Database denormalization

1. INTRODUCTION

Database developers often use normalization process to break raw databases into tables. It is an easy way to manage and organize the data. Normalization also guarantees that the data run through the database is accurate. The process reduces data redundancy. It helps to prevent the data from become functionally dependent so the database uploading is more efficient. However, normalizing database also has its downside. Normalization could cause a diminishing in database performance. It takes many joins of tables to combine information. When number of joins increases, it takes more time to run the database querying which slows down the completion process.

In order to have a faster retrieval, sometimes the choice of using denormalization process is considered. The reason of denormalizating a data is to achieve a better and faster performance. Denormalization is the opposite of normalization. The process puts one fact in numerous locations. It speeds up data retrieval with data modification as an expense.

Denormalization process can be done by fixing the table structures by creating constraints that synchronized the copies of redundant data in the normalized database. Taking down the normalization level one or two notch. Undo some steps in the normalization process, not completely. Another way is by copying data within tables in order to reduce the amount of tables that needs to be joined. The process allows controlling redundancy, while the joining increases during the database performance.

2. EXAMPLES OF DENORMALIZATION

A few examples of denormalizations techniques include storing the count of objects in multiple relations as in one relation, adding attributes from different relation to be joined, star schemas, OLAP cubes, and materialized views. Denormalization is not essentially required in performance improvement situation, because it is costly and expensive. And it also takes additional efforts to keep track of related data. If the joins exceed than five or six tables, denormalization should do the work.

2. 1. Conditions of Demoralization

There are more conditions that indicate the need of denormalization; important queries often rely upon data from more than one table and sometimes need online processing, a group processing is needed for group repetitions, applying calculations is needed to columns before answering the queries, accessing tables is needed in different ways by different users in the same time, primary keys slows in querying, and a large percentage queried by certain columns. Many different types of denormalizations may be used for different needs, such pre-joined tables, report tables, mirror tables, split tables, combined tables, redundant data, repeating groups, derivable data, and speed tables.

Denormalization, however, leaves a hazardous effect. Data redundancy is increased and application coding gives more complications because the data that have been spread is difficult to locate. And denormalization shows poor performance in inserting, updating, or deleting data, but improved in selecting or reading data.

Denormalization should be taken only when normalization failed to give a satisfying performance. Denormalization may solve one part of performance problem, but it possible that it creates more problems in several other areas. Total performance impact must be evaluated frequently because data integrity is at high risk.

Suppose account transaction is created to know the balance of an account.

For optimal performance, the query with denormalization is better. But it is noticeable only when there are more than 100. 000 transactions. Balance information in the tables with denormalization is acquired from one or more transactions. Everytime new transaction is being made; an update is needed for the corresponding balance. And when a transaction is removed, the transaction value from the corresponding balance column has to be subtracted. Denormalization reduces total performance of the system.

Denormalization is used when there are only few transactions with a high frequency of reporting accounts balances. Otherwise, denormalization can be considered unnecessary. From the example in the above tables normalization is more flexible than the one with denormalization. In figure 1, with denormalization, transactions are restricted in fixed number of periods. While in figure 2, reports can be done daily.

Denormalization costs a lot during transaction. A series of Inserting, updating, deleting attempt must be done over and over again, in a foolproof method. Avoiding this method can result in transactions and balances that may be different from each other. False report can certainly avoid by placing the login in the database for triggering insert, update and delete.

3. DENORMALIZATION APPLICATIONS

Denormalization application matches a very high demand of reporting functions. To perform denormalization safely, it is better to put the denormalized database into a separate data warehouse and updated periodically.

For example denormalization can take place in combining address, contact, email, and web and phone information. Those tables can be combined into one. However, putting the phone number information is a problem. It can be added by adding a field to the table. But combining it with the other four tables (address, contact, email and web) may cause an error that leaves blank in data output. It can be managed by modifying the table structures.

4. CONCLUSION

It is true that denormalization is essential in database as it makes original programming easier; however, it does not go that well with normalization. For small amount of databases normalization is efficient enough, despite of higher cost.