**Experiment's description:**

This experiment is similar to previous one, but here Select condition is changed.

In the java application a scenario is simulated, in which random input user data, including user's location and the time, is generated and buffered, and then bulk-loaded into DB. In the meanwhile, there are some user agents that send select queries. PostgreSQL is used for database, PostGIS extension is used for handling location-related data. Application uses only one database, containing two tables, one being the main table and the other used as temp table (Database index on both of Time and Location columns (using gist)).

The Java application consists of following threads:

-CreateTextFileThread:

In this thread, two text file is created that contains 10,000 rows of sample user data. Each row has 10 columns, including Location as longitude and latitude, Time and some other data (some varchar (constant value in all rows), some numeric (random value)).

As we would like to use some geographic functions provided by PostGIS, we needed to convert the longitude and latitude numbers into "Point" data type. So, we created a buffer text file as buffer for keeping longitude and latitude, and another buffer text file for saving the rest of data.

This is used as a simulation for having incoming user input data and buffering it into a file. This way, we can later use "COPY" operation for bulk-loading which is faster than series of "INSERT" operations for all rows. (See chapter 14. Performance Tips in PostgreSQL's documentation for more information.)

-InsertRowsThread:

Here, the created text files are copied into two tables of database separately.  In general this thread does the following steps: (See the Overview diagram.)

1. Loads the other text file containing rest of data into a main table ("COPY")
2. Loads the text file (txtfileXY) containing latitude and longitude into a temp table ("COPY")
3. Creates point data type out of latitude and longitude within temp table
4. Loads the point data types into main table
5. Runs "ANALYZE" command

Running "ANALYZE" after bulk loading, ensures that the planner has up-to-date statistics about the table, which results in better decision making during query planning. (See chapter 14. Performance Tips in PostgreSQL's documentation for more information.)

-The RequestTimeLocation Threads:

There are 20 samples of this thread created. Each of these 20 threads, requests for information about users who have been in one of 20 pre-defined geographic regions in past 10 seconds and then go to sleep for 1 second after getting the results of the query.

The CreateTextFile Thread and  InsertRows Thread take turns, so that data is loaded after new text file is created. Queries are sent simultaneously. Each time a thread operates and goes to sleep is called a cycle in excel files.

Other classes are there just to run threads or to save the results.

The excel files show the results of running the program for about 45 minutes.

1. CopyIntoDB: Shows how long does it take to copy the buffers into tables ( including the time to convert location to Point data type and copy temp table to main table) and run analyze.

2. SelectRequests: This file shows the number of results and duration of queries sent by a sample request agent.
   Results show that as the database grows, the duration of a select operation remains almost the same.
   20 threads are requesting user data, with a select command based on pre-defined location (the entire area is divided into 20 squares, each related to one of these threads) in past 10 seconds. First chart illustrates the duration of each of requests. Second chart shows average duration values for every 100 select queries.
   The average time for select request is 52 milliseconds.

3. TextBuffer: shows how long does it take (in milliseconds) to create the both of text buffer files (each containing 10,000 rows to be copied into DB).

In all excel files duration is reported in millisecond.