SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 000

System for tracking vehicle locations in real time

Arthur Dent

Zagreb, lipanj 2016.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Introduction

Vehicle location is a multidimensional data point, it can contain up to three dimensions of spatial data and even one dimension of temporal data. Dataset is comprised of many data points and can be either static or live – in terms of supporting updates of vehicle locations. Of course, data is not worth anything if it can't be queried. Some common queries can be: what points are contained in given rectangle, or even more general in an arbitrary polygon. A query can be ask what points are contained within certain radius of a given point, but then another question arises – how is distance defined in this space. A natural assumption is that the points are on planet Earth, but even Earth can be modelled in many ways, as a perfect sphere or as an ellipsoid, each one suitable for a different application in life. If temporal dimension is present then all of these queries can be appended with time interval to return only the points that are happened within that time.

1.1. Motivation

Spatial and spatio-temporal data sets are growing rapidly. Consider these spatio-temporal data sets: Facebook or Foursquare with their check-ins, or photos on Instagram. These are all live datasets because queries and updates happen concurrently. There are also examples of static data sets such as data collected by Telecom providers – a data point for each time a user's phone contacted the tower. This dataset is not live and could be used to perform analytics, usually in a MapReduce system, which has much different requirements on the design of the system.

Initial inspiration for the problem is Uber and similar applications. They have a large number of vehicles that update their location every few seconds. Each user's phone shows vehicles that are in a relatively small area of the map which gives a pretty clear picture of what the most relevant queries could be. Note that this is not a spatio-temporal dataset because only the last location of a vehicle is important, although one could easily imagine the version where the history of locations is stored and queried.

1.2. Problem definition and constraints

Let us define the problem more precisely. Each vehicle has a unique vehicle id and location. Location is encoded in a standard geographic coordinate system, as longitude and latitude pairs. Longitude is in [-180, 180] range and latitude [-90, 90].

Updating a location can be referred as *update* $(vehicle\_id, longitude, latitude)$.

The simplest possible query is to find a list of vehicles that are within a given rectangle, and is referred as

$$rectangle\_query(longitude\_min, longitude\_max, latitude\_min, latitude\_max)$$

A point with its longitude and latitude is inside a given rectangle iff $longitude\_min <= longitude <= longitude\_max$ and $latitude\_min <= latitude <= latitude\_max$.

Another interesting query is to search for points within certain distance of a given point. Let's refer to this query as

$$radius\_query(longitude, latitude, radius)$$

Notice that the rectangle query was defined in a general way, it could be applied to any two dimensional data set, but this query needs a definition of distance between two points. We will use the simplest Earth model, a perfect sphere with quadratic mean radius of 6372 km. This choice is acceptable because distances should be fairly small and precise geometrical calculations are not mission critical. In turn, this should make geometry calculations as simple and as fast as possible.

Number of vehicles and users in this system is not defined in advance but is expected to be large so the requirement is that the system shouldn't be constrained by only one computer, i.e. must be scalable. Persistency of the data is not important because all of the data is only relevant for a short period of time, which isn't good enough reason to further complicate the system with persistence considerations. Also, if persistence is needed it can be achieved by another system which can be developed independently from this one and will not be studied in this thesis.

Consistency - eventual - not 100% sure?

2. Background

Historically, relational databases (RDBMs) have been the most popular storage system, and still are today. Geographical data and the need to store and query it has also been present in computer science for a long time. This is why most of the solutions were developed on top of relational RDBMS. The most important indexing data structure in RDBMs are B- and B$^+$-trees and they work very well with one dimensional data. Those trees then formed the basis of R- and R*-trees which became the standard for spatial queries in RDBMs because this design enabled great interoperability with existing RDBMs.

RDBMs have trouble in keeping up with the massive data growth in recent years mainly because of its scalability issues. This is why there has been a rise of NoSql databases that are designed to scale much better. This of course comes with a cost of supporting only a fraction of features that RDBMs support and often lack the same consistency guarantees. To improve on those basic scalability issues it is necessary to completely move from RDBMs to distributed storage systems even though that means that classical approaches such as R*-trees won't work.

2.1. Traditional solutions

Lets make a step back and briefly describe the traditional solutions in RDBMS – R-tree Guttman (1984) and its improvement R*-tree Beckmann et al. (1990).

R-tree is a height-balanced tree similar to a B-tree with every node except root containing between m and M records. M is chosen to store the largest possible number of records before it grows to big to fit inside a single disk page. Every record contains information about minimal bounding box I it is "responsible" for. This means that all the data in its subtree is within this bounding rectangle I, but that doesn't mean that some other node doesn't contain any points from this area. Bounding box can be represented as $I = (I_x, I_y)$ where I_x and I_y are intervals $[a, b]$ defining what interval on x and y axes current record contains. Records in non-leaf nodes contain minimal

bounding box of its subtree. Leaf nodes contain pointers to actual geometrical objects stored, but minimal bounding box is also computed and stored in them.

The most basic search query is to search for points within a given rectangle R. Search starts from the root node and recursively traverses the tree but only to nodes whose bounding box B intersects with R. If R completely contains B than whole subtree should be returned, otherwise search continues in all of the children nodes.

Inserting a new point is similar to search. New record is always inserted in the leaf node. If a node gets too big it has to be split and splits propagate up the tree. There are many heuristics on how to choose the appropriate leaf node to insert a new record. R-tree, at each level, always chooses the node which needs the least enlargement to fit a new point. Splitting a node also employs heuristics because there are $2^M - 1$ combinations to split it.

R*-tree improves on the R-tree by introducing better heuristics to split a node and to find the best node when inserting a new point.

Estimating the complexity of inserting and search isn't easy, but the main idea is that the search algorithm on average won't need to retrieve many data points that are not inside the query box.

2.2. Geohash

3. Conclusion

Conclusion.

LITERATURA

Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, i Bernhard Seeger. *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles+*, 1990. URL https://dbs.mathematik.uni-marburg.de/publications/myPapers/1990/BKSS90.pdf.

Antonin Guttman. *R-TREES. A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING*, 1984. URL http://www-db.deis.unibo.it/courses/SI-LS/papers/Gut84.pdf.

System for tracking vehicle locations in real time

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.