COLUMN ORIENTED DATABASES

ARUNDHATI KANUNGO
Developer Associate, SAP Labs India Pvt. Ltd.
E-mail: arundhati.kanungo@sap.com

Abstract - A column oriented database, rather a columnar database is a DBMS (Database Management System) that stores data in columns instead of rows. A columnar database aims to efficiently write and read data to and from hard disk storage to speed up the time to execute a query. A column-store is a physical concept. Here, I primarily focus on what a columnar database is, how it works, its advantages, disadvantages and applications at current times. In due course, the top three market selling columnar databases are discussed with their features. Thus, it is seen that, columnar database is an emerging concept which has high prospect in coming future.

Index Terms - DBMS, Columnar stores, Vertical partitioning

I. DEFINITION: COLUMN ORIENTED DATABASES:

A column oriented database management system is one such DBMS that stores table data as columns of data rather than as rows of data. When compared to most relational DBMSs that store data in rows, it is quite unusual. This column oriented DBMS has advantages for data warehousing, customer relationship management (CRM), and library catalog systems, and other ad hoc enquiry where aggregates are computed over large numbers of similar data items.

It is possible to achieve some of the benefits of both column and row-oriented organization with any DBMS.[1] By denoting one system as column oriented, we refer to both the ease of expressing a column oriented structure and the focus on optimizations for column oriented workloads. This approach is different from row oriented or row store databases and correlation databases, which use a value-based storage structure.[10]

II. HISTORY OF COLUMN ORIENTED DATABASES:

From the early days of DBMS development, column stores or transposed files have been implemented. In 1969 with focus on information-retrieval in biology, the first application of a column oriented database storage system called TAXIR was developed. [11] In 1976, Statistics Canada implemented the RAPID system for processing and retrieval of the Canadian Census of Population and Housing as well as several other statistical applications. [12] The system was shared with other statistical organizations throughout the world for widespread use in the 1980s. It was used by Statistics Canada until the 1990s.

KDB was the first commercially available column oriented database developed in 1993 followed by Sybase IQ in 1995. However, the traditional one shave changed rapidly since 2005 with many open source and commercial implementations.

III. WORKING OF COLUMN ORIENTED DATABASES:

A relational DBMS provides data that represents a two-dimensional table of columns and rows. For example, a database might have this table:

<table>
<thead>
<tr>
<th>EmpId</th>
<th>LastName</th>
<th>FirstName</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Smith</td>
<td>Joe</td>
<td>40000</td>
</tr>
<tr>
<td>12</td>
<td>Jones</td>
<td>Mary</td>
<td>50000</td>
</tr>
</tbody>
</table>

This simple table includes an employee identifier (EmpId), name fields (LastName and FirstName) and a salary (Salary). This two-dimensional format exists only in theory. In practice, storage hardware requires the data to be serialized into one form or another.

The most expensive hard drive operations are seeks. To improve overall performance, related data must be stored in a way such that it minimizes the number of seeks. This is known as locality of reference, and the basic concept appears in several different contexts. Hard drives are organized into a series of blocks of fixed size, typically enough to store several table rows. By organizing the data into rows that fit within the blocks, and grouping related rows together, the number of blocks that need to be read or sought is minimized.

A column oriented database serializes all values of a column together, then the values of the next column, and so on. For our example table, the data would be stored as below.


In this layout, any one of the columns closely matches the structure of an index in a row-based system. This causes confusion about how a column-oriented store "is really just" a row-store with an index on every column. However, the data mapping differs dramatically. In a row-oriented indexed system, the primary key is the row-id that is mapped to indexed...
data. In the column oriented system, primary key is the data, mapping back to row-ids. This may seem subtle, but the difference can be seen in this common modification to the same store:

\[ \ldots ;\text{Smith:001, Jones:002, 004, Johnson:003}; \ldots \]

As two of the records store the same value, "Jones", it is possible to store this only once in the column store, along with pointers to all the rows that match it. For many common searches, like "find all the people with the last name Jones", the answer is retrieved in a single operation. Other operations, like counting the number of matching records or performing computations over a set of data, can be greatly improved through this organization.

Whether a column-oriented system will be more efficient in operation depends mostly on the workload being automated. Operations that retrieve data for objects would be slower, requiring multiple disk operations to collect data from numerous columns to build up the record. However, these whole row operations are generally rare. In most cases, only a limited subset of data is retrieved. In a rolodex application, for instance, operations collecting the first and last names from several rows to build a list of contacts is far more common than operations reading the data for any single address. This is even more true for writing back data into the database, especially if the data tends to be "sparse" with many optional columns. For this reason, column stores have demonstrated excellent real world performance despite of theoretical disadvantages. This is a simplification. Moreover, partitioning, indexing, caching, views, OLAP cubes, and transactional systems such as write ahead logging or multi-version concurrency control all dramatically affect the physical organization of both systems. That said, online transaction processing (OLTP)-focused RDBMS systems are more row-oriented, while online analytical processing (OLAP)-focused systems are a balance of row-oriented and column-oriented ones.

**IV. TOP 3 MARKET SELLING COLUMN ORIENTED DATABASES:**

- **Sybase**: This DBMS aims to deliver high end performance for critical analytics, business intelligence and data warehousing solutions leveraging highly optimized server dedicated for analytics. Its column oriented with grid-based architecture, patented data compression, and advanced query optimizer, and henceforth, delivers high performance, flexibility, and economy in challenging reporting and analytics environments.

Essentially a data partitioned, index based storage technology, Sybase IQ’s engine offers several key features as below:

- Communications & Security
- User Friendly CUI based Administration & Monitoring
- Multiplex Grid Architecture
- Information Live-cycle management
- Fast Data Loading
- Query Engine supporting Full Text Search
- Column Indexing Sub System
- Column Storage Processor
- Web enabled analytics

The Sybase IQ Very Large Data Base (VLDB) provides partitioning and placement where a table can have a specified column partition key with value ranges. This partition allows data that should be grouped together to be grouped together and separates data where they should be separated. The drawback to this methodology is that it is not always known which is which.

- **Infobright**: Offering both commercial (IEE) and free community (ICE) edition, the combination of a column oriented database with their Knowledge Grid architecture delivers a self-managed, scalable, high performance analytics query platform. Allowing 50T data using a single server, their industry-leading data compression (10:1 up to 40:1) considerably lowers storage requirements and expensive hardware infrastructures. Delivered as a MySQL engine, Infobright runs on multiple OSs (operating systems) and processors needing only a minimum of 4GB RAM (however 16GB is recommended). Avoiding partition schemes, Infobright stores data in data packs, each node containing pre-aggregated statistics about the data stored within them. The Knowledge Grid provides related metadata offering a high-level view of the entire database content. Indexes, projections, partitioning or aggregated tables are not needed as these metadata statistics are automatically managed. The granular computing engine processes queries using the Knowledge Grid information to optimize query processing, eliminate or significantly minimize the amount of data required for decompression and access to answer a query. Some queries may not need to access the data at all, and instead find the answer in the Knowledge Grid itself.
The Infobright Data Loader inserts data very fast. It is better to avoid updates unless necessary, design denormalized tables, and plan any deletes. A reject option is a new feature added to the loader which allows valid rows to commit while invalid rows are logged.

This is highly useful when loading millions of rows and only having a few rows with bad data. Without this feature, the entire data load would be rolled back.

- **Vertica**: HP recently acquired this purpose-built platform to enable data values having high performance realtime analytics needs. Diverse communities can develop and scale with a seamless integration ecosystem leveraging capabilities like data loading, queries, columnar storage, MPP architecture, and data compression features. Claiming scalability, performance, elasticity, and simplicity, the Vertica analytics platform uses transformation partitioning and parallelism to declare which rows belong together and speed respectively. Several key features include:
  - Columnar Storage & Execution
  - Real-Time Query & Loading
  - Extensible In-Database Analytics Framework
  - In-Database Analytics Library
  - Database Designer & Administration Tools
  - Native BI & ETL support for MapReduce & Hadoop
  - Scale-out MPP Architecture
  - Automatic High Availability
  - Aggressive Data Compression

The Vertica Optimizer produces optimal query execution plans where several choices exist. It does this through traditional considerations like disk I/O and further incorporates network, concurrency, parallelism factors, CPU, memory, and the unique details of the columnar operator and runtime environment.

Other implementations of column oriented databases include Greenplum Database, CalpontInfiniDB, Accumulo, Teradata, SenSage, EXASOL, MonetDB, RCFile, Sqrrl, etc.

V. ADVANTAGES OF COLUMN ORIENTED DATABASES

- Scalability and fast data loading for Big Data
- Accessible by many third-party BI analytic tools
- Simple systems administration
- High performance on aggregation queries (like COUNT, SUM, AVG, MIN, MAX)\(^4\)
- Highly efficient data compression and/or partitioning

VI. DISADVANTAGES OF COLUMN ORIENTED DATABASES:

- Record updates and deletes reduce storage efficiency
- Effective partitioning/indexing schemes can be difficult to design
- Transactions are to be avoided or just not supported
- Queries with table joins can reduce high performance

VII. ROW VS COLUMN ORIENTED DATABASES:
VIII. INDUSTRIES TO BENEFIT FROM COLUMN ORIENTED DATABASES

- Telecommunications: Helps in improving response time to the customer by reducing input and output
- Financial Services: Supports high performance, millisecond response time to queries required for inbound market
- Retail: Reads the data only referenced in question driving higher performance and lowering processing costs compared to reading all the columns in the table[7]

CONCLUSION

Thus, columnar database is a key technology that delivers high business value by helping enterprises adapt their information infrastructure to the evolving demands for timely, reliable intelligence to run the business. In addition, it has far-reaching implications for the design of systems, and offers major cost savings affecting higher power and cooling requirements.

FUTURE WORK

Columnar databases can be very helpful in big data project. Big data, today is one of the biggest problem ever faced. When we have volume and variety of random real-time data, we might want to use a columnar database to exploit its flexibility, performance and scalability. Till date, HBase is the only column oriented database that is used with big data. I look forward to carry out a comparative analysis of HBase performance and other columnar database performance when fed with big data.

REFERENCES