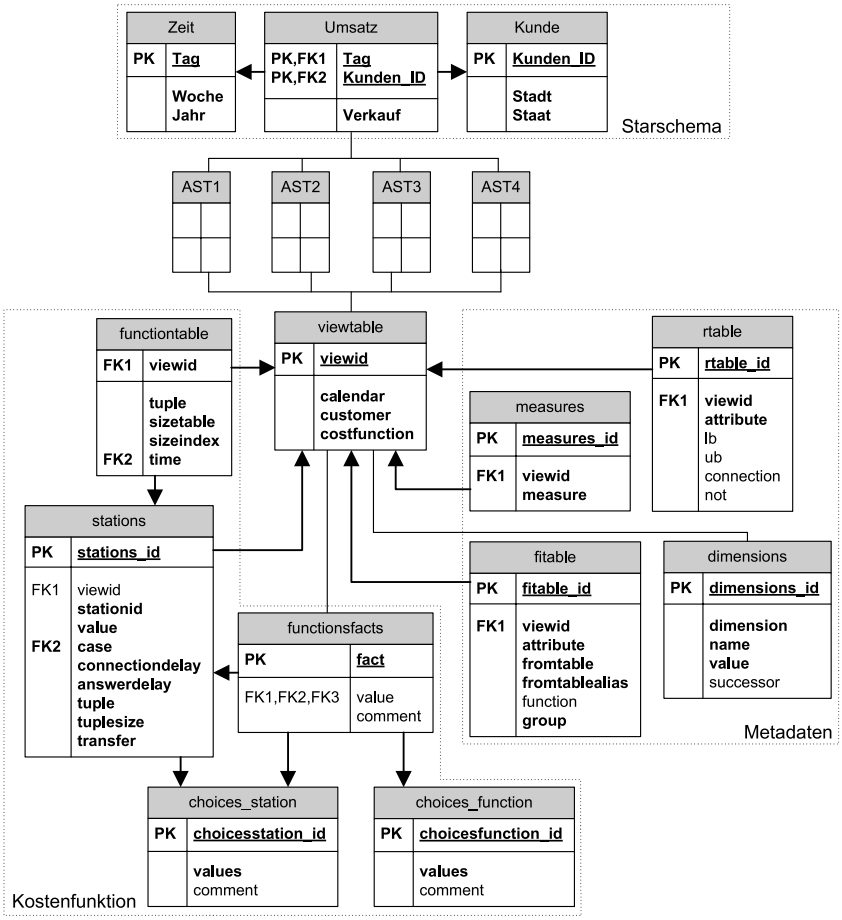

Abbildung 3: Struktur der Metadaten zur Beschreibung materialisierter Views

Es ist möglich, dass eine Menge von Views zur Beantwortung einer Anfrage Q verwendet werden können. In diesem Fall entscheidet eine *Kostenfunktion*, welche View im Vergleich zu

den anderen den geringsten Kostenfunktionswert aufweist. Dieser Wert wird entweder durch die Tupelanzahl, Seitenanzahl (die Summe der notwendigen Daten- und Indexseiten) oder durch die notwendigen Zugriffszeiten auf die entsprechenden Datensätze einer View bestimmt. Die zur Berechnung notwendigen Parameter sind wiederum in Relationen abgespeichert (siehe Abbildung 3).

Abbildung 4 veranschaulicht, wie der Prozess der Optimierung einer Anfrage vereinfacht ablaufen kann.

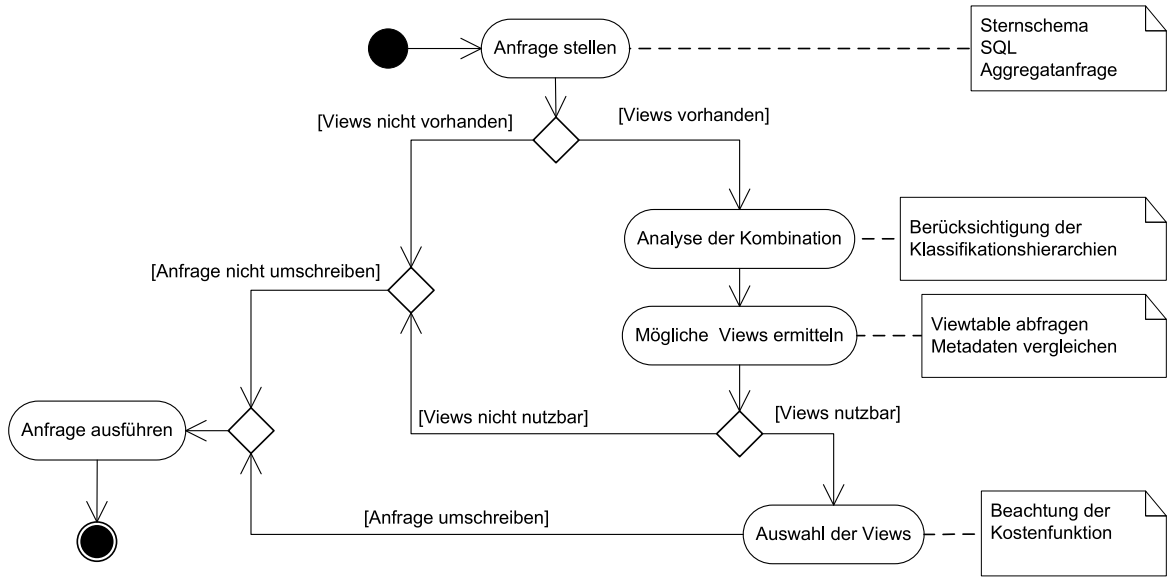

Abbildung 4: Anfragerealisierung unter Verwendung von materialisierten Views

4 Related work

Gupta et al. geben in [GM99] einen Überblick über die Gesamtthematik der materialisierten Views, wobei Ansätze zur Behandlung des in der Literatur bekannten „View Usability Problems“ vorgestellt werden. Eine Möglichkeit ist, dass die Anfragesprachen in deren Komplexität beschränkt werden.

Diesbezüglich behandelt Halevy in [Hal01] Algorithmen für „Conjunctive Queries“ (oder auch „Select-Project-Join Queries“ genannt), welche von Pottinger et al. in [PL00] aufgegriffen und zum MiniCon-Algorithmus erweitert werden. Ein kostenbasierten Ansatz derartiger Anfragen wird von Chaudhuri et al. in [CKPS95] vorgestellt, der mit Hilfe von Ersetzungsregeln einer „MapTable“ „Conjunctive Queries“ umformt.

Eine Erweiterung auf Aggregatanfragen stellen die Konzepte von Gupta et al. in [GHQ95] und Srivastava et al. in [SD96] dar. Es werden graphenbasierten Verfahren vorgestellt, mit denen Aggregatanfragen derartig umgeformt und umgeschrieben werden können, dass materialisierte Views anstelle von Basisrelationen verwendet werden.

Die Autoren von [NT04] beschreiben einen Metadaten-gestützten Ansatz zur Optimierung von Aggregatanfragen. Dabei wird ein Konzept zur Verwaltung von Views in einer zentralen Relation, zum Vergleich von Views untereinander mittels Kostenabschätzungen und zur Ausnutzung funktionaler Abhängigkeiten zwischen Views und Klassifikationsstufen vorgestellt.

5 Zusammenfassung und Weitere Arbeiten

Mit der hier vorgestellten Diplomarbeit sollten die Voraussetzungen geschaffen werden, um eine Anfrageoptimierung unter Verwendung von materialisierten Sichten in ein bestehendes Data-Warehouses-System zu integrieren. Das Data-Warehouse-System wird von der CBS Hanse mit Sitz in Kasnevitz (Insel Rügen) entwickelt. Es unterstützt Kunden in allen Phasen des Data Warehousings, beginnend von der Datenintegration aus Quellsystemen, über die Datenspeicherung, Anfragerealisierung, Datenanalyse und -repräsentation.

In [Nös09] ist detailliert nachzulesen, wie eine Anfrageoptimierung auf Basis materialisierter Sichten erfolgen kann und wie diese Komponente implementiert werden könnte.

In Zukunft wäre es als Erweiterung der bestehenden Umsetzung denkbar, dass nicht nur die Anfrageoptimierung an sich betrachtet wird, sondern dass folgende Funktionalitäten hinzu genommen werden könnten:

- Auswahl und Bestimmung materialisierter Views,
- Entwicklung und Umsetzung weiterer Kostenfunktionen,
- Test des Verfahrens anhand realer Daten.

Literatur

[CKPS95] Chaudhuri, Surajit, Ravi Krishnamurthy, Spyros Potamianos und Kyuseok Shim: *Optimizing queries with materialized views.* Seiten 190–200, 1995.

[GHQ95] Gupta, Ashish, Venky Harinarayan und Dallan Quass: *Aggregate-query processing in data warehousing environments.* In: *In Proceedings of the International Conference on Very Large Databases*, Seiten 358–369, 1995.

[GM99] Gupta, Ashish und Iderpal Singh Mumick (Herausgeber): *Materialized views: techniques, implementations, and applications.* MIT Press, Cambridge, MA, USA, 1999.

[Hal01] Halevy, Alon Y.: *Answering Queries Using Views: A Survey*, 2001.

[IWIP02] Inmon, W. H., John Wiley, W. H. Inmon und Wiley Computer Publishing: *Data Warehouse Third Edition Building the Data Warehouse Third Edition*, 2002.

[Nös09] Nösinger, Thomas: *Anfrageoptimierung in Data Warehouses durch Verwendung voraggregierter Views*, 2009.

[NT04] Nadeau, Thomas P. und Toby J. Teorey: *OLAP Query Optimization in the Presence of Materialized Views*, 2004.

[PL00] Pottinger, Rachel und Alon Levy: *A scalable algorithm for answering queries using views.* In: *In Proc. of VLDB*, Seiten 484–495, 2000.

[SD96] Srivastava, Divesh und Shaul Dar: *Abstract Answering Queries with Aggregation Using Views*, 1996.