

Институт ИТКН

Кафедра инженерной кибернетики

Направление подготовки: 01.03.04 прикладная математика

Квалификация (степень): бакалавр

Семестр: 6

Группа: БПМ-20-4

ОТЧЕТ

ПО КУРСОВОЙ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

на тему:

Сравнительный анализ применения методов
геопоиска и структур геоиндексации в
высоконагруженных системах.

Учащийся:


подпись

/

Ибрагимов П. И. /

Фамилия И.О.

Оценка: 5 (отлично)

Дата защиты: 25.06.2023

Руководитель КНИР


подпись

/

Тагиев Э. Р./

должность, уч., степень. Фамилия И.О.

Москва 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ	6
Расстояние между двумя точками	6
Евклидово расстояние	6
Расстояние гаверсина	7
Геодезическое расстояние	7
Структуры геоиндексации	8
K-d tree	8
KNN поиск по KD-tree	9
Поиск в круге по KD-tree	9
Geohash	10
KNN поиск по Geohash	11
Поиск в круге по Geohash	12
Иные древовидные структуры данных	13
Quadtree	13
R-tree	14
VP-tree	15
BSP-tree	15
Иные плоские структуры данных	17
Uber H3	17
S2 geometry	18
Сравнительный анализ	19
Качественный сравнительный анализ	19
ЗАКЛЮЧЕНИЕ	21
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	22

ГЛОССАРИЙ

Индекс — Абстрактная структура данных упрощающая операции поиска

Геоиндекс — Индекс, построенный для упрощения поиска по геоданным

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей КНИР применяют следующие сокращения и обозначения:

СУБД — Система управления базами данных

БД — База данных

SQL (Structured Query Language) — Язык структурированных запросов. Декларативный язык запросов для взаимодействия с СУБД. Используется в таких базах данных как: Postgres, MySQL, MSSQL, SQLite и тд.

RPS (Requests per second) — Запросов за секунду. Метрика оценки нагрузки системы по количеству запросов от клиентов серверу в секундах.

KNN (k-nearest neighbors) - K-ближайших соседей, задача поиска K ближайших соседей к точке A по метрике D.

ВВЕДЕНИЕ

В настоящее время геоданные являются неотъемлемой частью многих высоконагруженных систем, таких как поисковые системы, социальные сети, картографические сервисы и другие. Однако, обработка и хранение большого объема геоданных может стать проблемой для разработчиков. Для решения данной проблемы были разработаны методы геопоиска и структуры геоиндексации, которые позволяют эффективно работать с геоданными в высоконагруженных системах.

Цель данной дипломной работы - провести сравнительный анализ применения методов геопоиска и структур геоиндексации в высоконагруженных системах. В работе будут рассмотрены основные принципы работы данных методов, их преимущества и недостатки. Также будет проведено сравнение производительности данных методов на различных наборах геоданных.

Под высоконагруженной системой подразумевается система, в которой присутствуют как больше количество чтений, то есть запросов типа SELECT, если использовать язык SQL, так и изменений данных: UPDATE, DELETE, INSERT. Такие системы в основном в моменте обслуживают большой порядок пользователей и имеют большой RPS, например, 2000 запросов в секунду.

Основной проблемой работы с геоданными в высоконагруженных системах является неочевидность в выборе структур и методов хранения данных. Так, например, классический индекс B-Tree нельзя использовать при обращении к кортежам геоточек, потому что B-Tree умеет работать только с скалярными данными. При этом для работы с геоданным существует объемное количество индексов, например, R-Tree, KD-Tree, Quadtree, каждый из которых имеет свои плюсы и минусы. Помимо этого существует проблема сериализации и десериализации данных, которая заключается в том, что результаты анализа поиска, запросы на поиск и сами данные точек необходимо передавать по сети в как можно меньшем объеме и размере (в данном случае под объемом подразумевается количество передаваемых данных, а размер - вес в байтах).

Результаты данной работы могут быть полезны для разработчиков высоконагруженных систем, которые работают с геоданными, а также для специалистов в области геоинформатики.

АНАЛИТИЧЕСКИЙ ОБЗОР ЛИТЕРАТУРЫ

В данной работе будут анализироваться только 2 задачи поиска по геоданным. Задачи можно сформулировать следующим образом: Пусть есть множество точек, представляющих собой пару широта-долгота, отображающих некий объект на поверхности земли.

Надо:

а) Найти все объекты X , находящиеся на расстоянии Y от объекта Z . Эту задачу можно переформулировать следующим образом: найти все объекты множества X в окружности с радиусом Y и центром в Z .

б) Найти X ближайших объектов к объекту Z . Эта задача также известна как задача поиска ближайших соседей.

Расстояние между двумя точками

Указанные выше задачи требуют обусловить понятие расстояния между двумя геоточками. Для этого введем понятие геоточки: это точка, представляющая собой кортеж $A(x, y)$, где x и y - широта и долгота соответственно. Расстоянием между двумя точками есть наименьшая дуга на плоскости земного шара, при этом ее вычисление может быть разным в зависимости от положенной задачи.

Поставим задачу:

Даны геоточки $A(x_1, y_1), B(x_2, y_2)$. Требуется найти расстояние $d(A, B)$

Евклидово расстояние

В некоторых системах, например, расширение Postgis для СУБД Postgres для типа Geometry, используется евклидово расстояние:

$$d(A, B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Данный подход работает корректно только в части случаев, так, например, он выдаст правильный результат с точностью до 5 метров в случае поиска расстояния в пределах Москвы, но, расстояние в Африке будет отличаться в 2-3 раза относительно более корректных подходов. При этом важно отметить, что данная дистанция отвечает аксиомам метрики, поэтому ее корректно использовать при сравнении расстояний между двумя точками, например, при решении задачи KNN(К-ближайших соседей).

Расстояние гаверсинуса

Расстояние гаверсинуса - это способ определения расстояния между двумя точками на поверхности Земли, учитывающий кривизну Земли. Оно используется в геопоиске для определения расстояния между заданной точкой и объектами в заданном радиусе.

Формула расстояния гаверсинуса выглядит следующим образом:

$$d(A, B) = 2R \cdot \arcsin \left(\sqrt{\sin^2 \left(\frac{x_2 - x_1}{2} \right) + \cos(x_1) \cdot \cos(x_2) \cdot \sin^2 \left(\frac{y_2 - y_1}{2} \right)} \right)$$

где d - расстояние между двумя точками в километрах, R - радиус Земли (приблизительно 6371 км), x_1 и x_2 - широты двух точек в радианах, y_1 и y_2 - долготы двух точек в радианах.

Эта формула позволяет определить расстояние между точками с точностью до нескольких метров, что делает ее очень полезной для геопоиска. Однако, она может быть достаточно ресурсоемкой при работе с большими объемами геоданных, что делает ее медленнее по сравнению с Евклидовым расстоянием.

Геодезическое расстояние

Геодезическое расстояние - это расстояние между двумя точками на поверхности Земли, измеренное вдоль кратчайшей линии (геодезической линии) между этими точками. Геодезическая линия - это кривая на поверхности Земли, которая имеет наименьшую длину между двумя точками.

Геодезическое расстояние учитывает кривизну Земли и может отличаться от расстояния гаверсинуса, особенно на больших расстояниях и при использовании разных моделей формы Земли.

Для расчета геодезического расстояния используются различные методы, такие как метод Винсента, метод Гаусса-Крюгера и метод Хаверсина. Эти методы учитывают форму Земли и позволяют получить более точные результаты, чем простые формулы для расчета расстояния на плоскости.

Геодезическое расстояние широко используется в геопозиционировании, навигации, картографии и других областях, где требуется точное определение расстояния между двумя точками на поверхности Земли.

Формула вычисления геодезии:

$$d(A, B) = R \cdot \arccos(\sin(x_1) \cdot \sin(x_2) + \cos(x_1) \cdot \cos(x_2) \cdot \cos(y_2 - y_1))$$

Таким образом, при выборе формулы поиска расстояния надо учитывать требования.

Например, при разработке приложения кикшеринга, то есть аренды электронных самокатов, их можно комбинировать. На запросах, которые возвращают клиентам ближайшие самокаты от их текущей геопозиции по радиусу можно использовать метрику Евклида, так как можно банально искать на 10-20 метров больше, чем требуется и даже в случае ошибки, результат будет удовлетворительным. При этом, в ситуации, когда требуется построить аналитику данных, лучше использовать расстояние геодезии, потому что в данных задачах крайне важна точность результата.

Структуры геоиндексации

Как и с задачами поиска по *плоскому* массиву скаляров, для поиска по геоординатам используются специальные структуры данных, которые позволяют оптимизировать операции поиска.

Указанные структуры можно разбить на 2 типа: древовидные и плоские. К первому виду относятся: R-tree, VP-tree, BSP-tree, Quadtree, KD-tree. К плоским относятся: Geohash, S2 Geometry и Uber H3.

Рассмотрим принцип работы древовидных структур на примере KD tree.

K-d tree

K-d tree представляет собой дерево, позволяющее производить операции поиска в N-мерном пространстве. Рассмотрим двухмерное K-d tree, которое также можно назвать K-2 tree. Само K-D tree является обычным бинарным сбалансированным деревом. Алгоритм построения K-d tree довольно прост:

- а) Происходит поиск центральной точки, то есть той точки, которая будет находится на сумарно меньшем расстоянии от всех точек. Данная точка ставится в корень дерева k-d tree.
- б) Далее плоскость "разбивается" на 2 части по вертикальной оси
- в) "Слева" ищется "средняя точка" то есть та точка, которая по оси абсцисс(широте) находится на сумарно меньшем расстоянии до остальных. Данная точка записывается в левого ребенка корня дерева
- г) Аналогичная процедура повторяется справа.
- д) Аналогичная процедура повторяется для вновь созданных полотен, но уже с осью ординат(долготой)
- е) Данные процедуры повторяются со всеми точками.

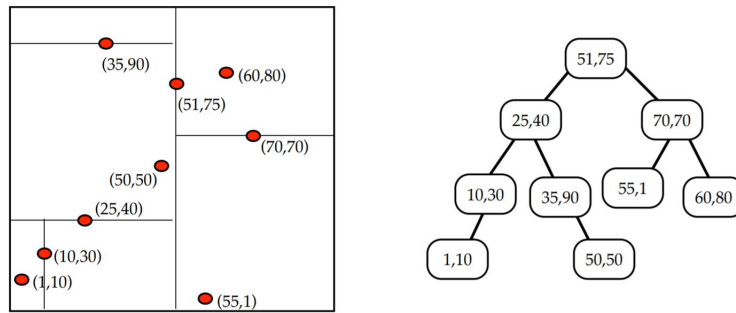


Рисунок 1 — Слева - пример визуализации разбиения плоскости через KD-tree.
Справа - результат построения дерева

KNN поиск по KD-tree

Процесс поиска K ближайших соседей по KD-tree заключается в следующих шагах:

- а) Строим KD-tree из набора точек.
- б) Находим ближайшую к заданной точке точку в дереве. Для этого спускаемся по дереву, сравнивая координаты заданной точки и текущей точки в узле дерева. Если координата текущей точки больше или равна координате заданной точки, то спускаемся в левое поддерево, иначе - в правое. При этом сохраняем расстояние между текущей точкой и заданной точкой.
- в) Добавляем найденную точку в список ближайших соседей.
- г) Проверяем, есть ли еще точки в дереве, которые могут быть ближе к заданной точке, чем уже найденные соседи. Для этого проверяем расстояние между заданной точкой и границей текущего поддерева (это можно сделать, используя формулу расстояния между точками). Если это расстояние меньше, чем расстояние до самого дальнего найденного соседа, то нужно проверить и другое поддерево.
- д) Повторяем шаги 2-4 для всех точек в дереве, пока не найдем K ближайших соседей или не пройдем по всему дереву.

В результате получаем список K ближайших соседей заданной точки. Этот алгоритм позволяет быстро находить ближайшие соседи в больших наборах данных и широко используется в геоинформационных системах для поиска ближайших объектов на карте.

Поиск в круге по KD-tree

Процесс поиска K ближайших точек и всех точек в заданном радиусе очень похож на процесс поиска по бинарному дереву за тем исключением, что при сравнении по четным нодам идет по широте, а по нечетным - по долготе.

Процесс заключается в следующих шагах:

- а) Строим KD-дерево на основе набора геоданных.
- б) Ищем листовой узел дерева, который содержит заданную точку. Для этого начинаем с корневого узла и спускаемся по дереву, выбирая каждый раз ту часть пространства, которая содержит заданную точку.
- в) Находим все точки, которые находятся в заданном круге с центром в заданной точке и радиусом R . Для этого проверяем каждую точку в листовом узле и всех его родительских узлах на расстояние до заданной точки. Если расстояние меньше или равно R , то добавляем точку в список найденных точек.
- г) Если листовой узел не содержит достаточного количества точек, то расширяем круг до тех пор, пока не найдем достаточное количество точек.
- д) Возвращаем список точек, которые находятся в заданном круге.

Этот алгоритм обеспечивает точность поиска в пределах заданного радиуса и может быть эффективным для поиска ближайших соседей в небольших наборах геоданных. Однако, он может быть менее эффективным для больших наборов данных или для поиска точек в нескольких кругах одновременно.

Geohash

Geohash(далее также геохеш), в отличие от древовидных структур представляет собой обычный массив, поиск по которому можно реализовывать через тривиальные методы, например, через бинарное дерево поиска, красно-черные деревья или b-tree. Самим геохешом называется строка, закодированная 32 разрядным алфавитом, перевод из 10-ой системы в указанный алфавит указан в таблице ниже.

Основание 10	0	1	2	3	4	5	6	7
Основание 32	0	1	2	3	4	5	6	7
Основание 10	8	9	10	11	12	13	14	15
Основание 32	8	9	b	c	d	e	f	g
Основание 10	16	17	18	19	20	21	22	23
Основание 32	h	j	k	m	n	p	q	r
Основание 10	24	25	26	27	28	29	30	31
Основание 32	s	t	u	v	w	x	y	z

Данная строка однозначно декодируется в кортеж геокоординат с некой точностью, зависящей от количества символов в строке. Примеры:

- а) строка `ucft` декодируется в прямоугольник площадью примерно 800 km^2 и центром в (55.81, 37.44).

б) строка `ucft943` имеет тот же центр - (55.8236, 37.3116), но меньшую площадь $23409m^2$

Как можно наблюдать, чем выше количество символов, используемых в геохеше, тем выше точность получаемых координат, но при этом выше затрачиваемая память. В данной работе не будет детально описываться процесс формирования геохеша за исключением базового принципа:

Сфера земли разбивается на практически равные прямоугольники, после чего каждому прямоугольнику присваивается номер в 32х ричной системе координат, номера присваиваются в порядке "змейкой сначала самый левый-верхний, далее ниже от него, далее справа от самого левого-верхнего и тд.

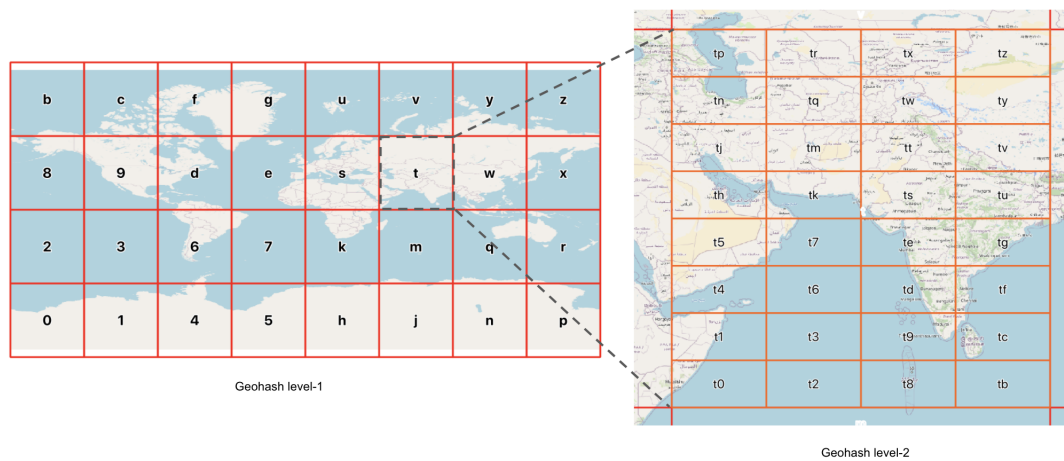


Рисунок 2 — Пример разбиения сферы земли через Geohash на первые 2 префикса

KNN поиск по Geohash

Процесс поиска K ближайших соседей по geohash заключается в следующих шагах:

- а) Преобразуем координаты заданной точки в geohash строку определенной длины.
- б) Ищем все точки, которые находятся в ячейках, соответствующих префиксу geohash строки заданной точки. Для этого используем функцию декодирования, которая преобразует geohash строку в координаты прямоугольной области, которая содержит все ячейки с данным префиксом.
- в) Сортируем найденные точки по расстоянию до заданной точки и выбираем K ближайших соседей.

г) Если количество найденных точек меньше K , то увеличиваем длину geohash строки и повторяем поиск, пока не найдем достаточное количество соседей.

д) Возвращаем список K ближайших соседей заданной точки.

Этот алгоритм позволяет быстро находить ближайшие соседи в больших наборах геоданных, используя простую и эффективную структуру индексации. Он широко применяется в геоинформационных системах и мобильных приложениях для поиска ближайших объектов на карте. Однако, он не обеспечивает точность поиска в пределах определенного радиуса, как это делает KD-tree, и может потребовать дополнительной обработки данных для учета искривления поверхности Земли.

При этом при поиске точек по заданному префиксу можно использовать структуру данных B-Tree, либо другие структуры, которые ускоряют поиск элементов в массиве по префиксу.

Поиск в круге по Geohash

Процесс поиска точек в заданном круге по Geohash заключается в следующих шагах:

а) Конвертируем заданные координаты в Geohash строку.

б) Определяем Geohash строки, которые находятся внутри заданного круга. Для этого используем формулу Хаверсина для вычисления расстояния между заданными координатами и каждой из ячеек Geohash, либо любую другую формулу вычисления расстояния в зависимости от заданных требований точности. Если расстояние меньше или равно радиусу круга, то добавляем эту Geohash строку в список.

в) Ищем все точки, которые соответствуют найденным Geohash строкам. Для этого используем индекс, который связывает каждую Geohash строку с набором геоданных. Ищем все точки в наборе, которые соответствуют найденным Geohash строкам.

г) Возвращаем список точек, которые находятся в заданном круге.

Этот алгоритм обеспечивает быстрый поиск точек в заданном круге и может быть эффективным для высоконагруженных систем, так как он позволяет быстро искать точки в больших наборах геоданных. Однако, он может быть менее точным, чем KD-Tree, так как Geohash строки имеют фиксированный размер и точность.

Иные древовидные структуры данных

Quadtree

Quadtree - это метод геоиндексации, который представляет пространство в виде дерева, где каждый узел представляет собой четыре дочерних узла. Этот метод позволяет быстро искать точки в заданном круге, разбивая пространство на более мелкие части и проверяя только те части, которые находятся внутри круга.

Процесс поиска точек в заданном круге по Quadtree заключается в следующих шагах:

а) Конвертируем заданные координаты в координаты узла Quadtree. Это делается путем разбиения пространства на сетку и назначения каждой ячейке уникальных координат узла Quadtree.

б) Определяем узлы Quadtree, которые находятся внутри заданного круга. Для этого используем формулу Хаверсина для вычисления расстояния между заданными координатами и каждым узлом Quadtree. Если расстояние меньше или равно радиусу круга, то добавляем этот узел в список.

в) Ищем все точки, которые соответствуют найденным узлам Quadtree. Для этого используем индекс, который связывает каждый узел Quadtree с набором геоданных. Ищем все точки в наборе, которые соответствуют найденным узлам Quadtree.

г) Возвращаем список точек, которые находятся в заданном круге.

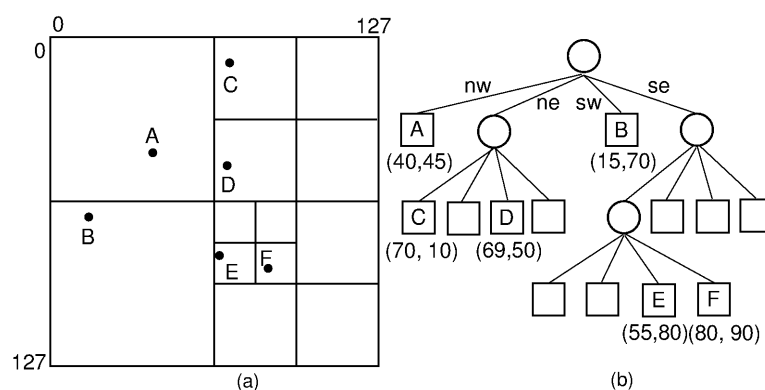


Рисунок 3 — Слева - пример визуализации разбиения плоскости через Quadtree.

Справа - результат построения дерева

Этот алгоритм обеспечивает быстрый поиск точек в заданном круге и может быть эффективным для высоконагруженных систем, так как он позволяет быстро искать точки в больших наборах геоданных. Однако, он может потребовать больше ресурсов для построения и поддержки индекса, чем Geohash.

R-tree

R-tree - это метод геоиндексации, который используется для хранения и поиска объектов в пространстве. R-tree представляет собой дерево, где каждый узел представляет собой прямоугольник, содержащий объекты. Этот метод позволяет быстро находить объекты, которые находятся в заданном прямоугольнике или близко к нему.

Процесс поиска объектов в заданном прямоугольнике по R-tree заключается в следующих шагах:

а) Конвертируем заданный прямоугольник в формат R-tree. Это делается путем определения минимального и максимального значения координат для каждого измерения.

б) Ищем узлы R-tree, которые пересекаются с заданным прямоугольником. Для этого используем алгоритм пересечения прямоугольников, который позволяет быстро определить, какие узлы R-tree находятся внутри или пересекаются с заданным прямоугольником.

в) Ищем все объекты, которые соответствуют найденным узлам R-tree. Для этого используем индекс, который связывает каждый узел R-tree с набором объектов. Ищем все объекты в наборе, которые соответствуют найденным узлам R-tree.

г) Возвращаем список объектов, которые находятся в заданном прямоугольнике.

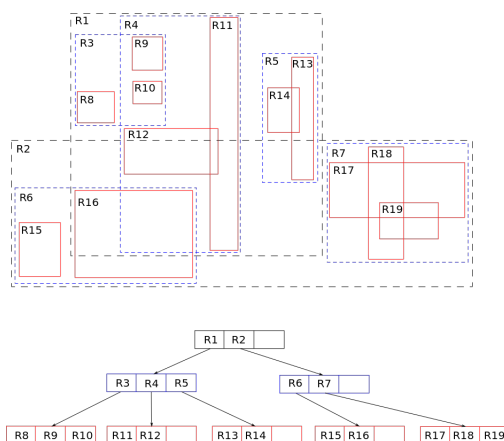


Рисунок 4 — Сверху - пример визуализации разбиения плоскости через R-tree.

Снизу - результат построения дерева

R-tree может быть эффективным для поиска объектов в пространстве, так как он позволяет быстро находить объекты, которые находятся в заданном прямоугольнике или близко к нему. Однако, он может потребовать больше ресурсов для построения и поддержки индекса, чем Geohash и Quadtree. Сравнительный анализ этих методов геоиндексации может помочь выбрать наиболее подходящий метод для конкретного приложения.

VP-tree

VP-tree (Vantage Point tree) - это метод геоиндексации, который используется для быстрого поиска ближайших соседей в многомерном пространстве. VP-tree представляет собой бинарное дерево, где каждый узел представляет собой точку-центр (вантажный пункт), от которой строятся два поддерева: левое поддерево содержит все точки, которые находятся внутри заданного радиуса от центра, а правое поддерево содержит все точки, которые находятся за пределами этого радиуса.

Процесс поиска ближайших соседей по VP-tree заключается в следующих шагах:

а) Строим VP-tree из набора точек. Для этого выбираем случайную точку-центр и разбиваем набор точек на две группы: те, которые находятся ближе к центру, и те, которые находятся дальше от него.

б) Ищем ближайшую точку к заданной точке. Для этого начинаем с корня VP-tree и рекурсивно спускаемся по дереву, выбирая поддерево, которое содержит более близкие к заданной точке точки. Если расстояние от центра поддерева до заданной точки меньше, чем текущее расстояние до ближайшей точки, то ищем ближайшую точку в этом поддереве.

в) Ищем k ближайших соседей к заданной точке. Для этого начинаем с корня VP-tree и рекурсивно спускаемся по дереву, выбирая поддерево, которое содержит более близкие к заданной точке точки. Если расстояние от центра поддерева до заданной точки больше, чем текущее расстояние до k -го ближайшего соседа, то не ищем в этом поддереве. Иначе ищем в этом поддереве и добавляем найденные точки в список ближайших соседей. После этого проверяем, есть ли еще поддеревья, которые могут содержать более близкие точки, и продолжаем поиск.

VP-tree может быть эффективным для поиска ближайших соседей в многомерном пространстве, так как он позволяет быстро находить ближайшие точки, используя разбиение на поддеревья. Однако, он может потребовать больше ресурсов для построения и поддержки индекса, чем Geohash и Quadtree.

BSP-tree

BSP-tree (Binary Space Partitioning tree) - это метод геоиндексации, который используется для разбиения пространства на бинарные подпространства. BSP-tree представляет собой бинарное дерево, где каждый узел представляет собой гиперплоскость, которая разделяет пространство на две части: левую и правую.

Процесс поиска ближайших соседей по BSP-tree заключается в следующих шагах:

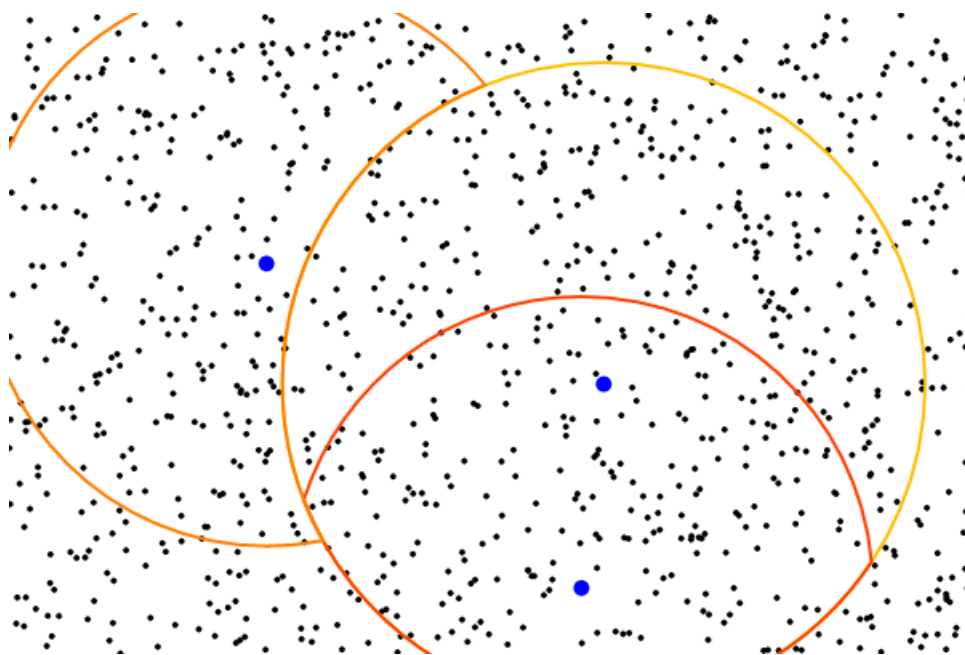


Рисунок 5 — Разбиения пространства на 3 подпространства через VP-Tree

а) Строим BSP-tree из набора точек. Для этого выбираем случайную гиперплоскость и разбиваем набор точек на две группы: те, которые находятся по одну сторону от гиперплоскости, и те, которые находятся по другую сторону.

б) Ищем ближайшую точку к заданной точке. Для этого начинаем с корня BSP-tree и рекурсивно спускаемся по дереву, выбирая поддерево, которое содержит более близкие к заданной точке точки. Если расстояние от гиперплоскости до заданной точки меньше, чем текущее расстояние до ближайшей точки, то ищем ближайшую точку в этом поддереве.

в) Ищем k ближайших соседей к заданной точке. Для этого начинаем с корня BSP-tree и рекурсивно спускаемся по дереву, выбирая поддерево, которое содержит более близкие к заданной точке точки. Если расстояние от гиперплоскости до заданной точки больше, чем текущее расстояние до k -го ближайшего соседа, то не ищем в этом поддереве. Иначе ищем в этом поддереве и добавляем найденные точки в список ближайших соседей. После этого проверяем, есть ли еще поддеревья, которые могут содержать более близкие точки, и продолжаем поиск.

BSP-tree похож по своим характеристикам на VP-tree, он может быть эффективным для поиска ближайших соседей в многомерном пространстве, так как он позволяет быстро находить ближайшие точки, используя разбиение на поддеревья. Однако, также как с VP-tree, он тяжеловесен - он может потребовать больше ресурсов для построения и поддержки индекса, чем Geohash и Quadtree.

Иные плоские структуры данных

Uber H3

Uber H3 - это сетка гексагональных ячеек, которая используется для геоиндексации и геопоиска в приложениях компании Uber, которая, собственно, и разработала данную систему. Каждая ячейка имеет уникальный идентификатор и может быть использована для определения местоположения объектов. Логика работы Uber H3 аналогична логике Geohash за основным исключением, что в Uber H3 используются шестиугольники, а в Geohash - прямоугольники.

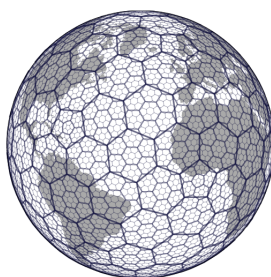


Рисунок 6 — Разбиение сферы земли на шестиугольники при использовании H3

Сравнение Uber H3 и Geohash:

- а) Размер ячеек: В Geohash размер ячеек зависит от уровня разбиения, тогда как в Uber H3 размер ячеек фиксирован и зависит от уровня разбиения.
- б) Форма ячеек: В Geohash ячейки имеют форму прямоугольников, тогда как в Uber H3 ячейки имеют форму правильных шестиугольников.
- в) Идентификаторы: В Geohash идентификаторы ячеек представляются строками, тогда как в Uber H3 идентификаторы представляются числами.
- г) Пространственная точность: В Geohash пространственная точность зависит от уровня разбиения, тогда как в Uber H3 пространственная точность зависит от размера ячеек.
- д) Поддержка многомерных данных: Uber H3 поддерживает многомерные данные, такие как время и высота, тогда как Geohash не поддерживает такие данные.
- е) Эффективность: Считается, что Uber H3 быстрее и эффективнее, чем Geohash, при работе с большими объемами данных. Данный тезис будет проверять в практической части данной работы.
- ж) Назначение: Geohash хорошо подходит для геопоиска и геоиндексации в небольших масштабах, тогда как Uber H3 хорошо подходит для геопоиска и геоиндексации в высоконагруженных системах с большим объемом данных.

Таким образом, выбор между Uber H3 и Geohash зависит от конкретных требований приложения. Если необходима высокая эффективность и поддержка многомерных данных, то лучше использовать Uber H3. Если же требуется простота и удобство использования в небольших масштабах, то Geohash может быть более подходящим выбором.

S2 geometry

S2 Geometry - это библиотека для работы с геометрическими объектами на сфере, разработанная компанией Google.

Основой S2 Geometry является иерархическая структура данных, называемая S2 Cell. Каждая ячейка S2 Cell представляет собой квадрат на сфере, который может быть разбит на более мелкие квадраты более высокого уровня. Уровень ячейки определяется числом n , которое указывает на количество разбиений квадрата на подквадраты. Чем больше значение n , тем меньше размер каждой ячейки.

S2 Geometry имеет ряд преимуществ перед Geohash. Во-первых, S2 Geometry использует более точную геометрическую модель, которая позволяет более точно представлять объекты на сфере. Во-вторых, S2 Geometry имеет более эффективный алгоритм поиска ближайших объектов, что делает ее идеальной для использования в приложениях, где требуется быстрый поиск объектов в заданном радиусе. В-третьих, S2 Geometry лучше масштабируется, что делает ее идеальной для использования в высоконагруженных системах.

Однако у Geohash есть свои преимущества. Во-первых, Geohash проще в использовании и имеет более простую структуру данных, что делает его более доступным для новичков. Во-вторых, Geohash имеет более широкую поддержку в различных языках программирования, что делает его более универсальным и гибким.

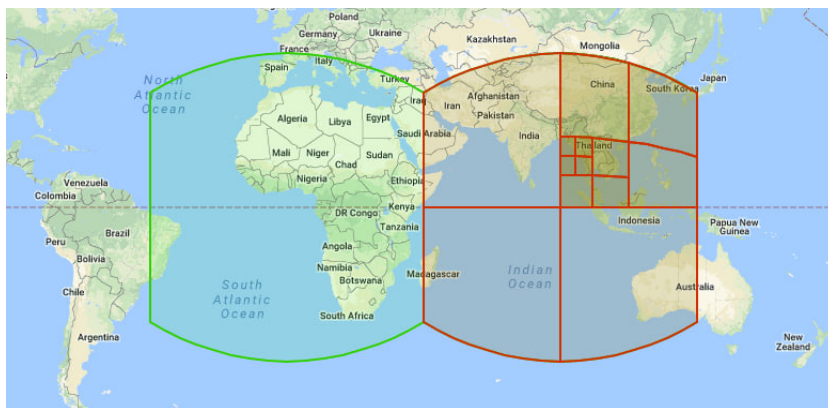


Рисунок 7 — Пример разбиения пространства на области с использованием S2 Geometry

Таким образом, выбор между S2 geometry и Geohash зависит от конкретных требований приложения. Если необходима высокая эффективность и поддержка многомерных данных, то лучше использовать S2 geometry. Но как и в случае с H3, Geohash более прост и популярен, поэтому требуется меньше человеческих ресурсов для разработки.

Сравнительный анализ

Чтобы дать корректный ответ на вопрос о том, какую структуру данных стоит использовать, и какие алгоритмы к ней применять, введем следующие критерии сравнения:

- а) Занимаемый объем оперативной памяти структурой данных - в байтах и в O-большой нотации
- б) Время предподготовки данных: в секундах и в O-большой нотации
- в) Время поиска K ближайших соседей: в секундах и в O-большой нотации
- г) Время поиска по радиусу и точке: в секундах и в O-большой нотации
- д) Время вставки точки: в секундах и в O-большой нотации
- е) Время удаления точки: в секундах и в O-большой нотации

При этом для чистоты эксперимента стоит добавить следующие ограничения:

- а) Не будет учитываться время записи на диск. Предполагается, что все операции производятся при использовании оперативной памяти.
- б) Требуется учитывать как теоретические показатели, то есть O-нотацию, так и практические результаты.

Качественный сравнительный анализ

Методы геопоиска и структуры геоиндексации могут использоваться в различных сценариях, включая навигационные приложения, сервисы доставки еды и товаров, системы мониторинга транспорта и т.д.

В навигационных приложениях методы геопоиска могут использоваться для определения местоположения пользователя и построения маршрута до заданной точки. В этом случае наиболее эффективным методом может быть использование алгоритмов S2 geometry или Geohash, которые позволяют быстро находить ближайшие объекты на карте.

В сервисах доставки еды и товаров методы геопоиска могут использоваться для определения ближайшего ресторана или магазина, а структуры геоиндексации - для поиска объектов в заданном радиусе от точки доставки. В этом случае наиболее

эффективными могут быть R-tree или Quadtree, которые позволяют быстро искать объекты в заданном радиусе.

В системах мониторинга транспорта методы геопоиска и структуры геоиндексации могут использоваться для отслеживания местоположения транспортных средств и определения ближайшего транспорта для пользователя. В этом случае наиболее эффективным методом может быть использование R-tree, который позволяет быстро искать объекты в заданном радиусе, а также определять их геометрические свойства (например, форму и размер).

Таким образом, выбор методов геопоиска и структур геоиндексации зависит от конкретной задачи и сценария использования. Необходимо учитывать требования к производительности, точности и эффективности поиска объектов на карте.

ЗАКЛЮЧЕНИЕ

В ходе исследования был проведен сравнительный анализ применения методов геопоиска и структур геоиндексации в высоконагруженных системах. Было выявлено, что все структуры имеют свои преимущества и недостатки, и выбор между ними зависит от конкретных задач и требований к системе.

Древесные методы R-tree, KD-tree позволяют быстро находить объекты на карте, но требуют больших вычислительных ресурсов при обработке больших объемов данных. Geohash, Uber H3, S2 Geometry, в свою очередь, позволяют эффективно хранить и обрабатывать большие объемы геоданных, но могут быть менее точными в поиске объектов.

Таким образом, выбор между методами геопоиска должен основываться на конкретных требованиях к системе и ее возможностях. Важно учитывать как скорость поиска объектов, так и эффективность использования ресурсов системы.

При этом важно отметить, что в высоконагруженных системах данные методы могут комбинироваться для достижения наилучших результатов, например, использовать Geohash для поиска в местах, где не требуется точность, а R-tree для аналитики данных через холодные хранилища.

В работе удалось полностью достичь поставленных целей.

В следующем семестре будет решаться более прикладная проблема, а именно сравнительный анализ конкретных реализаций структур с целью выявления узких мест в реализациях. Потребуется реализовать все приведенные структуры и провести нагрузочное тестирование, чтобы можно было найти все крайние случаи работы указанных подходов. Для решения данной задачи будут применяться научная дисциплина Информатика, для реализации структур и проведения нагрузочного тестирования будет использоваться язык Golang.

За время выполнения работы мною было познано знание относительно формирования геохешей, а также устройство работы многомерных деревьев: R-tree, KD-tree и тд.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- а) A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. Proceedings of ACM SIGMOD, pages 47-57, 1984
- б) N. Beckmann, H .P. Kriegel, R. Schneider and B. Seeger. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles. Proceedings of ACM SIGMOD, pages 323-331, May 1990.
- в) N. Roussopoulos, S. Kelley and F. Vincent. Nearest Neighbor Queries. ACM SIGMOD, pages 71-79, 1995.
- г) Sahr, Kevin (2019). Central Place Indexing: Hierarchical Linear Indexing Systems for Mixed-Aperture Hexagonal Discrete Global Grid Systems. Cartographica: The International Journal for Geographic Information and Geovisualization, 54(1), 16–29. doi:10.3138/cart.54.1.2018-0022
- д) Jiajun Liu, Haoran Li, Yong Gao, Hao Yu, Dan Jiang. [IEEE 2014 22nd International Conference on Geoinformatics - Kaohsiung, Taiwan (2014.6.25-2014.6.27)] 2014 22nd International Conference on Geoinformatics - A geohash-based index for spatial data management in distributed memory. 2014