Performance analysis of column-oriented database for data warehouse system

Kamod Kumar
kk_muz70@rediffmail.com
Vaishali Institute of Business and Rural Management,
Muzaffarpur, Bihar

Dr. Alok Ranjan Tripathy
artripathy0102@rediffmail.com
Babasaheb Bhimrao Ambedkar Bihar University,
Muzaffarpur, Bihar

ABSTRACT

Column-oriented database systems, also known as column-stores, have an important demand in the past few years. Basically, it is about storing each database column separately so that the attributes belonging to the same column would be stored contiguously, compressed and densely-packed in the disk. This method has advantages in reading the records faster as compared to Traditional row stores in which every row are stored one after another in the disk. These databases are more suitable for data warehousing system to get an analysis done faster as data is stored in columnar form. Indexes are much faster in column-oriented databases which results in faster data retrieval and hence data analysis. This is an alternate database technology over row-oriented database systems. There are two approaches to map database tables onto a one-dimensional interface: store the table row-by-row or store the table column-by-column. The row-by-row approach keeps all information about an entity together. In the customer example above, it will store all information about the first customer, and then all information about the second customer, etc. The column-by-column approach keeps all attribute information together: the entire customer names will be stored consecutively, then all of the customer addresses, etc. Both approaches are reasonable designs and typically a choice is made based on performance expectations. If the expected workload tends to access data on the granularity of an entity, then the row-by-row storage is preferable since all of the needed information will be stored together. On the other hand, if the expected workload tends to read per query only a few attributes from many records, then column-by-column storage is preferable since irrelevant attributes for a particular query do not have to be accessed.

Keywords: Column-oriented database, Information, Customer, Column-by-column and row-by-row Storage.

1. INTRODUCTION

Column-oriented database systems, also known as column-stores, have an important demand in the past few years. Basically, it is about storing each database column separately so that the attributes belonging to the same column would be stored contiguously, compressed and densely-packed in the disk. This method has advantages in reading the records faster as compared to Traditional row stores in which every row are stored one after another in the disk. These databases are more suitable for data warehousing system to get an analysis done faster as data is stored in columnar form. Indexes are much faster in column-oriented databases which results in faster data retrieval and hence data analysis. This is an alternate database technology over row-oriented database systems.

2. APPROACHES

There are two approaches to map database tables onto a one-dimensional interface:

i) Store the table row-by-row

ii) Store the table column-by-column.

The row-by-row approach keeps all information about an entity together. In the customer example above, it will store all information about the first customer, and then all information about the second customer, etc.
The column-by-column approach keeps all attribute information together: the entire customer names will be stored consecutively, then all of the customer addresses, etc. Both approaches are reasonable designs and typically a choice is made based on performance expectations. If the expected workload tends to access data on the granularity of an entity (e.g., find a customer, add a customer, delete a customer), then the row-by-row storage is preferable since all of the needed information will be stored together. On the other hand, if the expected workload tends to read per query only a few attributes from many records (e.g., a query that finds the most common e-mail address domain), then column-by-column storage is preferable since irrelevant attributes for a particular query do not have to be accessed (current storage devices cannot be read with fine enough granularity to read only one attribute formal row. The vast majority of commercial database systems, including the three most popular database software systems (Oracle, IBM DB2, and Microsoft SQL Server); choose the row-by-row storage layout.

3. DESIGN IMPLEMENTATION

The design implemented by these products descended from research developed in the 1970s. The design was optimized for the most common database application at the time: business transactional data processing. The goal of these applications was to automate mission-critical business tasks.

For example, a bank might want to use a database to store information about its branches and its customers and its accounts. Typical uses of this database might be to find the balance of a particular customer’s account or to transfer $100 from customer A to customer B in one single atomic transaction. These queries commonly access data on the granularity and entity (finds a customer, or an account, or branch information; adds a new customer, account, or branch). Given this workload, the row-by-row storage layout was chosen for these systems.

4. ORIGIN AND EXPANSION

Starting in around the 1990s, however, businesses started to use their databases to ask more detailed analytical queries. For example, the bank might want to analyze all of the data to find associations between customer attributes and heightened loan risks. Or they might want to search through the data to find customers who should receive VIP treatment. Thus, on top of using databases to automate their business processes, businesses started to want to use databases to help with some of the decision making and planning.

5. PROBLEMS AND ISSUES IN HANDLING AND PERFORMANCE ISSUE

However, these new uses for databases posed two problems.

i) First, these analytical queries tended to be longer running queries, and the shorter transactional write queries would have to block until the analytical queries finished.

ii) Second, these analytical queries did not generally process the same data as the transactional queries, since both operational and historical data (from perhaps multiple applications within the enterprise) are relevant for decision making.

Thus, businesses tended to create two databases (rather than a single one)

i) The transactional queries would go to the transactional database and

ii) The analytical queries would go to what is now called data warehouses.

6. CONCLUSION

This business practice of creating a separate data warehouse for analytical queries is becoming increasingly common. In fact, today data warehouses comprise $3.98 billion of the $14.6 billion database market 27% and are growing at a rate of 10.3% annually.

7. REFERENCES