# System Specification

The system used for this study is a MariaDB 10.0 MySQL database hosted on a cloud backend. This system is the same as the multiple databases I professionally work with, and the results of this paper are directly applicable to my current work. The topic of study for this paper is a direct comparison of real work design decisions made in a corporate atmosphere. The performance results of these decisions is the purpose of this analysis. Special permission was obtained from Dr. Tufte when this project was announced to study the performance of this specific database configuration.

# System Research

The research in this paper most closely resembles option 1, but both configurations will be within the MySQL system. These configurations study query performance with various schema design choices. These choices, specifically the decision to add columns to large indexed and non-indexed tables, may affect query performance in a significant way. For example, a table with ~5 million entries may be significantly impacted when a column of type VARCHAR(5000) is added to it. This paper examines the effects of various queries below.

The following configurations are examined:

1. 1 million entry table with 10 columns, indexed on primary key.
2. 1 million entry table with 11 columns, indexed on primary key, 11th column type VARCHAR(5000).
3. 1 million entry table with 12 columns, indexed on primary key, 11th and 12th column type VARCHAR(5000).
4. 1 million entry table with 11 columns, indexed on primary key, 11th column type is FK to secondary table containing index and VARCHAR(5000).
5. 1 million entry table with 11 columns, indexed on primary key, 11th and 12th column type is FK to secondary table containing index and VARCHAR(5000).
6. 1 million entry table with 11 columns, indexed on primary key, 11th and 12th column type is FK to two secondary tables, each containing index (to respective 11th and 12th columns) and VARCHAR(5000).
7. Similar configurations with larger VARCHAR sizes, depending if any different results are yielded in #2.
8. Similar configurations with varying populated VARCHAR data, depending if any different results are yielded in #2.
9. An attempt to set innodb\_buffer\_pool\_size at varying system buffer sizes, different from the default of 3/8.
10. An attempt to alter the innodb\_read\_ahead\_threshold, or buffer pool prefetching setting to ON/OFF.
11. An attempt to alter innodb\_old\_blocks\_pct and innodb\_old\_blocks\_time settings to change the LRU replacement policy and examine performance.

The results will be measure in query execution time. Buffer pool metrics will be reported when significant, using SHOW ENGINE INNODB STATUS.

# Performance Experiment Design

Using the configurations in the aforementioned list, the following experiments will be run:

1. On each configuration, a normal scan query will be made to identify any significant performance differences between schema configurations.
   1. SELECT \* FROM <table>
2. On each configuration, a normal index query will be made to identify any significant performance differences between schema configurations. With this query, it is expected the optimizer will ignore the VARCHAR columns and results will be similar.
   1. SELECT table.id,table.<non-varchar column> FROM <table>
   2. SELECT table.id,table.<non-varchar column>,<more columns> FROM <table>
3. On each configuration, a non-indexed query will be made to identify any significant performance differences between schema configurations. With this query, it is expected the optimizer will ignore the VARCHAR columns and results will be similar.
   1. SELECT table.<non-varchar column> FROM <table>
   2. SELECT table.<non-varchar column>,<more columns> FROM <table>
4. On each non-FK configuration, display the VARCHAR
   1. SELECT table.id,table.<varchar column> FROM <table>
5. On each non-FK configuration, display the VARCHAR column with an index range
   1. SELECT table.id,table.<varchar column> FROM <table> WHERE table.id BETWEEN X and Y
   2. SELECT table.id,table.<varchar column> FROM <table> WHERE table.id > X
   3. SELECT table.id,table.<varchar column> FROM <table> WHERE table.id < X
6. Use an indexed join with the FK configurations for comparing item 4.
   1. SELECT table1.id,table2.<varchar column> FROM <table1>,<table2> WHERE table1.<FK> = table2.<FK>
7. Use a join and a range with the FK configurations for comparing item 4.
   1. SELECT table1.id,table2.<varchar column> FROM <table1>,<table2> WHERE table1.<FK> = table2.<FK> AND table1.id BETWEEN X and Y
   2. SELECT table1.id,table2.<varchar column> FROM <table1>,<table2> WHERE table1.<FK> = table2.<FK> AND table1.id < X
   3. SELECT table1.id,table2.<varchar column> FROM <table1>,<table2> WHERE table1.<FK> = table2.<FK> AND table1.id > Y
8. Use SELECT \* on both tables and compare to item 6
   1. SELECT \* FROM <table>,<table2>
9. Use INNER JOIN with FK tables
   1. SELECT table1.id,table2.<varchar column> FROM table1 INNER JOIN table2 ON table1.FK = table2.FK
10. Use != null JOIN and compare to 9
    1. SELECT table1.id,table2.<varchar column> FROM <table1>,<table2> WHERE table1.<FK> = table2.<FK> AND table2.<varchar column> != NULL
11. Repeat simple joins for performance discussion with two varchar table configurations.

Data sets will be static sized, with the exception of #8 in the System Research section.

The expected results are that the operations will perform faster using the multiple-varchar table configuration, until the varchar data has to be accessed. This data should be used for information that is not frequently gathered. The scope of the experiment is to determine the magnitude of the performance difference.

# Learnings

In generating tests and configurations, there is a significant amount of queries and data that will need to be gathered. Some of these steps may be ignored if the initial findings are not fruitful, and does not appear to impact any results.

The cloud backend may limit buffer pool settings. If the backend is unable to accommodate alterations, they will be discarded for the scope of this study.