计算机科学
COMPUTER SCIENCE

学习索引研究综述

王艺潭，王一舒，袁野

引用本文

王艺潭，王一舒，袁野.学习索引研究综述[J]．计算机科学, 2023, 50(1): 1-8.

WANG Yitan, WANG Yishu, YUAN Ye. Survey of Learned Index[J]. Computer Science, 2023, 50(1): 1-8.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000149

学习索引研究综述

王艺潭[1]　王一舒[1]　袁　野[2]
1 东北大学计算机科学与工程学院　沈阳 110169
2 北京理工大学计算机学院　北京 100081
(2001826@stu.neu.edu.cn)

摘　要　大数据时代数据呈爆发式增长,传统索引结构难以处理庞大复杂的数据,为解决这一问题,学习索引应运而生,并成为当前数据库领域的研究热点之一。学习索引利用机器学习模型进行索引构建,通过对数据和物理位置之间的关系进行训练和学习得到学习模型,掌握二者之间的分布特点和规律,从而实现对传统索引的改进和优化。大量实验表明,与传统索引相比,学习索引可以适应大规模数据集,提供更好的搜索性能,具有更低的空间要求。文中详细介绍了学习索引的应用背景,梳理了现有的学习索引模型;根据数据类型的不同,将学习索引分为一维和多维两种类别,并对每种类别中学习索引模型的优缺点和可以支持的查询进行了详细的介绍和分析;最后对学习索引的未来研究方向进行了展望,以期为相关研究提供参考。

关键词:学习索引;机器学习;索引构建;数据结构;数据库
中图法分类号　TP311

Survey of Learned Index

WANG Yitan[1], WANG Yishu[1] and YUAN Ye[2]
1 School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China
2 School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Abstract　Due to the explosive growth of data in the era of big data, it is difficult for the traditional index structures to handle this huge and complex data. In order to solve this problem, the learned index has emerged and become one of the most popular research topics in the database. Learned indexes employ machine learning models for index construction. By training and learning the relationship between data and physical location, the learning model can be obtained so as to master the distribution characteristics between the two to realize the improvement and optimization of the traditional index. Extensive experiments show that learned indexes can adapt to large-scale data sets, and provide better search performance with lower memory requirements than traditional indexes. This paper introduces the applications of learned indexes and reviews the existing learned index models. According to data types, learned indexes are divided into two categories: one-dimensional and multi-dimensional. The advantages, disadvantages, and supported searches of learned index models in each category are also introduced and analyzed in detail. Finally, some future research directions of learned indexes are prospected to provide references for related researches.

Keywords　Learned index, Machine learning, Index construction, Data structure, Database

1　引言

索引指一种对数据库表中一列或多列的值进行排序的一种存储结构,它是某个表中一列或若干列值的集合,也是指向表中物理标识数据页的逻辑指针列表。索引的作用相当于图书的目录,人们可以根据目录中的页码快速找到所需的内容。由于内存中的高开销,传统的索引结构难以处理爆炸式增长的数据,难以在有限的系统资源下提供低延迟和高吞吐量的性能。常见的传统索引数据结构可以分为:基于哈希的结构[1]、基于树的结构[2]、基于位图的结构[3]和基于前缀树(Trie)的结构[4]。但是,基于哈希的索引不支持前置搜索或范围搜索;基于位图的索引存储、维护和解压缩成本可能很高[5];基于前缀树的索引大多是基于指针的。因此,基于树的索引是数据库系统中最主要的数据结构。然而,随着数据集越来越大,传统索引占有的存储空间也越来越大,查找速度变得越来越慢,为了解决这些问题,学习索引这个概念进入了人们的视野。

近年来,随着机器学习(Machine Learning)的飞速发展,大量使用机器学习来优化数据库系统的研究相继出现,其中学习索引结构受到了广泛的关注。学习索引使用机器

到稿日期:2021-10-20　返修日期:2022-06-07
基金项目:国家重点研发计划(2022YFB2702100);国家自然科学基金(61932004,62225203,U21A20516)
This work was supported by the National Key Research and Development Program of China(2022YFB2702100) and National Natural Science Foundation of China(61932004,62225203,U21A20516).
通信作者:袁野(yuan-ye@bit.edu.cn)

学习模型取代传统的索引结构(如 B 树[6]、B+树[7]、LSM 树[8])来进行索引。学习索引与传统索引相比有以下 3 个显著的优势。

(1)对庞大的数据更加友好。原因是,与传统索引相比,学习索引在构建索引的过程中充分利用了底层数据分布的规律和机器学习模型的自身优势,只需要简单的线性回归模型就可以定位到对应记录位置。

(2)更好的查找性能。前文提到,基于树的索引是数据库系统中最主要的数据结构,但是这种索引导致每次查询都需要访问从树根节点到叶子节点路径上的所有节点。随着数据量的增大,查找性能越来越低。学习索引使用学习模型来代替这种树形结构,这意味着学习索引在进行查找时,不必像传统索引一样遍历所有数据,而是通过训练得到模型,并给出一个预测值,学习索引在预测值附近展开搜索。例如,当数据键值从 1 递增到 n,为了搜索一个特定的键,我们可以使用键本身的规律。如果使用传统索引,查询的时间复杂度为 $O(\log n)$,而学习索引只需要 $O(1)$的复杂度。

(3)更低的空间要求。传统索引需要借助额外空间来索引原始的数据,而在学习索引中,叶子节点可以通过存储模型或者线性函数的斜率和截距来降低空间开销。在前文提到的示例中,对数据键值从 1 递增到 n 的数据构建索引,学习索引的空间复杂度只需要 $O(1)$,而传统索引需要 $O(n)$。

由于学习索引能够充分利用数据分布情况,因此与传统索引相比,学习索引的优势明显。这使得学习索引逐渐在文档检索[9]、物联网[10]、网络日志[11]、位置服务领域(LBS)[12]、最长前缀匹配[13]和提高查询召回率[14]中得到了广泛应用。在各应用中,学习索引目前主要用于解决点查询、范围查询和 kNN 查询问题,但 kNN 查询在学习索引中的应用不及点查询和范围查询广泛,仅部分多维学习索引支持 kNN 查询。

学习索引的显著优势使其得到了广泛的应用,但是使用机器学习模型替代传统索引结构仍然面临以下 3 个方面的巨大挑战。

(1)适应能力不足。因为学习索引是通过原始数据分布和记录位置之间的关系进行学习和分析得到,所以学习索引在面对有序数据的主键索引时索引效果是最优的,但是对于二级索引或者是多维索引来说,使用学习模型替代其传统的索引结构较为困难,数据的杂乱使得学习模型的构建变得更加复杂,且保证一定的准确率成为构建学习索引的挑战之一。

(2)难以提供误差保证。在传统 B+树索引中,可以对存储的键提供最小和最大误差保证,保证在查找已有键时,该键就存在于当前页面中。但是通过学习和训练构建索引在预测位置时存在误差。如何缩小误差,以及如何保证预测位置和实际位置之间的误差在一定范围内成为了构建学习索引的第二个挑战。

(3)更新索引困难。在插入数据时,插入的位置会影响查找的性能,并且插入数据可能会违反模型预先设定的误差阈值。除此之外,在索引更新后,若只进行小部分更新,则会降低查询的速率;若进行大量更新,新数据会改变键的分布,导致模型不适用于现有数据。因此如何在保证查询性能的情况下,提高插入的吞吐量,以及如何在大量更新后降低重新训练的成本成为了索引更新的挑战。

本文梳理了现有的学习索引,如表 1 所列,从处理数据维度的角度,可以将学习索引分为一维学习索引和多维学习索引。接下来从数据划分的角度,将一维和多维学习索引再进行细分。对于一维学习索引,数据划分策略包括将底层数据平均划分到根节点上的等分策略和固定误差阈值的贪心算法策略。对于多维学习索引,可分为通过映射策略将多维数据映射为一维数据和网格划分底层数据,以及使用现有数据重新定义布局这 3 类。除此之外,大多数一维学习索引均支持插入,插入方式包括就地插入和异地插入两种。但是多维学习索引更新困难,仅有 LISA 实现了就地插入。为详细介绍表 1 中的所有索引,本文其余部分的结构安排如下:第 2 节介绍了一维学习索引的构建和查询过程,第 3 节介绍了多维学习索引的构建和查询过程,最后总结全文并对学习索引的未来工作进行展望。

表 1 现有学习索引概要

Table 1 Summary of existing learning indexes

维度	一维学习索引			多维学习索引		
数据划分	不划分	等分	贪心算法	映射为一维数据	网格划分	自定义布局
不支持插入	RMI[9]	RS[15], PLEX[16]	—	SageDB[17], ZM[18], ML[19]	Flood[20], Tsunami[21], SPRIG[22]	LSBI[23]
就地插入	—	ALEX[24], LIPP[25], Hermit[26], CARMI[27]	FITing-Tree[10], AIDEL[11]	—	—	LISA[12]
异地插入	—	XIndex[28]	FITing-Tree[10], PGM[29], SIndex[30]	—	—	—

2 一维学习索引

Kraska 等[9]于 2018 年首次提出了一维学习索引,并将其命名为递归模型索引(Recursive Model Index, RMI)。RMI 作为最早的学习索引,为后续的研究打下了坚实的基础。表 2 列出了一维学习索引构建技术的逐步完善,从 RMI 开始,逐步实现了多种更新策略、强语义保证、二级索引和并发等技术,使得一维学习索引逐步完善,并且适用场景越来越广泛。一维学习索引结构通常分为内部节点和叶节点两个部分,内部节点使用 RMI 模型或多层 RMI 模型等学习模型预测分支,叶节点使用线性回归模型预测记录位置,再使用局部搜索对预测结果进行校正。底层数据划分策略的不同导致

内部节点形成不同的类型分支。本文按照数据划分策略的不同划分小节，2.1 节描述无底层数据划分的一维学习索引，如 RMI；2.2 节描述等分策略划分底层数据的一维学习索引，如 ALEX，LIPP，XIndex 等；2.3 节描述使用贪心算法划分数据的一维学习索引，如 FITing-Tree 和 AIDEL 等。

表 2 一维学习索引优化

Table 2 One-dimensional learned indexes optimization

一维学习索引	优化
RMI(2018 年)	使用学习模型代替 B 树
FITing-Tree(2019 年)	可更新、确认误差范围
AIDEL(2019 年)	模型独立性
Hermit(2019 年)	支持二级索引
ALEX(2020 年)	插入空隙、批量加载
PGM(2020 年)	模型压缩、学习查询负载
RS(2020 年)	使用样条插值法拟合数据
XIndex(2020 年)	可并发
SIndex(2020 年)	可并发、索引字符串键
LIPP(2021 年)	精确映射记录位置
CARMI(2021 年)	利用缓存提升性能
PLEX(2021 年)	只有最大误差一个超参数

2.1 无底层数据划分

RMI 采用分层的模型结构，每层模型之间彼此独立，即每层可以采用不同的模型。总体索引结构如图 1 所示，RMI 将内部节点和叶节点连接在一起形成一个层次结构。其中，叶节点用来存储实际数据。内部节点中每个节点描述一个机器学习模型，用于该模型预测输入键值的相应分支。给定要搜索的键值，通过 RMI 预测键的位置。由于这个预测结果并不精确，因此采用二分搜索在叶节点进行查找，来确定数据记录的实际位置，蓝线表示预测位置的过程。

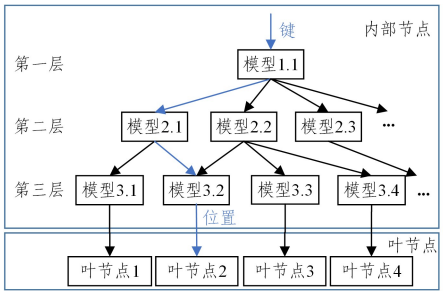

图 1 RMI 结构模型(电子版为彩图)

Fig. 1 RMI structure model

与传统的 B 树索引相比，RMI 具有以下两个优点：

(1)解决了“最后一英里”的准确性。RMI 设计最初只使用单个全连接神经网络模型来拟合整体数据分布，但由于数据本身具有随机性，因此很难准确拟合出个体数据。单个神经网络通常需要更多的空间和 CPU 时间才能减小个体的预测误差，即在单个数据级别的精确定位问题上面临困难，这个问题被称作是“最后一英里”的搜索准确性。分层的模型结构解决了缩小误差的问题，使得模型误差逐层递减，利用分层结构使“最后一英里”准确性的问题变得简单。

(2)构建模型的混合体。RMI 可以在不同层次上选择不同模型来实现不同的目的，例如图 1 中第一层可以使用神经网络，因为其能够学习大量复杂的数据分布，而第三层使用线性回归模型，因为它们在空间和执行时间上开销较小。利用这种混合体模型可以最大化每个模型的优势，从而提高模型的整体效率。实验证明，RMI 比传统 B 树访问效率高出 1.5～3 倍，空间占用仅是 B 树的 0.01～0.24 倍。虽然 RMI 在查询性能、存储上相比传统索引有所提高，但仍存在两点不足：1)RMI 只能处理对只读数据的查找，不支持更新操作；2)RMI 要求处理的数据是有序的。这两点不足使得 RMI 无法适用于实践中常见的动态读写工作负载。针对这两点不足，有大量工作对学习索引模型进行了改进。

2.2 等分划分数据

等分划分数据是将底层数据平均划分到根节点上，这使得其可以并行地在每段中训练模型。基于等分划分的索引一般采用“自上而下”的训练方式，首先拟合最顶层的模型，然后训练后续的层来纠正错误。

Ding 等基于 RMI 提出了 ALEX(Adaptive Learned Index)[24]索引。该索引基于 RMI 进行了许多改进，解决了在为包含点查询、范围查询、插入、更新和删除的混合工作负载实现学习索引时出现的实际问题。ALEX 索引为了实现更快的查找，不再采用固定的 RMI 结构，而是根据工作负载动态调整 RMI 的形状和高度，在叶节点使用线性回归模型对记录位置进行预测，并使用指数搜索纠正预测误差，图 2 中的蓝线箭头表示点查询时的查询路径。ALEX 索引为了实现插入和更新，使用间隙数组(Gapped Array)布局和基于模型的插入，如图 2 所示，ALEX 索引为每个数据节点(叶节点)使用了间隙数组，目的是将键插入到数据节点中模型所预测的该键应该在的位置。若插入位置是一个间隙，则直接插入，否则，在插入位置上通过向最近的距离方向移动一个位置来创建一个间隙，然后将元素插入到新创建的间隙中。此方法减小了模型的预测误差，而具有更好的搜索性能。在只读工作负载下，ALEX 的性能比 RMI 的性能高出 2.2 倍，索引大小比 RMI 高出 0.15 倍。但是同 RMI 一样，ALEX 索引默认数据按照索引列有序排列，并且无法提供插入性能的时间复杂度上限，这导致了大规模插入时性能的下降。

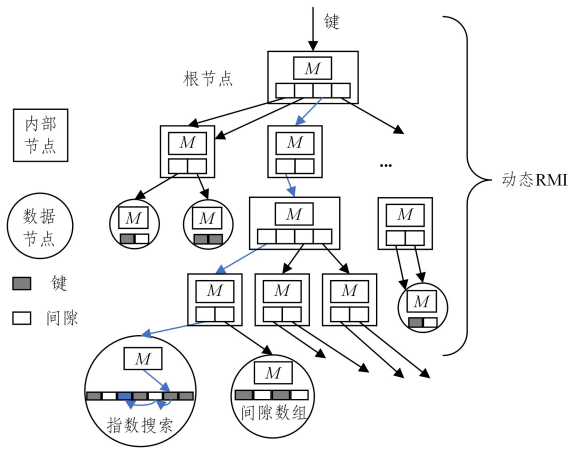

图 2 ALEX 的架构(电子版为彩图)

Fig. 2 Architecture of ALEX

为了解决上述问题，文献[25]提出了具有精确位置的可更新学习索引 LIPP(Learned Index with Precise Positions)。该团队认为 ALEX 虽然支持了更新操作，但更新操作会导致大量的元素移动，这些开销都是由学习模型的不精确预测带来的。因此 LIPP 改进了 ALEX 插入时不对树的高度进行约束的缺点，保证键到位置的映射是精确的。如果多个键映射到同一个位置，将创建一个新的子节点来保存这些键。为了

约束树状索引的高度，提出了核化线性模型，均匀地分配新插入的元素到位置的映射；以及利用一个轻量级的调整策略来保持树状高度的约束。LIPP 实现了比 ALEX 索引高 2.8 倍的查找性能。

Zhang 等对 RMI 的基本框架进行扩展，提出了 CARMI (Cache-Aware Recursive Model Index)[27]索引，它利用 CPU 缓存的行为特性来提高索引结构的性能。在 RMI 索引访问数据节点的过程中，对于每个树节点，至少需要一次内存访问来获取其内容。因此，内存访问所需的时间实际上会占用数据查找中的大部分时间。换句话说，该团队认为 RMI 的性能瓶颈是内存访问，因此他们对 RMI 进行了改进，提出了 CARMI 索引，通过强制每个节点的大小为 64 字节来实现查找数据时最小化内存访问次数。一方面，这种设计允许只通过一次内存访问来获取节点；另一方面，较大的节点大小允许它存储丰富的信息，这有助于减少树的平均深度，从而减少所需的内存访问数量。在索引构建上，CARMI 采用混合构建算法自动构建各种数据集和工作负载下的索引结构，无需任何手动调优。实验研究表明，与 ALEX 相比，CARMI 具有更好的时间效率和相似的空间使用量，二者在非线性和真实世界数据集上的差距更显著。

虽然 ALEX 索引和 LIPP 索引试图有效地处理学习索引的写操作，但它们在并发操作面前不能够保证正确性。为此，Tang 等提出了一种可更新的并发学习索引 XIndex[28]，它利用了两阶段压缩、读取-复制-更新(RCU)[31]和乐观的并发控制[32-33]的组合实现可更新的并发索引，这与 XIndex 的两层架构设计密不可分。顶层根节点使用学习的 RMI 模型来索引组，底层每个组节点使用学习的线性模型来索引其数据，组节点内包含线性模型、排序数组、缓冲区。缓冲区用于缓冲插入的记录，并通过模型索引的排序数组有条件地压缩，在执行并发操作时，缓冲区需要合并到排序数组，分为合并阶段和复制阶段。为了减少性能变化，XIndex 根据运行时工作负载特征调整其结构，通过结构更新操作(模型拆分/合并、组拆分/合并和根更新)保持错误界限和增量索引大小。评估表明，与现有传统并发索引[34-35]相比，XIndex 的性能优势高出 4 倍左右。

Wu 等提出了二级索引 Hermit[26]，它对 RMI 和 ALEX 等仅支持数据按照索引列有序排序做出了改进，利用隐藏在列中的软函数依赖关系[36-37]使其可以实现二级索引。二级索引可以显著提高涉及对非主键属性的选择的查询性能。与一级索引不同，非主键属性没有排序，可能包含重复项。Hermit 通过分层回归搜索树(Tiered Regression Search Tree，TRS)对数据库的同一数据表中的目标列 M 和主键列 N 之间的数据相关性进行建模，从而实现删除索引键访问的冗余结构。它利用分层回归方法在相关函数上自适应地动态执行曲线拟合。Hermit 能够高效地处理插入、删除和更新，并支持按需结构重组，以在系统运行时重新优化索引效率。

2.3 贪心算法划分数据

贪心算法旨在采用贪心的策略，保证数据划分满足规定的误差范围，从局部最优，最终达到全局最优。采用贪心算法划分数据的学习索引是“自下而上”训练的，首先将最底层拟合到一个固定的精度，然后构建后续层，以快速搜索最底层，找到合适的模型。

FITing-Tree[10]是最早的基于贪心策略划分数据的一维学习索引，它是基于传统的集群 B+树开发的，如图 3 所示，FITing-Tree 采用 B+树的内部节点，但与传统 B+树不同的是，FITing-Tree 叶节点使用分段线性函数的起始键和斜率来近似索引。FITing-Tree 支持非聚簇索引，其中关键的一步是通过索引键对数据进行排序，并按排序顺序物化指向每个数据项的指针数组来构建间接层。接下来，与聚簇索引的步骤一样，分段算法扫描间接层并生成有效的段集，然后将这些段集插入到上层树中。这里分段采用的是一种类似于 FSW[38-39]的贪心算法 ShrinkingCone，它具有较低的固定内存使用，并且能保证每个分段满足误差阈值，误差阈值由成本模型决定。FITing-Tree 支持插入和查找操作，每个段包含一个额外的固定大小的缓冲区，当新的键被插入缓冲区中，一旦缓冲区达到预定的大小，它将与段中的数据结合，然后使用分段算法重新创建一系列满足错误阈值的有效分段。实验表明，1 MB 内存的 FITing-Tree 能够与消耗超过 10 GB 内存的固定大小索引的性能相匹配，从而节省 4 个数量级的空间。

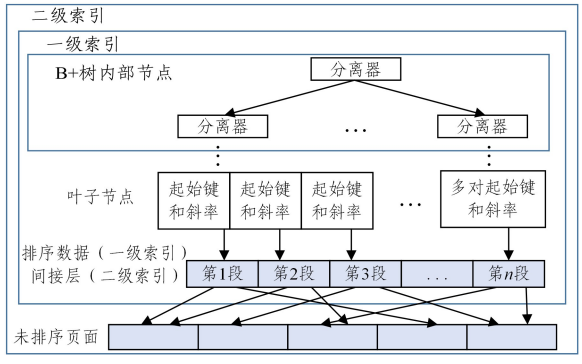

图 3 FITing-Tree 索引结构

Fig. 3 Structure of FITing-Tree index

AIDEL(Adaptive InDEpendent Linear regression models)[11]索引是基于 RMI 提出的一种可扩展的、具有有限预测误差的学习索引，它解决了 RMI 可扩展性差、维护开销大的问题。AIDEL 将数据划分为不同的部分，并使用多元线性回归模型来很好地学习每个部分。AIDEL 模型在内部是完全独立的，所有的模型都是通过学习探测算法(Learning Probe Algorithm，LPA)生成的，该算法利用贪心算法根据数据分布自适应地划分数据，然后分配线性回归模型。与 FITing-Tree 使用的分段算法不同，在 LPA 中，只有线性分布相同的数据才能被划分为相同的子数据集，因此，每个子数据集都可以很容易地用线性回归模型来表示。AIDEL 通过添加在现有数据后面的排序列表来实现可伸缩性，它可以轻松地处理插入并保证插入数据有序排列，从而有效地实现更新后的查询。AIDEL 中的这种设计允许在不影响整个结构的情况下更新任何模型。

PGM(Piecewise Geometric Model)[29]索引是一个完全动态的分段几何模型索引。不同于依赖经典的数据结构的 FITing-Tree，PGM 索引是基于 PLA 模型[40]提出在一个新的递归结构中固定最大误差容忍度来协调线性模型的最佳数量。此后，PGM 索引对键和段(即起始点和斜率)提供适当的无损压缩，对于起始键使用已有研究方法[41-42]进行压缩，对于斜率设计压缩算法。PGM 索引对键的分布和查询分布具有

适应性,形成了第一个已知的分布感知的学习索引。PGM 索引可以根据任何给定的空间或延迟要求有效地自动调整自身。为了支持插入,PGM 索引采用了 LSM 树的思想。空间占用率比 FITing-Tree 高 75%,构建速度相比 RMI 有所提升,同时不需要超参数调整。但其必须在所有不同大小的组件中搜索给定的键,导致查找性能严重降低。

RS(RadixSpline)[15]索引的构建过程分为两步。首先,线性样条拟合数据的 CDF,保证一定的误差范围;其次,构建一个基数表作为样条点的近似索引。RS 使用样条插值法[43]拟合数据来保证用户定义的误差范围,但随着数据集大小的增长,或者在大的异常值下,它的性能表现并不尽如人意。RS 查询性能与 RMI 类似,但是构建速度比 RMI 快约 6 倍。文献[44]对 RMI,PGM 和 RS 进行了实验对比,结果显示,对于最大的数据集,RMI,PGM 和 RS 的最快变体的构建时间分别为 80 s,38 s 和 20 s。RS 使用基数表对其样条点进行索引,然而,当数据集的键不能轻易地被基数表索引时,即它们的二进制表示的最长公共前缀很大时,RS 的性能可能会下降。

PLEX[16]通过优化基数树 HT(Hist-Tree)[45]替换基数表来解决上述问题。与 RS 一样,PLEX 也是"自下向上"构建的,首先在底层数据上构造一个错误边界样条,然后在 CHT 中对样条点进行索引。因为它只有一个超参数"最大误差 e",所以得到的索引结构具有构建速度快、错误有界查找和易于使用的特点。

现有的学习索引在整数键上表现出良好的性能。然而,它们的性能在查找字符串键时显著下降。实验表明,当键长度从 8 字节增长到 128 字节时,XIndex 的性能下降超过 66%。因此,Wang 等提出了专门处理可变长度字符串键工作负载的并发学习索引 SIndex[30],这也是目前唯一已知的针对字符串键进行优化的索引结构。它的整体架构与 XIndex 类似,为了实现在字符串键下的良好性能,SIndex 针对字符串键做出了 3 个重要的设计:1)SIndex 使用部分键来降低模型的计算成本和比较成本,部分键是原始键的保留顺序的子字符串,也是组内键的最短的固定长度的子串,目的是在这一个组中保持独特性,采用一种高效的算法来计算部分键;2)SIndex利用贪心算法策略自适应地将键划分为不同的组,这种分组方案通过减少组的数量来提高 SIndex,从而减少根节点的索引负担;3)SIndex 在根节点使用分段线性模型而不是 RMI 模型来索引组,使用分段线性模型有助于 SIndex 控制索引键范围。实验结果表明,对于所有键的长度,SIndex 都展现出了相当大的性能优势。当键长度从 8 字节增长到 128 字节时,SIndex 保持了其 8 字节性能的 62%,并且比 XIndex 高出 91%。

一维学习索引虽然一定程度上对传统索引进行了优化,但在实际应用中,因其只能处理单一维度的数据,导致应用场景受限严重。虽然可以使用多级一维索引对数据进行多个属性的查询,但这会产生巨大的存储开销,只能适用于较小的数据集。因此,为了更好地组织管理数据,学者们将研究方向转向了应用更广泛的多维学习索引。

3 多维学习索引

多维数据索引是一种用于高效数据访问的数据结构。与一维数据可以根据属性值进行排序不同,多维数据因为逻辑顺序不强而不易排序,这种不易排序的特性成为了将学习模型应用到多维索引的最大的挑战。表 3 列出了现有多维学习索引的优化对比和发展脉络,从初步实现对多维数据的索引到逐步实现各类查询、优化布局和动态更新。

表 3 多维学习索引优化对比

Table 3 Multi-dimensional learned indexes optimization

多维学习索引	优化
SageDB(2019 年)	实现点查询、范围查询
ZM(2019 年)	结合 Z 阶曲线和 RMI
LSBI(2019 年)	支持 NN 查询
ML(2020 年)	支持 kNN 查询
Flood(2020 年)	根据查询工作负载优化数据布局
Tsunami(2020 年)	根据数据相关性和查询倾斜优化数据布局
LISA(2020 年)	动态更新
SPGIN(2021 年)	对 kNN 查询进行剪枝处理

多维学习索引的迅速发展离不开一维学习索引打下的坚实基础。大部分多维学习索引的主要思想是将多维数据布局进行划分处理,在划分数据后普遍采用前面讨论过的 RMI 和分段线性回归模型对数据进行构建索引。因此,本文根据数据布局的不同详细介绍表内所有学习索引。第 3.1 节描述将多维数据映射为一维数据,如 SageDB,ZM,ML;第 3.2 节描述采用网格划分多维数据,如 Flood,Tsunami,SPRIG;第 3.3 节描述将多维数据转化为自定义布局,如 LISA 和 LSBI。

3.1 多维数据映射为一维数据

该类数据布局的多维索引主要思想为:首先将多维空间映射为一维空间;然后在一维空间建立索引;最后实现在多维数据上的查询。SageDB(SageDataBase)[17]和 ZM(Z-order Model)[18]索引是较早将学习索引应用到多维数据上的多维学习索引,它们实现了点查询和范围查询,但不支持 kNN 查询。SageDB 使用投影函数 L 将点投影到一维上,物理位置使用训练过的 CDF 模型进行分配预测,但是具有相同排序键的两个点可能相距很远,这导致围绕其中一个点的一个小查询矩形也会包含大量无关的点,使查询速度变慢。ZM 索引结合了 Z 阶空间填充曲线[46]和 RMI 来索引空间数据。ZM 索引应用 Z 阶空间填充曲线将多维数据映射到一维数据空间,使多维数据变为可排序数据后,构建 RMI 来学习数据分布,预测所需数据对象的位置。ZM 索引展现了更优的查询性能和更小的存储代价,同 SageDB 一样,ZM 索引的缺点也是在实现查询过程中会扫描到许多无关的点,降低了查询的效率。

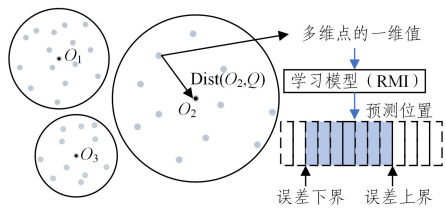

图 4 ML 索引方法

Fig. 4 ML-Index approach

为了改进这一问题,并实现 kNN 查询,文献[19]提出了第一个支持 kNN 查询的多维学习索引,即 ML(Multidimensional Learned)索引。ML 索引是一种高效的多维学习结构,用于处理点查询、范围查询和 kNN 查询。ML 索引由两个主要组件组成,如图 4 所示。它的创建分两个阶段进行:

第一阶段，基于 iDistance 索引[47]构建，使用参考点将多维数据缩放为一维值，以便排序，参考点的选择是使用 K-Means 算法完成；第二阶段是一个由简单回归模型组成的 RMI，用于学习缩放值的分布，以及用于搜索和存储数据的排序数组。ML 支持范围查询和 kNN 查询，使用距离近似方法[48]实现范围查询，使用基于 iDistance 的改进策略实现 kNN 查询。实验结果显示，在搜索二维点时，ZM 优于 ML；在范围查询时，ZM 的性能较差。

3.2 网格划分

上文提出了几种用于处理多维数据的学习索引，但这些方法的主要思想是利用现有的一维学习模型，这需要将多维数据转换为一维数据，因此以上方法不能利用或不能充分利用原始多维数据的空间分布信息。为了更直观地建立多维学习索引，需要将学习模型广泛应用于各个维度，然后分析多维数据并对其进行划分。为此，本小节介绍在多维数据上进行网格划分后建立学习索引。

Flood[20]索引同时优化数据存储布局和索引的结构，以此来获得优于其他多维索引的查询速度。Flood 索引的基本思想是在每个单元格内部使用分段线性回归模型拟合数据分布。为了优化网格划分，Flood 训练了一个随机森林回归模型[49]来预测成本模型的权重，以此选出最优的成本模型；为了实现数据样本平滑化，Flood 利用 CDF 使每个单元格内数据点近似平均；为了找到最佳排序维，Flood 迭代选择每个维度作为排序维，对其余 $d-1$ 维排序，对于 d 种可能用梯度下降法来找到最佳排序维。实验结果显示，Flood 的查询时间较多种空间索引[50-51]至少提高了 2.4～3.3 倍。在性能相当时，Flood 的索引占用空间仅为其他空间索引的 1/50。但 Flood 仍有两点不足：第一，它只对均匀查询进行优化，当查询不是均匀的时，性能会下降；第二，当数据关联时，Flood 基于模型的索引技术会导致单元格大小不同。

为了改进以上不足，Kraska 团队在 Flood 的基础上提出了 Tsunami[21]索引，目的是解决 Flood 在数据相关性和查询倾斜方面的局限性。首先，使用完整的数据集和样本查询工作负载优化网格树；其次，在优化后的网格树的每个区域中，构造一个只对与该区域相交的点和查询进行优化的增强网格。实验结果显示，Tsunami 的查询性能比 Flood 高出 6 倍，索引大小仅为 Flood 的 1/8。

SPRIG（SPatial inteRpolation functIon based Grid index）[22]索引是一个高效的基于空间插值函数[52]的网格索引，可以用来处理范围和 kNN 查询。索引建立首先基于成本模型找到最优的行和列生成网格，然后为每一个网格生成一个 ID，基于网格和 ID 生成拟合插值函数 Fin，将行列值作为输入，由拟合插值函数估算 ID。为了提升 kNN 查询性能，SPRIG 索引采用了两种修剪技术：一种是最近点剪枝技术，另一种是基于枢轴的过滤。在一系列的查询与优化下，实验结果表明，SPRIG 索引在查询时间上优于 Flood 索引，但消耗的存储更多。

3.3 自定义数据布局

该类多维学习索引在原有数据基础上重新定义了一种数据布局。LSBI（Learned Spatial Bucket Index）[23]索引在多维数据上应用桶装结构，并实现了范围查询和 NN 查询。图 5 给出了 LSBI 的模型架构，第 1－3 级表示第 1 维桶装，第 4 级表示第 2 维桶装。在此基础上，采用线性回归模型和单一神经网络两种模型，在图 5 中桶结构的第一级应用单一神经网络，通过学习每个桶内的值与其桶键之间的关系，快速缩小相关桶的区域，并对每个子桶建立线性回归模型，了解桶内数据点之间的关系及其在桶内的位置，快速定位每个桶内的相关值。实验结果显示，LSBI 范围查询和 NN 查询速度均优于 R 树[53]，但 LSBI 目前支持的查询类型比 R 树少，例如不支持 kNN 查询，这使得 LSBI 可进行的查询有限。

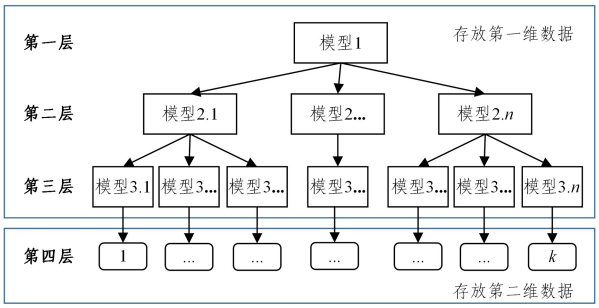

图 5 LSBI 的架构

Fig. 5 Architecture of LSBI

虽然以上多维学习索引在各个方面进行了优化，但它们都不支持插入。与一维索引相比，多维索引插入更难，因为多维数据杂乱无章，插入的位置会影响查询的性能。为了解决这个更新问题，Li 等提出了 LISA（Learned Index structure for Spatial dAta）[12]索引。LISA 在实现范围查询和 kNN 查询的基础上，实现了数据更新、插入和删除。它的核心思想是使用机器学习模型，以在磁盘页面中为任意空间数据集生成可搜索的数据布局。如图 6 所示，图 6(a)－图 6(b)是将多维点映射到一维空间，基于勒贝格测度（Lebesgue Measure）[54]的方法生成映射函数，图 6(b)－ 图 6(c)是通过多维点的映射值，学习由一系列分段线性函数组成的预测函数，以便为每个映射值分配一个碎片 ID，图 6(c)中每个彩色部分为一个碎片，这一系列的碎片类似于数据布局。最后，为每个碎片分配磁盘页面。不同于前两节讲述的数据布局固定的学习索引，LISA 使用学习模型来生成数据布局，因此它的数据布局是可以动态维护的。

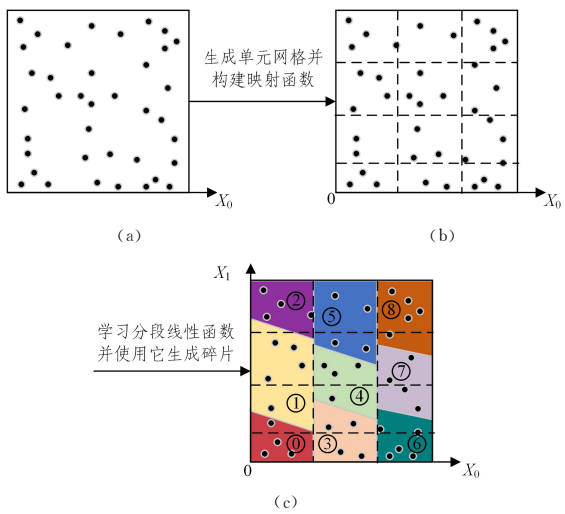

图 6 LISA 框架

Fig. 6 LISA framework

LISA 算法能有效地支持数据更新,即插入和删除。在插入操作时,通过预测函数计算插入点应该插入的磁盘页位置,若磁盘页面容量充足,则直接插入;若页面容量不足,则扩展磁盘页再插入。在执行删除操作时,找到应该删除该点的磁盘页面位置,若删除该点后页面点数为空,则释放当前页;若删除该点后页面容量过少,则合并磁盘页。除此之外,LISA 算法还支持范围查询,并使用格子回归(Lattice Regression)模型[55]结合范围查询实现 kNN 查询。实验结果表明,与 R 树相比,LISA 可以节省 10%~20%以上的 IO 消耗,且 LISA 所需的磁盘存储空间平均比 R 树少 5%~10%。

结束语 学习索引是一个非常有前景的研究领域,该领域已经成为了国内外学者的研究热点,并取得了许多优良的研究成果。本文调研了该领域相关的论文,并对研究成果进行了详细的划分,总结了现有研究成果的优势与不足。到目前为止,学习索引研究还处于初级阶段,依然有许多关键问题尚未解决。本文根据现有工作的不足,给出了未来学习索引值得探索的 3 个研究方向。

(1)多维学习索引的动态更新。在学习索引的研究中,一维学习索引普遍支持动态更新,但对于多维学习索引动态更新机制的研究仍处于起步阶段,如何在实现插入、删除的基础上,最大程度地保护模型的精度,以及在大量插入时实现局部重新训练模型,而非全局更新,是未来多维学习索引值得探索的研究方向。

(2)处理更多查询类型。虽然研究者们对学习索引处理点查询、范围查询和 kNN 查询已经有了初步的研究,但是很多在传统索引中已经进行深入研究的经典问题在学习索引上仍没有得到解决(如空间连接和最近点对查询)。这些问题在实际生活中都有很高的应用价值,如果能在学习索引中得到解决,学习索引的应用场景将更加广泛。

(3)学习索引与图数据库相结合。目前学习索引被大量应用于关系型数据库,尝试将其思想与图数据库建立连接,使其能被应用到图数据库中,并实现图上常见查询(如最短路径查询)。进一步研究学习索引在时序图上的应用,使其对动态数据友好,也是有待研究的课题方向。

参 考 文 献

[1] PAGH R,RODLER F F. Cuckoo Hashing[J]. Journal of Algorithms,2004,51(2):122-144.

[2] ATHANASSOULIS M,AILAMAKI A. BF-Tree:Approximate Tree Indexing[C]// International Conference on Very Large Databases. 2014:1881-1892.

[3] WITTEN I H,MOFFAT A,BELL T C. Managing Gigabytes:Compressing and Indexing Documents and Images[J]. SIGMOD Record,1999,33(2):113-114.

[4] AREF W G,BARBARÁ D,VALLABHANENI P,et al. The handwritten trie:indexing electronic ink[C]// ACM Sigmod International Conference on Management of Data. 1995:151-162.

[5] WANG J,LIN C,PAPAKONSTANTINOU Y,et al. An Experimental Study of Bitmap Compression vs. Inverted List Compression[C]// the 2017 ACM International Conference. 2017:993-1008.

[6] BENDER M A,DEMAINE E D,FARACH-COLTON M. Cache-oblivious B-trees[J]. Siam Journal on Computing,2000,35(2):300-409.

[7] JENSEN C S,LIN D,OOI B C. Query and update efficient B+-tree based indexing of moving objects[C]// Proceedings of the Thirtieth International Conference on Very Large Data bases-Volume 30. 2004:768-779.

[8] O'NEIL P,CHENG E,GAWLICK D,et al. The log-structured merge-tree(LSM-tree)[J]. Acta Informatica,1996,33(4):351-385.

[9] KRASKA T,BEUTEL A,CHI E H,et al. The Case for Learned Index Structures[C]// Proceedings of the 2018 International Conference on Management of Data. 2018:489-504.

[10] GALAKATOS A,MARKOVITCH M,BINNIG C,et al. FI-Ting-Tree:A Data-aware Index Structure[C]// Proceedings of the 2019 International Conference on Management of Data. 2019:1189-1206.

[11] LI P,HUA Y,ZUO P,et al. A Scalable Learned Index Scheme in Storage Systems[J]. arXiv:2102.06789,2019.

[12] LI P,LU H,ZHENG Q,et al. LISA:A Learned Index Structure for Spatial Data[C]// Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020:2119-2133.

[13] HIGUCHI S,TAKEMASA J,KOIZUMI Y,et al. Feasibility of longest prefix matching using learned index structures[J]. ACM SIGMETRICS Performance Evaluation Review,2021,48(4):45-48.

[14] RAMADHAN H,KWON J. Enhancing Learned Index for a Higher Recall Trajectory K-Nearest Neighbor Search[C]// 2021 IEEE International Conference on Big Data(Big Data). 2021:6006-6007.

[15] KIPF A,MARCUS R,VAN RENEN A,et al. RadixSpline:a single-pass learned index[C]// Proceedings of the Third International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2020:1-5.

[16] STOIAN M,KIPF A,MARCUS R,et al. PLEX:Towards Practical Learned Indexing[J]. arXiv:2108.05117,2021.

[17] KRASKA T,ALIZADEH M. SageDB:A Learned Database System[C]// 9th Biennial Conference on Innovative Data Systems Research. 2019:13-16.

[18] WANG H X,FU X Y,XU J L,et al. :Learned Index for Spatial Queries[C]// 2019 20th International Conference on Mobile Data Management. 2019:569-574.

[19] DAVITKOVA A,MILCHEVSKI E,MICHEL S. The ML-Index:A Multidimensional,Learned Index for Point,Range,and Nearest-Neighbor Queries[C]// EDBT. 2020:407-410.

[20] NATHAN V,DING J,ALIZADEH M,et al. Learning Multi-dimensional Indexes[C]// Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020:985-1000.

[21] DING J,NATHAN V,ALIZADEH M,et al. Tsunami:A Learned Multi-dimensional Index for Correlated Data and Skewed Workloads[J]. Proceedings of the VLDB Endowment,2020,14(2):74-86.

[22] ZHANG S,RAY S,LU R,et al. Spatial Interpolation-based Learned Index for Range and kNN Queries[J]. arXiv:2102.06789,2012.

[23] FOLMER M,NEUMANN R,KARUNANITHI T. A Learned Bucket Index Supporting Spatial Queries[D]. Aalborg:Aalborg University,2019.

[24] DING J,MINHAS U F,YU J,et al. ALEX:An Updatable

Adaptive Learned Index[C]// International Conference on Management of Data Conference. 2020:969-984.
[25] WU J,ZHANG Y,CHEN S,et al. Updatable Learned Index with Precise Positions[J]. Proceedings of the VLDB Endowment,2021,14(8):1276-1288.
[26] WU Y J,YU J,TIAN Y Y,et al. :Designing Succinct Secondary Indexing Mechanism by Exploiting Column Correlations[C]// Proceedings of the 2019 International Conference on Management of Data. 2019:1223-1240.
[27] ZHANG J,GAO Y. CARMI:A Cache-Aware Learned Index with a Cost-based Construction Algorithm[J]. arXiv: 2103.00858,2021.
[28] TANG C,WANG Y,DONG Z,et al. XIndex:A Scalable Learned Index for Multicore Data Storage[C]// Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2020:308-320.
[29] FERRAGINA P,VINCIGUERRA G. The PGM-index:a fully-dynamic compressed learned index with provable worst-case bounds[J]. Proceedings of the VLDB Endowment,2020,13(8):1162-1175.
[30] WANG Y,TANG C,WANG Z,et al. SIndex:a scalable learned index for string keys[C]// Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems. 2020:17-24.
[31] MCKENNEY P,KLEEN A,KRIEGER O,et al. Read-Copy Update[C]// Ottawa Linux Symposium. 2001:175-184.
[32] BRONSON N G,CASPER J,CHAFI H,et al. A practical concurrent binary search tree[J]. ACM Sigplan Notices, 2010,45(5):257-268.
[33] SANG K C,HWANG S,KIM K,et al. Cache-Conscious Concurrency Control of Main-Memory Indexes on Shared-Memory Multiprocessor Systems[C]// Proceedings of 27th International Conference on Very Large Data Bases. 2001:181-190.
[34] WU X,NI F,JIANG S,et al. Wormhole:A Fast Ordered Index for In-memory Data Management[C]// Proceedings of the Fourteenth EuroSys Conference. 2019:1-16.
[35] MAO Y,KOHLER E,MORRIS R. Cache Craftiness for Fast Multicore Key-Value Storage[C]// Proceedings of the 7th ACM European Confe-rence on Computer Systems. 2012:183-196.
[36] ILYAS I F,MARKL V,HAAS P,et al. CORDS:Automatic discovery of correlations and soft functional dependencies[C]// Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. 2004:647-658.
[37] KIMURA H,HUO G,RASIN A,et al. Correlation maps:a compressed access method for exploiting soft functional dependencies[J]. Proceedings of the VLDB Endowment,2009,2(1):1222-1233.
[38] LIU X,LIN Z,WANG H. Novel Online Methods for Time Series Segmentation[J]. IEEE Transactions on Knowledge and Data Engineering,2008,20(12):1616-1626.
[39] XU Z,ZHANG R,KOTAGIRI R,et al. An adaptive algorithm for online time series segmentation with error bound guarantee [C]// Proceedings of the 15th International Conference on Extending Database Technology. 2012:192-203.
[40] O'ROURKE J. An on-line algorithm for fitting straight lines between data ranges[J]. Communications of the ACM, 1981,24(9):574-578.
[41] OKANOHARA D,SADAKANE K. Practical entropy-compressed rank/select dictionary[C]// 2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX). 2007:60-70.
[42] MOFFAT A,TURPIN A. Compression and coding algorithms [M]. Berlin:Springer Science & Business Media,2002.
[43] NEUMANN T,MICHEL S. Smooth interpolating histograms with error guarantees[C]// British National Conference on Databases. 2008:126-138.
[44] MARCUS R,KIPF A,VAN RENEN A,et al. Benchmarking learned indexes[J]. arXiv:2006.12804,2020.
[45] CROTTY A. Hist-Tree:Those Who Ignore It Are Doomed to Learn[C]// CIDR. 2021:11-15.
[46] RAMSAK F,MARKL V,FENK R,et al. Integrating the UB-Tree into a Database System Kernel[C]// VLDB. 2000:263-272.
[47] JAGADISH H V,OOI B C,TAN K L,et al. iDistance:An adaptive B+-tree based indexing method for nearest neighbor search [J]. ACM Transactions on Database Systems(TODS),2005, 30(2):364-397.
[48] SCHUH M A,WYLIE T,LIU C,et al. Approximating High-Dimensional Range Queries with kNN Indexing Techniques[C]// Berlin:Springer International Publishing. 2014:369-380.
[49] BREIMAN L. Random forests[J]. Machine Learning,2001, 45(1):5-32.
[50] GAEDE V,GÜNTHER O. Multidimensional access methods [J]. ACM Computing Surveys(CSUR),1998,30(2):170-231.
[51] NIEVERGELT J,HINTERBERGER H,SEVCIK K C. The grid file:An adaptable,symmetric multikey file structure[J]. ACM Transactions on Database Systems(TODS),1984,9(1):38-71.
[52] MITAS L,MITASOVA H. Spatial interpolation [J]. Geographical Information Systems:Principles,Techniques,Management and Applications,1999,1(2):481-492.
[53] GUTTMAN A. R-trees:A dynamic index structure for spatial searching[C]// Proceedings of the 1984 ACM SIGMOD International Conference on Management of data. 1984:47-57.
[54] ROYDEN H L,FITZPATRICK P. Real analysis[M]. New York:Macmillan,1988.
[55] GARCIA E,GUPTA M. Lattice regression[J]. Advances in Neural Information Processing Systems,2009,22:594-602.

WANG Yitan, born in 1997, postgraduate. Her main research interests include data structure and temporal graph.

YUAN Ye, born in 1981, Ph.D, professor, Ph.D supervisor. His main research interests include graph databases, probabilistic databases, data privacy-preserving, and cloud computing.

(责任编辑:何杨)