DS4300 HW1
Jason Zhang and Marco Tortolani

DESIGN CHOICES

1. Added secondary indexes to user_id in both the tweets and follows tables.

PERFORMANCE TESTING

1. Approximately 2770.1 tweets can be posted per second. Because Twitter receives 6-10 thousand new tweets per second, MySQL cannot keep up
2. Approximately 872.8 home timelines can be retrieved per second. Our program definitely cannot keep up with the 200-300 thousand home timeline refreshes that happen per second

ANALYSIS AND REPORTING

Hardware configuration: 1.4 GHz Quad-Core Intel Core i5 processor
Software stack: MySQL8.0, InnoDB, Java (java.sql, java.util, java.io, java.time)

API Method	**API Calls/Sec**
postTweet	2770.1
getHomeTimeline	872.8

Factors that impacted our results:

- Concurrently running applications, particularly ones which require lots of RAM (IntelliJ, Youtube windows, etc.)
- Optimization: when comparing performance tests before and after implementing secondary indexing (on user_id for both tables), there were significant differences in database performance. Without secondary indexing, the post rate was 2907.0/sec and the retrieval rate was 30.1/sec. With secondary indexing, the post rate was 2770.1 and the retrieval rate was 872.8/sec. This demonstrates that, in this particular use case, secondary indexing had great benefits on our database performance
- Network connection impacts results because the queries through JDBC are run through the network, a weak/slow connection will negatively impact results
- Randomness: depending on which users are randomly selected to have their most recent tweets retrieved, it can take a much faster or much slower time depending on where their tweets are stored in the database