Maintaining Data Cubes Under Slowly Changing Dimensions

A thesis submitted for the degree of

Master Of Science

Rosette Kidwani, M.Sc,
School of Computer Science and Information Technology,
Science, Engineering, and Technology Portfolio,
RMIT University,
Melbourne, Victoria, Australia.

1st November, 2006
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Rosette H. Kidwani

School of Computer Science and Information Technology

RMIT University

1st November, 2006
Dedication

To my parents and my husband.
Acknowledgments

I wish to express my deepest gratitude to my supervisors, Dr. Xuzhen Zhang and Dr. Falk Scholer for their continuous help and guidance during my study. From them I learned valuable research skills, and they have always been a source of encouragement.
# Contents

Abstract 1

1 Introduction 3

1.1 Problems 6

1.2 Contributions 7

1.3 Thesis Organization 9

2 Background 11

2.1 The Multi-Dimensional Model and Data Cubes 12

2.2 Temporal Multi-Dimensional Models 16

2.2.1 Slowly Changing Dimensions 17

2.2.2 Other Temporal Multi-Dimensional Models 18

2.3 Data Warehouse Maintenance 20

2.3.1 Incremental Maintenance 21

2.3.2 The Summary-Delta Approach 23

2.4 Temporal Aggregation Maintenance 26
CONTENTS

2.5 View Maintenance ........................................... 27
2.6 Summary ..................................................... 29

3 A Temporal Multidimensional Model ......................... 30
3.1 Motivating Example ......................................... 31
3.2 Temporal Multi-Dimensional Model ..................... 33
  3.2.1 Temporal Dimension Table ............................ 34
  3.2.2 Temporal Fact Table .................................. 40
3.3 Temporal Aggregate Table ................................. 43
  3.3.1 Temporal Data Cube ................................. 48
3.4 Types Of Changes and Maintenance Strategy .............. 49
  3.4.1 Fact Table Insertions ............................... 50
  3.4.2 Dimension Insertions - New Dimension Values .... 51
  3.4.3 Dimension Insertions - Changes to Existing Dimension Values ..... 52
3.5 Summary ..................................................... 54

4 The Hash Base Aggregation Algorithm ....................... 56
4.1 Main Ideas .................................................. 56
4.2 Update Plan ............................................... 58
4.3 Summarizing the Changes ................................ 60
4.4 Refreshing the Data Cube ............................... 64
  4.4.1 Indexing Aggregates ............................... 68
4.5 Summary ..................................................... 74
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section/Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Experiments</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>5.1 Datasets</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>5.2 The Algorithms</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>5.3 Uniform Data and Uniform Mapping</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>5.3.1 Dimensions with multiple hierarchy levels: Product Dimension</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Insertions and Updates</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Insertions and Deletions</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>5.3.2 Dimensions of small size: Location Dimension</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Insertions and Updates</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>Insertions and Deletions</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>5.4 Skewed Data and Skewed Mapping</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>5.4.1 Dimensions with multiple levels: Product Dimension</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>Insertions and Updates</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>Insertions and Deletions</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>5.4.2 Dimensions with Less Hierarchy Levels: Location Dimension</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>Insertions and Updates</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>Insertions and Deletions</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>5.5 Skewed Distribution for Data and Uniform Mapping</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>5.5.1 Insertions and Updates</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>5.5.2 Insertions and Deletions</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>5.6 Discussion</td>
<td>100</td>
</tr>
</tbody>
</table>
CONTENTS

6  Conclusion  103
   6.1  Summary of Thesis Work  103
   6.2  Future Work  105

Bibliography  107
List of Figures

2.1 A star schema for a retail data warehouse .......................... 12
2.2 The aggregation hierarchies for the retail data warehouse dimensions ...... 14
2.3 The cube lattice of a data cube ........................................ 15

3.1 The temporal star schema for the retail data warehouse ...................... 34
3.2 The temporal dimension schema for the retail data warehouse ............... 38
3.3 The temporal data cube lattice ........................................... 48

4.1 The plan for updating the cube lattice ...................................... 59

5.1 Uniform distribution and uniform mapping: Multiple levels - Updates ......... 84
5.2 HBA vs. Recomputation on uniform data and uniform mapping ............... 85
5.3 Uniform distribution and uniform mapping: Multiple levels - Deletions ....... 87
5.4 Uniform distribution and uniform mapping: Small dimensions - Updates ...... 89
5.5 Uniform distribution and uniform mapping: Small dimensions - Deletions ...... 91
5.6 HBA vs. Recomputation on skewed data and skewed mapping ................. 93
5.7 Skewed Data and Uniform mapping: Insertions and Updates .................. 98
5.8 Skewed Data and Uniform Mapping: Insertions and Deletions . . . . . . . . . 99
List of Tables

3.1 DailySales table ........................................ 31
3.2 Result table 1 ........................................ 32
3.3 Result table 2 ........................................ 32
3.4 Result table 3 - based on the temporal model .......... 33
3.5 The Product Dimension ................................ 36
3.6 The Location Dimension ................................ 36
3.7 The Time Dimension .................................... 37
3.8 The DailySales fact table ................................ 42
3.9 The IS Summary table .................................. 44
3.10 The BS Summary table .................................. 44
3.11 The CS Summary table .................................. 45
3.12 The AS Summary table .................................. 45
3.13 Generating the two timestamps for the summary table .................................................. 46
3.14 The IS Summary table - updated (fact insertions) .................................................. 51
3.15 The IS Summary table - updated (dimension insertions) ........................................ 53
<table>
<thead>
<tr>
<th>Table Number</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Preparing the changes.</td>
<td>61</td>
</tr>
<tr>
<td>5.1</td>
<td>Database for the retail data warehouse</td>
<td>76</td>
</tr>
<tr>
<td>5.2</td>
<td>Uniform Distribution and Mapping: Performance Improvement for Updates</td>
<td>86</td>
</tr>
<tr>
<td>5.3</td>
<td>Uniform Distribution and Mapping: Performance Improvement for Deletions</td>
<td>88</td>
</tr>
<tr>
<td>5.4</td>
<td>Uniform Distribution and Mapping - Small Dimensions: Performance Improvement for Updates</td>
<td>90</td>
</tr>
<tr>
<td>5.5</td>
<td>Uniform Distribution and Mapping - Small Dimensions: Performance Improvement for Deletions</td>
<td>92</td>
</tr>
</tbody>
</table>
Abstract

In the multi-dimensional model of data warehouses, data drawn from multiple sources is organized into fact tables and dimension tables. Data cubes pre-compute the aggregations of measurements along combinations of dimensions to speed up the processing of Online Analytical Processing (OLAP) queries. When the underlying fact tables or dimension tables change over time, data cubes need to be incrementally maintained to reflect these changes. In this thesis, we study the problem of incrementally maintaining data cubes under slowly changing dimensions.

We argue that temporal features are essential in a data warehouse to avoid loss of historical information, and to help maintain data cubes to be consistent with the state of the underlying tables. We first characterize view maintenance problems based on changing dimensions. We then introduce our temporal multi-dimensional model that can describe changing dimension data, and we discuss strategies for incrementally maintaining data cubes based on that temporal model.

A maintenance algorithm is developed, and its performance is evaluated experimentally. The results show that our approach is highly effective and is more efficient than
re-materialization in some change cases. The results also show that our approach significantly outperforms existing incremental maintenance approaches.
Chapter 1

Introduction

Increasing market dynamics and competitiveness require organizations to be properly informed to make appropriate strategic planning and business decisions. Traditional database systems, often referred to as On Line Transactional Processing (OLTP), do not meet these needs since data is usually scattered among different systems, and each system is designed for a particular purpose. Data Warehousing, or On Line Analytical Processing (OLAP) technology, emerged to address these needs, and has become the essential element of decision support in large-scale environments. A data warehouse is a central repository, storing historical data spanning long periods of time for the purpose of data analysis and decision making without referring to operational systems (Chaudhuri and Dayal, 1997; Inmon, 1996; Vaisman, 2001; Widom, 1995). Queries to data warehouses are usually complex, and make extensive use of aggregate functions and the computation of statistical information to detect anomalies and to identify trends. For example, a typical query for order planning in a retail data warehouse would be: Compute the total sales of computers at Christmas for each of the
last five years for each brand and each store branch. Such queries require high-performance query processing and fast execution time.

OLAP applications provide fast, highly interactive visualized access to summarized information, in order to allow high-level users to query and analyze data from a number of perspectives and at various levels of depth, for the purpose of organizational planning and decision-making (Vaisman, 2001). Rather than navigating through multiple tables and rows, the user is allowed to navigate a visual *multi-dimensional* view of data, based on aspects of analysis that are of interest to the user (Vaisman, 1998). In the multi-dimensional model, a fact is viewed as a mapping from a point in a space of dimensions into one or more spaces of measure or numerical value. For example, a retail data warehouse may organize its business data along dimensions such as *Product*, *Location*, *Customer* and *Time*. The fact table may hold a transaction record for each sale of a product identified by *ItemId*, sold in store *StoreId*, by customer *CustomerId* on instant *TransDate*. *Sales* is the measure of the fact table. The numerical value of the fact table, *Sales*, is the attribute that is of importance to users and is the subject of analysis. It can be aggregated over one or more dimensions, such as finding the total sales for each store or for each item, using aggregate functions such as \( \text{SUM}(\text{sales}) \).

Dimension attributes may be organized into *hierarchies* of detail. For example, in a *Product* dimension of a retail warehouse, a particular product may be identified by its name and may belong to a unique product brand, which in turn may belong to a particular product category. As such, the *Product* dimension is said to have a hierarchical relationship among its elements, where the product name is the lowest level of the hierarchy, and *All* is the
highest level. Every dimension value is part of the ALL value. In a dimension hierarchy, each level represents the aggregated total of the data from the level below.

To speed up query processing, results of aggregate queries are stored for later use. These pre-computed aggregations are known as aggregate views or summary tables. A data warehouse stores these aggregates as materialized views to be queried and analyzed in OLAP, without accessing remote databases. If a query is repeatedly posed over base relations, it would be more efficient if it is posed over a defined pre-aggregated view rather than the base table, since they require less scanning time. In OLAP systems, aggregation can be performed over one or more dimension levels and stored into the multi-dimensional cells in data cubes. In data cubes, the fact table is the intersection of the axis of the cube, and the dimension tables contain the details for each axis. A data cube is the generalization of the GROUP-BY operator over all possible combinations of dimensions involving aggregates over various granularity levels (Agarwal et al., 1996; Gray et al., 1997). A data cube has a numerical measure fact table attribute, such as Sales. The measure attributes of the records are computed as a result of an aggregate function, such as SUM(). Sales values at varying granularities can be analyzed by selecting cells, planes or subcubes from the base GROUP-BY.

Changes may occur to the underlying fact and dimension tables, and Kimball (1996) termed dimension changes as slowly changing dimensions. There are three approaches to handle dimension changes: overwriting the original data, adding a row with every dimension modification, or keeping two columns for the two most recent modifications. These different approaches correspond to different models, where each model applies different aggregate maintenance strategies. While the first approach updates the model with the new version,
the second approach keeps all versions, and the third approach keeps only the two latest versions.

When the underlying fact tables or dimension tables change over time, the data cubes should be maintained to reflect these changes. One approach to solve this problem is to re-materialize all the aggregations that are related to the updated underlying tables. However, the space and time costs of re-materialization are extremely high, as the number of summary tables are typically large, and base table changes may occur frequently (Gupta and Munick, 1995). A better solution is the incremental maintenance approach. Incremental maintenance is the process of updating the aggregations by computing only the changes in the underlying relations (Vaisman, 1998). In data warehouses, source changes are usually deferred and applied to the summary tables in batch mode, during which time the summary tables cannot be accessed for analysis. The large size of the data cubes requires efficient techniques to maintain data cubes. In this thesis, we study incremental maintenance of data cubes under slowly changing dimensions. We focus on the consistency of aggregate results in the aggregate cube views given changes to dimension tables.

1.1 Problems

Dimensions may change over time. For example, a store dimension, where stores are grouped into states, may change, as new stores are established or existing stores are re-located from a state to another. Assume a store is re-located from NSW to Victoria. A query asking for the total sales of all stores of state Victoria may be interpreted in two different ways: the query could be asking for the total sales by the stores that are currently located in Victoria
as if they have always been located there, or for the total sales of these stores since they located in Victoria. The two answers reflect two different perspectives. Aggregation results may greatly vary depending on the interpretation. In this example, the total sales of a store currently located in Victoria while previously located in NSW should only be added under the first interpretation. In a non-temporal model which keeps only the most recent version of modelled reality, the previous state of data is lost, causing inconsistent aggregate results which can lead to wrong conclusions.

To ensure the consistency of aggregate results, it is essential that the changes to the underlying tables are taken into account. We argue for temporal features in data warehouses in order to keep track of underlying base table changes. Our work is concerned with the consistency of aggregation in the data cube. Based on a temporal representation of data evolution, the data cube can be updated to reflect the underlying changes and produce consistent aggregate results.

1.2 Contributions

We propose a temporal multi-dimensional model for keeping track of changes of historical information in a data warehouse. A temporal multi-dimensional model extends the multi-dimensional model by adding a temporal feature to keep track of changes. In this model, we introduce timestamps in the fact and dimension tables. The fact table has a timestamp to denote when an event occurred. To track changes in dimension tables, each new dimension change is identified by a unique artificially-generated key known as a surrogate key, while relying on the unchanged natural key to identify the history of the changes for the same
CHAPTER 1. INTRODUCTION

dimension element (Kimball, 1996). We give each dimension tuple effective and expiry dates to denote when a dimension row is valid. Several models have been introduced to represent the history of changes in the data warehouse (Bluhm et al., 1998; Body et al., 2002; Kimball, 1996; Pedersen et al., 1999; Mendelzon and Vaisman, 2000). However, these approaches do not address aggregation consistency. We address the problem of updating data cubes to produce consistent aggregate results, given changes to dimension data. In the temporal multi-dimensional model we propose, we also give the data cube two timestamps to denote when the aggregation is valid. We will show that maintaining data cubes based on our temporal multi-dimensional model will prevent the loss of previous states of the data warehouse, and will ensure the consistency of aggregation.

Based on our temporal multi-dimensional model, we introduce a new maintenance approach that uses the Propagate and Refresh paradigm (Mumick et al., 1997) to maintain data cubes given dimension changes, and yields consistent aggregate results. This new approach supports fast searching and efficient updating. We show how to efficiently update multiple views that are related to one another by a parent-child relationship.

In particular:

- We propose a temporal multi-dimensional model that can keep track of dimension data evolution by introducing timestamps for every fact and dimension change to represent when the data is valid. The temporal multi-dimensional model also ensures consistent aggregation by introducing temporal features for the data cube to denote when the aggregation is valid.

- We introduce a Hash Base Aggregation (HBA) algorithm that efficiently maintains
the data cube given dimension changes. The HBA algorithm uses the Propagate and Refresh paradigm (Mumick et al., 1997) where only the aggregate changes are computed before being applied on the data cube. We discuss maintenance strategies based on the temporal multi-dimensional model to update data cubes that summarize dimensional levels. We describe how timestamps can be manipulated in the update process in order to allow for the consistency of aggregate results. In the existing incremental maintenance approaches (Hurtado et al., 1999; Mumick et al., 1997; Vaisman, 2001), multiple joins and sequential traversals are necessary to apply the changes on the data cube, causing the update process to be slow. To address this problem, the HBA approach applies indexing techniques on the combination of dimension keys to provide fast search time, quick data retrieval capabilities, and quick maintenance time.

- We conduct a comprehensive series of experiments to evaluate the performance of the HBA approach against re-materialization and other existing incremental maintenance approaches. Our experiments show that incremental data cube maintenance using the HBA approach based on our temporal multi-dimensional model is more efficient and has better performance than recomputation and other existing incremental maintenance approaches, depending upon the type of dimension modification.

1.3 Thesis Organization

We present definitions and preliminaries, and review research background in Chapter 2. We also describe different proposed multi-dimensional models and maintenance techniques in data warehousing. In Chapter 3, we present our temporal multi-dimensional model. We also
discuss view maintenance issues in response to modifications to underlying fact and dimension tables based on that model. In Chapter 4, we present our maintenance approach and describe the HBA algorithm. We show that maintaining data cubes based on the temporal multi-dimensional model using the HBA algorithm is efficient and ensures consistent aggregation. In Chapter 5, we describe a series of experiments to evaluate the performance of the HBA algorithm, and we present the results of comparing the HBA approach with recomputation and existing incremental maintenance techniques, such as the Summary-Delta approach (Munick et al., 1997). Our evaluation demonstrates the efficiency of our algorithm in comparison with re-materialization and other incremental maintenance approaches. Finally, Chapter 6 summarizes our work, and presents directions for future work.