International Journal of Computer Science
and Engineering (IJCSE)
ISSN(P): 2278-9960; ISSN(E): 2278-9979
Vol. 5, Issue 2, Feb-Mar 2016, 1-4
© IASET

THEORITICAL APPROACH TO DATABASE INTERNAL TABLE STRUCTURE FOR JOIN FREE QUERYING AND ELIMINATING REDUNDANT DATA ENTRY

S. MEENAKSHISUNDARAM

Department of Information Management System, Sri Venkateswara College of Engineering Pennalur,
Irrunkattukotai, Sriperumbudur (T.K), Tamilnadu, India

ABSTRACT

This paper discusses a database table structure toeliminate the redundancy without normalization. Also studies the problems of very large tables and views. We try to provide a database table model to provide a single table with joins and without redundant data. Also study the possibility of the physical connectivity between rows of different tables. This model suggests data storage structure of a table.

KEYWORDS: Database Table Structure

INTRODUCTION

Handling very large data table is a problematic one. It consumes more time to manipulate a required data in a normalized environment. DBAs are normally denormalizing the table to get the desired result quickly. The application developer has some burden in developing applications based on denormalized environment. We create views, the Precompiled frequently used SQL statements for reusability. Cost of the joins is high for very large tables. Temporary table or table variables are very often created to aggregate table values in stored procedure. Access time is very less if we avoid joins. But joins are the only way to display meaningful data fetched from normalized database. When query with aggregate functions along with joins will increase manipulating time when comparing same query from a single table.

All the RDBMS package developers have their own internal structure to store the data in row. Data structure to keep the table is different from vendor to vendor. But they have some common theory to manipulate them. In normalized environment we join various tables to retrace desired result. It would be useful if we can get the desired result without using join and views. Constructing a structure to get such a result is our problem.

The redundancy data of duplicated data cause many problems, mainly typography errors. Most of the research works are mainly concentrated on denormalizing methods rather than a table physical structure. We built warehouse data for cubes performance

RELATED WORKS

The internal storage or the physical implementation of row storage is deferred from RDBMS package developer to developer. Most of the works are intended in indexing. Mostly data storage in heaps, sorted files and hashed files are discussed. Also methods to keep the lengthy row are discussed. None of the work is existing or discussed about eliminating the redundancy without storing the repeated data. To the best of my knowledge no work has previously been done on physical implementation model for our goal. A lot of works has been done in optimizing the data retrieval and table partitioning.

Selinger P.G *et.al*. have presented methods of accessing the records using relations. Typically, there is more than one way to retrieve tuples because of the availability of indexes and the potential presence of conditions specified in the query for selecting the tuples [5].The use of materialized views will result in a better or a worse plan depends on the query and the statistical properties of the database [4].

Denormalizing is often discussed in performance tuning of the data retrieval. Yma Pinto has derived a systematic solution for denormalization and states that detailed authorization and access matrix stored along with denormalized view will increase performance [1]. Foto N. Afrati had studied to generate a search space of rewritings that is guaranteed to include a rewriting with an optimal physical plan and cost models for rewritings with the minimum number of sub goals [2].Also they have investigated the problem of generating efficient rewritings using views to answer a query such that a search space that is guaranteed to include an optimal rewriting [3].

A comprehensive approach to solving the problem of optimization in the presence of materialized views and the solution is syntax independent and cost based was presented by Surajit Chaudhuri et.al [4].The rewriting the query and optimizing the execution plan is a task of retrieving the data from large data. Our study is to eliminate this task. Can we get the data directly from the table? We integrate data with joins. Joins are cost effective.

Is it possible to get desired integrated data from multiple tables without join? This paper presents a simple logic that often used to achieve this.

THE SUGGESTED INTERNAL STRUCTURE

Normally we provide input FK [Foreign Key] to refer the redundant data. The redundant data is stored in other table. The table page header has information about the rows. A query will fetch the data from the redundant table and show the combined result as per our prerequisite. Let us assume that, if the relevant physical address of the required row of a table is directly linked to the FK key column then the table itself has the data as the part of the table.

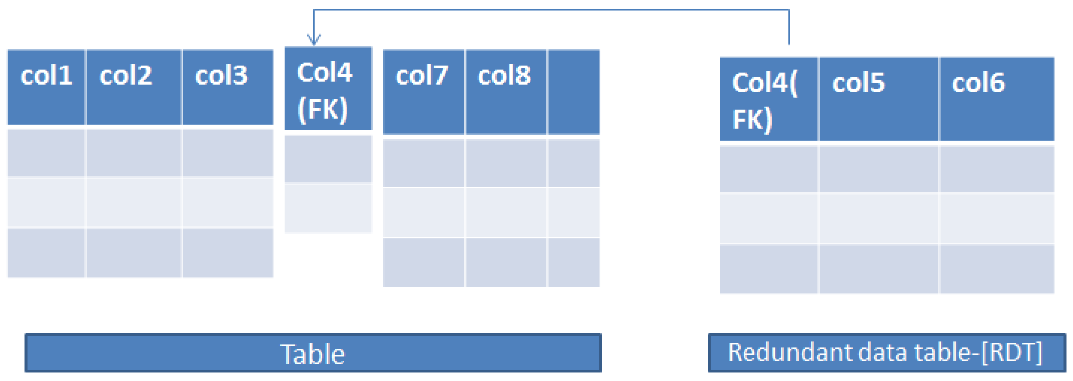

Figure 1

In other words logically the RDT is inserted within the primary table or it becomes part of the primary table. Now all the fields are part of RDT. Now we can access including all the columns of RDT. The primary table can be viewed as single object which consist of all relevant fields. The tuple will have required integrated relevant data.

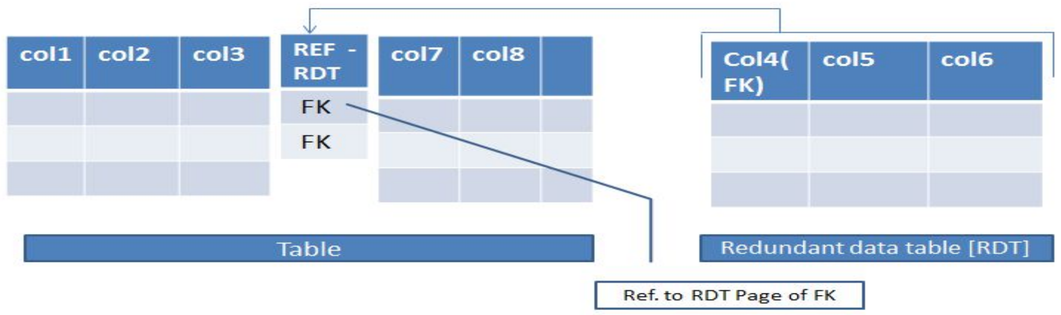

Figure 2

A view is a virtual table. Its performance becomes poor when the table size grows very large. If we use materialized view then the overhead cost increases. The above structure will reduce the cost of joins. In other words we never need a join query. The implementations may be very complicated one. But we require such join less tables to improve the large data access. We have to follow a structured input method to overcome this implementation hurdles. In NoSQL we create joins at the time of collection creation. Similarly we should create the table columns with reference to the RDT. So that any data inserted will have the physical link and become as part of the table.

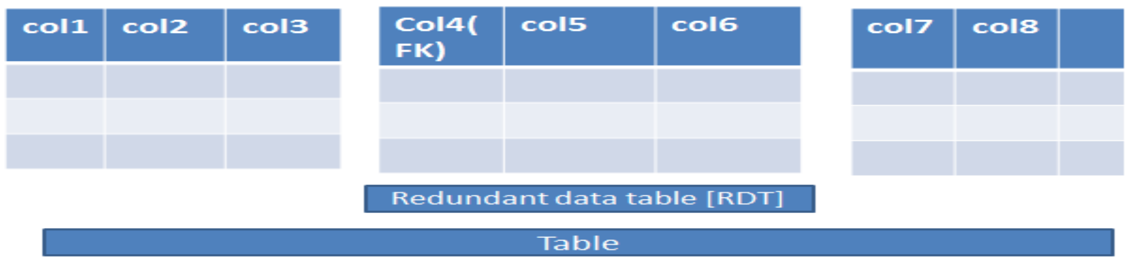

Figure 3

While creating the table the RDT columns should be mentioned as referral in the primary table. Here data base engine may take care of inserting the physical link of the referral table. The FK column is a referral columns and the data in this column is referral to the relevant physical row. The table contains the columns of RDT and becomes part of the primary table.

CONCLUSIONS

This model suggests data storage structure of a table. We can execute a join less query to get all data including the redundant data. The repeated data will not redundant in physical. The cost of data retrieval will be reduced. No need to combine tables for data retrieval direct query will get all relevant data spread over multiple tables.

REFERENCES

1. Yma Pinto, "A Framework for Systematic Database Denormalization", Global Journal of Computer Science and Technology, Vol 9, No 4, 2009

2. F. Afrati, C. Li, and J. D. Ullman. "Generating efficient plans for queries using views". In SIGMOD, pages 319–330, 2001.

3. F. Afrati, C. Li, and J. D. Ullman. "Using views to generate efficient evaluation plans for queries" Journal of Computer and System Sciences, Volume 73, Issue 5, August 2007, Pages 703–724
4. S. Chaudhuri, R. Krishnamurthy, S. Potamianos, K. Shim, " Optimizing queries with materialized views " , ICDE (1995), pp. 190–200
5. Selinger P.G., Astrahan M.M., Chamberlin D.D., Lorie R.A., Price T.G. Access path selection in a relational database management system. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1979, pp. 23–34.