智能数据库学习型索引研究综述

蔡 盼[1),2)] 张少敏[1),2)] 刘沛然[1),2)] 孙路明[1),2)] 李翠平[1),2)] 陈 红[1),2)]

[1)](中国人民大学数据工程与知识工程教育部重点实验室 北京 100872)

[2)](中国人民大学信息学院 北京 100872)

摘 要 建立高效的索引结构是提升数据库存取性能的关键技术之一. 在数据呈爆发式增长、海量聚集、高维复杂的大数据环境下，传统索引结构（例如 B^+树）处理海量数据时面临空间代价高、查询效率低、存取开销大等难题. 学习型索引技术通过对底层数据分布、查询负载等特征进行建模和学习，有效的提升了索引性能，并减少了访存空间开销. 本文从学习型索引技术的基础模型入手，对 RMI 基础模型实现原理、构造和查询过程进行了分析，并总结了基础模型的优点和存在的问题；以此为基础，按照索引结构特点对学习型索引技术进行分类，从索引创建方式和更新策略两方面对学习型索引技术进行了系统梳理，并对比分析了典型学习型索引技术的优点及不足之处. 另外，本文总结了学习型索引技术的扩展研究. 最后，对学习型索引的未来研究方向进行了展望.

关键词 机器学习；学习型索引；索引结构；RMI 模型；智能数据库

中图法分类号 TP391 **DOI 号** 10.11897/SP.J.1016.2023.00051

An Overview of Learned Index Technologies for Intelligent Database

CAI Pan[1), 2)] ZHANG Shao-Min[1), 2)] LIU Pei-Ran[1), 2)] SUN Lu-Ming[1), 2)]
LI Cui-Ping[1), 2)] CHEN Hong[1), 2)]

[1)](*Key Laboratory of Data Engineering and Knowledge Engineering of the Ministry of Education, Renmin University of China, Beijing* 100872)

[2)](*School of Information, Renmin University of China, Beijing* 100872)

Abstract Data access is an important operation in the database system. It is critical to improve the performance of the database system by increasing the speed of data access. Therefore, both academia and the industry have been devoted to establishing efficient index structures to improve the performance of data access in database systems over the past decades. Nevertheless, the traditional index structures (e.g., B+-tree) face the challenges of the high space cost, the low query efficiency, and the more access overhead in the era of big data, which is characterized by explosive data growth, massive aggregation, and high dimensional complexity. Consequently, to deal with the mentioned issues, machine learning methods are applied to the traditional index structure to advance the research of learned indexes, which gradually becomes one of the research hotspots in the database field. The core of the learned index is to approximate the cumulative distribution function of the underlying data through machine learning methods to realize the mapping between keys and record positions. During the query process, the learned index predicts the location of the record through the key. The tree traversal operation of the traditional index structures is replaced by model prediction, which effectively improves the query speed and reduces the storage overhead.

收稿日期：2021-09-30；在线发布日期：2022-05-25. 本课题得到国家自然科学基金（62072460, 62076245, 62172424,62276270）、北京市自然科学基金（4212022）、中国人民大学科学研究基金（中央高校基本科研业务费专项资金）（22XNH189）资助. **蔡 盼**，博士，主要研究领域为学习型索引、数据库系统. E-mail: 2020000878@ruc.edu.cn. **张少敏**，硕士，主要研究领域为索引技术、机器学习. **刘沛然**，硕士，主要研究领域为内存数据库、学习型索引. **孙路明**，博士，主要研究领域为数据库系统、机器学习. **李翠平**（通信作者），教授，主要研究领域为主要研究领域为社交网络分析、社会推荐、大数据分析及挖掘. E-mail: licuiping@ruc.edu.cn. **陈 红**，教授，主要研究领域为数据库技术、新硬件平台下的高性能计算.

Learned indexes provide new research ideas for improving the performance of traditional index structures. Until now, a considerable amount of literature has sought to investigate the learned index structures. Therefore, it is worthwhile to present the state-of-the-art of the learned index. This paper firstly introduces the research background of the learned index. Then, we present a review of current research and analyze the proposed methods in terms of the fundamental and extension issues. All learned indexes are associated with the Recursive Model Index (RMI) model as a clue. Specifically, the basic problems of the learned indexes, mainly include: (1) The basic RMI model that this paper systematically summarizes the implementation principle, index structure, and query of the RMI model process, and summarizes and analyzes the advantages and existing problems of the RMI model. (2) Learned indexes for one-dimensional data, according to whether it adopts the same structure as the RMI model, it is divided into a learned index with a hierarchical structure and a learned index with a non-hierarchical structure. For each type of index, further subdivision and summary analysis are made based on key technologies. (3) Learned indexes for multi-dimensional data that all use the same structure as the RMI model. Therefore, only according to the key technology of indexing, the learned indexes of multi-dimensional data are divided into the index based on the dimensionality reduction method and the non-dimensionality reduction method. For each category, this paper summarizes their key technologies, and index structures, and compares and analyzes the advantages and disadvantages of different learned indexes. Next, this paper summarizes the expansion of learned indexes, which mainly include six aspects that namely: (1) Learned indexes of other query types that involve point queries of hash indexes and existence queries of Bloom filters. (2) The traditional index structures are assisted by machine learning, which is improved by machine learning methods rather than completely replacing the traditional index structures. (3) Secondary indexes that mainly use machine learning methods to deal with disordered data. (4) Spatio-temporal learned index applies machine learning methods to improve Spatio-temporal indexes. (5) Test benchmarks, which provide a unified test platform for learned indexes to evaluate their performance. (6) Parameter tuning provides a visual tool to intuitively show the parameter tuning process of the learned index for the user. Finally, this paper looks forward to the future research direction of learned indexes.

Keywords machine learning; learned index; index structure; RMI model; intelligent databases

1 引 言

索引技术对于数据库系统实现数据的高效访问至关重要[1]，一直以来都是数据库领域的研究热点之一. 自上世纪 70 年代以来，工业界和学术界开展了大量的索引技术研究[2-5]，用于提高内存利用率、缓存命中率或 CPU 效率. 典型索引技术有面向范围查询的 B 树索引、面向点查询的哈希索引，以及面向存在性查询的布隆过滤器等. 然而，大数据时代的到来，数据呈爆发式增长、海量聚集，数据库需要处理 PB 级，甚至 ZB 级的数据，致使传统索引结构在处理海量数据时，面临空间代价高（例如，B^+树索引需要借助 $O(n)$规模的额外空间来索引大规模的数据集）、查询效率低（例如，B^+树每次查询时需要遍历根节点到叶节点路径上的所有节点，导致在大规模数据集中查询性能下降）的问题. 其主要原因有两方面：第一，传统索引结构是通用数据结构，没有利用底层数据的分布规律，不能够灵活处理特定的数据集或者查询负载；第二，传统索引结构的查询操作往往是顺序扫描，未充分利用现代加速器的高并行处理能力来提升查询效率. 针对上述传统索引结构的问题，研究人员将目光投向人工智能技术和机器学习方法，研究了学习型索引技术.

学习型索引技术的思想来源于智能数据库系统的研究. 近些年，将人工智能技术和机器学习应用到传统数据库系统，通过捕获底层数据的分布规律或者查询负载的特征等，建立深度学习模型或者强化学习模型来加强或替换数据库的核心组件（参数调优、查询优化、索引结构等），使得传统数据库系统拥有智能，成为了数据库领域的研究热点[6]. 在

2018 年的 SIGMOD 会议上，Kraska 等人[7]首次采用机器学习模型替换传统索引结构，学习底层数据的分布规律，建立键与位置之间的映射关系，提出了学习型索引的概念. 学习型索引技术因其自身的显著优势（例如，学习型索引通过数学计算替代树遍历操作有效减少了存储开销，提高了查询效率），引起了数据库领域研究人员的广泛关注，在 Kraska 等人提出的学习型索引技术的基础上，开展了大量的研究工作. 本文从基础问题-扩展问题这两个层次对学习型索引技术进行梳理、总结和分析. 具体地，针对基础问题，本文首先系统概述学习型索引技术的基础模型 RMI 模型，分析了其原理、构造和查询过程，并总结了 RMI 模型的优点和存在的问题. 然后，以 RMI 模型为线索，依据索引（模型）组织方式和优化策略对学习型索引技术进行分类和总结. 对每一类技术，从索引创建方式、更新策略等方面展开详细阐述. 并从多个角度对比分析了不同学习型索引技术之间的优缺点. 针对扩展问题，本文总结了 6 个研究方向. 最后，总结分析了学习型索引技术存在的问题并对未来的研究方向进行展望.

相关工作：由于学习型索引技术研究尚处于起步阶段，关于这方面的综述性文章不多. 文献[6, 8-10]介绍了学习型索引技术，但是它们的侧重点是机器学习改进的数据库系统. 文献[11]和[12]是已知的仅针对学习型索引技术进行综述的文章. 文献[11]是国外首篇综述学习型索引技术的文章，但是文章侧重于技术罗列且较早期，后续涌现的许多新成果未纳入其中. 文献[12]是国内首篇综述学习型索引技术的文章，文章以查询类型、测试基准为分类框架，总结了学习型索引技术. 文献[11]和[12]均侧重于一维学习型索引的更新策略，缺乏对一维和多维学习型索引技术细致的分类和对比分析. 本文提出了一个新的研究框架：基本问题-扩展问题，并以索引（模型）组织方式、优化策略来划分和总结不同的学习型索引技术，比上述文献更加综合全面.

2　学习型索引的基础模型

2.1　RMI模型（Recursive Model Index）

（1）RMI 模型的实现原理

B 树常用来处理一维范围查询，其索引操作包括查询、插入和删除等. 给定一个查询键，B 树通过树遍历算法，逐步缩小查询范围，直到找到包含查询键的叶子节点为止. 在叶子节点内部，数据是按照顺序存储在数组中，因此通过顺序扫描或者二分查找可快速定位到查询键对应的记录在磁盘页中的位置. Kraska 等人将 B 树看作是键映射到记录位置的一种模型，因此可用机器模型（例如神经网络、线性回归等）替换. B 树将查询键映射到一个具有误差约束（最小范围是 0，最大范围是磁盘页的大小）的范围内，如果查询键在误差约束范围内存在，则保证一定能够在这个范围内检索到查询键的准确位置. 因此，采用模型替换 B 树时，需要考虑以下两点：第一，模型需要提供与 B 树索引相似的误差保证，才能够保证找到查询键；第二，模型需要解决“最后一英里搜索”问题，机器学习模型不能够完全拟合数据的累积分布函数，模型预测误差往往很大，如果想要实现较小的预测误差则比较困难. 例如，将 100 MB 的模型误差减少为 10 kB，则会牺牲更多的 CPU 时间和空间.

基于上述分析，受到混合专家的启发[13]，Kraska 等人设计了 RMI 模型[7]. RMI 模型是一种递归回归模型，为了减小模型预测误差（预测位置与实际位置之间的距离），它采用了多个模型进行预测，并按照层次结构的方式组织模型. RMI 模型的基本思想是学习模型来近似有序数组键的累积分布函数 CDF (Cumulative Distribution Function). 模型表示键与记录位置之间映射关系的近似函数，即 $pos = F(key) * N$，其中，N 表示键的总数量；$F(key)$ 表示键的近似累积分布函数，用来估计键小于或者等于查询键的概率；pos 表示查询键的预测位置. RMI 模型通过计算模型的最大误差（Max_err）和最小误差（Min_err）来提供与 B 树索引类似的误差保证，即[$pos - Min\_err, pos + Max\_err$]. 如果查询键存在，则在误差保证的范围内一定能够找到查询键.

（2）RMI 模型的构造过程

RMI 模型的结构如图 1 所示，模型主要由根模型、中间层模型和叶模型组成. RMI 模型采用自上而下的方式构造，其中，根模型只包含一个模型，由整个数据集训练而成. 具体地，根模型将数据集划分成多个子集，每个子集表示一个分段. 然后，对每个分段重新训练模型，直到训练完所有分段为止. 如果分段模型拟合效果较差，则继续划分分段，直到模型可以拟合最小范围内的数据分布，成为叶模型为止. RMI 模型采用混合模型，由于根模型是在整个数据集上进行训练，数据分布可能较为复杂，因此采用神经网络作为根模型. 而下层模型（中间层模型或者叶模型）是在较小的子数据集上训练，为了节省空间开销和构造成本，可以使用简单的线性回归模型. 同时，如果叶模型始终无法有效拟合数据分布，则可由 B 树替换叶模型.

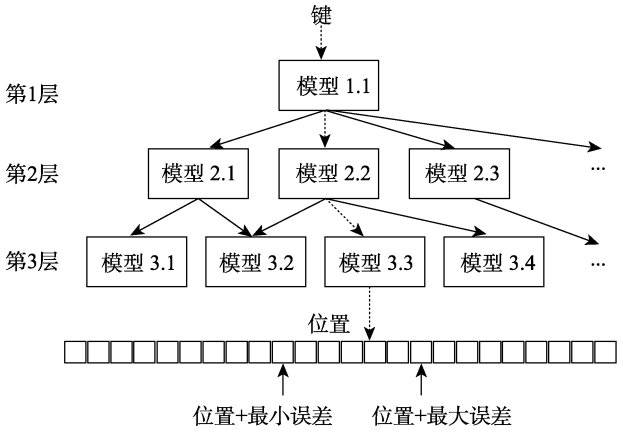

图 1 RMI 模型

（3）RMI 模型的查询过程

与 B 树索引范围查询的原理相似，RMI 模型通过层数的递增不断收紧搜索范围. 在执行查询时，首先从根模型开始，输入目标键，根模型的输出结果用来选择下一层的子模型，而叶模型输出最终的预测位置. 最后，在误差保证的范围内，基于预测位置执行二分搜索定位查询键的准确位置.

（4）RMI 模型的优点

① RMI 模型内存开销小：RMI 模型的大小是一个常数，由所有模型的参数总量以及每个模型需要存储的误差数量来计算. 而 B 树的大小则是与数据集基数成线性关系.

② RMI 模型比 B 树查询效率高：第一，RMI 模型在查询时通过计算查找目标键的位置，无需遍历操作. 而 B 树需要遍历大量的中间节点来查找包含目标键的叶子节点；第二，B 树的每个节点都是固定大小，而 RMI 模型不需要预先指定模型大小，不同层的模型所覆盖的数据数量并不相同. 因此，当目标键落在数据数量少的子模型范围内时，局部搜索时间要少于 B 树；第三，RMI 模型利用层次结构来减小预测误差，它将数据集划分成多个子范围，通过不断缩小搜索范围来查找目标，与 B 树相比，仅需要少量的操作可完成“最后一英里搜索”.

2.2 面临的挑战

尽管 RMI 模型具备显著优势，为索引性能的提升带来裨益. 但是在实际实现中，采用 RMI 模型替代 B 树时，还需要面临以下挑战：

（1）最坏情况下的查找性能问题. 由于 RMI 模型根据键的数量均匀地划分数据集，因此，它无法确定模型在每个数据分段上的最大误差. 如果模型预测误差很大，则本地搜索的代价将会很高.

（2）动态更新问题. B 树索引能够有效支持更新操作，而 RMI 模型仅支持静态查询，无法处理更新操作. 因为更新操作容易引起底层数据分布的改变，一旦数据分布发生改变，RMI 模型的叶模型以及与其相连的上层模型均需要重新训练，将会产生昂贵的训练成本.

3 一维学习型索引

围绕如何限制最坏情况下的查找性能问题以及如何实现动态更新两个问题，研究人员继续深入研究学习型索引. 这些研究工作都是基于一维有序数据进行，所以统称为一维学习型索引. 接下来，本节首先按照索引（模型）组织方式将一维学习型索引划分为层次结构（3.1 节）和非层次结构（3.2 节）. 然后，针对 RMI 模型面临的问题，按照优化策略对两类结构又加以细分. 对每种类型的技术，总结了索引创建方式（数据划分和模型选择）和更新策略. 接着，本章在 3.3 节阐述了两种学习型索引的辅助方法. 最后，在 3.4 节部分比较和分析了不同一维学习型索引之间的差异以及优缺点.

3.1 基于层次结构的索引

这类学习型索引的组织方式与 RMI 模型相同，都是按照层次结构来组织多个模型. 根据优化策略的不同，将它们划分为以下四类：基于线性回归模型优化 RMI 模型、基于并发技术优化 RMI 模型、基于永久内存优化 RMI 模型以及基于插值法优化 RMI 模型.

3.1.1 基于线性回归模型优化 RMI 模型

与 RMI 模型采用混合模型不同，这类学习型索引主要选择比较简单的线性回归模型拟合键的累积分布函数，以此来减少模型复杂度和存储开销.

（1）FITing-TREE 索引

Galakatos 等人[14]提出了 FITing-TREE 索引，该索引的基本思想是预先定义一个误差阈值，保证每个模型的最大预测误差不会超过给定的误差阈值，从而约束最坏情况下的查找性能. FITing-TREE 由内部节点和叶子节点组成，它的索引结构如图 2 所示. FITing-TREE 采用类似 FSW（Feasible Space Window）方法[15,16]的贪心算法将已排序的数据按范围划分成多个不相交的分段(segment)；然后，每个分段采用线性函数进行拟合. 其中，每个分段的模型预测误差都不会超过给定的误差阈值. 结束分段后，所有分段对应的模型由 B^+树组织，树的每个节点存储模型的起始键、斜率以及每个分段的边界. 另外，叶子节点还存储一个指向该节点对应分段的指针.

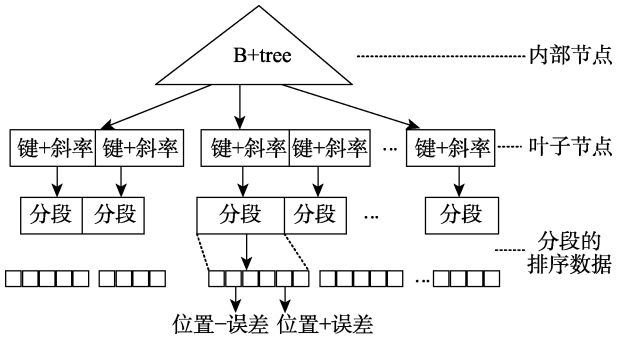

图 2　FITing-TREE 索引结构

FITing-TREE 的查询、插入操作与 B^+树的相同，区别在于叶节点. 对于查询操作，FITing-TREE 利用斜率、查询键与起始键之间的距离计算查询键的近似位置. 然后，在给定的误差阈值范围内，基于近似位置执行二分搜索找到查询键的准确位置. 对于插入操作，FITing-TREE 提出了两种插入策略：第一种是就地插入策略（In-Place Insert Strategy），即在数组的左右两端预留一些空白间隙用来存储新插入的键. 如果插入位置已满，则需要对插入位置所属的分段重新划分和训练模型；第二种是基于缓冲区的插入策略（Delta Insert Strategy），该策略在每个叶子节点中额外设置一个固定大小的缓冲区. 当有新的键插入时，查找到该键所属的分段，如果当前分段对应的缓冲区有空间，则插入该键；否则，重新分段. 两种插入策略均需保证新的键插入后，对应分段的预测误差不会超过给定的误差阈值. Galakatos 等人[14]的实验表明，FITing-TREE 索引的查询性能与 B^+树相近，但是空间开销节省了 4 个数量级. 而插入性能略差于 B^+树(假定 B^+树结构同样使用基于缓冲区的插入策略)，但是在索引的构造速度上，FITing-TREE 要快于 B^+树.

（2）ALEX 索引（Adaptive Learned index）

Ding 等人[17]提出了自适应学习型索引 ALEX，该索引预先确定了叶节点的最大键数量，且允许自适应的确定 RMI 模型的深度以及叶模型的数量. ALEX 的索引结构如图 3 所示：索引包含一个根节点（是一个两层的 RMI 模型）以及一些预先确定数量的叶节点，所有模型采用线性回归模型. 构造 ALEX 索引时，首先为根节点确定若干分区，使得每个分区恰好包含最大数量的键；然后，为每个非根节点提供固定数量的分区，这些分区针对每个子数据集进行学习. 如果一个分区包含的键数量超过分区所包含的最大键数量，则创建中间节点，将当前分区看作是新的根节点，递归地划分当前分区. 否则，将当前分区设为叶节点.

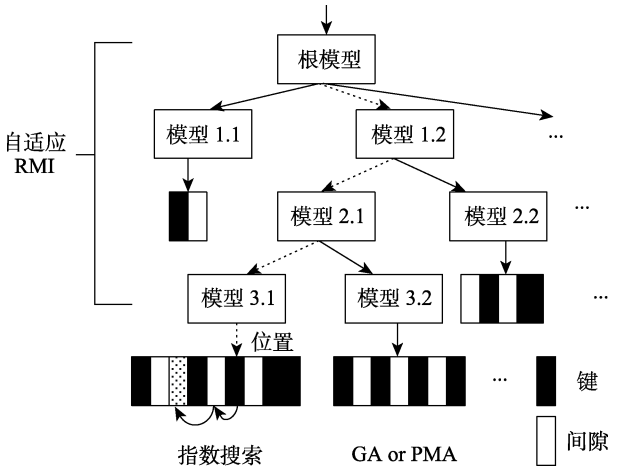

图 3　ALEX 索引结构

为了支持插入操作，ALEX 在叶子节点设计两种数据结构 GA（Gapped Array）和 PMA（Packed Memory Array）[18]来存储键. GA 在相邻的键之间填充空闲间隙，当有新的键插入时，使用模型预测键的插入位置，如果位置恰好是空闲间隙则直接插入. 否则，使用指数搜索距离当前位置最近的空闲间隙插入. GA 的缺陷是随着键的不断插入，空闲间隙越来越少，导致插入过程中需要移动大量的键，从而降低插入性能. 针对此问题，ALEX 设计了备选方案 PMA. 为了避免移动大量的键，PMA 将空闲间隙均匀的分布在数组中，并在插入过程中动态的维护这种分布. PMA 的插入过程与 GA 类似，将不再赘述. 在插入过程中，当叶节点的空闲间隙全部填满时，ALEX 分裂当前叶节点，创建一个新的中间节点以及固定数量的叶节点，原叶节点的数据将会分布到新的叶节点中. 与 B 树不同，ALEX 不需要保持树结构平衡. 由于 ALEX 限定了叶节点的最大键数量，因此，能够限制叶子层模型的数量以及 RMI 模型的增长上界. Ding 等人[17]的实验表明，ALEX 的查询性能和插入性能要比 B^+树高达 3 倍左右；在空间消耗上，ALEX 节约了 5 个数量级. 与 RMI 模型相比，ALEX 查询性能和空间开销均得到改善.

（3）PGM 索引（Piecewise Geometrice Model Index）

FITing-TREE 采用 B^+树结构组织各个模型，当数据集基数过大时，B^+树的节点将会增多，导致空间成本增加，查询性能下降. 针对此问题，Ferragina 等人[19]在 FITing-TREE 的基础上进行改进，提出了 PGM 索引. PGM 索引自底向上构建，采用 PLA（Piecewise Linear Approximation model）模型拟合数据分布. PGM 索引在进行分段时，采用流算法对 PLA 进行优化，限定了分段的最大数量以及每个分

段最少覆盖的键的数量，得到了一个最优的分段序列. 在组织分段时，PGM-index 使用 PLA 模型递归的索引每个分段，首先，提取每个分段的第一个键存放到新的数组；然后，对新数组递归地采用流算法得到优化的 PLA 模型，直到 PLA 模型覆盖的段为一个为止.

PGM 基于 LSM 的策略实现插入操作，但在他们的工作中未介绍实现细节. 除了解决查询和插入，PGM-index 还提出了压缩技术，将每个分段的斜率和截距做进一步的压缩来减少存储成本. 同时，PGM 将多标准优化与数据结构结合，设计了多标准 PGM. 此外，PGM 利用键的分布规律和查询键的频率调整自身结构来适应查询操作的分布，是第一个基于分布感知的学习型索引. Ferragina 等人[19]的实验表明，在空间消耗上，PGM 索引比 FITing-TREE 节约了 63.3%，比 B$^+$树少了 4 个数量级. 同时，PGM 索引实现了与 FITing-TREE、B$^+$树相当甚至更好的查询性能.

（4）LIPP 索引（Updatable Learned Index with Precise Positions）

Wu 等人[20]提出了可更新学习型索引 LIPP，该索引的核心是避免模型的不精准预测，也就是解决冲突问题（即两个不同的键具有相同的预测位置）. LIPP 的索引结构如图 4 所示：LIPP 索引是一种树形结构，与一般树形索引不同，它没有叶子节点，所有的节点都具有相同的结构. LIPP 的每个节点都由学习模型、一个数组（用来存储项）和一个位向量（用来指示项的类型）组成. 其中，项的类型包括 NULL（表示空闲项）、DATA（表示项存储的记录）和 NODE（表示指向下一层的一个孩子节点的指针）三种类型. 如果某项存在冲突，则为当前项创建孩子节点，并将项的类型修改为 NODE. 为了解决冲突问题，LIPP 设计 FMCD 算法（Fastest Minimum Confilict Degree）通过计算最小冲突度（即预测位置为同一个位置的键的数量），并构造相应的模型来消除预测误差. 具体地，给定最小冲突度的约束条件、初始最小冲突度以及一个有序键集，FMCD 算法按照顺序依次遍历和划分有序键集. 若在当前划分的子集中，所有键均满足约束条件，则返回当前子集对应的模型以及最小冲突度. 否则，增加最小冲突度，重新训练模型，直到满足约束条件为止.

LIPP 索引采用就地插入策略实现键的插入，与其他索引不同的是，LIPP 在插入键后，需要检查冲突的数量是否满足约束条件. 若不满足，则需要创建新的节点，同时修改节点中项的类型. LIPP 采用

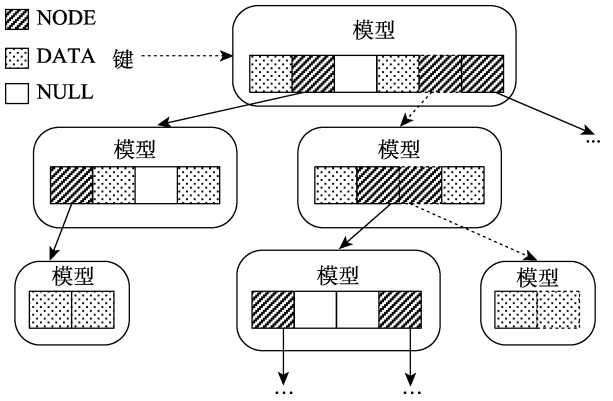

图 4 LIPP 索引

核化的线性模型，该模型能够将新插入的键均匀的映射到相应的位置，从而限制 LIPP 索引结构的深度，并通过调整策略维护索引结构的深度不超过最大增长上界. Wu 等人[20]的实验表明，在只读工作负载上，LIPP 的查询性能比 ALEX 索引提高了 2.8 倍，比 PGM 索引提高了 6.3 倍；在动态工作负载上，LIPP 和 ALEX 所消耗的空间成本几乎相同，但 LIPP 索引的插入性能要优于 ALEX 索引 2.9 倍.

3.1.2 基于并发技术优化 RMI 模型

这类方案虽然也是采用线性回归模型学习底层数据分布，但是在索引结构中考虑了并发问题. Tang 等人[21]提出了 XIndex，该索引采用已有的乐观并发控制[22-24]、细粒度锁[24-26]和 RCU(Read-Copy-Update)技术[27]，设计了一种两阶段合并算法来支持并发操作. XIndex 的索引结构如图 5 所示，该索引由两层架构组成：顶层是根节点，由一个两层的 RMI 模型和指向底层组节点的指针构成；底层是组节点，用来存储分组对应的模型、数据以及增量索引. 在 XIndex 索引中，根节点的 RMI 模型由整个数据集训练，然后，RMI 模型将数据集划分到多个分组中，在每个分组中继续使用线性模型进行训练. 为了支持插入操作，XIndex 索引在每个分组中关联一个增量索引用来进行缓冲插入，插入过程与 FITing- TREE 相似. Tang 等人[21]的实验表明，XIndex 与 Masstree[24]

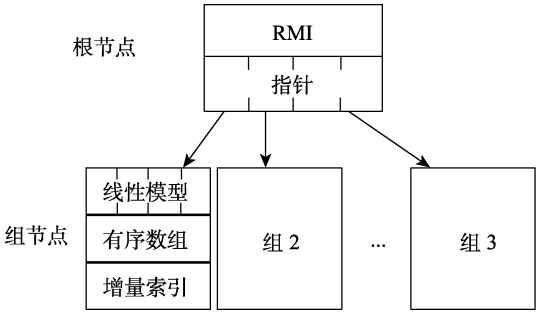

图 5 XIndex 结构

相比，索引性能提升了 3.2 倍，与 Wormhole[26]相比，索引性能提升了 4.4 倍.

3.1.3　基于永久内存优化 RMI 模型

内存索引的可扩展性受限于 DRAM 的高成本和低容量，而永久性内存 PM（Persistent Memory）能够提供高性能（略低于 DRAM）、支持永久性存储，且成本更低. 因此，PM 具有替代 DRAM 的潜力. Lu 等人[28]基于 PM 提出了 APEX 索引（A High-Performance Learned Index on Persistent Memory），该索引在 ALEX 索引的基础上进行改进以适应 PM，结合 PM 和学习型索引的优势（存储开销小、查询效率高等）提升索引性能. APEX 设计的基本原则是减少 PM 的访问次数以及存储开销. APEX 的索引结构由内部（模型）节点和数据（叶子）节点组成，内部节点包含一个线性回归模型和一个指针数组. 数据节点由两部分组成：第一部分是主数组和隐藏数组（用来存储溢出的记录），用来存储记录，这部分内容在 PM 中存储. 第二部分是元数据和加速器等，为了减少 PM 访问次数，这部分内容存储在 DRAM 中. 其中，一个加速器包括 16 字节的指纹（用来指示键是否存在），16 比特的槽位图（用来记录记录槽位是否空闲）以及 48 比特的指针. 为了减少存储开销，在主数组中，每 16 个记录共享一个加速器.

APEX 内部节点的查询方法与其他学习型索引相同，区别在于数据节点，APEX 通过一种探针-隐藏机制（probe-and-stash mechanism）进行查询. 具体地，给定一个查询键，APEX 预测查询键的位置，如果在预测位置中未找到查询键，则使用“探针”在主数组内继续查找. 如果在约束距离内“探针”未找到查询键，则检查隐藏数组. APEX 索引也支持更新操作，它将数据节点看作是哈希表结构，训练线性模型作为哈希函数来实现基于模型的插入. Lu 等人[28]的实验表明，APEX 索引与最优的 PM 索引相比，其吞吐量高出 15 倍，同时，APEX 索引可以在 42ms 的时间内完成故障恢复.

3.1.4　基于插值法优化 RMI 模型

该类方案中，学习型索引采用插值法替代线性函数来拟合数据分布.

Kipf 等人[29]提出 RS 索引（RadixSpline index），它采用样条插值法[30]拟合数据. RS 索引由一组样条点和一个基数表构成. 其中，键的子集组成样条. 基数表用来检索查询键的正确样条点. RS 索引采用自底向上的方式进行构造：首先，在底层构造一个具有误差边界的样条点；然后，将样条点的索引存储到基数表中. 与 RMI 模型相比，使用插值法拟合数据只需对数据进行一趟扫描即可完成自上而下的索引构建. Kipf 等人[29]的实验表明，RS 索引的构建速度与 ART[31]和 B 树索引几乎相当，但是要快于与 RMI 模型. RS 索引也仅支持静态查询负载，查询速度上，RS 索引优于 ART 和 B 树大约 1 到 2 个数量级，稍逊于 RMI 模型.

Setiawan 等人[32]采用多项式函数取代神经网络模型学习数据的分布，通过切比雪夫插值法[33]或伯恩斯坦插值法[34]求解多项式系数，从而获取键的预测位置. 与 RMI 模型相比，采用多项式函数构造学习型索引，查询时间基本相同，但是空间开销降低了约 2 个数量级，准确率提高了 30%~40%. 其主要原因在于需要调优的参数数量不同，多项式函数需要调优的参数只有一个，而 RMI 模型则有大量的参数需要调优，因此导致模型构造成本增加. 但是，切比雪夫插值法和伯恩斯坦插值法要求学习模型是分段线性的、平滑的，然而面对复杂的数据分布，线性模型很难捕获数据特征.

3.2　基于非层次结构的索引

由于 RMI 模型的分层结构复杂，需要训练的模型参数太多，模型训练成本较高，所以基于非层次结构的索引抛弃了 RMI 模型的层次组织方式，采用单层的方式构建学习型索引. 基于非层次结构的索引按照优化策略分为两类：基于模型解耦的学习型索引和非模型解耦的学习型索引.

3.2.1　基于模型解耦的学习型索引

RMI 模型的不同层之间存在较强依赖，下层模型需要由上层模型来选择，不能够有效支持更新. 因此，为了减少模型之间的依赖性，一些研究工作提出了基于模型解耦的学习型索引，通过消除模型之间的依赖型，建立单层模型来优化索引性能.

（1）AIDEL 索引（Adaptive InDEpendent Linear regression models）

为了减少分层模型之间的依赖关系，并能够提供高效的可伸缩性，实现低开销的索引更新，Li 等人[35]提出了一种自适应独立线性回归模型 AIDEL. AIDEL 索引采用 LPA（Learning Probe Algorithm）根据数据分布自适应的构造学习模型. LPA 将数据分布相同的数据划分到同一个分区中；然后，针对每一个分区训练模型，并保证每个分区的模型误差不超过给定阈值. 为了使得模型之间相互独立，LPA 通过构造键-值存储对来索引模型. 其中，键表示模型覆盖的第一个数据的记录，值表示指向模型的一个指针. 为了支持有效的更新操作，AIDEL 索引在

现有数据（键-值存储对）后面附加了一个排序列表结构，用来存放新插入的数据. 由于 AIDEL 索引所使用的模型是独立的，所以重新训练时只需要训练新插入的数据对应的模型，而不会影响其他的模型，从而减少重训练模型的成本. Li 等人[35]的实验表明，AIDEL 索引与 B^+树相比，插入性能提高了 1.3 倍到 2.7 倍，查询性能提高了 2 倍多. 与 RMI 模型相比，查询性能相当，但是 AIDEL 索引能够有效地实现可扩展性.

（2）Dabble 模型

高远宁等人[36]提出了 Dabble 模型，索引结构如图 6 所示. 该模型的构建过程为：首先，采用 K-means 聚类方法将数据集划分成 k 个区域；然后，分别在 k 个区域上采用神经网络学习数据的分布，k 个区域之间互相独立. 在进行模型训练时，Dabble 模型将数据访问的热点信息融入到神经网络模型中，提高模型对热点数据的预测精度. 在处理插入时，模型利用 LSM 树[37]延迟的思想来实现数据写入. 当插入新数据时，将数据放入一个 B 树的缓存中. 若缓存达到一定阈值，则将缓存中的数据一次性归并到相应的数据分区中. 然后，重新训练该分区的模型. 为了使得模型之间互相独立，Dabble 模型在预测部分增加了一个中间层，模型的预测结果是键在对应数据分区中的相对位置. 因此，在重新训练模型时只需要训练新键插入区域所对应的模型，无需训练所有模型，从而提高了模型的可扩展性. 高远宁等人[36]的实验表明，Dabble 模型的查询性能、插入性能以及空间代价均优于 B^+树和 ALEX 索引. 但是 Dabble 模型的索引性能受分区数量的限制，分区数量过多或者过少，都会影响模型的性能.

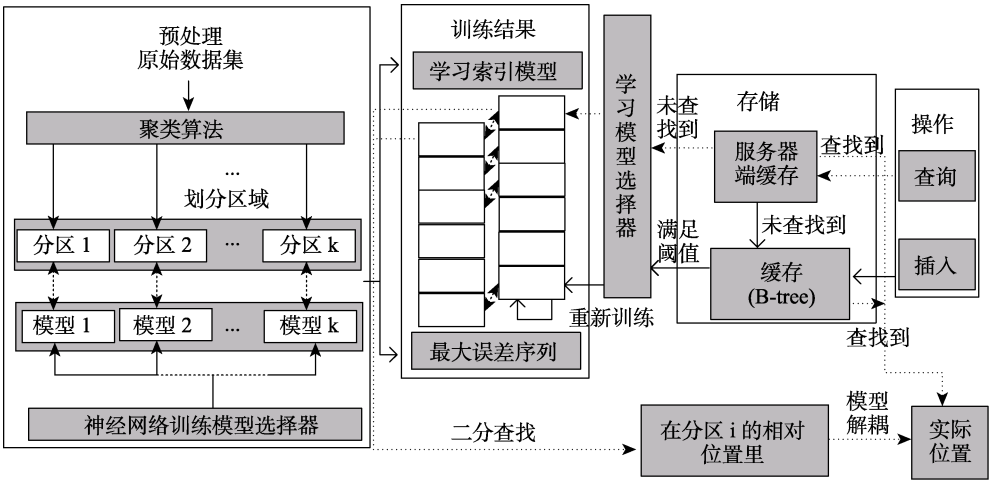

图 6 Dabble 模型

3.2.2 非模型解耦的学习型索引

Li 等人[38]提出了一种单层学习型索引 ASLM 索引（Adaptive Single Layer Model），该索引在数据划分中考虑了数据之间的欧式距离. 具体地，ASLM 索引首先计算每两个相邻数据之间的距离，并按照距离降序存储所有数据. 然后，选择前 k 个数据点作为划分的边界点，并保证划分后的每个分区大小不超过给定的阈值. 如果大于给定阈值，则重新划分. 最后，为每个分区创建和训练模型，记录每个子模型的最小和最大误差. 所有子模型组织在一层，而不是与 RMI 模型一样进行分层组织.

为了支持更新操作，ASLM 为每个子模型维护一个缓冲区. 如果插入键的误差小于等于平均误差，则直接插入数据；否则，需要检查缓冲区. 如果缓冲区超过给定阈值，则插入键后需要重新训练对应的子模型，并更新相应的平均误差、最小误差、最大误差以及缓冲区. 对于删除操作，移除键不会影响模型的精确度，因此不需要重新训练模型. 但对于插入和删除操作，都需要检查子模型对应的分区大小是否超过给定阈值，分区大小过大（过少）时，需要重新划分（合并）数据分区. Li 等人[38]的实验表明，ASLM 索引的预测精度比 RMI 模型提高了 50%. 在不同的动态工作负载上，ASLM 索引的插入性能表现稳定.

3.3 基于辅助方法的学习型索引

为减少学习模型的复杂度或设计更有效的插入策略，研究人员设计了两种辅助方法来提升学习型索引的性能.

Hadian 等人[39]提出了一种基于偏移表（Shift-Table）的方法，通过增加偏移表来纠正模型误差，

从而减少模型数量，降低学习模型的复杂度和规模. 偏移表包含一个查找表，表中存储了键的预测位置、偏移值以及具有相同预测位置的索引键的数量. 增加了偏移表的学习型索引结构如图 7 所示. 偏移表与学习模型之间是相互独立的，即学习型索引可以根据需要来选择是否采用偏移表. 通过偏移表，学习型索引能够快速的减小预测误差，提高查询速度. Hadian 等人[39]的实验表明，增加偏移表的 RMI 模型和 RS 索引在查询性能上提升了 1.5 倍到 2 倍多.

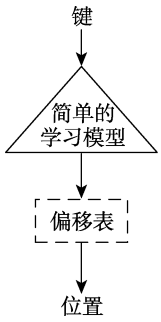

图 7　学习模型+偏移表

Li 等人[40]量化分析了学习型索引的“学习”能力，提出了一种基于最小描述长度原则的学习框架. 该框架可用来衡量学习型索引，对不同场景下选择合适的学习型索引有所裨益. 具体地，该框架将学习索引过程中的“预测”和“修正”两个阶段进行量化分析，采用最小描述长度构建目标函数. 此外，针对大规模数据集，该框架提出了采样技术，通过采样数据集的子集进行学习，从而提高学习型索引的泛化能力. 针对动态工作负载，从学习效果的角度出发，研究了结果驱动的间隙插入技术. Li 等人[40]的实验表明，他们提出的学习框架将现有学习型索引（RMI、FITing-TREE 和 PGM）的构建速度提高了 78 倍，在静态和动态负载下，将查询速度提高了 1.59 倍.

3.4　一维学习型索引方法的对比分析

一维学习型索引主要是从分区策略、误差保证以及更新操作等方面对 RMI 模型进行优化，旨在减小误差和提高索引性能. 基于上述分析，本节从索引结构、数据划分、学习模型、误差保证、插入策略、模型重训练以及索引优缺点这 7 个方面总结和比较一维学习型索引技术，详细对比如表 1 所示. 其中，索引的删除过程与插入过程类似，因此，不在表中进行赘述. 为了便于理解，对表 1 中的部分内容进行解释说明：

（1）学习模型

目前采用最多的是 RMI 模型框架. RMI 模型采用混合模型，即上层（根模型）使用神经网络模型，下层（中间层和叶模型）使用简单的线性回归模型. 由于神经网络训练的代价以及参数调优的成本较高，比线性回归模型更复杂. 因此，大多数索引方法采用线性回归模型，然后通过分区技术来提高线性模型的预测精度. 除了线性回归模型，使用插值函数也能达到与线性回归模型相当的性能.

（2）误差保证

模型预测做不到完全精确，预测位置和实际位置之间存在误差，对索引的查询性能影响较大. 如果误差过大，则大大增加本地搜索的时间，导致学习型索引查询性能低于 B 树. 因此，需要限制误差大小. 一维学习型索引约束误差的方法是预先设置阈值（例如，限制分区大小等），通过预先指定阈值能够有效控制索引最坏情况下的查询性能.

（3）插入策略

插入策略主要两种：第一种是就地插入策略，一般是在数组中预留一些空隙用来插入新数据；第二种是异地插入策略，通过额外设置缓冲区或者溢出块来存储新数据. 两种方法相比，理论上异地插入策略要优于就地插入策略. 其原因在于，异地插入策略需额外开辟空间存储新数据. 但 Galakatos 等人[14]的实验表明，就地插入策略在插入的数据量和预测误差都较小的情况下性能更优，原因是在数据集和预测误差较小的情况下，就地插入策略空闲的间隙较多，搜索范围小，因此需要移动的键的数量少，并占用较少的空间. 而异地插入策略，由于开辟了固定大小的空间，所以要比就地插入策略占据更多的空间.

4　多维学习型索引

随着一维学习型索引的不断发展，将学习型索引的思想应用到多维数据吸引了不少研究人员的关注. 处理多维数据的传统索引技术主要包括以下三类[41,42]：基于数据划分的索引（例如网格文件[43-45]、Kd 树[46]、四叉树[47]、八叉树[48]等），基于空间划分的索引（例如 R 树及其各种变体[49-52]），以及基于映射的索引[53-55]（例如 Z-order 曲线、希尔伯特曲线等）. 将索引看作是模型，则传统多维索引结构也可以用机器模型替换，例如，R 树可看作是一个将目标键映射到 MBR（Minimum Bounding Rectangle）的函数. 然而，较之一维数据，在多维数据上进行建模和学习是非常困难的. 因为模型更容易学习有序数据的分布特征，而多维数据大多是杂乱无序的，直接采用 RMI 模型去学习多维数据的累积分布函数

表 1 一维学习型索引比较

索引	组织方式	数据划分	学习模型	误差保证	插入策略	模型重训练		优点	不足
						何时训练	训练规模		
RMI[7]	层次结构	不划分	混合模型	不固定	不支持	数据分布改变	所有模型	替换树遍历操作查询效率提高	不支持动态更新数据分区不合理分层模型复杂
FITing-TREE[14]	树形结构	贪心算法	分段线性函数	预先定义	缓冲区插入/就地插入	分段超过指定误差	部分模型/所有模型	构建速度快；保证最坏情况下的查询性能	B^+树结构组织数据分区，需要树遍历操作，查询速度下降；
ALEX[17]	层次结构	模型划分	线性回归模型	不指定	GA/PMA	满足数组扩充条件	部分模型	自适应结构，约束RMI模型的深度上限	缺乏误差约束，并牺牲查询时间来支持更新操作
PGM[19]	层次结构	贪心算法	优化的PLA模型	预先定义	LSM	缓存块达到阈值	部分模型	单趟构建，索引构建速度快. 压缩技术，减少空间存储开销	牺牲查询时间来支持更新操作
LIPP[20]	树形结构	FMCD	线性回归模型	最小冲突度	就地插入	满足触发条件时	部分节点	预测精度高	模型结构复杂存储开销大
XIndex[21]	层次结构	范围划分	线性回归模型	不固定	缓冲区插入	分组数量超过界限	部分模型	两阶段机制并发控制	模型结构复杂存储开销大
RS[29]	层次结构	样条点插值	插值函数	预先定义	不支持	未讨论	未讨论	自底向上构造，单趟构建，构建速度快	不支持更新操作
AIDEL[35]	非层次结构	贪心算法	线性回归模型	预先定义	有序列表	有序列表长度超过界限	所有模型/部分模型	模型互相独立	模型精度不高
Dabble[36]	非层次结构	k-means	神经网络	不固定	LSM	缓存块达到阈值	缓存块对应的模型	模型互相独立	模型性能受分区数量的限制
ASLM[38]	非层次结构	贪心算法	神经网络	不固定	缓冲区插入	缓冲区满	缓冲区对应的模型	模型结构简单查询效率高	模型精度不高

很困难. 如图 8 所示，如果简单的学习多维 CDF，则不同的数据之间可能会有相同的映射值. 但是具有相同映射值的数据之间的实际距离可能很远，导致在查询时需要检查大量不相关的数据，降低查询性能. 文献[56]首次提出了多维学习型索引，但是他们的研究工作中并没有介绍具体的实现细节. 后续关于多维学习型索引的研究也开展了不少的工作[57-59]，这些工作都是基于 RMI 模型的层次结构设计了不同的多维学习型索引技术. 本文按照索引构造方式将这些索引技术划分为基于降维方式的索引（4.1 节）和非降维方式的索引（4.2 节）进行梳理和总结，并从多个角度进行对比和分析（4.3 节）.

4.1 基于降维方式的学习型索引

这类方案是将多维数据投影到一维空间，按照一维数据的方式创建学习型索引. 根据投影策略的不同，将基于降维方式的学习型索引细划分为以下三类：基于勒贝格测度的索引、基于空间填充曲线的索引和基于数据划分的索引.

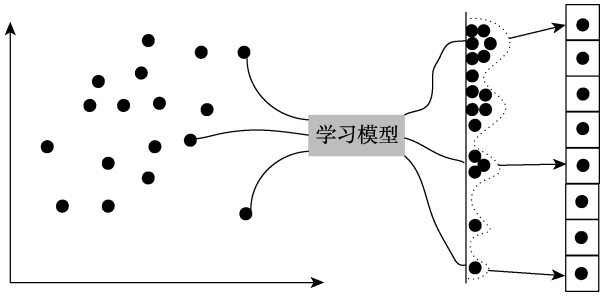

图 8 多维 CDF

4.1.1 基于勒贝格测度的索引

Li 等人[57]提出了一种基于磁盘的多维学习型索引 LISA（Learned Index Structure for Spatial Data）. 该索引首先将多维数据空间划分为网格，数据按照网格单元有序；然后，采用基于勒贝格测度(Lebesgue

Measure)[60]的方法构建函数将多维数据转化成一维值；接着，使用一个单调分片预测函数将一维数据空间划分成多个分片，将每个一维值映射到一个分片 ID 中. 为了减少模型复杂度，预测函数采用分段线性模型；最后，训练一个局部模型将每个分片分配到相应的磁盘页中，模型的输入是分片所包含的一维值，输出是该一维值对应的页地址. LISA 能够处理范围查询和 KNN 查询，在处理 KNN 查询时，采用格回归模型(lattice regression model)[61]将 KNN 查询转化为范围查询进行处理. LISA 也支持更新操作：当插入数据时，通过模型预测数据的页地址. 如果当前磁盘页已满，则创建新的磁盘页，否则将数据插入到对应的位置. 删除操作与插入操作相似，因此，不再赘述. Li 等人[57]的实验表明，LISA 索引的查询效率要高于 R 树和 Kd 树，空间开销低于 R 树，略高于 Kd 树.

4.1.2　基于空间填充曲线的索引

（1）ZM 索引（Z-order Model index）

针对空间数据，Wang 等人[58]提出了 ZM 索引. 该索引利用 Z-order 曲线[53]将多维数据转化成一维有序的 Z-address. 其中，Z-order 曲线是一种空间填充曲线，它可以将多维向量空间完全填充，使所有的数据映射到一维空间. 每个多维向量均可转换成一个唯一的整数，即 Z-address. 同时，这种转换能够保证高维数据之间原有的大小关系，即当一个数据的每个维度都大于另一个数据时，它对应的 Z-address 也一定大于另一个数据. 如此可保证将高维的范围查询转换成一维数据上的范围查询.

将多维数据转换成一维数值 Z-address 后，ZM 索引采用与 RMI 模型相似的多阶段模型索引 Z-address. 其中，模型输入是 Z-address，输出结果是 Z-address 在有序数组中的预测位置. 为了查找到精确位置，ZM 索引利用 MBS 方法（Model Biased Search，MBS 是二分查找的一种变体方法）进行查找. ZM 索引结合 Z-order 空间填充曲线和 RMI 模型有效地解决了多维空间数据的索引问题. Wang 等人[58]的实验表明，ZM 索引与传统空间索引相比，在查询性能和空间消耗上均得到较大的改进. 但是在处理范围查询时，ZM 索引仍然需要检查大量不相关的数据.

（2）RSMI 索引（Recursive Spatial Model Index）

采用 Z-order 曲线将多维数据转换成一维数值的方法，虽然能够处理多维数据. 但是转换后的一维数据分布不均匀，相邻的一维数值之间可能存在较大的间隙，影响模型性能. 针对该问题，Qi 等人[59]提出了 RSMI 索引，它在采用空间填充曲线降维之前，首先采用封装策略（Packing Strategy）[62]将原始多维数据空间映射到一个网格排序空间（Rank Space），根据原始数据的 x 坐标（y 坐标）对数据进行排序，其排序位置作为它在排序空间中的 x 坐标（y 坐标）. 将原始数据转换到排序空间后，再利用空间填充曲线获取每个空间数据的曲线值，依据该值对数据进行排序. 将排完序的空间数据按照顺序依次划分到固定大小的块中. 最后，RSMI 索引采用多层感知器 MLP（MultiLayer Perceptron）来学习一维数值的分布. 其中，模型输入是键，输出是键对应的块 ID.

为了提高索引的可扩展性，RSMI 索引采用了递归划分的方式处理大规模数据. 首先，RSMI 将数据集进行分区；然后，针对每个分区训练模型，如果分区仍然很大，则继续划分分区，并递归执行该过程. 为了减轻倾斜数据分布的影响，在 RSMI 的高层子模型中使用非规则网格和模型预测来对数据进行划分. RSMI 能够处理点查询、范围查询以及 KNN 查询，并提出了相应的查询算法. 同时，RSMI 也能够处理插入与删除操作，在执行插入操作时，新插入的数据存储到块中，如果当前块已满，则创建新的块并进行标记，然后递归地更新插入键对应的块的父模型. RSMI 索引与 ZM 索引都是利用空间填充曲线进行降维处理，但是 RSMI 索引采用排序空间技术，使得一维数值之间的间隙更加均匀. Qi 等人[59]的实验表明，RSMI 索引在查询性能上优于 R 树和 ZM 索引一个数量级以上.

4.1.3　基于数据划分的索引

针对空间数据，Davitkova 等人[63]也提出了一种内存只读多维学习型索引 ML（Multidimensional Learned structure）. 与 ZM 索引和 RSMI 索引采用空间填充曲线将多维数据转化成一维数值的方法不同，ML 索引采用类似于 iDistance 方法[64]的方法进行降维. ML 索引的结构如图 9 所示，索引主要由两部分构成：第一部分是一组参考点；第二部分由一个学习模型和一个有序数组构成. 更具体地，给定一组有序的参考点，ML 索引首先计算每个参考点与数据之间的欧式距离，将与参考点距离相近的数据划分到参考点对应的分区中；然后，根据分区偏移量以及每个数据与所在分区参考点之间的距离来计算每个数据的一维键值. 分区偏移量由当前分区的最大半径以及排序在当前分区之前的最大半径之和获得；最后，采用 RMI 模型来学习一维键值的分布，所有模型都是线性模型. 与 iDistance 方法相比，

ML 索引提出的投影方法不会产生重叠的分区，同时能够减小不同分区之间的间隙，有助于学习模型的构建. ML 索引使用基于数据的范围近似方法[65]处理范围查询.

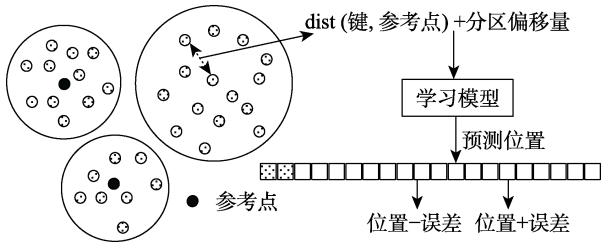

图 9 ML 索引结构

4.2 非降维方式的学习型索引

基于降维方式的索引能够有效处理多维数据，但是这种方法不能够充分利用原始多维数据空间的分布信息，使得模型预测精度不够准确. 为了更好的学习多维数据的分布特征，获取更多的有用信息，其他工作研究了非降维的方法.

（1）Flood 索引

Vikram 等人[66]提出了 Flood 索引，它是一种内存只读优化多维学习型索引. Flood 索引通过自动调整数据布局和索引结构来适应不同的数据分布和查询负载. Flood 索引的系统架构如图 10 所示，它包含离线阶段和在线阶段. 其中，离线阶段完成数据布局、布局优化以及索引的构造；在线阶段执行查询操作，包括投影、精确以及扫描操作. 更具体地，在离线阶段，Flood 索引设计了两种不同的数据布局，分别是等间距列的布局和针对倾斜数据的扁平化布局（Flatten Layout）. 两种布局都是以一个 d-1 维的网格去覆盖 d-1 维的属性，然后选择剩余的维对数据点进行排序. 其中，扁平化布局利用 RMI 模型划分数据. 为了选择最优的布局（也就是每个维应该选择划分的列数量以及选择哪个维作为排序维），Flood 索引设计了一个代价模型，通过随机森林模型训练来求解最优布局. 在线查询阶段中，由于模型预测存在误差，因此，需要在误差保证的范围内执行本地搜索. 为了加快搜索速度，Flood 索引采用 PLM 模型（Piecewise Linear Model）替换二分查找，具体地，PLM 模型在排序维的属性值上学习每个单元格（cell）的 CDF 模型来估计物理索引范围.

Flood 的一个显著优点就是能够调整自身结构以适应不同的数据分布和查询工作负载，而其他多维学习型索引仅利用数据的分布特征构建学习模型，没有考虑查询工作负载. Vikram 等人[66]的实验表明，Flood 索引的查询性能要优于传统多维索引 30~400 倍，且消耗较少的内存空间.

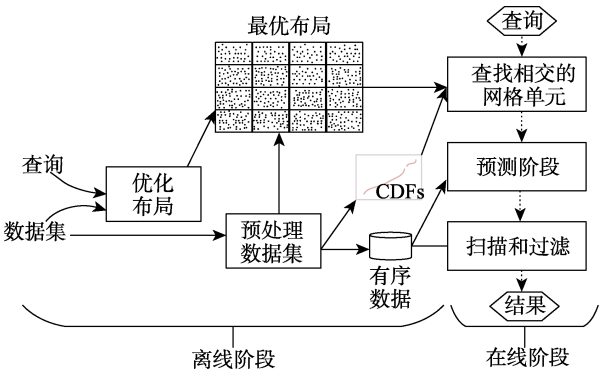

图 10 Flood 系统架构

（2）Tsunami 索引

针对倾斜工作负载和索引关联数据的问题，Ding 等人[67]在 Flood 基础上提出了一种新的内存只读优化多维学习型索引 Tsunami. 为了解决数据倾斜分布的问题，Tsunami 构造了一种轻量级决策树——网格树（Grid Tree），通过将空间划分为不重叠的区域来减少数据分布倾斜的情况. 对于属性关联问题，Tsunami 提出了增强网格（Augmented Grid）为每个区域建立索引，通过函数映射和条件 CDFs 两种技术来捕捉关联属性之间的相关性信息，有效解决了属性关联的问题. Tsunami 索引也是根据数据和查询工作负载自动优化数据存储组织和索引. Ding 等人[67]的实验表明，Tsunami 索引在查询性能上比 Flood 提高了约 6 倍，索引大小要比 Flood 小 8 倍多. 与传统索引相比，Tsunami 索引在查询速度和空间代价上均占优势.

（3）SPRIG 索引（Spatial Interpolation Function Based Grid Index）

Zhang 等人[68]采用双线性插值函数来学习空间数据的分布，提出了 SPRIG 索引. 该索引采样空间数据来构建自适应网格，使用样本数据作为输入来拟合插值函数，输出为网格单元的 ID. 给定一个空间查询键，首先通过空间插值函数来获取键的预测位置；然后，基于最大估计误差，执行一个局部二分搜索来查找准确位置. SPRIG 的索引结构包括三个部分：一个 $n \times m$ 大小的网格布局、一张映射表以及基于空间插值函数的学习模型. 映射表用来存储每个网格单元的 ID、每个网格单元覆盖的第一个数据的记录以及所覆盖的记录的数量. SPRIG 能够处理范围查询和 KNN 查询，并且提出了一种基于枢纽

（Pivot）的方法来改进 KNN 查询的性能. Zhang 等人[68]的实验表明，与 Flood 和传统空间索引相比，SPRIG 在查询速度上占有优势. 在空间代价上，SPRIG 索引小于传统空间索引，但高于 Flood 索引，主要是因为 SPRIG 需要存储一个 $n\times m$ 大小的网格，而 Flood 不需要. SPRIG 的优势在于其优秀的查询性能可以平衡存储开销，并且能够直接学习空间数据的分布. 但是该学习型索引使用双线性插值函数，因此只能够处理二维数据.

4.3　多维学习型索引对比分析

本小节从数据分区方式、学习模型、查询类型以及优缺点四个方面对多维学习型索引进行总结和比较，详细内容如表 2 所示. 为了便于理解，对表中部分内容进行解释说明：

（1）排序方式：对于有序数据，学习模型更容易拟合数据的累积分布函数. 因此，在构造多维学习型索引之前，需要对多维数据进行排序，排序方法有两种：第一种方法是通过将多维数据转换成一维数据进行排序，该方法的关键是投影策略的选择，多维学习型索引主要采用的投影策略包括勒贝格测度、空间填充曲线、距离度量函数；第二种方法是采用非降维的方式，利用网格单元对多维数据排序.

（2）学习模型：在学习多维数据的分布时，两种学习方式的本质还是将多维数据转化成一维数据进行学习，区别是基于降维的方式是将多维数据转化到一维空间进行学习，而非降维的方式则是直接学习多维数据每一维的累积分布函数. 因此，多维学习型索引选择的模型与一维学习型索引的相同，大部分是采用线性回归模型，少数则是采用插值函数.

表 2　多维学习型索引比较

索引	数据分区/投影策略	学习模型	点查询	范围查询	KNN 查询	优点	不足
LISA[57]	网格划分+基于勒贝格测度的投影	分段线性回归模型	未讨论	支持	支持	支持更新操作 支持磁盘存储	更新成本高
ZM[58]	Z-order 曲线	RMI	支持	支持	不支持	空间代价小	只能处理二维数据；不支持更新和 KNN 查询
RSMI[59]	排序空间+空间填充曲线	RMI	支持	支持	支持	支持更新操作 预测误差小	递归划分索引空间，维护成本高
ML[63]	K-means+改进的 iDistance 策略	MLP	支持	支持	支持	无分区重叠	不支持更新
Flood[66]	基于代价模型的网格划分	分段线性回归模型	支持	支持	支持	索引结构能够自适应不同的工作负载	倾斜工作负载中性能差，不支持更新操作
Tsunami[67]	基于代价模型的网格划分	分段线性回归模型	支持	支持	支持	支持倾斜工作负载和关联数据	不支持更新操作
SPRIG[68]	自适应网格	双线性插值函数	支持	支持	支持	KNN 查询效率高	只能处理二维数据；不支持更新

5　学习型索引的扩展研究

除了上述工作，研究人员还在其他方面对学习型索引进行拓展研究，主要包括以下 6 个方面.

5.1　面向其他查询类型的学习型索引

除了范围查询之外，学习型索引也可以应用于其他查询类型，例如点查询、存在性查询等.

（1）面向点查询的学习型索引

1）哈希索引

哈希索引通常用来处理点查询问题，哈希索引通过哈希函数把键映射到数组中的随机位置. 创建哈希索引的关键是避免冲突，即避免将不同的键映射到相同的位置. 机器学习方法为哈希索引减少冲突提供了一个可行的方案，Kraska 等人[7]通过学习键的分布来学习哈希函数，在学习模型时采用 RMI 模型，他们在实验中结合基于链表的散列映射来测试学习哈希索引的性能.

2）倒排索引

倒排索引[69-72]是文本索引中常用的一种典型索引结构，通常用来处理文档、网页等类型的数据. Oosterhuis 等人[73]结合传统倒排索引的查询优化方法[74,75]，使用深度神经网络模型学习一个布尔函数

来判断单词是否存在于文档中. 文献[76]利用多层递归神经网络模型替代倒排索引中的哈希索引部分，提出了一种倒排学习型索引结构 Pavo.

（2）面向存在性查询的学习型索引

布隆过滤器通常用来解决存在性查询问题，Kraska 等人将存在性查询问题看作是一种二分类任务[7]，通过训练递归神经网络或者卷积神经网络来学习布隆过滤器，它们在字符串建模中多次证明是有效的[77,78]. 除此之外，Mitzenmacher 等人[79]改进了学习布隆过滤器，提出了三明治结构的学习布隆过滤器. 文献[80]利用 RNN 模型输出的概率评分的分布，自适应调整不同区域的哈希函数数量. 针对动态更新，Rae 等人[81]提出神经布隆过滤器，通过使用元学习[82]和记忆增强神经网络[83]来实现数据的动态更新. Bhattacharya 等人[84]通过重新训练异地插入的数据和动态分区布隆过滤器[85]两种方案来解决学习布隆过滤器中的插入问题. 文献[86]将上述问题扩展到多维空间，通过学习布隆过滤器来处理多维数据的存在性查询问题.

5.2 机器学习辅助的传统索引结构

这类索引结构不是严格意义上的学习型索引，它们主要是通过机器学习方法来改进而不是替换传统索引结构. Llaveshi 等人[87]在 B^+树节点内部使用线性回归模型加快搜索速度. IFB-Tree[88]采用插值法构建 B^+树，在进行插值搜索时，保证每个节点中的数据搜索误差小于给定阈值. Qu 等人[89]使用线性回归模型进行预测，在构造索引时将数据中的离群值排除，然后存储到 B^+树中. 针对多维索引，文献[90]提出了一种 RLR-tree，在 R 树的子树选择阶段和节点划分阶段采用 RNN 来构造. 文献[91]将 Flood 索引的分区技术应用于传统多维索引结构（Fixedgrid[92]、Adaptive-grid[45]、Kd 树[46]、Quadtree[47]和 STR[93]），并在精确查找阶段采用学习模型（例如 RS）取代传统的搜索方法.

5.3 面向辅助索引的学习型索引

目前，在一维学习型索引和多维学习型索引中都是针对有序数据构建索引. 但也有一些工作关注了无序数据，文献[94]针对无序数据提出了 Hermit 索引机制，它将学习型索引作为辅助索引. 该索引构造了一棵 TRS-tree，通过分层回归方法学习关联属性之间的软函数依赖关系，允许离群值的存在，并在叶子节点设置缓冲区存储离群值. 文献[95]是将学习型索引与辅助索引混合使用，使用学习型索引处理关联属性之间的软函数依赖关系，采用辅助索引处理离群值.

5.4 面向时空查询的学习型索引

文献[96]针对移动对象提出了 NEIST（Neural-Enhanced Index for Spatio-Temporal Queries）来处理时空查询. NEIST 采用递归神经网络 RNN 预测基于观测轨迹的运动目标在未来某段时间的可能位置，并提出一种后缀树将运动轨迹相似的移动对象划分到一组，从而减少预测开销. 在每段时间内，采用传统的线性预测模型建立 TPR-tree 来支持时空查询.

5.5 学习型索引的测试基准

目前学习型索引之间的性能分析和比较，大多都是采用各自的合成数据集，在不同的环境中进行. 因此，对于学习型索引性能是否真的优于传统索引结构存疑. 针对此问题，Kipf 和 Marcus 等人[97-98]研究了学习型索引的测试基准，提供了统一的测试平台来比较不同学习型索引以及学习型索引与传统索引结构之间的性能.

Kipf 和 Marcus 等人[97]提出了 SOSD（Search On Sorted Data Benchmark）框架. 该框架分别在 4 种合成数据集（对数正态分布、正态分布、密集整数集以及均匀分布的稀疏整数集）和 4 种真实数据集（图书销售人气数据、Facebook 用户 ID 数据集的上采样版本、Open Street Map 的小区 ID 以及维基百科的编辑时间戳）上，从索引创建时间、预测误差、缓存丢失以及查询性能等方面测试了学习型索引 RMI 模型和 RS 索引. 并提供三类传统索引技术作对比分析，它们分别是：动态算法（BS、IS、TIP[99]）、辅助索引（RBS）、传统索引结构(ART[31]、B 树和 FAST[100]). 此后，Kipf 和 Marcus 等人[98]又对 SOSD 进行扩展，主要包括：（1）测试数据集不同，仅采用 SOSD 的四种真实数据集进行评估. 因为采用合成数据集评估学习型索引存在问题，现有的合成数据或是完全随机的，无法学习到基础数据的有效模型；或是从已知分布中提取，学到的模型无法表示整体的数据分布情况. （2）进行对比分析的索引技术除了 SOSD 中的传统索引结构，还增加了 IBTree[101]（Interpolating B-Tree）、FST[102]（Fast Succint Tree）和 Wormhole 索引. （3）参与测试的学习型索引增加了 PGM 索引. （4）测试了学习型索引在多线程系统中的性能和构建时间.

两种测试基准为学习型索引技术提供了统一的评测环境，并且公开源代码，为后续学习型索引技术的研究给予了帮助. 同时，新的学习型索引技术

可直接通过两种测试基准进行性能评估. 目前，两种测试基准还存在局限性，它们仅测试了静态查询负载下的一维学习型索引.

5.6　学习型索引的参数调优

为了提高模型预测精度，RMI 模型需要不断地调优参数. 参数的大小以及选择都由具体的数据集或者工作负载确定，选择合适的参数才能使得模型拟合效果达到最佳. 因此，为了直观理解模型参数调优对 RMI 模型性能的影响情况，Ryan Marcus 等人[103]设计了一种可视化工具 CDFShop. CDFShop 提供了两种可视化场景：第一种场景是在 RMI 模型参数的不同配置下，将查询时间以及内存的变化情况进行可视化. 在这种场景下，用户可以直观地观察到 RMI 模型近似 CDF 的过程，以及不同参数配置对模型性能的影响情况. 第二种场景是将 RMI 模型优化的每个过程进行可视化，用户可以直接观察每个优化过程所带来的性能效益. CDFShop 将 RMI 模型参数调优的过程进行可视化，方便研究者直观地查看参数调优对 RMI 模型的影响，为后续更好的探索 RMI 模型的优化技术提供了帮助.

6　未来研究方向

通过上述分析和比较，可以看出学习型索引技术取得了一定的成果，但面对海量、多维、异构数据的高效索引需求时，现有的学习型索引技术适用场景有限，采用学习型索引技术替换传统索引结构仍然面临着许多问题和挑战，还需进一步的研究：

（1）如何实现学习型索引的语义保证. 学习型索引技术的核心是建立学习模型. 然而，机器学习技术使用的数据和传统索引结构处理的数据存在差别. 传统索引结构处理的数据具有复杂的语义，并且数据之间一般存在关联性，例如关系型数据库中的数据，不同属性之间存在依赖关系. 而机器学习技术通常假设数据是独立同分布的，即数据之间不存在关联性. 因此，采用学习型索引替换传统索引结构时，需要考虑数据自身表达的语义以及数据之间的关联性.

（2）如何减少学习型索引的更新代价和维护成本. 学习型索引依赖于数据分布，更新操作将会引起数据分布发生改变，需要重新训练模型来维护模型拟合的效果. 但是模型重训练成本和维护代价很高，尤其是对于大规模数据集. 然而，数据的更新是数据库系统中必不可少的操作，如何平衡模型重训练成本和学习型索引的效用以及效率，是学习型索引技术还需要继续研究的关键问题.

（3）如何将学习型索引整合到数据库系统. 当前学习型索引研究聚焦于学习模型，未涉及在数据库系统中的应用. 将学习型索引技术集成到数据库系统中具有挑战性，这不仅要求保证学习型索引的优势，同时又要提升数据库系统的性能. 索引作为数据库系统的核心组件之一，与其他组件之间存在一定的联系. 当数据库系统中某个组件的性能改变时，往往会对其他组件产生影响，不利于数据库系统的性能提升. 同时，将学习型索引技术集成到数据库系统涉及来自数据库系统、机器学习模型以及编程的挑战.

（4）多维学习型索引的扩展研究. 在大数据环境下，数据呈现海量、高维、复杂等的特点. 采用学习型索引技术替换传统多维索引结构来处理数据存在很大的优势. 然而，多维学习型索引的工作大多关注中低维空间数据的静态查询，在动态更新、不同的查询需求以及更高维数据等方面尚处于空白，还需要进一步探索和研究.

（5）学习型索引的评测技术. 提供统一的测试平台公平的比较和分析不同学习型索引之间的性能，能够帮助研究人员更深入的研究和发展学习型索引技术. 为了更好的挖掘学习型索引技术的潜在优势，测试平台应该尽可能的包含多方面的测试内容. 然而，现有的测试基准仅考虑静态查询负载下的一维学习型索引，而动态查询负载和多维学习型索引方面的评测尚缺乏研究.

（6）机器学习辅助的索引结构. 学习型索引技术的研究尚处于探索阶段，当前提出的技术缺乏理论保证. 而传统索引技术发展成熟，具有一套相对完整的理论体系，同时能够有效地应用于实际的数据库系统中. 因此，可结合传统索引结构的优势，利用机器学习模型改进而不是取代传统索引结构.

7　总　结

大数据环境下，索引面临空间代价高、查询效率低、存取开销大等难题. 学习型索引可充分挖掘底层数据的分布规律，利用机器学习构建索引模型，提升索引性能并降低空间代价. 本文对现有学习型索引技术进行系统综述，首先，从学习型索引技术的基础模型入手，对其实现原理、构造和查询过程进行了剖析，并总结了基础模型的优点和存在的问题. 其次，按照索引结构特点对不同学习型索引进行分类，对一维学习型索引和多维学习型索引从创建方式、优化策略以及优缺点等方面进行梳理和总结. 此外，本文总结了学习型索引的扩展研究，包

括其他查询类型的学习型索引、测试基准以及参数调优等. 最后,探讨了学习型索引的发展趋势. 学习型索引作为数据库领域的一个研究热点，目前该领域仍有许多问题需要做深入的研究.

参 考 文 献

[1] Garcia-Molina H, Ullman J D, Widom J. Database System Implementation. Englewood Cliffs, USA: Prentice-Hall, 2000

[2] Richter S, Alvarez V, Dittrich J. A seven-dimensional analysis of hashing methods and its implications on query processing. Proceedings of the VLDB Endowment, 2015, 9(3): 96-107

[3] Fan B, Andersen D G, Kaminsky M, et al. Cuckoo filter: Practically better than bloom//Proceedings of the ACM International on Conference on emerging Networking Experiments and Technologies. Sydney, Australia, 2014:75-88

[4] Alexiou K, Kossmann D, Larson P Å. Adaptive range filters for cold data: Avoiding trips to siberia. Proceedings of the VLDB Endowment, 2013, 6(14): 1714-1725

[5] Graefe G, Larson P A. B-tree indexes and CPU caches//Proceedings of the International Conference on Data Engineering. Heidelberg, Germany, 2001:349-358

[6] Sun Lu-Ming, Zhang Shao-Min, Ji Tao, Li Cui-Ping, Chen Hong. Survey of data management techniques powered by artificial intelligence. Journal of Software, 2020, 31(3): 600-619 (in Chinese)
(孙路明, 张少敏, 姬涛, 李翠平, 陈红. 人工智能赋能的数据管理技术研究. 软件学报, 2020, 31(3): 600-619)

[7] Kraska T, Beutel A, Chi H, Dean J, Polyzotis N. The case for learned index structures//Proceedings of the 2018 International Conference on Management of Data. Houston, USA, 2018: 489-504

[8] Chai Ming-Ke, Fan Ju, Du Xiao-Yong. Learnable database systems: challenges and opportunities. Journal of Software. 2020, 31(3): 806-830 (in Chinese)
(柴茗珂, 范举, 杜小勇. 学习式数据库系统:挑战与机遇. 软件学报, 2020,31(3):806-830)

[9] Li Guo-Liang, Zhou Xuan-He, Sun Ji, Yu Xiang, Yuan Hai-Tao, Liu Jia-Bin, Han Yue. A survey of machine learning based database technology. Chinese Journal of Computers, 2020, 43(11): 2019-2049 (in Chinese)
(李国良, 周煊赫, 孙佶, 余翔, 袁海涛, 刘佳斌, 韩越. 基于机器学习的数据库技术综述. 计算机学报, 2020, 43(11): 2019-2049)

[10] Meng Xiao-Feng, Ma Chao-Hong, Yang Chen. Survey on machine learning for database systems. Journal of Computer Research and Development, 2019, 56(9): 1803-1820 (in Chinese)
(孟小峰, 马超红, 杨晨. 机器学习化数据库系统研究综述, 计算机研究与发展, 2019, 56(9): 1803-1820)

[11] Ferragina P, Vinciguerra G. Learned data structures//Recent Trends in Learning From Data - Tutorials from the INNS Big Data and Deep Learning Conference. Sestri Levante, Italy, 2019: 5-41

[12] Zhang Zhou, Jin Pei-Quan, Xie Xi-Ke. Learned indexes: current situations and research Prospects. Journal of Software, 2021, 32(04): 1129-1150 (in Chinese)
(张洲, 金培权, 谢希科. 学习索引:现状与研究展望.软件学报, 2021, 32(04): 1129-1150)

[13] Shazeer N, Mirhoseini A, Maziarz K, Davis A, Le QV, Hinton GE, Dean J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv: 1701.06538, 2017

[14] Galakatos A, Markovitch M, Binnig C, et al. Fiting-tree: a data-aware index structure//Proceedings of the International Conference on Management of Data. Amsterdam, The Netherlands, 2019: 1189-1206

[15] Liu X, Lin Z, Wang H. Novel online methods for time series segmentation. IEEE Transactions on Knowledge & Data Engineering, 2008, 20(12): 1616-1626

[16] Xu Z, Zhang R, Kotagiri R, Parampalli U. An adaptive algorithm for online time series segmentation with error bound guarantee// Proceedings of the International Conference on Extending Database Technology. Berlin, Germany, 2012: 192-203

[17] Ding J, Minhas U F, Yu J, et al. ALEX: an updatable adaptive learned index//Proceedings of the ACM SIGMOD International Conference on Management of Data. Portland, USA, 2020: 969-984

[18] Bender MA, Hu H. An adaptive packed-memory array. ACM Transactions on Database Systems, 2007, 32(4): 1-43

[19] Ferragina P, Vinciguerra G. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. Proceedings of the VLDB Endowment, 2020, 13(8): 1162-1175

[20] Wu J, Zhang Y, Chen S, et al. Updatable learned index with precise positions. Proceedings of the VLDB Endowment, 2021, 14(8): 1276-1288

[21] Tang C, Wang Y, Dong Z, et al. XIndex: a scalable learned index for multicore data storage//Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. San Diego, USA, 2020: 308-320

[22] Bronson N, Casper J, Chafi H, Olukotun K. A practical concurrent binary search tree. ACM SIGPLAN Notices, 2010, 45(5): 257-268

[23] Cha SK, Hwang S, Kim K, Kwon K. Cache-conscious concurrency control of main-memory indexes on shared-memory multiprocessor systems//Proceedings of the International Conference on Very Large Data Bases. Roma, Italy, 2001: 181-190

[24] Mao Y, Kohler E, Morris R. Cache craftiness for fast multicore key-value storage//Proceedings of the Seventh EuroSys Conference on Computer Systems. Bern, Switzerland, 2012: 183-196

[25] Binna R, Zangerle E, Pichl M, Specht G, Leis V. HOT: A height optimized trie index for main-memory database systems// Proceedings of the International Conference on Management of Data. Houston, USA, 2018: 521-534

[26] Wu X, Ni F, Jiang S. Wormhole: A fast ordered index for in-memory data management//Proceedings of the Fourteenth EuroSys Conference on Computer Systems. Dresden, Germany, 2019: 18:1-18:16

[27] McKenney P E, Appavoo J, Kleen A, et al. Read-copy update// Proceedings of the AUUG Conference. Sydney, Australia, 2001: 175-184

[28] Lu B, Ding J, Lo E, et al. APEX: A high-performance learned index on persistent memory. Proceedings of the VLDB Endowment. 2021, 15(3): 597-610

[29] Kipf A, Marcus R, van Renen A, et al. RadixSpline: a single-pass learned index//Proceedings of the Third International Workshop

on Exploiting Artificial Intelligence Techniques for Data Management. Portland, USA, 2020: 5:1-5:5

[30] Neumann T, Michel S. Smooth interpolating histograms with error guarantees//Proceedings of the British National Conference on Databases. Cardiff, UK, 2008: 126-138

[31] Leis V, Kemper A, Neumann T. The adaptive radix tree: ARTful indexing for main-memory databases//Proceedings of the International Conference on Data Engineering (ICDE). Brisbane, Australia, 2013: 38–49

[32] Setiawan NF, Rubinstein BI, Borovicagajic R. Function interpolation for learned index structures//Proceedings of the Australasian Database Conference. Melbourne, Australia, 2020: 68-80

[33] Brisebarre N, Joldes M. Chebyshev interpolation polynomial-based tools for rigorous computing//Proceedings of the Symbolic and Algebraic Computation, International Symposium. Munich, Germany, 2010: 147-154

[34] Alda F, Rubinstein B I P. The bernstein mechanism: Function release under differential privacy//Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. San Francisco, USA, 2017: 1705-1711

[35] Li P, Hua Y, Zuo P, et al. A scalable learned index scheme in storage systems. arXiv preprint arXiv:1905.06256, 2019

[36] Gao Yuan-Ning, Ye Jin-Biao, Yang Nian-Zu, Gao Xiao-Feng, Chen Gui-Hai. A middle layer based scalable learned index scheme. Journal of Software, 2020, 31(3): 620-633 (in Chinese)
(高远宁,叶金标,杨念祖,高晓泓,陈贵海. 基于中间层的可扩展的学习索引技术.软件学报, 2020, 31(3): 620-633)

[37] Oneil P, Cheng EY, Gawlick D, Oneil E. The log-structured merge-tree (LSM-tree). Acta Informatica, 1996, 33(4): 351-385

[38] Li X, Li J, Wang X. ASLM: Adaptive single layer model for learned index//Proceedings of the International Conference on Database Systems for Advanced Applications. Cham,Swizerland: Springer,2019: 80-95

[39] Hadian A, Heinis T. Shift-Table: a low-latency learned index for range queries using model correction//Proceedings of the International Conference on Extending Database Technology. Nicosia, Cyprus, 2021: 253-264

[40] Li Y, Chen D, Ding B, et al. A pluggable learned index method via sampling and gap insertion. arXiv preprint arXiv: 2101. 00808, 2021

[41] Ooi B C, Sacks-Davis R, Han J. Indexing in spatial databases. National University of Singapore, Singapore, 1996

[42] Singh H, Bawa S. A survey of traditional and mapreducebased spatial query processing approaches. ACM SIGMOD Record, 2017, 46(2): 18-29

[43] Hinrichs K. Implementation of the grid file: Design concepts and experience. BIT Numerical Mathematics, 1985, 25(4): 569-592

[44] Hutflesz A, Six H W, Widmayer P. Twin grid files: Space optimizing access schemes. ACM SIGMOD Record, 1988, 17(3): 183-190

[45] βNievergelt J, Hinterberger H, Sevcik KC. The grid file: an adaptable, symmetric multikey file structure. ACM Transactions on Database Systems, 1984, 9(1): 38-71

[46] Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975, 18(9): 509-517

[47] Finkel RA, Bentley JL. Quad-trees: a data structure for retrieval on composite keys. Acta Informatica, 1974, 4(1): 1-9

[48] Meagher D. Geometric modeling using octree encoding. Computer Graphics and Image Processing, 1982, 19(2): 129-147

[49] Guttman A. R-trees: a dynamic index structure for spatial searching//Proceedings of the International Conference on Management of Data. Boston, USA, 1984: 47-57

[50] Beckmann N, Kriegel HP, Schneider R, Seeger B. The R*-tree: an efficient and robust access method for points and rectangles// Proceedings of the International Conference on Management of Data. Atlantic City, USA, 1990: 322-331

[51] Sellis T, Roussopoulos N, Faloutsos C. The R+-Tree: A dynamic index for multi-dimensional objects//Proceedings of the International Conference on Very Large Data Bases. Brighton, UK, 1987: 507-518

[52] Jung H R, Kim Y S, Chung Y D. QR-tree: An efficient and scalable method for evaluation of continuous range queries. Information Sciences, 2014, 274: 156-176

[53] Sagan H. Space-filling curves. Springer-Verlag GmbH, 2014, 12(8): 133–135

[54] Sagan H. Hilbert's space-filling curve(Space-filling curves). New York,USA: Springer, 1994

[55] Ramsak F, Markl V, Fenk R, et al. Integrating the UB-tree into a database system kernel//Proceedings of the International Conference on Very Large Data Bases. Cairo, Egypt, 2000: 263–272

[56] Kraska T, Alizadeh M, Beutel A, et al. Sagedb: A learned database system//Proceedings of the Biennial Conference on Innovative Data Systems Research. Asilomar, USA, 2019:1-10

[57] Li P, Lu H, Zheng Q, et al. LISA: A learned index structure for spatial data//Proceedings of the ACM SIGMOD international conference on management of data. Portland, USA, 2020: 2119-2133

[58] Wang H, Fu X, Xu J, et al. Learned index for spatial queries// Proceedings of the IEEE International Conference on Mobile Data Management (MDM). Hong Kong, China, 2019: 569-574

[59] Qi J, Liu G, Jensen C S, et al. Effectively learning spatial indices. Proceedings of the VLDB Endowment, 2020, 13(12): 2341-2354

[60] Royden HL, Fitzpatrick PM. Real Analysis, New York,USA: Macmillan, 1988

[61] Garcia EK, Gupta MR. Lattice regression//Proceedings of the Annual Conference on Neural Information Processing Systems. Vancouver, British Columbia, Canada, 2009: 594-602

[62] Qi J, Tao Y, Chang Y, et al. Theoretically optimal and empirically efficient R-trees with strong parallelizability. Proceedings of the VLDB Endowment, 2018, 11(5): 621-634

[63] Davitkova A, Milchevski E, Michel S. The ML-Index: A multidimensional, learned index for point, range, and nearest-neighbor queries//Proceedings of the 23rd International Conference on Extending Database Technology. Copenhagen, Denmark, 2020: 407-410

[64] Jagadish H V, Ooi B C, Tan K L, et al. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. ACM Transactions on Database Systems (TODS), 2005, 30(2): 364-397

[65] Schuh MA, Wylie T, Liu C, Angryk RA. Approximating high-dimensional range queries with kNN indexing techniques// Proceedings of the International Conference on Computing and Combinatorics. Atlanta, GA, USA, 2014: 369-380

[66] Nathan V, Ding J, Alizadeh M, et al. Learning multi-dimensional indexes//Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. Portland, USA, 2020: 985-1000

[67] Ding J, Nathan V, Alizadeh M, et al. Tsunami: A learned multi-dimensional index for correlated data and skewed workloads. Proceedings of the VLDB Endowment. 2020, 14(2): 74-86

[68] Zhang S, Ray S, Lu R, et al. Spatial interpolation-based learned index for range and kNN queries. arXiv preprint arXiv: 2102. 06789, 2021

[69] Zobel J, Moffat A, Ramamohanarao K. Inverted files versus signature files for text indexing. ACM Transactions on Database Systems (TODS), 1998, 23(4): 453-490

[70] Moffat A, Zobel J. Self-indexing inverted files for fast text retrieval. ACM Transactions on Information Systems (TOIS), 1996, 14(4): 349-379

[71] Zobel J, Moffat A. Inverted files for text search engines. ACM Computing Surveys, 2006, 38(2):6

[72] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(6): 734-749

[73] Oosterhuis H, Culpepper JS, De Rijke M. The potential of learned index structures for index compression//Proceedings of the Australasian Document Computing Symposium. Dunedin, New Zealand, 2018:7:1-7:4

[74] Rossi C, De Moura ES, Carvalho A, Silva AS. Fast document-at-a-time query processing using two-tier indexes//Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval. Dublin, Ireland, 2013: 183-192

[75] Goodwin B, Hopcroft M, Luu D, et al. BitFunnel: Revisiting signatures for search//Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. Shinjuku, Tokyo, Japan, 2017: 605-614

[76] Xiang W, Zhang H, Cui R, Chu X, Li K, Zhou W. Pavo: A RNN-based learned inverted index, supervised or unsupervised?. IEEE Access, 2019, 7: 293-303

[77] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks///Proceedings of the Annual Conference on Neural Information Processing Systems, Montreal, Canada, 2014:3104-3112

[78] Graves A. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013

[79] Mitzenmacher M. A model for learned bloom filters and optimizing by sandwiching//Proceedings of the Annual Conference on Neural Information Processing Systems. Montreal, Canada, 2018: 462-471

[80] Dai Z, Shrivastava A. Adaptive learned bloom filter (ada-bf): Efficient utilization of the classifier with application to real-time information filtering on the web. Advances in Neural Information Processing Systems, 2020, 33: 11700-11710

[81] Rae JW, Bartunov S, Lillicrap T. Meta-learning neural bloom filters// Proceedings of the 36th International Conference on Machine Learning, Long Beach, USA, 2019: 5271-5280

[82] Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T. Meta-learning with memory-augmented neural networks//Proceedings of the 33nd International Conference on Machine Learning. New York City, USA, 2016: 1842-1850

[83] Vinyals O, Blundell C, Lillicrap T, Kavukcuoglu K, Wierstra D. Matching networks for one shot learning//Proceedings of the Annual Conference on Neural Information Processing Systems, Barcelona, Spain, 2016: 3630-3638

[84] Bhattacharya A, Bedathur S, Bagchi A. Adaptive learned bloom filters under incremental workloads//Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. Hyderabad, India, 2020: 107-115

[85] Negi S, Dubey A, Bagchi A, et al. Dynamic partition bloom filters: A bounded false positive solution for dynamic set membership. arXiv preprint arXiv:1901.06493, 2019

[86] Macke S, Beutel A, Kraska T, et al. Lifting the curse of multidimensional data with learned existence indexes//Proceedings of the 32nd Conference on Neural Information Processing Systems. Montreal, Canada. 2018:1-6

[87] Llaveshi A, Sirin U, Ailamaki A, West R. Accelerating B+-tree search by using simple machine learning techniques//Proceedings of the International Workshop on Applied AI for Database Systems and Applications. California, CA, USA.2019: 1-10

[88] Hadian A, Heinis T. Interpolation-friendly B-trees: Bridging the gap between algorithmic and learned indexes//Proceedings of the International. Lisbon, Portugal, 2019: 710-713

[89] Qu W, Wang X, Li J, et al. Hybrid indexes by exploring traditional B-tree and linear regression// Proceedings of the International Conference on Web Information Systems and Applications. Cham,Swizerland: Springer, 2019: 601-613

[90] Gu T, Feng K, Cong G, et al. The RLR-tree: a reinforcement learning based R-tree for spatial data. arXiv preprint arXiv: 2103.04541,2021

[91] Pandey V, van Renen A, Kipf A, et al. The case for learned spatial indexes//Proceedings of the 2nd International Workshop on Applied AI for Database Systems and Applications, Tokyo, Japan, 2020:1-8

[92] Bentley J L, Friedman J H. Data structures for range searching. ACM Computing Surveys (CSUR), 1979, 11(4): 397-409

[93] Leutenegger S T, Lopez M A, Edgington J. STR: A simple and efficient algorithm for R-tree packing//Proceedings of the International Conference on Data Engineering. Birmingham, UK, 1997: 497-506

[94] Wu Y, Yu J, Tian Y, Sidle RS, Barber RJ. Designing succinct secondary indexing mechanism by exploiting column correlations//Proceedings of the International Conference on Management of Data. Amsterdam, Netherlands, 2019: 1223-1240

[95] Ghaffari B, Hadian A, Heinis T. Leveraging soft functional dependencies for indexing multi-dimensional data. arXiv preprint arXiv: 2006.16393, 2020

[96] Wu S, Pang Z, Chen G, et al. NEIST: A Neural-Enhanced Index for Spatio-Temporal Queries. IEEE Transactions on Knowledge and Data Engineering, 2019, 33(4): 1659-1673

[97] Kipf A, Marcus R, van Renen A, et al. SOSD: A benchmark for learned indexes. arXiv preprint arXiv:1911.13014, 2019

[98] Marcus R, Kipf A, van Renen A, et al. Benchmarking learned indexes. Proceedings of the VLDB Endowment, 2020, 14(1): 1-13

[99] Van Sandt P, Chronis Y, Patel J M. Efficiently searching in-memory sorted arrays: Revenge of the interpolation search?// Proceed-

ings of the International Conference on Management of Data. Amsterdam, Netherlands, 2019: 36-53
[100] Kim C, Chhugani J, Satish N, et al. FAST: fast architecture sensitive tree search on modern CPUs and GPUs//Proceedings of the ACM SIGMOD International Conference on Management of data. Indianapolis, USA, 2010: 339-350
[101] Graefe G. B-tree indexes, interpolation search, and skew//Proceedings of the 2nd international workshop on Data management on new hardware. Chicago, USA, 2006: 5
[102] Zhang H, Lim H, Leis V, et al. Surf: Practical range query filtering with fast succinct tries//Proceedings of the International Conference on Management of Data, Houston, USA, 2018: 323-336.
[103] Marcus R, Zhang E, Kraska T. Cdfshop: Exploring and optimizing learned index structures//Proceedings of the ACM SIGMOD International Conference on Management of Data. Portland, USA 2020: 2789-2792

CAI Pan, Ph.D. Her research interests include the learned index and database.

ZHANG Shao-Min, M.S. candidate. His research interests include index and machine learning.

LIU Pei-Ran, M.E. candidate. Her research interests include database and learned index.

SUN Lu-Ming, Ph.D. His research interests include database and machine learning.

LI Cui-Ping, professor,Ph.D. supervisor. Her research interests include Social network analysis, social recommendations, data analysis, and mining.

CHEN Hong, professor, Ph.D. supervisor. Her research interests include database and the new hardware platform for high-performance computing.

Background

The speed of data access has been one of the concerns in the database field. The most fundamental way for a database system to gain access to the data is via scanning all the recorded datasets. However, it takes significant time overhead in this way. Therefore, the index structure is proposed to improve the speed of data access. Although the index structure can improve the data access speed, the corresponding storage overhead increases, especially with large-scale data sets. Subsequently, the subsequent research proposes the compression scheme of index structure, i.e., space-saving by using prefix/suffix truncation, and dictionary compression. The traditional index structure (e.g., B+-tree) faces the challenges of the high space cost, the low query efficiency, and the more access overhead in the era of big data, which is characterized by explosive data growth, massive aggregation, and high dimensional complexity. When the dataset is large, the size of the tree index structure even exceeds the dataset itself. At the same time, the number of intermediate nodes to be traversed by the tree index structure during the query process increases, which increases the query overhead. Whereas, the compression scheme of indexes has little performance gain in the big data environment.

With the development and wide application of artificial intelligence and machine learning, researchers have found that artificial intelligence and database systems can be mutually beneficial. Therefore, it has become a hot research topic in the database field to apply artificial intelligence and machine learning methods to the database system to improve or replace traditional database system components to enhance the performance of traditional database systems, such as query optimization, and index structure. Based on this, learned indexes are proposed to solve the problems of traditional index structures in big data environments. Learned indexes use the underlying data distribution patterns to improve index performance and enhance data access speed and are one of the main applications to improve database system performance through artificial intelligence methods in recent years.

Learned indexes provide a new perspective for improving the performance of traditional index structures. In recent years, many research results have emerged around learned indexes. This paper summarizes these research results from the basic and extended problem levels. Then use the basic model as a clue to connect all the learned indexes, and classify and summarize the learned index according to the characteristics of the index structure and key technologies. This paper aims to provide a guide and help for relevant researchers interested in learned indexes.

This work is supported by the Fundamental Research Funds for the National Natural Science Foundation of China (62072460, 62076245, 62172424,62276270), Beijing Natural Science Foundation (4212022), Central Universities and the Research Funds of Renmin University of China (22XNH189).