Available online at www.sciencedirect.com
ScienceDirect

Decision Support Systems 42 (2006) 267–282

Decision Support Systems

www.elsevier.com/locate/dsw

Denormalization strategies for data retrieval from data warehouses

Seung Kyoon Shin[a,*], G. Lawrence Sanders[b,1]

[a]College of Business Administration, University of Rhode Island, 7 Lippitt Road, Kingston, RI 02881-0802, United States
[b]Department of Management Science and Systems, School of Management, State University of New York at Buffalo, Buffalo, NY 14260-4000, United States

Available online 20 January 2005

Abstract

In this study, the effects of denormalization on relational database system performance are discussed in the context of using denormalization strategies as a database design methodology for data warehouses. Four prevalent denormalization strategies have been identified and examined under various scenarios to illustrate the conditions where they are most effective. The relational algebra, query trees, and join cost function are used to examine the effect on the performance of relational systems. The guidelines and analysis provided are sufficiently general and they can be applicable to a variety of databases, in particular to data warehouse implementations, for decision support systems.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Database design; Denormalization; Decision support systems; Data warehouse; Data mining

1. Introduction

With the increased availability of data collected from the Internet and other sources and the implementation of enterprise-wise data warehouses, the amount of data that companies possess is growing at a phenomenal rate. It has become increasingly important for the companies to better manage their data warehouses as issues related to database design for high performance are receiving more attention. Database design is still an art that relies heavily on human intuition and experience. Consequently, its practice is becoming more difficult as the applications that databases support become more sophisticated [32].Currently, most popular database implementations for business applications are based on the relational model. It is well known that the relational model is simple but somewhat limited in supporting real world constructs and application requirements [2]. It is also readily observable that there are indeed wide differences between the academic and the practitioner focus on database design. Denormalization is one example that

* Corresponding author. Tel.: +1 401 874 5543; fax: +1 401 874 4312.
E-mail addresses: shin@uri.edu (S.K. Shin), mgstand@mgt.buffalo.edu (G.L. Sanders).
[1] Tel.: +1 716 645 2373; fax: +1 716 645 6117.

0167-9236/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2004.12.004

has not received much attention in academia but has been a viable database design strategy in real world practice.

From a database design perspective, normalization has been the rule that should be abided by during database design processes. The normal forms and the process of normalization have been studied by many researchers, since Codd [10] initiated the subject. The objective of normalization is to organize data into normal forms and thereby minimize update anomalies and maximize data accessibility. While normalization provides many benefits and is indeed regarded as the rule for relational database design, there is at least one major drawback, namely "poor system performance" [15,31,33,50]. Such poor performance can be a major deterrent to the effective managerial use of corporate data.

In practice, denormalization techniques are frequently utilized for a variety of reasons. However, denormalization still lacks a solid set of principles and guidelines and, thus, remains a human intensive process [40]. There has been little research illustrating the effects of denormalization on database performance and query response time, and effective denormalization strategies. The goal of this paper is to provide comprehensive guidelines regarding when and how to effectively exercise denormalization. We further propose common denormalization strategies, analyze the costs and benefits, and illustrate relevant examples of when and how each strategy can be applied.

It should be noted that the intention of this study is not to promote the concept of denormalization. Normalization is, arguably, the mantra upon which the infrastructure of all database systems should be built and one of the foundational principals of the field [20,25]. It is fundamental to translating requirement specifications into semantically stable and robust physical schema. There are, however, instances where this principal of database design can be violated or at least compromised to increase systems performance and present the user with a more simplified view of the data structure [39]. Denormalization is not necessarily an incorrect decision, when it is implemented wisely, and it is the objective of this paper to provide a set of guidelines for applying denormalization to improve database performance.

This paper is organized as follows: Section 2 discusses the relevant research on denormalization and argues the effects of incorporating denormalization techniques into decision support system databases. Section 3 presents an overview of how denormalization fits in the database design cycle and develops the criteria to be considered in assessing database performance. Section 4 summarizes the commonly accepted denormalization strategies. In Section 5, relational algebra, query trees, and join cost function approach are applied in an effort to estimate the effect of denormalization on the database performance. We conclude in Section 6 with a brief discussion of future research.

2. Denormalization and data warehouses

2.1. Normalization vs. denormalization

Although conceptual and logical data models encourage us to generalize and consolidate entities to better understand the relationships between them, such generalization does not guarantee the best performance but may lead to more complicated database access paths [4]. Furthermore, normalized schema may create retrieval inefficiencies when a comparatively small amount of data is being retrieved from complex relationships [50]. A normalized data schema performs well with a relatively small amount of data and transactions. However, as the workload on the database engine increases, the relational engine may not be able to handle transaction processing with normalized schema in a reasonable time frame because relatively costly join operations are required to combine normalized data. As a consequence, database designers occasionally trade off the aesthetics of data normalization with the reality of system performance.

There are many cases when a normalized schema does not satisfy the system requirements. For example, existing normalization concepts are not applicable to temporal relational data models [9] because these models employ relational structures that are different from conventional relations [29]. In addition, relational models do not handle incomplete or unstructured data in a comprehensive manner, while for many real-world business applications such as data warehouses where multiple heterogeneous data sources are consolidated into a large data repository, the data are frequently incomplete and uncertain [19].

Download English Version:

https://daneshyari.com/en/article/554856

Download Persian Version:

https://daneshyari.com/article/554856

Daneshyari.com