Report

The fundamental difference between a from-scratch column-store and a row-store using the column-oriented physical design without exploring alternate physical designs for the row-store system. The goal of this paper is to answer this question in a systematic way. Important optimization techniques specific to column-oriented DSMs include: 1) Late Materialization, 2) Block Iteration, 3) Column-specific compression techniques.

Performance: The performance of these various techniques to the baseline performance of the open-source C-Store database on the SSBM, showing that despite the ability of the above methods to emulate the physical structure of a column-store inside a row- store, their query processing performance is quite poor. Hence, one contribution of this work is showing that there is in fact something fundamental about the design of column-store systems that makes them better suited to data-warehousing workloads.

Advantages: 1) Benefits of a column-store using a row-store are either by vertically partitioning the schema or by indexing every column so that column can be accessed independently. 2)Changes must be made to both the storage layer and the query executor to fully obtain the benefits of a column-oriented approach. 3)Row stores can write data very quickly, whereas a column store is awesome at aggregating large volumes of data for a subset of columns. One of the benefits of a columnar database is its crazy fast query speeds.

Disadvantages: 1) Columnar database is not suited for incremental data loading, 2) Online Transaction Processing (OLTP) applications are not suitable in column-oriented databases. 3)User queries against only a few rows cannot give you any benefits in columnar databases.

Applications: It is used in predicate application algorithms. Each predicate can be applied in Parallel and results merged using fast bitmap operations. At a basic level, row stores are great for transaction processing. Column stores are great for highly analytical query models. Row stores can write data very quickly, whereas a column store is awesome at aggregating large volumes of data for a subset of columns.

Conclusion: The conclusion of this work is not that simulating a column store in a row-store is impossible. Rather, it is that the simulation performs poorly on today's row-store systems. It also showed that attempts to emulate the physical layout of a column-store in a row-store via techniques like vertical partitioning and index-only plans do not yield good performance. Attribute this slowness to high tuple reconstruction costs, as well as the high per-tuple overheads in narrow, vertically partitioned tables.