# LISA: A Learned Index Structure for Spatial Data

**摘要:** 在空间查询处理中，流行的 R 树索引可能会导致大量的存储消耗和高IO成本。受最近学到的索引[17]的启发，该索引用机器学习模型替代了B树，我们研究了空间数据的类比问题。我们提出了一种新颖的用于空间数据的学习索引结构（简称LISA）。其核心思想是使用机器学习模型，通过多个步骤，为任意空间数据生成可搜索的磁盘页数据布局。特别是，LISA 包括一个映射函数，将空间键（点）映射到一维映射值，一个学到的碎片预测函数，将映射空间划分为碎片，以及一系列将碎片组织成页面的本地模型。基于 LISA，设计了一种范围查询算法，然后是一个格回归模型，使我们能够将 KNN 查询转换为范围查询。还为 LISA 设计了处理数据更新的算法。大量实验证明，LISA 在存储消耗和查询的IO成本方面明显优于 R 树和其他替代方案。此外，LISA 能够高效处理数据插入和删除。

**1.介绍**

为了支持高效的查询，几十年来，空间数据库一直依赖于精细的索引。其中，R 树[13]是最流行的空间索引之一，用于修剪与查询无关的数据，即避免访问没有所需数据的数据块。然而，在大空间数据时代，这种经典的以索引为中心的范式越来越受到挑战。

一方面，不断增长的数据量需要大型的 R 树。有时，R 树比其底层数据集更大，特别是如果数据具有更多的空间维度、非空间属性较少和/或 R 树的磁盘页利用率[21]较小。这给空间数据库带来了严重的存储压力，并通过过多的树节点访问减缓了搜索算法。另一方面，大空间数据的速度，例如位置的快速更新，导致 R 树非常频繁地过时。为了保证 R 树的新鲜度，我们必须对树执行频繁且高效的更新。这使得实现和维护 R 树变得困难，而在速度和体积问题共存时影响更加严重。因此，今天 R 树及其变体[4, 15]在为大空间数据建立索引方面存在不足。空间数据库需要超越传统思维、改变范式的创新，以替代 R 树，以满足大空间数据的需求。

最近的一项工作[17]将 B 树索引的一维数据替换为递归模型索引（RMI），该索引包含一系列分层的机器学习模型。RMI 假定数据已排序并保存在内存密集数组中。给定搜索键 x，RMI 预测 x 的数据在数组中的位置，带有一些误差界限。位置预测等同于近似所有搜索键的累积分布函数（CDF）。RMI 能够处理一维空间中的点查询和范围查询。使用学习的 CDF F(x)，要获取所有位于 [x1, x2] 的键，RMI 计算 F(x1) 和 F(x2)。存储在位置 F(x1) - ϵ 到 F(x2) + ϵ 之间的键形成范围查询的预测答案，其中 ϵ 是误差界限。

然而，在空间数据的背景下，RMI 的思想并不适用。首先，空间数据使得 RMI 所需的一维数据假设失效。尽管可能学习多维 CDFs，但这样的 CDFs 无法提供仅通过模型预测进行的范围查询的简洁下界/上界。相反，这样的 CDFs 将导致在一个维度上合格但在所有维度上不合格的本地区域的搜索。而且，当数据驻留在磁盘上时，即使有一个小的误差界限 ϵ，也意味着为定位一个不在预测页上的搜索键而访问几个额外的页面。在处理需要加载多个页面的空间查询时，RMI 将不得不访问许多页面，产生相当大的IO成本。

最近的 Z-order 模型（ZM）[29]将 Z-order 空间填充曲线[25]和 RMI 结合起来以对空间数据进行索引。然而，它需要为范围查询检查许多不相关的键，并且不支持KNN查询或数据更新。

在本文中，我们提出了一种新颖的面向空间数据的学习索引结构（LISA）。其核心思想是利用机器学习模型，通过几个精心设计的步骤，为任意空间数据集在磁盘页面上生成可搜索的数据布局。它采用了一种简单而有效的策略——基于数据沿一系列轴的分布，将空间划分为一系列网格单元，并沿着这些轴对这些单元进行编号。LISA根据单元的边界构建了一个部分单调函数M，将数据从Rd映射到R。如果两个空间点，即键x0和x1分别在单元Ci和Cj中，且i < j，则保证M(x0) < M(x1)成立。此外，基于键的映射值，LISA学习了一个单调函数SP，由一系列分段线性函数组成，为每个映射值分配一个碎片ID。通过调整SP的参数，每个碎片包含相似数量的键。通过构建一个本地模型来管理每个碎片的键，属于同一碎片的键存储在一个或多个连续的磁盘页面上，而具有不同碎片ID的两个键存储在两个不同的页面上。总体而言，本地模型负责分配页面以存储键，并分裂或合并页面以更新数据。因此，我们的LISA也能够处理包括插入和删除在内的数据更新。与RMI [17]不同的是，后者首先固定数据布局，然后学习模型来近似布局，而LISA则使用学到的模型生成数据布局（碎片）。

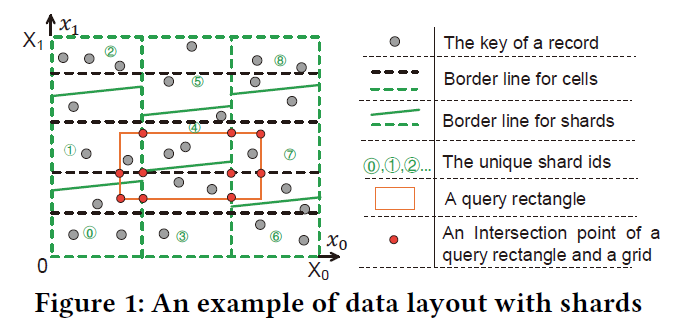


图1展示了LISA中的碎片布局示例。要访问在范围查询矩形qr中的所有键，我们只需要搜索与该矩形重叠的所有单元。对于每个这样的单元Ci，假设qri = qr ∩ Ci = [l0,u0) × · · · × [ld−1,ud−1)]。我们计算ml = M(l0, . . . , ld−1)和mu = M(u0, . . . ,ud−1)，使用SP获取与M^(-1) [ml ,mu )相交的相关碎片，然后使用本地模型访问落在qri中的键的数据。

在LISA中，KNN查询被转换为范围查询。我们提出了一个格回归模型LR，用于估计给定查询点和K的范围查询矩形大小。随后，执行具有估计矩形大小的范围查询。如果范围查询返回少于K个最近邻居，将为另一个范围查询扩充范围，直到找到足够的邻居。

LISA的存储消耗比需要根据MBRs和父子关系构建包含所有节点和条目的树的R树大得多。相比之下，LISA只保留用于KNN查询的M、SP、本地模型和LR的参数。具体而言，M的参数仅包含几个数字和一个小列表，SP由一系列分段线性函数组成，其参数是一些系数。本地模型的参数通常也只是几个数字，而存储格回归模型所需的空间很小。

总体而言，与传统的R树相比，LISA在多个方面都取得了显著的改进。首先，LISA的存储消耗要小得多。其次，在处理范围和KNN查询时，LISA的IO成本要少得多。第三，LISA高效支持数据更新，即插入和删除。

本文的贡献总结如下。

• 我们设计了LISA，一种面向磁盘存储的空间数据的新颖学习索引结构。据我们所知，这是空间数据的第一个完整的学习索引。（第3节）

• 我们设计了一种使用LISA处理范围查询的高效算法，然后是一个格回归模型，使我们能够在LISA中将KNN查询处理为一系列范围查询。（第4节）

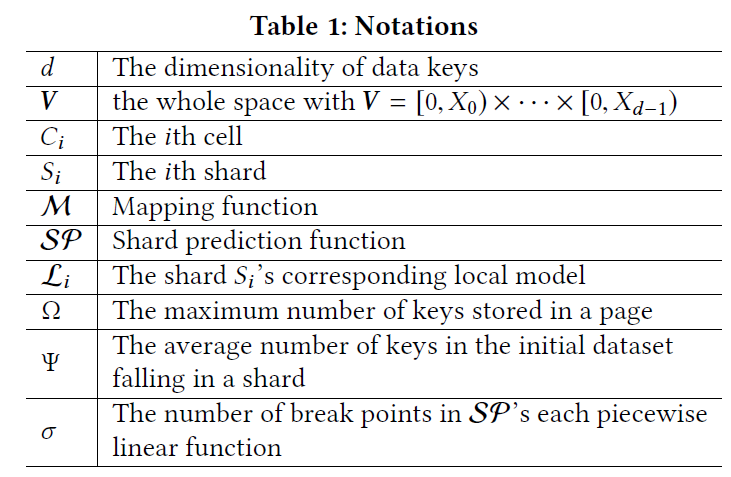
• 我们设计了高效的算法，以更新LISA以进行键插入和删除。（第5节）

• 我们使用真实和合成数据进行了广泛的性能评估。结果表明，在范围和KNN查询的存储消耗和IO成本方面，LISA优于替代方法。此外，LISA能够高效处理数据更新。（第6节）

此外，第2节对研究问题进行了阐述并介绍了我们的解决框架，第7节回顾了相关工作，第8节总结了论文并指出了未来的研究方向。

**2.前言**

本节介绍 LISA 的定义、基线方法和概述。 表 1 列出了重要的符号。



**2.1定义**

为了不失一般性，我们工作在一个d维空间V=[(0,X0)×…×[0,Xd-1]] ⊆Rd

定义1(键) 键k 是数据记录的唯一标识符，其中 k = (x0, ..., xd−1) ∈ Rd 。

我们有0 ≤ xi < Xi（0 ≤ i < d.键(x0, ..., xd−1)也是V中的一个点.我们用网格将V划分为单元

定义2(单元) 一个网格单元是V中的一个高维矩形，其下角和上角分别是点(l0, . . . , ld−1)和点(u0, . . . , ud−1)。也就是cell=[l0,u0) × · · · × [ld−1,ud−1).

除了网格之外，我们还使用映射函数将每个 d 维点映射为连续顺序。

定义3(映射函数)映射函数M是一个区域V上非负区间局部单调的函数, 也就是M:[0,X0] ×…×[0,Xd-1）->[0,+∞））,M((x0,…,xd-1))≤M(((y0,…,yd-1)))当x0≤y0,…,xd-1≤yd-1.

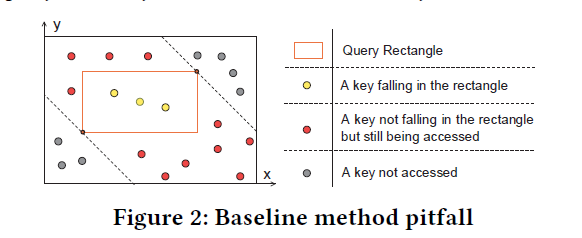
在我们的设置中，映射值被用来组织空间数据库序列化磁盘页上的空间键。对于页P，P.keys表示存在P上的键集合的映射结果，即M(P) ≜ M(P.keys).对于页P和键k=(x0,…,xd-1),如果inf M(P) ≤M(k) ≤sup M(P)1,在我们的设置中键k必须被存在页P.在这种情况下，我们称页P包含键k

有了这些基本定义，我们设计了一种基线方法

2.2 基线方法

给定映射函数M,我们可以将标量值[17]上的范围查询的学习型索引拓展至空间数据。基线方法工作如下.我们将所有的键根据他们的映射值排序，将它们存储在多个页面中，每个页面都被充分利用，并将这些页面的地址存储在一个密集数组中。假设d = 2。如果一个点(x,y)的映射值大于前j页中存储的键的值，也就是, ,我们将(x,y)存在页Pj,也就是，(x,y)的页地址索引是j. 这里，前 j 页的地址恰好是密集数组中的前 j 项。 随后，对于查询矩形 qr = [l0,u0) × [l1,u1)，我们只需要预测 i1 和 i2，分别为 (l0, l1) 和 (u0,u1) 的索引，加载 i2 - i1 + 1 页，并扫描这些页以查找落在查询矩形 qr 中的键。

然而，基线存在一个严重的问题——可能会加载许多与查询矩形无关的页面，从而产生大量不必要的 IO。 图2显示了映射函数为M的示例，其中映射函数为M(x,y)=x+y。因此，数据集被划分为三个部分。 查询矩形完全落在第二部分内。 这会导致仅包含三个相关键的范围查询访问许多（即 11 个）不相关点。



假设G=M-1([M((l0,l1)),M((u0,u1))]).在基线方法中，G\qr可能很大并且包含很多键，这提高了范围查询的IO消耗.查询矩形的位置和形状并没有提前固定。当固定映射函数 M 时，Lebesgue测度 [24] μ(G \ qr ) 对于大多数矩形来说都很大，这导致许多键被不必要地加载。 为了解决这个问题，我们设计了一种适用于任意空间数据集的通用学习索引方法 LISA。

2.3 LISA 概述

我们的 LISA 远不止映射函数 M 生成的单元格。相反，我们也对映射到 [0, +∞) 范围内的已知区间的密钥集感兴趣。

定义4(碎片) 分片 S 是映射函数 M 下区间 [a, b) ⊆ [0, +∞) 的原像，即S=M-1([a,b])

给定一个初始数据集，我们基于数据分布构建一个映射函数M. 在获得所有键的映射值后，学习单调分片预测函数 SP 将键划分到不同的分片中。假设映射值mi=inf SP1-1([i,i+1])和mi+1=inf SP-1([i+1,i+2])。第i个碎片Si≜ M-1([mi-1,mi))

在我们的设置中，所有分片都表现出相对于映射范围内相应间隔的总顺序。 此外，所有分片彼此不相交，它们的并集包含在 V 中。 根据映射函数和分片的定义，显然：

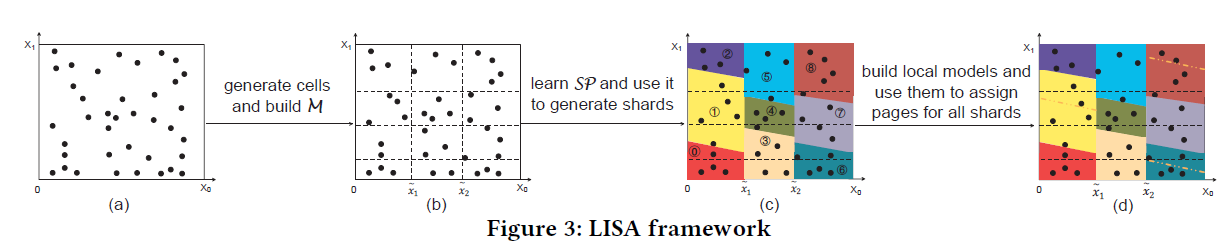


此外，分片中的键存储在多个磁盘页面中，而一页中的键必须来自同一分片。 相反，网格单元中的键可以对应于多个页面，反之亦然。 换句话说，分片和页面之间是一对多的关系，单元格和页面之间是多对多的关系。 如果页面 P 与网格单元 C ⊆ V 重叠，则 P 中存储的点落在 C 中。如果分片 S ⊆ V 包含页面 P，则 P 中存储的任何密钥落在 S 中。我们使用局部模型来确定 分片中键的页面地址。

定义 5（本地模型）。 本地模型Li是处理分片Si内的操作的模型。 它保留动态结构，例如 Si 包含的页面地址。

给定一个空间数据集，我们生成映射函数 M、分片预测函数 SP 和一系列局部模型。 在此基础上，我们构建了索引结构LISA来处理范围查询、KNN查询和数据更新。 与现有索引（例如 R-Tree）相比，LISA 旨在减少存储消耗和 IO 成本。

本质上，LISA 由四个部分组成：网格单元的表示（第 3.1 节）、映射函数 M（第 3.2 节）、分片预测函数 SP（第 3.3 节）和所有分片的局部模型（第 3.5 节）。 如图3所示，构建LISA的过程由构建四个部分的步骤组成。



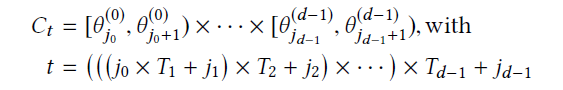
为了获取与查询矩形 qr 重叠的所有页面，我们首先将 qr 分解为许多较小的矩形，每个矩形仅与一个单元格相交。 对于每个这样的单元 Ci ，我们使用 SP 来选择与 qr 重叠的 Ci 中的分片。 最后，我们使用本地模型来获取与 qr 重叠的页面的地址。 详细信息在第 4.1 节中给出。 在LISA中，KNN查询被转换为一系列范围查询。 我们构建了一个点阵回归模型来估计查询点和 K 的适当查询范围。如果在范围查询中找到少于 K 个邻居，则查询范围会增加。 详细信息请参见第 4.2 节

要插入或删除键 k，第一步是计算 k 的映射值。 接下来，通过SP获得k的分片。 随后，本地模型将在相应的分片中定位相应的磁盘页面。 对于插入（第 5.1 节），如果页面已满，本地模型将拆分它。 对于删除（第 5.2 节），本地模型将计算要删除 k 的页面中存储的键。 可以执行不同的策略，例如，如果连续的页面各自包含的键太少，则可以合并连续的页面。

**3.LISA的设计和训练**

3.1生成网格单元

在每个维度上，我们沿着xi轴将空间V划分为Ti个部分这样初始数据集的键值能够均匀地被每一部分覆盖.我们使用Θi=[θ1(i),…, θTi(i)]来表示由 xi 轴的分区操作生成的边界点。整个空间V可以由一系列不相交单元的并集来表示也就是,其中



请注意，某部分沿 xi 轴的每个分区操作也可以使用一维单调回归模型执行。 模型的输入仅仅是该部分中按键的 xi 分量。 其输出范围为[0,Ti]。 当 Ti 很大时，保存模型可能比保留 θi 需要更少的存储空间。 实际上，每个Ti设置为一个较小的数字（小于300）。 在本文中，我们只保留这些边界点的列表。