基于列式数据库的大规模违建房屋面数据管理

吴然

广州市城市规划勘测设计研究院

摘　要：针对违建治理信息化系统中大规模房屋面数据存储效率瓶颈的问题，设计了基于主流列式数据库的空间房屋面矢量存储策略，同时设计了空间索引存储结构与对应的空间检索策略，以获取高效的房屋面数据存储、检索效率。通过开源房屋面数据集进行了效率验证试验，结果表明方案具备较高的实际效率。

关键词：城市违建治理；空间大数据；Geohash；列式数据库

【DOI】10.12252/j.issn.2096-6288.2022.12.239

一、引言

随着我国现代化建设的逐步推进，各大中心城市特别是一二线经济发达城市违法建设现象频出，对经济社会的健康发展与人民群众的人身安全造成了严重的威胁，对违法建设行为的监督、查处、整治也成了各地城市管理部门的重要任务。随着近年来数字政府、数字城市的建设，各级城市管理部门也引入了信息化手段来为城市违法建设执法提供支撑，相关的信息化系统需要支撑海量具有时空信息、复杂的关联属性的房屋面数据存储[1]。传统的关系型数据库（如PostgreSQL+PostGIS、Oracle Spatial等）在面对海量空间矢量数据的存储时，容易出现效率瓶颈[2][3]，难以支撑时空大数据的快速分析、检索任务。本文主要研究基于非关系型数据库进行大规模空间矢量数据存储的解决思路，利用列式数据库为基础，对其进行空间特性改造，并利用该存储策略测试其在大规模城市建筑房屋面数据下的效率。

二、存储策略设计与实现

1.数据结构设计

本文选取了ClickHouse[8]数据库来提供存储服务，ClickHouse是一款面向OLAP的列式存储DBMS，其具有轻量级、分列存储、高压缩等特点，使其天然擅长进行大批量数据的写入、检索操作，同时又极易使用。ClickHouse内部实现了多种数据库引擎，不同的引擎对数据的写入、压缩、索引建立等进行了不同的设计，使其各自能够适用于不同的应用场景，其中MergeTree族表引擎下的VersionedCollapsingMergeTree就实现了数据状态的异步合并，可以支持数据的“状态”记录，需要借此特性来间接实现数据的更新、删除操作。

房屋面数据主要分为空间数据、属性数据两部分，其中空间数据包括空间描述以及对应的空间索引信息，实现空间索引较为主流的方式包括格网索引、Geohash[4][5]、Google S2[6][7]等。本文选取了Geohash作为空间索引的生成算法，为能够保证空间索引在城市房屋面存储与检索等业务上具备足够的精度，采用了15级的Geohash索引划分，即对应的Geohash二进制编码需要采用30比特进行存储，采用ClickHouse内部支持的Int32类型字段进行索引信息的存储，前30位用于实际编码数值的存储，剩余的2位则填充为0，防止编码的十进制值发生负数突变。由此设计，由于单个房屋面可能涉及多个Geohash区块，即需要通过多个Geohash索引值代表其空间位置，所以在房屋面的数据表的实现上，采用Int32数组类型字段进行房屋面空间索引的存储。

根据以上设计，房屋面数据表的结构如表1所示，其中sign与version字段分别控制数据的删除状态与更新状态（创建数据表时需要声明，如式1所示），默认情况下，新录入的数据sign设置为1，version为1。当需要删除该数据时，插入一条UUID一致且sign为-1的数据记录，VersionedCollapsingMergeTree引擎会自动对数据库进行数据折叠，清除对应的数据记录；当需要更新数据时，插入一条UUID一致且version递增的数据记录，数据请求时取version值为最大的数据为最新数据记录。

```
CREATE TABLE table1
(
`UUID` UInt64,
`indexhash` Array（Int32）,
`timestamp` DataTime,
`properties` String,
`sign` Int8,
`version` UInt8
）ENGINE = VersionCollapsingMergeTree
（sign，version）
ORDER BY DataID
```

（式1）

表1 房屋面数据表结构

字段名	字段类型	说明
UUID（*）	uInt64	数据的唯一ID，主键
indexhash	Array（Int32）	Geohash索引序列
timestamp	DateTime	数据时间戳
properties	String	存储相关属性的JSON结构
sign	Int8	数据状态
version	UInt8	数据版本

至此本文实现了基于ClickHouse的房屋面空间数据与对应索引信息的存储，基于列式数据库的存储特性，将设计对应的数据检索策略，以利用列式数据库的存储特点提升空间数据操作的效率。

2.空间检索策略

ClickHouse数据库支持针对大规模数组进行SQL-IN语法操作，基于此特性，根据检索区域转化得到对应的Geohash序列后，对数据表中的空间索引字段进行SQL-IN即可实现数据的快速筛查。但由于空间检索可能会涉及较大的空间范围，即对应的检索区域Geohash序列会是一个长数组，数组的生成以及SQL-IN的效率都会受到明显的影响。所以这里采用对检索范围分情况处理的方法优化检索策略。

在进行小范围的数据检索时，所采用的检索策略如图1所示，即直接将该检索区域转化为对应的Geohash序列，同时利用ClickHouse支持的arrayJoin内置函数将房屋面数据拆解为多个房屋面记录序列，直接与检索区域转化的Geohash序列进行SQL-IN操作即可得到检索结果，如表2所示，蓝色区域为检索框生成的Geohash序列，红色与绿色框分别为不同的房屋面拆解后得到的Geohash序列，通过分析序列的叠加情况即可实现空间索引层面的检索。实现的SQL语句如式2所示。

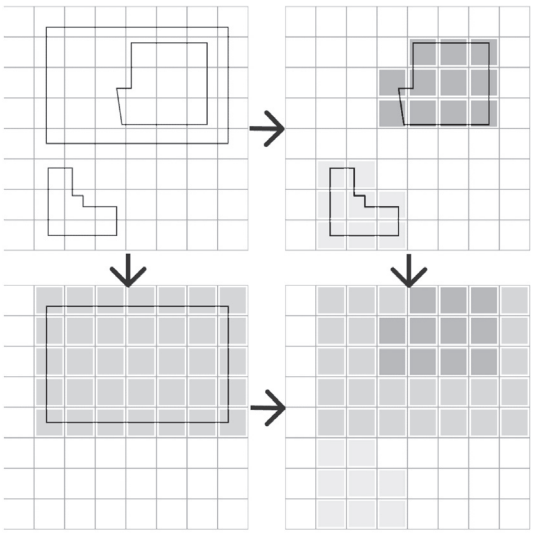

图1 小范围检索示意

表2 arrayJoin操作效果

arrayJoin前		arrayJoin后	
UUID	Indexhash	UUID	Indexhash
1	[3492882, 3492883]	1	3492882
		1	3492883
2	[5810101, 5810102, 5810103]	2	5810101
		2	5810102
		2	5810103

```
SELECT
  UUID
FROM `default`.table1
WHERE UUID IN (
  SELECT
    DISTINCT UUID
  FROM (
    SELECT
      UUID,
      arrayJoin (indexhash) AS split_index
    FROM `default`.table1
    WHERE split_index IN (G1,G2,…GN)
  ) sub_tab1
) c
GROUP BY UUID, version
HAVING sum (sign) >0
```

（式2）

当检索的区域为较大的区域时（实际实现时判断标准为检索范围外包框边界最长边超过0.5°），此时不直接进行检索区域的Geohash序列的求解，采用Geohash编码值连续性的特点进行数据检索优化。Geohash编码值在某一层下的划分区块范围在后续层级下的划分都为连续的编码序列，如图2所示，蓝色区块由于分别出于完整的层级1的区块下，所以其各自后续内部的划分可表示为连续的编码序列，而红色区块则无法实现连续，会存在编码值的突变。

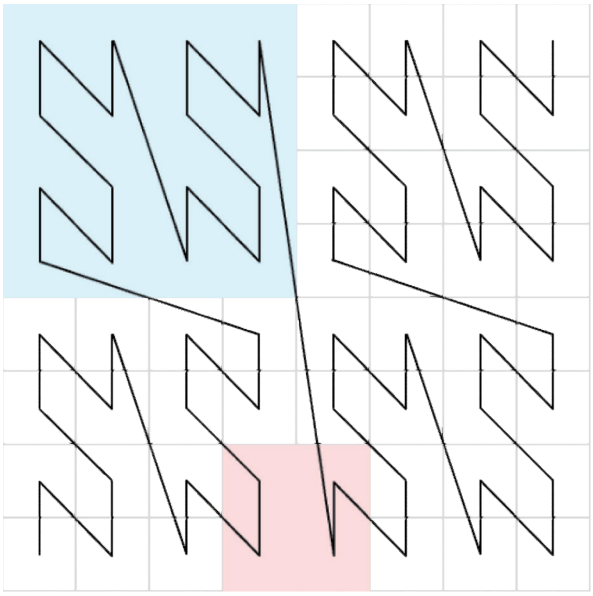

图2 Geohash子区块编码连续性示意图

利用这个连续的特性，将检索区域划分为内部连续的子区域序列，如图3所示，由于各子区域内部为连续，所以只需要采用子区块下的编码最大值与最小值即可代表该区域范围。子区块的编码最大值与最小值的获取方式为对当前完成编码的二进制序列的剩余未分割的编码位分别填充1与0实现，假设当前子区块为第三层级下的子区域，此时已完成分割的编码序列（即前6位）为100111，则只需要将其剩余的编码位填充为1即可得到最大编码，填充为0即可得到最小编码。

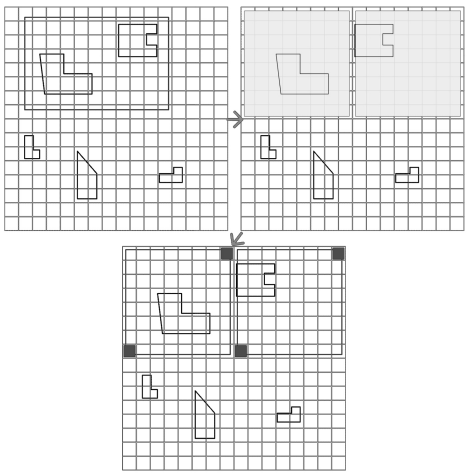

图3 大范围检索示意

完成利用子区域的最大最小编码对该区域的表示后，只需要对arrayJoin拆解后的离散房屋面记录序列采用数值范围检索即可实现空间数据检索，实现的SQL语句如式3所示。

```
SELECT
  UUID,
FROM `default`.table1
WHERE UUID IN (
  SELECT
    DISTINCT UUID
  FROM (
    SELECT
      UUID,
      arrayJoin (indexhash) AS split_index
    FROM `default`.table1
    WHERE
      (split_index >=G_min1 AND split_index <= G_max1) OR
      (split_index >= G_min2 AND split_index <= G_max2) OR
      ⋮
      (split_index >= G_minN AND split_index <= G_maxN)
  ) sub_tab1
) c
GROUP BY UUID, version
HAVING sum (sign) >0
```

（式3）

在实际的实现中，首先会对大范围的检索区域进行最小外框计算，寻找可利用单个分割区域表示该检索区域的最高分割层级，基于此分割层级的基础上加2作为该检索区域的子区域的分割层级，基于此方式在大多数情况下可获得尽量小的子区域范围，实现更精确的空间数据检索。

三、实验验证

基于以上的设计思路，本文将具体的数据存储流程、索引生成方式、数据入库流程、数据检索接口等基于jdk11进行了封装实现，采用的Clickhouse版本为v22.5.1。同时为了验证大规模房屋面数据下本框架的检索效率，基于开源的OpenStreetMap数据集搭建了房屋面数据仓库，房屋面数量达1000万，所采用的运行环境参数如表3所示。实验分别在亚洲经纬度范围内随机抽取10个区域，分别按照大范围检索、小范围检索的场景生成检索框进行空间检索实验，平均耗时如表4所示。

表3 实验环境

序号	配置项	配置
1	处理器	Intel（R）Core（TM）i7-5960X @ 3.00GHz
2	内存	64.0GB

表4 实验结果

类型	生成方式	平均耗时（毫秒）
小范围检索	最大跨度0.01°～0.05°	1674
大范围检索	最大跨度1°～5°	1852

以上可见，在千万级的房屋面数据量下，本文存储策略可实现较高的响应速度，可支撑实际业务中对房屋面数据的实时检索、获取需求。

四、结语

本文主要基于违法建设治理信息化建设中，大规模房屋面数据存储、检索效率瓶颈问题，设计了一种基于主流列式数据库的大规模房屋面存储管理与检索方案，实验表明设计思路具备可行性且能够实现较高的检索效率。本文仍存在部分不足，主要是框架所提供的空间操作较为单一，后续实践研究中会基于此思路继续丰富完善架构体系。

参考文献

[1]李德仁.展望大数据时代的地球空间信息学[J].测绘学报，2016，45（4）：379-384

[2]龚健雅，李德仁.论地球空间信息服务技术的发展[J].测绘通报，2008，No.374（05）：5-10.

[3]李清泉，李德仁.大数据GIS[J].武汉大学学报·信息科学版，2014，39（6）：641-644.

[4]金安，程承旗，宋树华，陈波.基于Geohash的面数据区域查询[J].地理与地理信息科学，2013，29（05）：31-35.

[5]胡晨希，刘会侠，乐鹏，王艳东.一种可扩展的线性可排序四叉树空间索引[J].测绘信息与工程，2010，35（05）：1-3.

[6]封旗杰. 基于Google S2算法的大规模实时查找与并行优化研究[D]. 湖南大学，2020.

[7]冯凌峰.一种基于Google S2 空间索引组织的海量轨迹数据区域查询方法，CN112347109A[P].2021.

[8]Alexey，et al.ClickHouse is a free analytics DBMS for big data[EB/OL].[2019-4-8].https: //github.com/ClickHouse/ClickHouse.

作者简介：吴然，1994年7月生，男，助理测绘工程师，广州市城市规划勘测设计研究院，主要从事测绘地理信息工作。