Information Systems 34 (2009) 353–370

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Query optimization using restructured views: Theory and experiments

Dongfeng Chen [a], Rada Chirkova [a,*,1], Fereidoon Sadri [b,2]

[a] Computer Science Department, NC State University, Raleigh, NC 27695, United States
[b] Computer Science Department, University of North Carolina at Greensboro, Greensboro, NC 27402, United States

ARTICLE INFO

Article history:
Received 29 March 2007
Received in revised form
31 December 2007
Accepted 1 October 2008
Recommended by: Y. Ioannidis

Keywords:
Query optimization
Restructured views
Query optimization using materialized views

ABSTRACT

We study optimization of relational queries using materialized views, where views may be regular or *restructured*. In a restructured view, some data from the base table(s) are represented as metadata—that is, schema information, such as table and attribute names—or vice versa.

Using restructured views in query optimization opens up a new spectrum of views that were not previously available, and can result in significant additional savings in query-evaluation costs. These savings can be obtained due to a significantly larger set of views to choose from, and may involve reduced table sizes, elimination of self-joins, clustering produced by restructuring, and horizontal partitioning.

In this paper we propose a general query-optimization framework that treats regular and restructured views in a uniform manner and is applicable to SQL select-project-join queries and views without or with aggregation. Within the framework we provide (1) algorithms to determine when a view (regular or restructured) is usable in answering a query and (2) algorithms to rewrite queries using usable views.

Semantic information, such as knowledge of the key of a view, can be used to further optimize a rewritten query. Within our general query-optimization framework, we develop techniques for determining the key of a (regular or restructured) view, and show how this information can be used to further optimize a rewritten query. It is straightforward to integrate all our algorithms and techniques into standard query-optimization algorithms.

Our extensive experimental results illustrate how using restructured views (in addition to regular views) in query optimization can result in a significant reduction in query-processing costs compared to a system that uses only regular views.

1. Introduction

Query optimization using materialized views has become standard practice in commercial database systems in recent years [1]. *Restructuring*, that is, representing the same information in different structures where data in one representation can be represented as metadata in another, has also been studied for relational databases (see, e.g., [2,3]). In this paper we extend query optimization using views to *restructured views*: We study optimization of relational queries using materialized views, where views may be regular or *restructured*. In a restructured view, some data from the base table(s) are represented as metadata—that is, schema information, such as table and attribute names—or vice versa. Often, data from one or multiple tables can be restructured into a

* Corresponding author. Tel.: +1 919 247 3271.
E-mail addresses: dchen3@ncsu.edu (D. Chen), chirkova@csc.ncsu.edu (R. Chirkova), sadri@uncg.edu (F. Sadri).

[1] Supported by NSF grants Career 0447742 and IIS 0307072.
[2] Supported by NSF grant IIS 0083312.

0306-4379/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2008.10.002

new representation with significant reduction in the overall data size. Intuitively, by moving data values that appear frequently and repeatedly in a relation to metadata in the view, we can represent the information in the view more compactly than by using a regular view. This size reduction, together with *physical clustering* that results from the restructuring (to be discussed in Sections 2 and 5), can achieve orders-of-magnitude improvement in query-processing time for certain classes of queries.

Consider, for example, the `salesInfo` relation of Fig. 1, which lists stores and their monthly sales information over a number of years. Fig. 2 shows a restructured view of `salesInfo`. This `salesInfoView` relation represents the same information as `salesInfo`, but the months (`Jan,Feb,...,Dec`) now play the role of attribute names, and the sales values for the months of each year are organized "horizontally" into a single tuple for each store. Note that each tuple of the view `salesInfoView` represents exactly the information represented by 12 tuples of the base table `salesInfo`. Fig. 3 shows another view, `annualSalesView`, which involves aggregation and restructuring. Each tuple of `annualSalesView` represents the complete annual sales values for each store for all years.

Using restructured views in query optimization opens up a new spectrum of views that were not previously available, and can result in significant additional savings in query-evaluation costs. The savings can be obtained due to a significantly larger set of views to choose from, and may involve reduced table sizes, elimination of self-joins, clustering produced by restructuring, and horizontal partitioning.

storeID	year	month	sales
1	2005	Jan	120000
1	2005	Feb	100000
...	...	...	...
1	2005	Dec	150000
2	2005	Jan	300000
...	...	...	...

Fig. 1. Base relation salesInfo.

storeID	year	Jan	Feb	...	Dec
1	2005	120000	100000	...	150000
2	2005	300000	...	...	...

Fig. 2. Restructured view salesInfoView.

storeID	...	2003	2004	2005	...
1	...	1500000	1650000	1770000	...
2	...	3000000	2870000	2800000	...

Fig. 3. Restructured view annualSalesView.

Commercial database systems are adding facilities for the definition and materialization of restructured views [2]. Our goal is to tap into this new capability for query optimization. Our experiments, both in this paper (Section 7) and in previous work [4], show a significant reduction in query-processing costs using materialized restructured views for certain workloads. Our work also formalizes certain ad hoc approaches to query optimization and execution, such as approaches based on horizontal partitioning, and integrates them into the general framework of optimization using materialized views.

Semantic information, such as knowledge of the key of a view, can be used to further optimize a rewritten query. We consider the problem of determining the key of a (regular or restructured) view, and show how this information can be used to further optimize a rewritten query.

In addition to general query optimization, restructured views may prove very beneficial in data-warehousing applications by providing a significantly extended space of schemas to store, represent, and query summarized data. Another natural application of restructured views is in e-commerce and other areas where flexible data modeling is required [5]. A convenient data model in these applications is a "vertical representation" (also called "property-table representation"), where a uniform schema (tupleID, attribute, value) is used for all data. Our restructured-view approach can easily accommodate this representation, either as schema of the physical storage (with the regular "wide table" defined as view) or, conversely, as a view of the stored "wide table".

1.1. Our contributions

- We propose a general query-optimization framework that treats regular and restructured views in a uniform manner and is applicable to SQL select-project-join (SPJ) queries and views without or with aggregation. Our remaining contributions are all within the context of this framework. It is straightforward to integrate all our proposed algorithms and techniques into standard query-optimization algorithms.
- We propose criteria and algorithms for determining when a view (possibly restructured) is usable in answering a SQL query.
- We propose algorithms to rewrite SQL queries in terms of usable views. Our view-usability criteria and rewriting algorithms are applicable to non-aggregate queries and views, as well as to queries and views that involve group-by and aggregation operations.
- We show how knowledge of key constraints can be used to optimize a rewritten query, and propose algorithms to determine the key of a restructured view.
- Using extensive experiments (see Section 7 and [4]), we show how using restructured views in addition to regular views in query optimization can result in a significant reduction in query-processing costs compared to a system that uses only regular views.