Ethics Pledge

Consistent with the above statements, all homework exercises, tests and exams that are designated as individual assignments MUST contain the following signed statement before they can be accepted for grading.

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

Signature: Haodong Zhao Date: Mar 5th. 2019

Please note that assignments in this class may be submitted to

www.turnitin.com, a web-based anti-plagiarism system, for an evaluation of their originality.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Reading review**

**Column-Stores vs. Row-Stores: How Different Are They Really?**

"Column storage" is a column-oriented database system that performs better than traditional database systems ("row storage"). This is because the row store only needs to read those properties accessed by the query from disk (or memory).

This article demonstrates that the following assumptions are wrong: row storage can be done in vertical partition mode or by indexing each column so that columns can be accessed independently. By analyzing the performance differences, it was found that there were significant differences between the two systems except the storage tier level. The storage tier and the query must be modified to make the row store fully benefit from the column-oriented approach.

The paper mentions a test, implements a variety of different physical database designs, and explores as much "column-oriented" design as possible, including:

1. A set of two lists consisting of pairs of vertical partitions in the system
2. Using an index plan
3. Use a collection of materialized views

Tests have shown that important optimizations for column-oriented DBMSs include:

1. Late materialization
2. Block iteration
3. Column-specific compression
4. Invisible connections improve connection performance in post-matching column storage

To analyze which of these is the most important, remove these column-specific optimizations one by one to carefully measure the different variants of the C-Store database and break down the factors that contribute to their performance. It was then found that compression can provide orders of magnitude gain when possible, post-implementation provides approximately three times performance gain, and other optimized draws provide 1.5 times performance improvement.

The three contributions of this article:

1. Indicates that attempting to simulate column storage in row storage does not produce good performance results
2. Propose a new technique for improving connection performance in column storage called invisible connections
3. Explore the contribution of different optimized overall system performance

Column-oriented simulations that want to make today's ‘row store’ successful require some important improvements to the system.