软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(1):279−300 [doi: 10.13328/j.cnki.jos.007449] [CSTR: 32375.14.jos.007449]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

智能查询优化算法研究综述*

何家豪[1,3], 王嘉辰[1,3], 王 晓[1,3], 张喜盈[2,3], 李翠平[1,3], 陈 红[1,2]

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

[2](数据库与商务智能教育部工程研究中心 (中国人民大学), 北京 100872)

[3](中国人民大学 信息学院, 北京 100872)

通信作者: 李翠平, E-mail: licuiping@ruc.edu.cn

摘 要: 查询优化是数据库系统中至关重要的环节, 查询优化器通过找出一条查询语句对应的最佳查询计划来减少查询执行的代价. 传统优化器依赖固定规则或简单启发式算法加工并筛选候选计划. 然而随着实际应用中关系模式和查询逐渐复杂, 传统的查询优化器已经难以满足应用需求. 智能查询优化算法将机器学习技术应用到查询优化领域, 通过学习查询计划与复杂关系模式的特征来协助传统优化器完成查询优化. 此类算法在代价模型、连接优化、计划生成和查询改写等方面都提出了创新有效的解决方案. 梳理上述 4 类智能查询优化算法近年来的研究成果和发展脉络, 并对智能查询优化未来的研究方向进行展望, 希望研究者可以全面了解智能查询优化算法的研究现状, 以助于其后续科研工作的开展.

关键词: 查询优化; 人工智能; 强化学习; 数据库系统; 数据管理

中图法分类号: TP311

中文引用格式: 何家豪, 王嘉辰, 王晓, 张喜盈, 李翠平, 陈红. 智能查询优化算法研究综述. 软件学报, 2026, 37(1): 279–300. http://www.jos.org.cn/1000-9825/7449.htm

英文引用格式: He JH, Wang JC, Wang X, Zhang XY, Li CP, Chen H. Survey on Learned Query Optimization Algorithms. Ruan Jian Xue Bao/Journal of Software, 2026, 37(1): 279–300 (in Chinese). http://www.jos.org.cn/1000-9825/7449.htm

Survey on Learned Query Optimization Algorithms

HE Jia-Hao[1,3], WANG Jia-Chen[1,3], WANG Xiao[1,3], ZHANG Xi-Ying[2,3], LI Cui-Ping[1,3], CHEN Hong[1,2]

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

[2](Engineering Research Center of Database and Business Intelligence of the Ministry of Education (Renmin University of China), Beijing 100872, China)

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

Abstract: Query optimization is a critical component in database systems, where execution costs are minimized by identifying the most efficient query execution plan. Traditional query optimizers typically rely on fixed rules or simple heuristic algorithms to refine or select candidate plans. However, with the growing complexity of relational schemas and queries in real-world applications, such optimizers struggle to meet the demands of modern applications. Learned query optimization algorithms integrate machine learning techniques into the optimization process. They capture features of query plans and complex schemas to assist traditional optimizers. These algorithms offer innovative and effective solutions in areas such as cost modeling, join optimization, plan generation, and query rewriting. This study reviews recent achievements and developments in four main categories of learned query optimization algorithms. Future research directions are also discussed, aiming to provide a comprehensive understanding of the current state of research and to support further investigation in this field.

Key words: query optimization; artificial intelligence; reinforcement learning; database system; data management

---

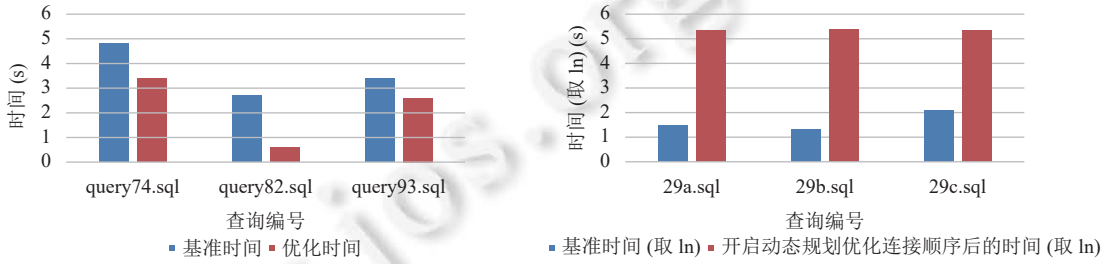
* 基金项目: 国家重点研发计划 (2023YFB4503600); 国家自然科学基金 (U23A20299, U24B20144, 62172424, 62276270, 62322214)
收稿时间: 2025-01-13; 修改时间: 2025-03-16; 采用时间: 2025-04-20; jos 在线出版时间: 2025-08-20
CNKI 网络首发时间: 2025-08-20

1 引 言

数据库系统是现代信息技术的重要组成部分, 广泛应用于各类数据密集型场景中. 数据库系统已从早期简单的存储系统发展为支持数据共享、决策与隐私保护的复杂系统. 数据库系统的核心任务是高效、可靠地存储、管理和检索数据, 而在数据库系统中促进高效完成这一系列任务的核心组件就是查询优化器. 查询优化器作为数据库系统中的重要组件, 它的主要功能是找出一条查询语句所对应的最优查询计划, 从而最小化查询执行的代价.

传统查询优化器会采用相对静态的优化策略以保证在尽可能多的应用场景下发挥作用, 其优化方式的缺陷主要体现在以下 3 点.

(1) 采用固定的优化规则进行流水线式地优化. 这些规则都是由数据库专家基于多年来的经验手动编写的. 例如 Spark SQL 的 Catalyst 查询优化器就包含 50 余条这样的规则. 然而在部分查询语句的优化中, 这些固定的规则并不一定总是会起到正面的作用, 对于 TPC-DS[1]数据集中的第 74、82、93 条查询, 分别关闭 reorder join、column pruning、eliminate outer join 规则, 反而可以得到更短的执行时间 (实验是在 Spark SQL 3.4.0 上进行的, 本节实验环境后同, 图 1(a)).



(a) TPC-DS 中部分查询在关闭个别优化规则前后的执行时间 (b) JOB 中部分查询在开启动态规划优化连接顺序前后的执行时间

图 1 部分查询在关闭优化规则与开启连接顺序优化前后的执行时间

(2) 采用启发式算法求解优化问题. 例如 PostgreSQL 会根据查询涉及的表数来决定采用动态规划还是一种名为 GEQO[2]的遗传算法来求解连接顺序. 这样的方法往往会因为自身过大的开销和基数估计的误差错过全局的最优解. 比如对于 JOB[3]数据集中的 29a、29b、29c 这 3 条查询, 其本身直接执行只需要花费 3–8 s 左右, 但是只是使用动态规划算法优化他们的连接顺序就需要花费 210 s 左右, 并且得到的还不一定是最优的连接顺序 (图 1(b)).

(3) 不会从过往的经验中学习. 当传统优化器在一个场景下犯了错误之后, 下一次它依然会犯同样的错误, 导致总体性能长期无法提升.

在这样的背景下, 智能查询优化算法应运而生. 智能查询优化算法指的是一系列运用机器学习模型辅助或代替传统查询优化器做出优化行为的算法, 其中包括代价估计算法、连接优化算法、计划生成算法以及查询改写算法等. 随着近年来智能查询优化算法的逐渐成熟, 许多商业数据库中也开始逐渐将这样的技术集成进自己的产品中, 比如华为的 OpenGauss 和 GaussDB 上实现了一种基于机器学习的基数估计方法[4]; 微软的 SCOPE 中采用了一种基于提示的智能查询优化算法[5]; Oracle 和 RedShift 中也有利用机器学习生成物化视图的算法[6].

虽然在本文之前已有一些提到智能查询优化算法的总结性工作, 比如文献 [7−12] 都有提到智能查询优化算法, 但是上述文献的综述对象是完整的智能数据库系统, 智能查询优化算法作为智能数据库系统的子模块并未得到详尽的阐述. 此外, 文献 [13] 专门对智能查询优化算法进行了综述, 但未提及许多新兴的研究方向 (如基于排序学习的代价模型等). 文献 [14] 虽然覆盖了大部分的计划表示方法, 但缺乏对于完整的查询优化算法的系统描述.

因此, 本文将综述代价模型、连接优化、计划生成、查询改写这 4 个方面近年来取得的研究进展, 总结出智能查询优化领域目前面临的挑战, 并指出未来可能探索的研究方向. 上述 4 个方面是智能查询优化算法的核心技术, 其中, 代价模型为优化决策提供量化依据, 连接优化解决多表关联的效率瓶颈, 计划生成实现完整执行方案的智能构建, 查询改写则从查询语句源头提升查询质量. 它们相互协同, 共同推动传统优化器向动态适应的智能化方向演进.

本文后续内容的关系如图 2 所示.

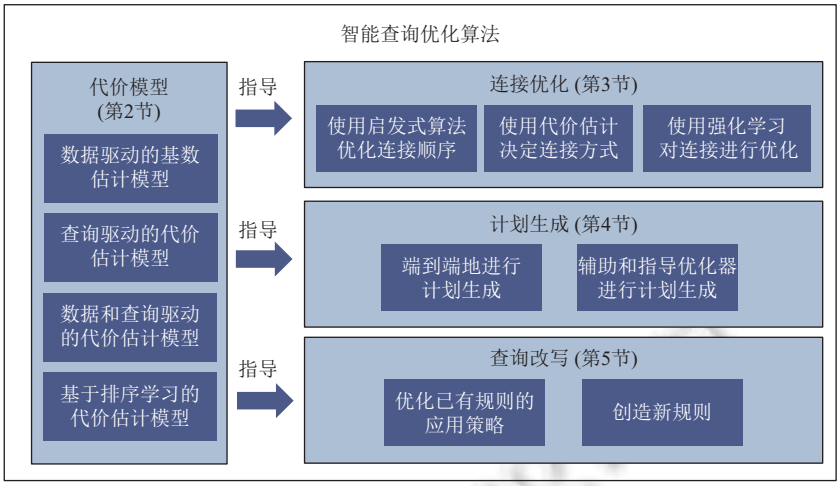


图 2 智能查询优化算法逻辑关系图

第 2 节介绍各类用于评估候选计划优劣的代价模型. 此类模型通过对查询进行建模作为输入, 然后采用机器学习的模型来评估查询计划的优劣. 虽然代价模型本身并不直接对查询进行优化, 但是许多算法都需要有代价模型来对其枚举出的候选方案进行评估, 所以代价模型是为它们提供指导的重要组成部分. 其中主要包括数据驱动的基数估计模型, 查询驱动的代价模型, 结合数据和查询驱动的代价估计模型以及基于排序学习的代价估计模型.

第 3 节介绍一系列连接优化算法. 此类方法的优化目标包括连接顺序和连接方式. 连接顺序和连接方式是决定查询性能的两个重要特征, 也是早期研究者们研究查询优化问题的主要优化目标. 在进入强化学习的时代之后, 连接优化算法也迎来了新的突破.

第 4 节介绍计划生成方法. 在机器学习模型可以单独解决代价估计和连接优化问题之后, 研究者们试图用机器学习方法为一条查询语句直接生成完整的计划, 而并非仅优化计划中某些单独的特征. 达成此目的方法主要分为两类: 端到端地进行计划生成与辅助和指导传统优化器进行计划生成.

第 5 节主要介绍查询改写算法. 查询改写的关注点并不是查询计划上的缺陷, 而是查询语句结构上的缺陷. 这些问题大多只能通过查询在真正输入数据库之前的查询改写环节解决, 所以查询改写是查询优化过程中不可替代的重要步骤. 查询改写一般基于规则进行, 其优化方式主要分为: 优化已有规则的应用策略和创造新的规则.

第 6 节总结本文的主要内容, 并对未来的研究方向进行展望.

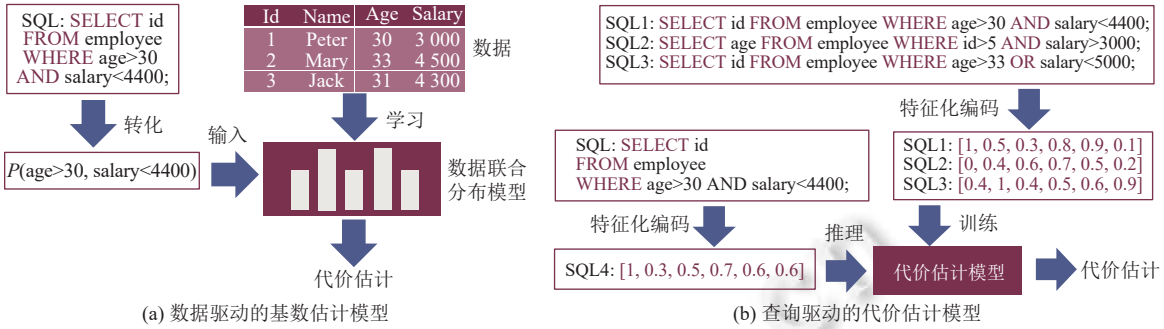
2 代价模型

代价模型输入一个 (子) 查询计划, 输出其对应的代价值. 虽然代价模型实现的功能比较简单, 但是它对其他各种智能查询优化算法具有指导性意义, 比如: 它可以判断哪种连接顺序的代价更低, 某一次扫描操作选择哪种算子代价更低等. 代价模型就通过这样的定量分析对其他查询优化算法 (如连接优化, 计划生成) 起指导作用, 它的性能可以直接影响其他优化算法的效果. 传统优化器中的代价模型设计比较简单, 它们一般会直接采用一个固定的公式将统计信息中的基数, 磁盘 I/O 开销等变量进行简单建模并计算得到预测代价. 这类代价模型的预测值与真实值往往偏差较大, 从而使得传统优化器会错误地选择次优计划. 而基于机器学习的代价模型可以从数据分布和查询计划中考虑更丰富的输入特征, 更复杂的关联关系, 从而达到更高的预测精度, 引导优化器选择更优的查询计划.

2.1 数据驱动的基数估计模型

基数估计指在一个子计划或算子执行之前事先预测出其输出的元组数. 虽然基数估计相对于代价估计而言缺少了对物理运算符和其他系统环境的感知, 但它还是影响着代价估计的最重要因素, 以至于一些传统优化器就是直接使用基数估计的结果来作为代价估计值. 所以研究者们最先开始改进的也是基数估计模型[15−17]. 其中数据驱

动的方法是其中的一个关键分支. 数据驱动的方法的核心思想是对数据的分布进行联合概率建模, 然后依照判断符合条件的元组在表中出现的概率对基数进行估算 (图 3(a)).



(a) 数据驱动的基数估计模型　　(b) 查询驱动的代价估计模型

图 3　数据驱动的基数估计模型和查询驱动的代价估计模型

传统的数据驱动方法仅利用单列信息并且假设表中列与列之间是独立的, 通过这些简单的假设收集信息来建立粗粒度模型. 随着研究的不断深入, 更多细粒度的数据建模模型逐渐出现, 其中包括基于深度自回归模型的[18]方法和基于关系型求和乘积网络 (RSPN) 的方法[19]等 (术语对照见附录 A). 这些模型的主要贡献在于更好地考虑了数据库列与列之间的联合分布以及函数依赖等相关性信息. 不过数据驱动的方法存在的不足之处在于: 一是数据驱动的模型一般参数量较大, 预测开销高; 二是只能对固定结构的查询 (如 SPJ 查询) 预测出其基数, 难以对包含大量聚合、子查询和复杂连接方式的查询进行代价预测. 表 1 中对比了一系列数据驱动的基数估计模型.

表 1　数据驱动的基数估计模型对比

方法	数据建模模型	动态更新方式	采样方式	是否支持多表查询
KDE[20]	核密度模型	维护样本	随机采样	—
MOSE[21]	格回归模型	不支持	随机采样	—
QuickSel[22]	均匀混合模型	依据查询更新	基于查询采样	—
DLM[23]	自回归模型	依据查询更新	适应性加权采样	—
Naru[24]	自回归模型	增量或重新训练	渐进式采样	—
NeuroCard[25]	自回归模型	增量或重新训练	渐进式采样	√
FLAT[26]	FSPN	增量更新FSPN	无采样	√
ASM[27,28]	自回归模型	增量或重新训练	随机采样	—
Grid-AR[29]	自回归模型	不支持	无采样	√
ICE[30]	多维索引模型	通过索引维护	排序空间采样	—
DeepDB[19]	RSPN	增量更新RSPN	无采样	√
LHist[31]	多维直方图	通过更新直方图	无采样	—
Chow-Liu Trees[32]	贝叶斯网络	重新计算CPD	无采样	—

2.2　查询驱动的代价估计模型

为了更深入地考虑查询计划的特征, 研究者们设计了查询驱动的代价估计模型. 对于查询驱动的代价模型而言, 无论模型输出代价的实际意义是基数还是端到端的时延, 或是其他某种抽象的代价值, 都只需要在训练时更换其标注值即可, 不影响模型设计. 因此, 后文提到的基数估计模型、代价估计模型等说法在方法上无本质区别.

对查询进行建模需要表示查询计划的树状结构、查询语句中的连接信息和每一个操作符内部的信息, 其建模难度更高. 但是使用擅长学习复杂特征分布的机器学习模型依然可以达到更好的效果. 因此查询驱动的基数估计方法因为其模型更加轻便, 且表现不亚于数据驱动的方法而逐渐被学者们深入研究 (图 3(b)). 但是因为查询驱动的方法[33,34]没有对数据进行建模, 所以难以适应数据分布的变化. 同时, 查询驱动的方法在面对未学习过的特征表

现较差, 模型性能下降时一般只能采用重新训练的方式适应新的环境.

2.3 数据和查询驱动的代价估计模型

研究发现, 单纯的数据驱动或查询驱动都无法达到理想的效果, 所以结合了二者的方法逐步出现. 这类方法会先将查询的特征和数据的特征 (统计直方图、样本位图等) 一起整合进行编码为某种向量, 然后将其作为后续模型的输入进行回归预测其代价. 这样的方法比数据驱动的方法更加轻便, 比查询驱动的方法更加具有适应性.

2.3.1 特征编码

早期的特征编码会直接将一条查询编码成一个向量[35]. 后来为了使得代价模型可以更深入地理解查询的树状结构, 后来的特征编码会将一条查询编码为结构较为复杂的张量. 对于一条查询而言, 可以编码的特征有: 分类型特征、数值型特征和谓词型特征.

● 分类型特征: 分类型特征是查询中最常见的特征. 查询中涉及的表名, 列名以及操作符类型都属于分类型特征. 这一类型的特征在编码时往往采用独热编码的方式. 然而这样的方式有几点不足: 一是在编码时就需要确定每一类特征会出现的所有情况, 导致后续拓展性较差 (比如有新的表, 或表中有新的列出现); 二是独热编码的向量维度比较高, 有效信息分布比较稀疏. 一些方法针对这一不足进行了改进, 如 QueryFormer[36]中会为分类型变量的独热编码额外进行一次嵌入操作, 将高维稀疏的独热编码向量映射到低维稠密的嵌入向量中; ALECE[37]则采用了一种特殊的二进制编码, 将表名特征编码的空间复杂度压缩到了 $O(\log_2 n)$.

● 数值型特征: 数值型特征包括传统优化器给出的预测值 (基数估计、代价估计和 IO 次数), 结点在查询计划中的高度等. 一般对于数值型特征的编码方式比较简单, 大多采用直接拼接或正则化后拼接在对应特征向量尾部的方式.

● 谓词型特征: 谓词型特征指在查询中出现的筛选条件, 形式一般为<列名, 关系运算符, 值>的三元组. 对于谓词的编码固然可以简单地将其中出现的条件运算符和列名当作分类型特征, 被判断的数值作为数值型特征来处理[38]. 但是考虑到谓词对于扫描算子和部分连接算子的基数估计而言至关重要, 许多方法对于谓词的编码做了特别的改进. 比如文献 [39−41] 中对谓词的编码采用了自然语言处理中的词嵌入技术, 将不同的谓词编码成了固定长度的嵌入向量. 这样的编码可以使得相似的筛选条件在高位嵌入空间中依然是相近的, 同时可以兼容所有类型的筛选, 包括字符串的 LIKE 条件和 IN 的判断, 但是它们没有考虑到数据分布的特征. 所以又有一些方法比如 Robust-MSCN[42], CEDA[43]等, 就会一并将样本位图、统计直方图等信息通过一些预先的筛选或自注意力机制进行编码, 整合到整体查询的编码之中. 这样的编码方式一定程度上反映了数据分布的特征, 实现了查询和数据的统一建模.

值得注意的是, 对于上述提到的诸多可编码的特征, 在实际运用中并非编码的越多, 使用的编码方式越复杂越好, 而是需要在不同的目标场景下进行灵活的运用. 比如当代价模型的目标只是降低代价估计的误差时, 将更多的统计信息进行编码可能会更有效; 而当代价模型运用于计划生成问题上时, 使用传统数据库的预测值可能更加重要. 对于一些任务而言, 选择正确的编码特征和编码方式甚至比选择正确的模型架构更加重要[14].

2.3.2 模型架构

以特征编码得到的张量为输入, 预测的代价值为输出的机器学习模型有很多种. 例如使用支持向量机作为代价估计模型的文献 [44], 利用核典型相关分析模型作为代价估计模型的文献 [34,45], 以及构建了运算符级的回归树模型的代价估计模型的文献 [46]. 然而, 相比于较为简单的机器学习模型, 深度学习模型可以学习到更复杂的数据特征, 在近期的研究中也更受到研究者们的青睐. 本节将主要关注用于进行代价估计的深度学习模型, 其中包括 Tree-RNN[47]、Tree-LSTM[48]、Tree-CNN[49]和 Attention-based[50]的方法.

● Tree-RNN (tree-structured recursive neural network): RNN[51]一般用于一维线性结构的输入, 所以研究者们为了对树状的查询计划编码输入使用 RNN, 对其进行了结构上的改进. RNN 会用节点的状态来表示模型学习到的信息, 对于一维线性结构的简单 RNN 而言, 第 i 个节点的状态一般仅取决于第 i–1 个节点的状态. 即 $H_i=f(W_{i-1}H_{i-1}+B_i)$, 其中 f 为激活函数, W_{i-1} 为第 i 个节点位置模型学习到的参数, H_i 为第 i 个节点的状态, B_i 为模型对第 i 个节点位置学习到偏置值. 但对于树状结构 (Tree-RNN) 而言, 一个节点的状态需要同时考虑其左右子节点的状态, 即

$H_i=f(W_lH_l+W_rH_r+B_i)$, 其中 W_l, W_r 为节点 i 左、右子节点位置学习到的参数. Tree-RNN 通过这样的方式从叶子节点开始不断向上递归, 将底层节点的信息逐步传递到高层节点, 从而捕获数据中的全局结构信息, 得到整棵树的根节点的表示.

● Tree-LSTM (tree-structured long short-term memory) 是一种特殊的 Tree-RNN 方法, 它在普通的 Tree-RNN 基础上使用了 LSTM[52]单元, 通过引入输入门 i_t、遗忘门 f_t 和输出门 o_t 机制来捕捉长距离的复杂依赖关系. 相比于普通的 LSTM 方法, Tree-LSTM 方法在计算门时, 也需要考虑左右子节点的隐藏状态 h^{L}_{t-1}、h^{R}_{t-1} 和单元状态 c^{L}_{t-1}、c^{R}_{t-1}. 同时, 该方法还为左右子节点分别引入了独立的遗忘门 f^{L}_t、f^{R}_t [53]. 文献 [38] 采用 Tree-LSTM 方法构建代价估计模型, 递归地处理树中的节点和子树, 以捕捉查询的树形结构特征, 包括父子节点的依赖关系和自底向上的信息流通路径.

● Tree-CNN (tree-based convolutional neural network) 是卷积神经网络[54]的一种特殊变体, 专门针对树形结构数据进行设计. 它模拟了卷积神经网络在图像数据中的卷积操作, 然后将这些本来应用于矩形数据上的操作扩展到了三角形的树形结构上, 有效地捕捉了节点之间的层次关系和局部特征. 在 Tree-CNN 中, 每个节点都与其子节点相关联, 并且每个节点都有一个表示其特征的向量. 卷积核沿着树的结构在树的节点上滑动. 需要注意的是, Tree-CNN 与普通的 CNN 不同的一处在于, Tree-CNN 在卷积的过程中不会缩小树的结构, 只会将树中每个结点的特征维度降低. 所以为了将一个向量树变换成一个一维向量进行线性回归得到代价估计值还需要进行一步池化操作. Tree-CNN 一般会选择动态池化的方法, 即: 在每一个维度选择各个节点中该维度的最大值作为输出一维向量对应维度的值.

● Attention-based 的方法: Attention-based[55]模型是一种用于处理序列到序列任务的深度学习模型. 此类模型的优点在于能够有效地捕捉序列中的长距离依赖关系和更高的并行计算性能. 借助自注意力机制可以计算序列元素 i 对元素 j 的注意力分数 A_{ij}, 从而反映元素间的相关性. 基于自注意力机制的回归预测模型中的代表工作是 QueryFormer[36], 它通过扩展位置编码为树高编码和使用树偏自注意力机制来聚合单个节点, 使得 Attention-based 的架构能够适用于查询计划的向量树结构. 同时也有研究者发现, 注意力机制的计算本身就涉及将序列中的元素分别映射到 (Query, Key, Value) 的空间中, 这与查询语句在某种特定的数据分布上执行查询的思想暗合. 所以 ALECE[37]利用了这一点, 它首先将来自数据的统计直方图编码进行自注意力运算, 使每一张表的直方图编码结合了各个其他表的相关性信息. 随后, ALECE 会将这一编码映射到 Key 和 Value 部分, 再与映射到 Query 部分的查询编码进行注意力计算. 这一过程相当于在 Transformer 架构内部执行了一次“元查询”操作, 最终将计算结果通过线性变换后输出.

从深度学习的角度来讲, 研究者们长期以来做出的一系列模型架构方面的改进可以被称为代表了对查询优化有用的归纳偏差: 也就是说网络的结构, 而不仅是其参数, 都是为了代价估计任务而生的.

2.3.3 模型设计异同分析与性能对比

表 2 列举并对比了若干查询优化算法中所使用的代价模型以及其特征编码, 模型架构和算法的优化目标.

表 2 各类代价模型使用的特征编码与模型架构

方法	特征编码						模型架构	优化目标
	操作符	表名	谓词	样本位图	直方图	预测值		
AVGDL[56]	√	√	√	—	—	—	LSTM	视图选择
Fauce[57]	√	√	√	—	—	—	DNN	基数估计
AIMeetsAI[58]	√	—	—	—	—	√	Hybrid DNN	索引推荐
ReJOIN[59]	√	√	√	—	—	—	FCNN	连接优化
Robust-MSCN[42]	√	√	√	√	—	—	MSCN	基数估计
Plan-Cost[47]	√	—	—	—	—	√	Tree-RNN	代价估计
E2E-Cost[38]	√	√	√	√	—	—	Tree-LSTM	代价估计
RTOS[60]	√	√	√	—	—	—	Tree-LSTM	连接优化

表 2 各类代价模型使用的特征编码与模型架构 (续)

方法	特征编码						模型架构	优化目标
	操作符	表名	谓词	样本位图	直方图	预测值		
Neo[39]	√	√	√	—	—	√	Tree-CNN	计划生成
Bao[61]	√	—	—	—	—	√	Tree-CNN	计划生成
DeepO[40]	√	√	√	—	—	—	Tree-CNN	计划生成
Prestroid[41]	√	√	√	—	—	—	Tree-CNN	代价估计
QueryFormer[36]	√	√	√	√	√	—	Transformer	多目标
ALECE[37]	√	√	√	—	√	—	Transformer	基数估计
PRICE[62]	√	√	√	—	√	—	Transformer	基数估计
CEDA[43]	√	√	√	—	√	—	Transformer	基数估计

从表 2 中可以看出, 优化目标不同的各类智能查询优化算法在选用和设计代价模型时具有一定的倾向性. 比如对于需要精确估计基数或代价的方法 (QueryFormer, ALECE 等) 就更倾向于采用更丰富的特征编码和更复杂的模型架构. 对于输出不只是代价估计或是基数估计的方法 (ReJOIN, AIMeetsAI 等) 就更倾向于采用比较简单的特征编码和轻量化的模型架构. 这是因为产生精确的基数估计或代价估计通常就需要编码更多包括样本位图或直方图等关于数据分布的信息. 而对于连接优化、索引推荐等优化目标而言, 通常只需要在若干个候选优化措施中选择正确的一项即可. 因为这样的动作空间比较离散, 所以就可以进而放弃一些细节的编码, 选择轻量化的模型换取在优化效率上的提升.

不过, 除了因为不同的优化目标而有倾向性的选择代价模型之外, 每个方法也会依据它们自身的特点设计独特的模块. 比如 Bao 为了追求极致的轻量化, 甚至放弃了编码表名和谓词, 从而达到可以在实际场景中落地级别的实用性; 又如 QueryFormer 为了达到其多目标优化的通用性, 不惜特征编码的复杂性而选择同时支持样本位图和直方图的编码. 这也说明了代价模型的设计不存在固定的“公式”, 所有的设计都应根据实际的问题场景调整.

除了在模型设计方面的差异之外, 各类模型在性能上也有差异. 表 3 中对比了一些具有代表性的代价估计模型在不同数据集上的性能数据[14]. 为了方便对比优化目标不同的各种模型, 表中的数据统一采用 Q-Error (Q-Error 的计算方式为: $Q\text{-}Error_i=\max(C_i/C_i', C_i'/C_i)$. 其中 C_i 为数据集中第 i 条查询的真实代价值, C_i' 为代价模型对数据集中第 i 条查询预测出的代价值) 的中位数和最大值表示. 其中 JOB-Light[3]和 STATS[63]数据集是来源于真实世界的电影信息和经过匿名转储的某网络论坛上的用户信息; TPC-H 和 TPC-DS[1]中的数据是相关领域专家根据业务经验合成的.

表 3 不同代价模型在不同数据集上的性能表现

方法	JOB-Light[3]	STATS[63]	TPC-H[64]	TPC-DS[1]
ReJOIN[59]	3.276–2 746	8.941–96 690	1.052–14.93	1.044–5.326
Plan-Cost[47]	2.070–189.5	8.035–30 380	2.711–79.52	2.016–295.3
E2E-Cost[38]	1.771–433.0	3.831–1 270	3.253–13.46	2.100–50.18
RTOS[60]	4.013–1853	3.793–32 650	1.071–1.961	1.064–6.615
Neo[39]	3.924–716.0	2.622–34 630	1.027–2.240	1.034–5.669
Bao[61]	2.242–316.0	2.028–457.2	1.008–1.331	1.085–3.956
Prestroid[41]	4.324–584.4	2.695–17 600	1.011–8.196	1.212–9.333
QueryFormer[36]	1.408–380.0	1.322–132.1	1.028–1.259	1.053–7.499
PostgreSQL	2.742–455.9	6.431–8 565	2.064–1 131	2.972–116 900

从表 3 可以看出, 不同代价模型在不同来源的数据集上的性能差异比较显著. 在 TPC-H 和 TPC-DS 等合成数据集上, 数据分布均匀且相关性低, 大多数模型表现较好. 例如, Bao、Prestroid 等模型的 Q-Error 中位数在 TPC-H

上接近 1, 而传统优化器 PostgreSQL 在 TPC-DS 上的最大 Q-Error 高达 116 900, 暴露了传统方法在大规模复杂查询中的不足. 然而, 在真实数据集 JOB-Light 和 STATS 中, 数据分布呈现高偏斜与强相关性, 模型性能分化也更加明显. 比如 QueryFormer 在 STATS 上的 Q-Error 最大值仅为 132.1, 显著优于其他模型, 突显出其处理复杂数据的鲁棒性; 但是轻量化模型如 ReJOIN 和 Plan-Cost 在真实数据集上表现相对较差, 因为其简单特征编码难以应对高偏斜场景.

深入分析性能产生的差异的原因可以发现, 代价估计的精准度与特征编码密切相关, 尤其是对谓词的精细编码和与传统数据库预测值的结合. 表现较优的 QueryFormer 和 Bao 都采用了精细的谓词编码或整合了传统数据库的基数估计等统计信息, 这样的设计使得他们在面对高偏斜数据时能够通过多维度特征缓解单一预测的偏差. 但是方法如 ReJOIN 的编码仅涵盖操作符、表名和简单谓词编码, 并未充分挖掘谓词逻辑或利用优化器的预测值, 导致其误差可能会在复杂查询中被放大.

2.4 基于排序学习的代价估计模型

对于大部分查询优化算法的目标而言, 代价模型的核心任务并不是追求精准地估计单个查询计划的代价值, 而是正确判断若干个候选计划之间的相对优劣. 因此, 很多情况下代价估计的数值本身并不是最重要的, 重要的是候选计划间的排序关系, 本节将介绍基于排序学习的代价模型训练方法[4,65−67].

一般的代价模型的训练流程是: 先执行训练数据集中的查询, 然后从数据库的执行记录中收集每条查询对应的查询计划和代价信息, 最后将查询计划中的特征编码作为回归预测模型的输入, 代价信息作为监督信号对模型进行训练[68]. 基于排序学习的代价模型对此流程的改进主要在两个方面.

(1) 划分训练批次的方式不同: 一般的代价模型训练的批次数据是通过打乱所有数据之后取 k 等份 (一般每份的数据量为 2 的整数次方), 然后每份作为一个批次输入模型; 而基于排序学习的代价估计模型是按照产生查询计划的语句对批次进行划分的. 也就是说一般的代价模型在一个批次里可能同时有 a 查询语句产生的第 i 个候选计划和 b 查询语句产生的第 j 个候选计划, 但是在基于排序学习的模型的训练批次中 a 查询语句产生的候选计划只会和 a 查询语句产生的其他候选计划在一个批次中. 这样划分训练批次更加有助于模型理解查询计划之间的相对优劣关系.

(2) 使用的损失函数不同: 一般用于代价模型的损失函数是 MSE 等常见于回归问题建模的损失函数. 而用于训练排序学习的代价估计模型所使用的损失函数不需要考虑每条数据本身的绝对误差, 而需要考虑数据之间相对关系判断的误差. 这样的损失函数主要分为两类: 成对 (pairwise) 的和成列表 (listwise) 的[69,70]. 成对的方法关注的是两个样本间的相对顺序 (大于或小于), 其通常使用交叉熵来作为损失函数. 成列表的方法则考虑的是所有样本间的整体排序, 其通常使用可以体现列表位序的损失函数. 以 COOOL[67]中成列表的损失函数为例:

$$L_{\text{list}}(\theta) = -\sum_{q\in Q} \ln \Pr_{\text{PL}}(\sigma_q;\theta),$$

其中, θ 是当前代价模型的参数, Q 是数据集中所有的查询, σ_q 是当前一条查询产生的若干查询计划代价值的真实排序, $\Pr_{\text{PL}}$ 的定义如下:

$$\Pr_{\text{PL}}(t_1^q > t_2^q > \ldots > t_n^q;\theta) = \prod_{j=1}^{n} \frac{\exp\left(s_j^q\right)}{\sum_{m=j}^{n} \exp\left(s_m^q\right)},$$

其中, s_j^q 是模型参数为 θ 时对查询语句 q 的第 j 个计划的代价 t_j^q 的预测得分, n 是查询语句 q 产生的计划数量.

基于排序学习的代价估计方法不直接预测精确的代价值, 而是专注于样本之间的相对优先级关系. 这种特点使其在面对训练数据中的噪声和不确定性时, 表现出了更强的适应性, 同时也可以节省大量的计算和内存资源. 所以基于排序学习的方法在处理大规模数据集或高维度特征时, 表现出了更高的效率和更好的性能.

2.5 与代价估计相关的其他工作

除了对输入的 (子) 查询计划进行评估的主线任务之外, 还有一些其他与代价估计相关的工作.

● 一些方法关注于某些特定查询结构细节的代价估计, 比如对于 SUBSTR、GROUP BY 的基数估计: 文献 [71] 提出了一个对于单表且包含 HAVING 子句的基数估计方法; 文献 [72,73] 提出了两种对包含子串筛选 (LIKE) 查询的基数估计方法; 文献 [74] 提出了一种可以对包含列属性为集合的查询进行基数估计的方法.

● 一些方法提出了对于收集代价模型训练资料的优化方向. 比如: DataFarm[75]可以从查询负载需求、数据需求和资源限制这 3 个角度出发生成量身裁定的数据; Hit the Gym[68]通过“宏加速”和“微加速”两种手段加速了训练数据的获取, 使得不必再通过真实执行查询语句来获取每条数据的标注值.

● PACE[76]提出了一个研究基数估计模型安全性的问题. 它指出如果商业产品中使用了基于机器学习的代价模型, 那么恶意攻击者只需要通过执行 SELECT 语句就可以攻击到代价模型的性能, 从而影响业务执行的整体效率. PACE 的作者还实现了一个恶意攻击代价估计模型的算法, 它可以有效降低现行的多种代价估计模型的效果.

3 连接优化

当一条查询需要从多个表中取得其需要的数据并进行分析时, 连接 (join) 操作是不可避免的. 而在连接涉及的数据量较大、数据相关性较强时, 连接操作的性能就会显得至关重要. 在 OLAP 应用场景中, 一条查询往往都会涉及大量数据以及多个表的连接操作, 一些对连接操作敏感的查询可能会因为没有进行有效的连接优化而导致性能下降数十倍. 因此, 连接优化多年来一直是被广泛研究的课题[77]. 连接优化算法在智能查询优化研究中具有奠基性意义: 通过智能化手段解决多表连接这一在查询优化领域中的经典问题成为机器学习技术应用于查询优化领域的关键突破口, 为后续更多的智能查询优化算法打下了基础. 目前, 连接优化的主流做法是使用强化学习优化连接顺序和方式.

3.1 使用启发式算法对连接顺序进行优化

连接顺序的优化是谈及连接优化问题时第 1 个可以直观地联想到的优化问题, 因为连接顺序的优化可以仅通过改变逻辑查询计划中表的连接顺序来实现. 连接顺序问题的解空间是一棵排列树, 传统优化器中的连接优化方法主要通过遍历这棵排列树, 同时结合代价估计对排列树进行剪枝来完成连接顺序优化. 这样的优化方式即使在代价估计足够准确的情况下也伴随着相当大的计算开销 ($O(n!)$)[78]. 更何况对于传统优化器的代价估计而言, 产生足以让动态规划算法得到最优连接顺序的代价估计几乎是不现实的. 其他如 GEQO[2]、QuickPick-1000[79]等启发式方法可以加速连接顺序的生成, 但通常无法生成最佳的连接顺序. 同时类似 System R、PostgreSQL 中的传统优化器在决定一个连接顺序并执行后就会忘记它们曾经优化过的这个连接顺序. 所以即使传统优化器在某一次连接优化结束后得到了最佳的连接顺序, 也会由于缺乏反馈, 重复地进行连接优化操作, 选择相同的次优顺序, 不会从之前或好或坏的选择中学习到经验.

3.2 使用代价估计值决定连接方式

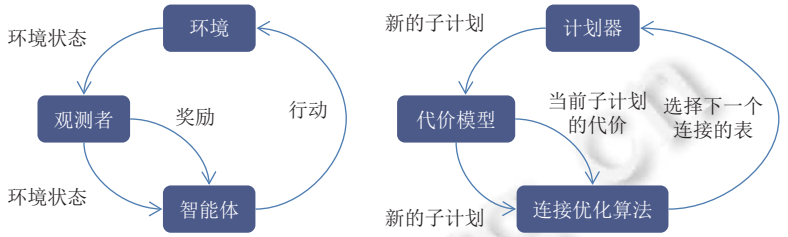
确定每一次连接操作的连接方式是在确定了连接顺序之后需要解决的第 2 个问题. 它决定了每一次逻辑上的连接 (join) 将采用什么样的物理方式实现, 比如归并连接 (merge join)、哈希连接 (hash join) 等. 它相较于连接顺序而言是一个相对简单的问题, 但也经常会影响性能. 传统优化器一般会根据它自身并不准确的代价估计选择连接方式, 所以常常会选择出次优的连接方式.

比如, 哈希连接在内存空间足够的前提下是一个不错的连接方式, 因为其时间复杂度是 $O(n)$; 但在内存空间无法容纳下构建侧构建出来的哈希表时, 它的性能就会因为频繁访问不在内存中的数据而退化到不如归并连接. 所以当错误的代价估计导致优化器认为内存空间足够, 但实际上内存空间不足时, 优化器就会错误地选择哈希连接导致性能退化.

类似的问题在分布式场景下的数据库系统中会更加严重, 因为分布式场景下还需要考虑到每一个具体的连接方式在节点之间传输数据量的成本, 比如即使都是哈希连接, 分布式场景下也需要考虑要采用的是将整个小表广播出去的广播哈希连接 (broadcast hash join) 还是按照键分布组合在每个节点上只完成部分连接任务的混洗哈希连接 (shuffle hash join). 连接方式的错误在严重的情况下可能会导致部分节点内存溢出而宕机.

3.3 使用强化学习对连接进行优化

为了解决这些挑战, 近年来出现了基于强化学习的连接优化算法. 这些方法利用强化学习模型从过去的实例中获取经验, 从而减少了因为代价估计错误而造成的连接顺序和方式错误. 同时它们的开销也比传统优化器的动态规划算法更低. 强化学习通过智能体与环境的交互逐步习得最优策略, 其基本模式如图 4(a) 所示, 将其应用于连接优化的基本模式如图 4(b) 所示. 这些方法有一部分只关注了连接顺序的问题, 有一部分同时也关注了连接方式, 但大致思路是一致的, 不再细做分类.



(a) 强化学习的基本模式 (b) 使用强化学习解决连接优化问题的基本模式

图 4 强化学习和使用强化学习解决连接优化问题的基本模式

ReJOIN[59]是第 1 个应用强化学习的连接优化算法. ReJOIN 算法将每张表与连接操作进行编码, 每个状态代表当前的连接树的子集, 每个动作则是将两个子树通过连接合并, 奖励基于代价模型给出, 目的是最小化最终选择的计划的代价. 系统通过这种方式逐步构建出完整的连接顺序. 在 ReJOIN 中, 每次选择表的连接都被视为一个回合, 从初始状态开始, 将当前状态输入神经网络后, 输出下一步的连接动作. 系统利用策略梯度方法在多个这样的回合中学习, 通过学习任务的持续反馈来优化神经网络的参数, 以寻找最优的连接顺序. 这样的方法相较于动态规划算法的显著改进之处如下.

(1) 找到的连接顺序更优: 采用基于机器学习的代价模型, 代价估计更准确, 更容易选择出正确的低代价连接顺序和连接方式.

(2) 效率更高: 动态规划算法即使使用了代价模型去进行剪枝, 也需要枚举大量的候选子计划, 算法的复杂度为 O(n!). 强化学习模型的推理开销显著低于动态规划算法, 更容易在有限开销内完成推理, 得到最终的连接顺序.

在 ReJOIN 的基础上, DQ[80]进一步优化了学习方法和在代价估计偏差较大时的微调策略, 同时证明了 DQ 只需要对于现有数据库进行很少的代码修改就可以集成. FOOP[81]在 DQ 的基础上进一步优化了状态的编码, 同时引进了双深度 Q 网络、近邻策略优化、优先级经验回放等更前沿的强化学习方法. 之后的 RTOS[60]通过引入了 Tree-LSTM 优化了子计划的表示来进行代价估计, 同时支持了数据库模式的变更. SOAR[82]则采用了图注意力网络来对子计划进行了表示. JOGGER[83]则进一步引入了图卷积神经网络和课程学习策略逐步增加训练集的难度, 从而加速模型的收敛. 这一系列连接优化方法的发展脉络如图 5 所示.

图 5 连接优化算法发展过程

基于强化学习的方法可以将连接优化问题解决得更好, 主要是因为强化学习与连接优化的任务高度契合, 体现于以下 3 点.

(1) 传统优化方法依赖于固定的规则和历史数据, 而强化学习具有平衡探索与利用的特点, 能够动态地在不同的查询场景中进行探索, 逐步发现更优的连接顺序和方式. 而且通过适当的探索, 可以避免陷入局部最优解;

(2) 数据库的工作负载是动态变化的, 不同的查询在不同的时间点上可能会有不同的最优执行计划. 强化学习具有动态适应性, 可以不断学习和调整策略以适应这些变化, 从而提供持续的优化效果;

(3) 连接优化通常涉及多个步骤, 强化学习具有延迟奖励机制, 可以综合考虑这些步骤的最终效果, 逐步优化整个查询计划, 而不仅是局部的某一步优化.

3.4 与连接优化相关的其他工作

除了使用强化学习找出最优的连接顺序和连接方式的主线任务之外, 还有一些与连接优化相关的其他工作.

● L1-error[84]提出了一种识别低效率的连接顺序的方法, 并指出对于大部分查询只要前 4 个表顺序是正确的, 整体性能就已经接近最优了. 所以在实际应用中只需要花比较小的开销规避掉最差的几种连接顺序就可以达到不错的性能效果, 不一定每次都花费很高的开销去寻找最优的连接顺序.

● ADOPT[85]提出了一种多表连接优化算法, 即一次连接多个连接条件在同样键上的表, 而不是每次仅多连接一个表. 这样连接表顺序的问题就变成了连接键顺序的问题, ADOPT 可以确保在最差情况下连接操作的复杂度是 O(n) 的.

4 计划生成

基于机器学习的方法在代价估计和连接优化两个特定方面取得不错的成效之后, 研究者们希望可以更进一步, 从生成整个计划的角度端到端地解决查询优化问题. 此方向的研究与单独地进行连接优化的最主要区别在于: 在计划生成算法中评估的对象通常是完整的查询计划, 而不是仅选择下一个表的连接顺序或者连接方式. 计划生成算法在智能查询优化研究中具有里程碑意义: 它突破了传统的单环节优化 (如仅解决代价估计或连接顺序问题) 的局限, 开始试图构建一个完整的智能查询优化系统, 标志着智能查询优化从模块化改进向全流程智能化的转变.

在人工智能赋能的查询优化算法尚未兴起之前, 前人就在此方向做过了许多研究. 比如: 文献 [86] 提出了一种对查询进行聚类, 然后对于在同一个聚类中的查询采取类似的查询计划的方案; 文献 [87] 中提出了一种特定的编码方式, 将优化关系型查询的问题建模成了一个删除自由的优化问题, 然后将贪心算法应用其中.

这些算法的研究从不同维度提升了系统性能. 有一些方法专注于直接降低查询执行的成本, 有一些则着眼于优化查询优化过程本身的资源消耗. 这些方法通常能够提供可观的优化成果, 实现相对简便, 并且多数采用可解释的白盒模型. 但是由于这类方法多依赖于预设的规则和静态的数据统计信息, 往往难以深入挖掘数据和查询内容中潜藏的复杂关系和模式. 所以这些传统算法在面对频繁变化的查询负载和相关性比较复杂的底层数据时会面临一定的挑战.

基于深度学习的技术开始应用于查询优化领域之后, 很大程度上克服了传统算法在处理复杂查询和动态数据环境中的局限性. 现在计划生成方向的工作基本可以分为两类: 一是选择抛弃传统数据库的查询优化器, 采用一个全新的算法来端到端地构造查询计划[39,88−92]; 二是采用较为渐进的策略, 将智能算法作为传统优化器的辅助和指导, 修正传统优化器产生的查询计划, 以期在保留现有系统可解释性的基础上引入智能算法的优势[5,40,61,93−95].

4.1 端到端地进行计划生成

虽然传统优化器具有种种缺陷 (见第 1 节), 但是想要完全用机器学习模型取代它还是有不小的难度[88]. 这主要体现在以下 3 点.

(1) 机器学习模型在训练的初期往往效果不佳, 需要很久才可以达到传统优化器的水平.

(2) 与其他机器学习任务不同, 对于计划生成任务而言收集训练数据需要实际执行若干查询并收集相关信息, 开销较大.

(3) 计划生成任务涉及的输出空间 (执行生成计划的动作) 极大, 收敛更加困难.

为了克服这些挑战, 后续端到端地取代现有传统优化器的方法基本上都应用了以下两种策略: 一是在训练初期依赖传统优化器的策略, 借此提高机器学习模型的初始表现; 二是采用基于代价模型的奖励值计算方法, 避免直接执行查询计划, 以加速模型的收敛.

Neo[39]是第 1 个基于强化学习的端到端的查询优化器, 它利用现有优化器的优化模型作为起点, 并持续从新查询中学习如何生成更优的查询执行计划. Neo 在一个基于 Tree-CNN 的代价模型的指导下执行一系列构造完整计划的决策. 开始时, Neo 会先初始化一个最小堆, 堆中的每个元素都是一个未经指派扫描方式的叶子节点. 然后 Neo 会逐步为这些元素选择对应的扫描方式, 然后再在他们之间逐步选择连接方式, 将它们从一个个叶子节点出发, 拼接成一个完整的查询计划. 在这个过程中 Neo 每次会选择堆中所有元素代价的最小值进行操作 (分配一个扫描方式或连接方式和连接的位置). 一个元素 (实际上是一个子计划) 代价定义为所有可能包含该元素的完整计划中代价的最小值. 从此处也可以看出, 计划生成类方法虽然也是在一步步构造计划, 但其每次评估的出发点却是整个计划. 以此类推, 直到达到收敛条件为止. 即使是从一个非常简单的优化器作为起点的 Neo, 也能学习出与当前最先进的商业优化器性能相当的效果, 甚至在某些情况下可以超过它们. Neo 证明了完全依赖于强化学习驱动的, 端到端的查询优化器是可行的.

继 Neo 之后, 若干类似的工作延续了这样的思路: Balsa[89]从启动策略和训练策略的角度优化了模型的收敛效率, 缩短了学习周期; LOGER[90]通过反向指定禁止操作符和奖励加权机制, 有效地解释了更优的计划为什么更优的问题, 从而降低了模型发挥效果的波动性; GLO[91]通过聚类数据库的统计信息和增加与传统数据库产生的计划进行对比, 大幅增加了泛化性, 降低了尾部性能退化. 表 4 对比了近年来端到端地进行计划生成的方法.

表 4 端到端地计划生成方法对比

方法	启动引导	选择计划生成动作的方式	代价模型	与传统优化器的关系
Neo[39]	传统优化器生成	直接指定	Tree-CNN	依赖
Balsa[89]	模拟器生成	直接指定	Tree-CNN	无关
LOGER[90]	无	反向指定	Graph Transformer	利用但不依赖
GLO[91]	无	直接指定	类QueryFormer	利用但不依赖

4.2 辅助和指导优化器进行计划生成

虽然研究者们在改进端到端的方法上做出了许多贡献, 但是第 4.1 节提到的 3 点缺陷一直未得到根治. 所以一些研究者们决定在计划生成的方向上后退一步, 不再试图完全取代整个传统优化器, 而是辅助和指导传统查询优化器进行计划生成. 其中的理论依据在于, 当传统优化器在进行计划生成时犯了一个小错误 (比如一处连接顺序选择错误), 虽然这个错误可能会导致性能骤降很多倍, 但是这终究是一个小错误导致的, 只需要对产生的计划进行少数几处的修正就可以使其达到最优效果, 不必完全取代传统优化器[93]. 这样的方法的优势如下.

(1) 起点是传统优化器产生的计划, 下限较高, 无需担心很久效果才能收敛至与传统优化器相当.

(2) 辅助和指导操作的输出空间比端到端地构造计划的输出空间更小, 更容易收敛.

(3) 方法可解释性强, 可以通过模型的辅助和指导行为推断出传统优化器具体在哪方面发挥较差.

(4) 工程实现简便, 易于集成进现有的数据库系统中, 生产环境更容易接受.

目前辅助和指导传统优化器的主要方式是“提示”传统优化器中某些规则的开关.

4.2.1 使用“提示”辅助和指导传统优化器进行计划生成

Bao 是此类方法的代表[61]. Bao 的作者在研究中揭示了一个新的发现: 对某些查询, 禁用某些传统优化器中的优化规则可以产生更优的查询计划. 基于这一发现, 该文献提出了一种查询优化指导系统 Bao, 它通过提示传统优化器要使用哪些规则来生成更优的查询计划. 具体而言, Bao 针对 PostgreSQL 数据库将 6 条特定优化规则的启用与否作为“提示”, 形成了 48 种不同的“提示集”. 这些规则包括 3 种指定连接方式的规则和 3 种指定扫描方式的规则. 在面对一个新的查询时, Bao 首先会进行规则开关的试验, 收集由这些规则组合引起的不同查询计划; 随后, 利用基于 Tree-CNN 的代价模型估算每个计划的执行时间, 最终选择最优计划执行. 这种方法比传统的基于固定规

则集的优化更加灵活, 并且与端到端的计划生成方法相比, 它更简便、收敛速度更快, 且极少出现“尾部灾难”现象.

Bao 对传统优化器的辅助和指导可以理解为通过使用“提示”指导传统优化器进行优化. 这样的方式需要首先构造出一个可以对计划生成过程产生影响的“提示集”, 然后再从“提示集”中为每条到来的查询选择适当的“提示”对传统优化器进行指导. 之后的 DeepO[40]在此基础上, 又将“提示集”推广到了连接顺序上, 并且还提供了带有置信度的输出以供用户更加清晰地认知到其辅助和指导的过程.

随着“提示集”的规模逐渐扩大, 研究者们发现如何为每一条查询确定一个合理的“提示集”探索空间成为一个新的问题. 比如在微软的 SCOPE 系统中有多达 256 条这样可开关的规则, 而且很多规则之间还存在依赖关系, 如果为每一条查询都去枚举 2^{256} 的数量级的规则开关组合的“提示”, 然后再找出最优的提示集是不现实的[5]. 所以 FASTgres[93]和 SCOPE 的研究者就提出了对查询进行分组, 在每一个组内定制一个相对小一点的“提示集”, 然后对组内的查询就只用在其组所对应的“提示集”内进行枚举的思路. 区别在于 SCOPE 采取的分组策略是根据“提示”对查询计划产生影响的特性将查询分组, 每组查询通过启用或禁用相应的“提示集”的子集, 能够产生不同的查询计划, 而无需考虑与组不相关的“提示”; FASTgres 则是依据参与连接的表和连接谓词作为分类依据将输入的查询语句进行分组, 基于这种分组, 系统再进一步提取各组查询的谓词作为特征, 针对每个特定的查询组独立训练一个专用的小型机器学习模型输出其相应的“提示”.

除了利用分组的思想限制“提示”的搜索空间之外, 研究者们还在实验过程中发现了“提示”的两个特点.

(1) 对于一条查询有提升的“提示”的子集也是有提升的“提示”.

(2) 一般有提升的“提示”的规模不会太大 (不超过 4).

所以探索“提示”是一个同时具有贪心选择性质和最优子结构的问题, AutoSteer[94]便从这个角度出发, 提出了一种贪心算法, 即: 在探索过程中如果发现单独关闭规则 R_1 和规则 R_2 都有提升, 那么下一步 AutoSteer 就会探索同时关闭规则 R_1 和规则 R_2 的“提示”. AutoSteer 会按照这样的方式探索, 直到最优规则集无法再进一步拓展为止. AutoSteer 通过这样的方式也一样可以有效地限制“提示”的搜索空间.

图 6 概括了使用“提示”对传统优化器进行辅助和指导的方法的发展流程.

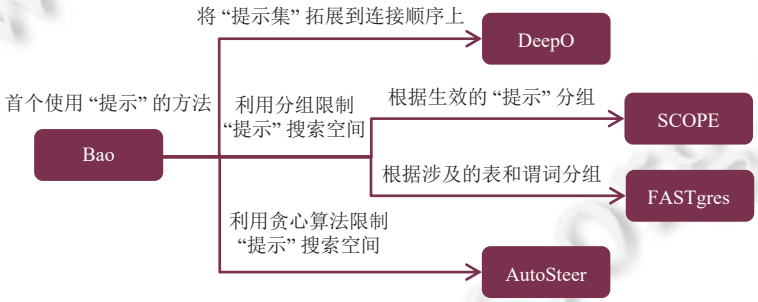


图 6 使用“提示”对传统优化器进行辅助和指导的方法的发展流程

4.2.2 通过其他方式辅助和指导传统优化器进行计划生成

● 在主流算法大多在考虑使用传统优化器中规则的开关作为“提示”时, Lero[65,66]选择将“增大或缩小传统优化器中的基数估计 10^n 倍”作为“提示”. 对于单条查询而言, 这样修改基数估计的“提示”虽然不一定完全精准, 但是根据排序学习 (第 2.4 节) 的思想, 对于即使不是完全正确的“提示”, 只要对其枚举出的候选计划之间的相对关系判断是正确的, 就依然可以选择出最优计划. 所以 Lero 在使用了这样的“提示”之后又使用了一个成对 (pairwise) 的代价模型去从枚举出的计划中选择最优的.

● FOSS[95]则放弃了在传统优化器开始优化之前就去进行“提示”的做法, 而是在传统优化器生成计划之后去再进行修补. 每一次的修补涉及一次连接方式、扫描方式的选择或是一次连接顺序的调整等. 这样的做法也有效地达到了不亚于在优化之前进行“提示”的方法的效果, 为辅助和指导传统优化器提供了新思路.

5 查询改写

查询改写是一类特殊的查询优化方法, 与前几种方法不同的是, 它的优化点并非是优化查询计划上的缺陷, 而是查询语句结构上的缺陷. 由于一条查询有无穷多种等价的 SQL 表述方式, 不同专家和应用程序写出的 SQL 各不相同, 所以查询改写可以优化的问题非常广泛. 并且其中有许多问题传统优化器难以注意到, 比如查询优化器很难通过某种固定的规则去分辨哪一层嵌套的子查询是冗余的, 或者哪一处的聚合其实是不必要的. 所以这些问题只能通过查询在真正输入数据库之前的查询改写环节解决, 即查询改写算法对于智能查询优化算法的研究而言具有不可替代的意义: 因为一旦查询语句通过了语法解析生成了对应的逻辑计划, 后续的查询优化过程会默认计划中每一步的操作都是必要的, 从而很少进行语义等价的替换优化.

查询改写是基于规则进行的[96]. 以广泛使用的 Apache Calcite 改写库为例, 它其中包含了 50 余条常见的改写规则, 每一条规则都代表着一种对查询语句的等价变换, 但这些规则并不保证一定会带来正优化. 所以对于查询改写方向的研究一方面在于如何决定要应用哪些规则, 以及用什么样的顺序应用这些规则; 另一方面在于创造新的规则.

5.1 优化已有规则的应用策略

在优化已有规则的应用策略方面, LLM-R2[97]通过提示大语言模型来为查询选择适合应用的规则集合, 其提示的核心原则有二: 一是它会用基于 QueryFormer[36]的方法来判断已经进行过改写的查询语句有哪些是与当前待改写的查询语句相似的; 二是与当前待改写的查询语句相似的查询语句改写效果如何, 然后让将这些信息传递给大语言模型, 让其判断对于当前这条待改写的查询而言什么样的改写规则集合更适合它.

但是 LLM-R2 没有考虑到应用改写规则的顺序. 相比而言, LearnedRewrite[98]将查询改写过程建模为一个蒙特卡洛树搜索的过程. 它将初始计划作为搜索树的根节点, 然后每一个节点的子节点都是从该节点出发可以通过应用互不冲突的改写规则得到的新计划. 通过这种方式, 策略树逐步展开, 每个分支代表了一系列不同的改写规则应用序列. 在每次迭代中, 蒙特卡洛树搜索会基于已探索节点的代价估计结果来决定下一步最有希望的探索方向.

从优化空间上来讲, LearnedRewrite 相比 LLM-R2 多考虑了可以按照不同顺序多次调用查询改写规则的情况(反映为在蒙特卡洛树搜索中的不同分支), 在都使用同样的 Calcite 查询改写规则库的基础上带来了更高的优化上限; 不过从优化效果上来讲, LearnedRewrite 使用的蒙特卡洛树搜索方式中的剪枝策略依赖于代价估计的结果, 而代价模型的准确度对于 LearnedRewrite 来说是一个较大的瓶颈, 在有些时候可能会因为次优的剪枝从而错失最佳的查询改写方案, 而 LLM-R2 通过评估查询之间的相关性只需要给出相关或不相关的判断即可, 相对错误率较低.

所以在 LLM-R2 和 LearnedRewrite 的前车之鉴下, 有研究者就开始考虑如何可以同时借用大语言模型的理解能力并且不丧失按照不同顺序多次调用查询改写规则的优化上限. 其中的代表工作 R-Bot[99]收集了数据库文档、查询优化规则代码和论坛问答内容作为“重写证据”. 然后与 LLM-R2 类似, 它会从这些“重写证据”中选择与要优化的查询相关的“证据”帮助大语言模型选择要使用的规则. 同时它也考虑到了有一些规则是在另外一些前置规则应用之后再应用才会有效果. 所以又与 LearnedRewrite 类似地, 它会采用迭代的方式逐步去改写一条查询, 这样就可以避免选择了正确的规则集合, 但是应用顺序不正确的问题.

5.2 创造新规则

经过多年来专家经验的积累, 现有改写规则库中的查询改写规则已经比较完备, 所以很多运用机器学习方法创造的新规则只适用于一些特定场景或特定结构的查询语句, 比如谓词之间有隐含的线性关系的查询 (SIA[100]) 或是 ORM 产生的查询 (WeTune[101]) 等. 同时创造新规则的查询改写方法需要证明其创造的规则满足逻辑等价性, 这通常需要严格的数学推导证明或者在每次改写完成之后判断改写前后的逻辑等价性. 这个过程虽然一般由谓词逻辑求解器完成, 但是由于求解的查询结构不同和求解成本的限制, 验证逻辑等价性依然是这类方法的瓶颈. 关于验证查询逻辑等价性的算法也有许多, 不过与查询优化算法的关系比较疏远, 本文不再列举.

对于一些谓词之间存在隐含线性关系的查询, SIA 可以通过查询中谓词之间隐含的线性关系合成出新的谓词.

SIA 合成出的谓词可以让传统优化器正常下推从而减少查询过程中处理的数据量. SIA 会先利用 SMT 求解器生成满足当前查询谓词的正例和不满足谓词的反例, 然后使用这些样本训练一个线性支持向量机模型, 该模型试图区分满足和不满足原始查询谓词的数据集. 然后再将这个支持向量机模型还原成若干条谓词添加到查询结尾.

目前主流的 Web 应用后端一般都会使用 ORM 来对数据库进行操作, 而不是程序员直接将操作数据库的查询模板写在代码里. 这样的方式对上层的程序员隐藏了底层的数据库操作, 对绝大多数开发人员带来了很大的便利, 但同时也为数据库的查询改写任务提出了新的挑战, 即: 改写 ORM 产生的查询语句, 而非经过人类先行优化过的查询语句. WeTune 是一种通过暴力穷举创造新查询改写规则的方法, 这种方法自动化地探索所有可能的逻辑查询计划变换, 以发现以往可能被忽视的查询改写规则. WeTune 会首先枚举所有有效的逻辑查询计划, 然后尝试发现在特定约束条件下等价但更高效的查询计划. 与 SIA 类似, WeTune 在鉴定可用性的过程中也采用了基于 SMT 求解器的验证方法.

同时, 有一些方法对规则的形式进行了创新. 现有规则库中的规则一般都采用正则表达式的方式编写, 然后通过专门的正则表达式执行器执行改写, 对人类而言可读性较差. GenRewrite[102]提出了一种叫作 NLR2 的改写规则, 它由自然语言编写, 大语言模型负责执行改写. GenRewrite 将自然语言改写规则作为其提示和知识转移的手段, 帮助大语言模型提供更准确的改写建议, 还通过迭代修正自然语言改写规则的提示显著减少了改写查询中的语法和语义错误.

以上 3 种方法都从创造新的查询改写规则角度推进了查询改写技术的发展, 但是它们也存在一定的弊端: SIA 只能通过拟合谓词的方式来促进“选择下移”过程, 对于不需要拟合谓词或连接的数据量不大时的优化效果有限, 同时对每一条查询都训练一个支持向量机的做法存在开销过高的隐患; WeTune 的应用场景主要专注于 Web 应用后端的查询场景, 此类查询的主要问题一般是由 ORM 程序在自动生成查询语句时产生的, 导致有许多冗余的嵌套和操作符, 但是对于人类专家撰写的查询可能优化空间有限; GenRewrite 虽然优化效果突出, 但是其优化过程中需要与大语言模型进行多次交互, 其时间开销与经济成本的限制导致 GenRewrite 几乎不可能作为在线算法使用, 只能用于优化有限数量且反复使用的查询.

综上, 本节提到的各种查询改写方法的详细对比如表 5 所示.

表 5　查询改写方法对比

方法	适用范围	优化方式	语义等价	算法特点
LLM-R2[97]	所有查询	应用已有规则	无需确保	提示大语言模型选择规则集合
LearnedRewrite[98]	所有查询	应用已有规则	无需确保	调整规则的运用顺序
R-Bot[99]	所有查询	应用已有规则	无需确保	迭代提示大语言模型使用规则
SIA[100]	有选择下移的查询	创造新规则	SMT求解器	SVM模型拟合新谓词
WeTune[101]	ORM产生的查询	创造新规则	SMT求解器	穷举合理改写方案的对应规则
GenRewrite[102]	所有查询	创造新规则	质询大语言模型	提示大语言模型产生规则

6　总结与展望

6.1　本文内容总结

近年来, 智能技术开始广泛地应用于解决数据库领域的问题, 迸发出了一系列新兴的研究成果. 其中查询优化作为数据库领域的重要研究方向, 也逐渐走上了智能化的道路. 本文首先介绍了在多种不同优化算法中都有使用的代价模型, 随后按照优化目标的不同, 将智能查询优化算法分为了连接优化、计划生成和查询改写这 3 个研究方向, 并对它们进行了系统地综述. 纵观这些方向可以发现, 目前将智能技术融入查询优化的方式基本上可以归纳为以下两点: 一是通过改进评估候选计划的精确度来在已有的解空间内找到更优的计划; 二是通过改进解空间的构造, 使其囊括更高的上限与包含更少的次优解, 以此来达到改进性能和降低开销的目的. 这些智能查询优化算法既可以从历史经验中学习, 又能运用这些经验应对未来可能出现的未知任务, 将传统优化器从一个相对静态系统

变成了一个可以动态更新、自我调整的智能化系统.

6.2 未来研究方向展望及挑战

数据库是现代信息系统中的关键基础软件, 而查询优化器作为数据库系统的核心组件, 其性能直接影响数据处理效率和系统整体表现. 与人工智能技术在其他领域的应用相比, 将人工智能技术引入查询优化面临更多独特的挑战. 其中最关键的挑战就是: 智能查询优化算法在走向实践的过程中需要证明其在某些极端情况下的性能上限高于传统优化器, 又需要确保其在普适的环境中的性能下限不低于传统优化器. 这就要求智能查询优化算法不仅需要创新, 也需要兼顾很高的可用性, 从而实现更稳健、更智能的优化能力. 在这一背景下, 智能查询优化算法在实际应用中依然面临很多问题, 解决以下 5 点瓶颈有望成为推动智能查询优化算法发展的关键.

● 降低查询优化模型训练和推理的开销: 传统优化器本身的启发式算法复杂度较高, 而当研究者们开始用机器学习模型替换这些算法的时候, 经常会强调其模型在收敛之后得到的优化效果相较于传统优化器提高了若干倍, 而很少关注其优化模型本身的训练和推理开销. 实际上, 对于一些智能查询优化算法而言, 只是训练机器学习模型就需要花费大量的时间和空间, 还有一些算法在推理阶段需要枚举比传统优化器多数倍的候选计划进行代价评估. 通过花费这样的高代价才能获取到一个性能超越传统优化器的模型在实际的应用场景中很难被接受. 未来的研究中应当注意此方面的问题, 比如可以使用一些轻量化的机器学习模型或采用知识蒸馏等机器学习技术来降低开销.

● 适应查询负载和数据分布的漂移: 使优化算法可以应对查询负载和数据分布的漂移是确保其下限不低于传统优化器下限的关键能力. 现有的智能查询优化算法有很多是在固定的数据集上进行训练, 然后再在固定的数据集上进行测试. 在此过程中数据库中的数据几乎不会发生增删改查, 用于训练的查询语句也只是结构比较固定的 SPJ 查询. 即使有少数方法通过实验证明可以做到对抗一定程度的波动 (比如部分关系模式发生变化, 查询语句的格式和部分数据发生变化等), 但其能对抗的波动程度也不及实际场景中的会发生的波动. 如果不能很好地解决适应性的问题, 这些智能查询优化算法应用于实际场景中就需要频繁的重新训练或增量学习. 未来的研究可以更加关注如何将迁移学习、元学习和领域泛化等技术应用于智能查询优化算法中, 从而使其具备更强的适应能力.

● 高效获取高质量的带标注的物理计划: 没有高质量的训练数据, 就没有高质量的优化模型. 而为了收集高质量的数据, 研究者们经常需要在数据库中频繁地执行查询语句然后抽取其对应的物理计划和代价作为数据和标注. 相较于一些其他可以通过人工标注获取数据的任务而言, 在数据库中执行查询语句然后进行标注的速度是非常慢的. 同时对于许多查询优化任务, 研究者们需要来自更多样化场景的数据, 比如覆盖金融、医疗、电子商务等多个方面的关系模式和数据分布, 而不只是现有常用的几个固定的数据集 (TPC-DS[1], JOB[3]等). 这些高质量的数据在现阶段获取的难度依然较高, 是智能查询优化算法发展的一个瓶颈. 未来的研究除了研究算法与模型本身, 应适当关注如何高效获取高质量的训练数据.

● 理解实际复杂生产环境中的语义信息: 查询优化面临的问题经常包括很复杂的语义信息, 有时单独使用固定的编码方式和深度学习模型有限的理解能力很难考虑周全. 比如一般的深度学习模型很难注意到在硬件环境、操作系统、数据分布、查询负载都不相同的两个优化场景在某一步上可以进行的优化操作可能是同理的. 而这些语义信息对于数据库中优化任务而言又是至关重要的, 所以现在一些研究者已经开始使用大语言模型助力数据库的优化从而有效利用这些信息. 比如使用大语言模型助力数据库优化过程的研究已经在参数调优[103,104], 运维诊断[105]以及与数据系统交互的方面[106]都有了一些初步的尝试, 而在查询优化方面的应用还处于比较初级的阶段, 未来的研究或可在此方向继续进行尝试.

● 增强智能查询优化的安全性: 现有研究多聚焦于优化效率与性能提升, 而并未充分关注智能查询优化算法的安全性问题. 例如, 基于机器学习的代价模型可能因对抗样本攻击而失效[76], 恶意用户可通过构造特定查询诱导优化器选择高代价计划, 导致服务降级. 此外, 隐私保护场景下 (如医疗、金融数据库), 虽然智能查询优化算法不会直接泄露具体的数据, 但是不能排除它可能会在训练或推理阶段泄露敏感数据的分布特征 (如均值、方差等)[107,108]. 未来可能需探索对抗训练、差分隐私[109]等技术在查询优化中的应用, 同时设计可解释性更强的模型以

检测异常优化决策, 确保智能优化算法在有安全约束下的可靠性.

References

[1] Nambiar RO, Poess M. The making of TPC-DS. In: Proc. of the 32nd Int'l Conf. on Very Large Data Bases. Seoul: VLDB Endowment, 2006. 1049–1058.

[2] Horng JT, Kao CY, Liu BJ. A genetic algorithm for database query optimization. In: Proc. of the 1st IEEE Conf. on Evolutionary Computation. Orlando: IEEE, 1994. 350–355. [doi: 10.1109/ICEC.1994.349926]

[3] Leis V, Gubichev A, Mirchev A, Boncz P, Kemper A, Neumann T. How good are query optimizers, really? Proc. of the VLDB Endowment, 2015, 9(3): 204–215. [doi: 10.14778/2850583.2850594]

[4] Chen X, Chen HT, Liang ZB, Liu SC, Wang JH, Zeng K, Su H, Zheng K. Leon: A new framework for ML-aided query optimization. Proc. of the VLDB Endowment, 2023, 16(9): 2261–2273. [doi: 10.14778/3598581.3598597]

[5] Negi P, Interlandi M, Marcus R, Alizadeh M, Kraska T, Friedman M, Jindal A. Steering query optimizers: A practical take on big data workloads. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2557–2569. [doi: 10.1145/3448016.3457568]

[6] Ahmed R, Bello R, Witkowski A, Kumar P. Automated generation of materialized views in Oracle. Proc. of the VLDB Endowment, 2020, 13(12): 3046–3058. [doi: 10.14778/3415478.3415533]

[7] Sun LM, Zhang SM, Ji T, Li CP, Chen H. Survey of data management techniques powered by artificial intelligence. Ruan Jian Xue Bao/Journal of Software, 2020, 31(3): 600–619 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5909.htm [doi: 10.13328/j.cnki.jos.005909]

[8] Li GL, Zhou XH, Cao L. AI meets database: AI4DB and DB4AI. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2859–2866. [doi: 10.1145/3448016.3457542]

[9] Chai MK, Fan J, Du XY. Learnable database systems: Challenges and opportunities. Ruan Jian Xue Bao/Journal of Software, 2020, 31(3): 806–830 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5908.htm [doi: 10.13328/j.cnki.jos.005908]

[10] Meng XF, Ma CH, Yang C. Survey on machine learning for database systems. Journal of Computer Research and Development, 2019, 56(9): 1803–1820 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2019.20190446]

[11] Li GL, Zhou XH, Sun J, Yu X, Yuan HT, Liu JB, Han Y. A survey of machine learning based database techniques. Chinese Journal of Computers, 2020, 43(11): 2019–2049 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2020.02019]

[12] Song YM, Gu Y, Li FF, Yu G. Survey on AI powered new techniques for query processing and optimization. Journal of Frontiers of Computer Science and Technology, 2020, 14(7): 1081–1103 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.1911063]

[13] Lan H, Bao ZF, Peng YW. A survey on advancing the DBMS query optimizer: Cardinality estimation, cost model, and plan enumeration. Data Science and Engineering, 2021, 6(1): 86–101. [doi: 10.1007/s41019-020-00149-7]

[14] Zhao Y, Li ZDH, Cong G. A comparative study and component analysis of query plan representation techniques in ML4DB studies. Proc. of the VLDB Endowment, 2023, 17(4): 823–835. [doi: 10.14778/3636218.3636235]

[15] Wang H, Sevcik KC. A multi-dimensional histogram for selectivity estimation and fast approximate query answering. In: Proc. of the 2003 Conf. of the Centre for Advanced Studies on Collaborative Research. Toronto: IBM Press, 2003. 328–342.

[16] Gunopulos D, Kollios G, Tsotras VJ, Domeniconi C. Selectivity estimators for multidimensional range queries over real attributes. The VLDB Journal, 2005, 14(2): 137–154. [doi: 10.1007/s00778-003-0090-4]

[17] Zhao ZY, Christensen R, Li FF, Hu X, Yi K. Random sampling over joins revisited. In: Proc. of the 2018 Int'l Conf. on Management of Data. Houston: ACM, 2018. 1525–1539. [doi: 10.1145/3183713.3183739]

[18] Nash C, Durkan C. Autoregressive energy machines. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 1735–1744.

[19] Hilprecht B, Schmidt A, Kulessa M, Molina A, Kersting K, Binnig C. DeepDB: Learn from data, not from queries! Proc. of the VLDB Endowment, 2020, 13(7): 992–1005. [doi: 10.14778/3384345.3384349]

[20] Heimel M, Kiefer M, Markl V. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In: Proc. of the 2015 Int'l Conf. on Management of Data. Melbourne: ACM, 2015. 1477–1492. [doi: 10.1145/2723372.2749438]

[21] Sun LM, Li CP, Ji T, Chen H. MOSE: A monotonic selectivity estimator using learned CDF. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(3): 2823–2836. [doi: 10.1109/TKDE.2021.3112753]

[22] Park Y, Zhong SC, Mozafari B. QuickSel: Quick selectivity learning with mixture models. In: Proc. of the 2020 ACM SIGMOD Int'l

Conf. on Management of Data. Portland: ACM, 2020. 1017–1033. [doi: 10.1145/3318464.3389727]

[23] Hasan S, Thirumuruganathan S, Augustine J, Koudas N, Das G. Deep learning models for selectivity estimation of multi-attribute queries. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. Portland: ACM, 2020. 1035–1050. [doi: 10.1145/3318464.3389741]

[24] Yang ZH, Liang E, Kamsetty A, Wu CG, Duan Y, Chen X, Abbeel P, Hellerstein JM, Krishnan S, Stoica I. Deep unsupervised cardinality estimation. Proc. of the VLDB Endowment, 2019, 13(3): 279–292. [doi: 10.14778/3368289.3368294]

[25] Yang ZH, Kamsetty A, Luan SF, Liang E, Duan Y, Chen X, Stoica I. NeuroCard: One cardinality estimator for all tables. Proc. of the VLDB Endowment, 2020, 14(1): 61–73. [doi: 10.14778/3421424.3421432]

[26] Zhu R, Wu ZN, Han YX, Zeng K, Pfadler A, Qian ZP, Zhou JR, Cui B. FLAT: Fast, lightweight and accurate method for cardinality estimation. Proc. of the VLDB Endowment, 2021, 14(9): 1489–1502. [doi: 10.14778/3461535.3461539]

[27] Kim K, Lee S, Kim I, Han WS. ASM: Harmonizing autoregressive model, sampling, and multi-dimensional statistics merging for cardinality estimation. Proc. of the ACM on Management of Data, 2024, 2(1): 45. [doi: 10.1145/3639300]

[28] Lee S, Kim K, Han WS. ASM in action: Fast and practical learned cardinality estimation. In: Companion of the 2024 Int'l Conf. on Management of Data. Santiago: ACM, 2024. 460–463. [doi: 10.1145/3626246.3654728]

[29] Gjurovski D, Davitkova A, Michel S. Grid-AR: A grid-based booster for learned cardinality estimation and range joins. arXiv:2410.07895, 2024.

[30] Li YZ, Liu XL, Wang HZ, Zhang KX, Wang ZX. Updateable data-driven cardinality estimator with bounded Q-error. arXiv:2408.17209, 2024.

[31] Liu QY, Shen YY, Chen L. LHist: Towards learning multi-dimensional histogram for massive spatial data. In: Proc. of the 37th IEEE Int'l Conf. on Data Engineering. Chania: IEEE, 2021. 1188–1199. [doi: 10.1109/ICDE51399.2021.00107]

[32] Halford M, Saint-Pierre P, Morvan F. An approach based on Bayesian networks for query selectivity estimation. In: Proc. of the 24th Int'l Conf. on Database Systems for Advanced Applications. Chiang Mai: Springer, 2019. 3–19. [doi: 10.1007/978-3-030-18579-4_1]

[33] Kipf A, Freitag M, Vorona D, Boncz P, Neumann T, Kemper A. Estimating filtered group-by queries is hard: Deep learning to the rescue. In: Proc. of the 1st Int'l Workshop on Applied AI for Database Systems and Applications. Los Angeles, 2019.

[34] Akdere M, Çetintemel U, Riondato M, Upfal E, Zdonik SB. Learning-based query performance modeling and prediction. In: Proc. of the 28th IEEE Int'l Conf. on Data Engineering. Arlington: IEEE, 2012. 390–401. [doi: 10.1109/ICDE.2012.64]

[35] Dutt A, Wang C, Nazi A, Kandula S, Narasayya V, Chaudhuri S. Selectivity estimation for range predicates using lightweight models. Proc. of the VLDB Endowment, 2019, 12(9): 1044–1057. [doi: 10.14778/3329772.3329780]

[36] Zhao Y, Cong G, Shi JC, Miao CY. QueryFormer: A tree Transformer model for query plan representation. Proc. of the VLDB Endowment, 2022, 15(8): 1658–1670. [doi: 10.14778/3529337.3529349]

[37] Li PF, Wei WQ, Zhu R, Ding BL, Zhou JR, Lu H. ALECE: An attention-based learned cardinality estimator for SPJ queries on dynamic workloads. Proc. of the VLDB Endowment, 2023, 17(2): 197–210. [doi: 10.14778/3626292.3626302]

[38] Sun J, Li GL. An end-to-end learning-based cost estimator. Proc. of the VLDB Endowment, 2019, 13(3): 307–319. [doi: 10.14778/3368289.3368296]

[39] Marcus R, Negi P, Mao HZ, Zhang C, Alizadeh M, Kraska T, Papaemmanouil O, Tatbul N. Neo: A learned query optimizer. Proc. of the VLDB Endowment, 2019, 12(11): 1705–1718. [doi: 10.14778/3342263.3342644]

[40] Sun LM, Ji T, Li CP, Chen H. DeepO: A learned query optimizer. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 2421–2424. [doi: 10.1145/3514221.3520167]

[41] Kang JKZ, Gaurav, Tan SY, Cheng F, Sun SX, He BS. Efficient deep learning pipelines for accurate cost estimations over large scale query workload. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1014–1022. [doi: 10.1145/3448016.3457546]

[42] Negi P, Wu ZN, Kipf A, Tatbul N, Marcus R, Madden S, Kraska T, Alizadeh M. Robust query driven cardinality estimation under changing workloads. Proc. of the VLDB Endowment, 2023, 16(6): 1520–1533. [doi: 10.14778/3583140.3583164]

[43] Wang ZL, Zeng QX, Wang N, Lu HW, Zhang Y. CEDA: Learned cardinality estimation with domain adaptation. Proc. of the VLDB Endowment, 2023, 16(12): 3934–3937. [doi: 10.14778/3611540.3611589]

[44] Akdere M, Cetintemel U, Riondato M, Upfal E, Zdonik S. The case for predictive database systems: Opportunities and challenges. CIDR. In: Proc. of the 5th Biennial Conf. on Innovative Data Systems Research. Asilomar, 2011. 167–174.

[45] Ganapathi A, Kuno H, Dayal U, Wiener JL, Fox A, Jordan M, Patterson D. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In: Proc. of the 25th IEEE Int'l Conf. on Data Engineering. Shanghai: IEEE, 2009. 592–603. [doi: 10.1109/

ICDE.2009.130]

[46] Li JX, König AC, Narasayya V, Chaudhuri S. Robust estimation of resource consumption for SQL queries using statistical techniques. Proc. of the VLDB Endowment, 2012, 5(11): 1555–1566. [doi: 10.14778/2350229.2350269]

[47] Marcus R, Papaemmanouil O. Plan-structured deep neural network models for query performance prediction. Proc. of the VLDB Endowment, 2019, 12(11): 1733–1746. [doi: 10.14778/3342263.3342646]

[48] Zhu XD, Sobhani P, Guo HY. Long short-term memory over recursive structures. In: Proc. of the 32nd Int'l Conf. on Machine Learning. Lille: JMLR.org, 2015. 1604–1612.

[49] Mou LL, Li G, Zhang L, Wang T, Jin Z. Convolutional neural networks over tree structures for programming language processing. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. Phoenix: AAAI, 2016. 1287–1293. [doi: 10.1609/aaai.v30i1.10139]

[50] Li Y, Wang LW, Wang S, Sun Y, Peng ZY. A resource-aware deep cost model for big data query processing. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala: IEEE, 2022. 885–897. [doi: 10.1109/ICDE53745.2022.00071]

[51] Elman JL. Finding structure in time. Cognitive Science, 1990, 14(2): 179–211. [doi: 10.1207/s15516709cog1402_1]

[52] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9(8): 1735–1780. [doi: 10.1162/neco.1997.9.8.1735]

[53] Tai KS, Socher R, Manning CD. Improved semantic representations from tree-structured long short-term memory networks. In: Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int'l Joint Conf. on Natural Language Processing (Vol. 1: Long Papers). Beijing: Association for Computational Linguistics, 2015. 1556–1566. [doi: 10.3115/v1/P15-1150]

[54] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc. of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

[55] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6000–6010.

[56] Yuan HT, Li GL, Feng L, Sun J, Han Y. Automatic view generation with deep learning and reinforcement learning. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering. Dallas: IEEE, 2020. 1501–1512. [doi: 10.1109/ICDE48307.2020.00133]

[57] Liu J, Dong WQ, Zhou QQ, Li D. Fauce: Fast and accurate deep ensembles with uncertainty for cardinality estimation. Proc. of the VLDB Endowment, 2021, 14(11): 1950–1963. [doi: 10.14778/3476249.3476254]

[58] Ding BL, Das S, Marcus R, Wu WT, Chaudhuri S, Narasayya VR. AI meets AI: Leveraging query executions to improve index recommendations. In: Proc. of the 2019 Int'l Conf. on Management of Data. Amsterdam: ACM, 2019. 1241–1258. [doi: 10.1145/3299869.3324957]

[59] Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration. In: Proc. of the 1st Int'l Workshop on Exploiting Artificial Intelligence Techniques for Data Management. Houston: ACM, 2018. 3. [doi: 10.1145/3211954.3211957]

[60] Yu X, Li GL, Chai CL, Tang N. Reinforcement learning with tree-LSTM for join order selection. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering. Dallas: IEEE, 2020. 1297–1308. [doi: 10.1109/ICDE48307.2020.00116]

[61] Marcus R, Negi P, Mao HZ, Tatbul N, Alizadeh M, Kraska T. Bao: Making learned query optimization practical. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1275–1288. [doi: 10.1145/3448016.3452838]

[62] Zeng TJ, Lan JW, Ma JH, Wei WQ, Zhu R, Zhou YL, Li PF, Ding BL, Lian DF, Wei ZW, Zhou JR. PRICE: A pretrained model for cross-database cardinality estimation. Proc. of the VLDB Endowment, 2024, 18(3): 637–650. [doi: 10.14778/3712221.3712231]

[63] Han YX, Wu ZN, Wu PZ, Zhu R, Yang JY, Tan LW, Zeng K, Cong G, Qin YZ, Pfadler A, Qian ZP, Zhou JR, Li JN, Cui B. Cardinality estimation in DBMS: A comprehensive benchmark evaluation. Proc. of the VLDB Endowment, 2021, 15(4): 752–765. [doi: 10.14778/3503585.3503586]

[64] Transaction processing performance council (TPC). TPC-H Version 2 and Version 3. 2021. http://www.tpc.org/tpch/

[65] Zhu R, Chen W, Ding BL, Chen XG, Pfadler A, Wu ZN, Zhou JR. Lero: A learning-to-rank query optimizer. Proc. of the VLDB Endowment, 2023, 16(6): 1466–1479. [doi: 10.14778/3583140.3583160]

[66] Chen XG, Zhu R, Ding BL, Wang SB, Zhou JR. Lero: Applying learning-to-rank in query optimizer. The VLDB Journal, 2024, 33(5): 1307–1331. [doi: 10.1007/s00778-024-00850-3]

[67] Xu XH, Zhao ZB, Zhang TY, Kang R, Sun LM, Chen JJ. COOOL: A learning-to-rank approach for SQL hint recommendations. arXiv:2304.04407, 2023.

[68] Lim WS, Ma L, Zhang W, Butrovich M, Arch S, Pavlo A. Hit the gym: Accelerating query execution to efficiently bootstrap behavior models for self-driving database management systems. Proc. of the VLDB Endowment, 2024, 17(11): 3680–3693. [doi: 10.14778/

3681954.3682030]

[69] Cao Z, Qin T, Liu TY, Tsai MF, Li H. Learning to rank: From pairwise approach to listwise approach. In: Proc. of the 24th Int'l Conf. on Machine Learning. Corvalis: ACM, 2007. 129–136. [doi: 10.1145/1273496.1273513]

[70] Liu TY. Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval, 2009, 3(3): 225–331. [doi: 10.1561/1500000016]

[71] Moerkotte G. Cardinality estimation for having-clauses. Proc. of the VLDB Endowment, 2024, 18(1): 28–41. [doi: 10.14778/3696435.3696438]

[72] Kwon S, Jung W, Shim K. Cardinality estimation of approximate substring queries using deep learning. Proc. of the VLDB Endowment, 2022, 15(11): 3145–3157. [doi: 10.14778/3551793.3551859]

[73] Aytimur M, Reiner S, Wörteler L, Chondrogiannis T, Grossniklaus M. LPLM: A neural language model for cardinality estimation of like-queries. Proc. of the ACM on Management of Data, 2024, 2(1): 54. [doi: 10.1145/3639309]

[74] Meng ZZ, Cao X, Cong G. Selectivity estimation for queries containing predicates over set-valued attributes. Proc. of the ACM on Management of Data, 2023, 1(4): 261. [doi: 10.1145/3626755]

[75] van de Water R, Ventura F, Kaoudi Z, Quiané-Ruiz JA, Markl V. Farming your ML-based query optimizer's food. In: Proc. of the 38th IEEE Int'l Conf. on Data Engineering. Kuala Lumpur: IEEE, 2022. 3186–3189. [doi: 10.1109/ICDE53745.2022.00294]

[76] Zhang JT, Zhang C, Li GL, Chai CL. PACE: Poisoning attacks on learned cardinality estimation. Proc. of the ACM on Management of Data, 2024, 2(1): 37. [doi: 10.1145/3639292]

[77] Zhou XH, Chai CL, Li GL, Sun J. Database meets artificial intelligence: A survey. IEEE Trans. on Knowledge and Data Engineering, 2020, 34(3): 1096–1116. [doi: 10.1109/TKDE.2020.2994641]

[78] Ioannidis YE, Kang YC. Left-deep vs. bushy trees: An analysis of strategy spaces and its implications for query optimization. In: Proc. of the 1991 ACM SIGMOD Int'l Conf. on Management of Data. Denver: ACM, 1991. 168–177. [doi: 10.1145/115790.115813]

[79] Waas F, Pellenkoft A. Join order selection (good enough is easy). In: Proc. of the 17th British National Conf. on Databases: Advances in Databases. Exeter: Springer, 2000. 51–67. [doi: 10.1007/3-540-45033-5_5]

[80] Krishnan S, Yang ZH, Goldberg K, Hellerstein J, Stoica I. Learning to optimize join queries with deep reinforcement learning. arXiv:1808.03196, 2019.

[81] Heitz J, Stockinger K. Join query optimization with deep reinforcement learning algorithms. arXiv:1911.11689, 2019.

[82] Zhou WQ, Zhan SY, Dai B, Guo L. SOAR: A learned join order selector with graph attention mechanism. In: Proc. of the 2022 Int'l Joint Conf. on Neural Networks (IJCNN). Padua: IEEE, 2022. 1–8. [doi: 10.1109/IJCNN55064.2022.9892450]

[83] Chen J, Ye GY, Zhao Y, Liu SC, Deng LW, Chen X, Zhou R, Zheng K. Efficient join order selection learning with graph-based representation. In: Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Washington: ACM, 2022. 97–107. [doi: 10.1145/3534678.3539303]

[84] Izenov Y, Datta A, Tsan B, Rusu F. Sub-optimal join order identification with L1-error. Proc. of the ACM on Management of Data, 2024, 2(1): 17. [doi: 10.1145/3639272]

[85] Wang JX, Trummer I, Kara A, Olteanu D. ADOPT: Adaptively optimizing attribute orders for worst-case optimal join algorithms via reinforcement learning. Proc. of the VLDB Endowment, 2023, 16(11): 2805–2817. [doi: 10.14778/3611479.3611489]

[86] Ghosh A, Parikh J, Sengar VS, Haritsa JR. Plan selection based on query clustering. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: VLDB Endowment, 2002. 179–190. [doi: 10.1016/B978-155860869-6/50024-X]

[87] Robinson N, McIlraith S, Toman D. Cost-based query optimization via AI planning. In: Proc. of the 28th AAAI Conf. on Artificial Intelligence. Québec City: AAAI, 2014. 2344–2351. [doi: 10.1609/aaai.v28i1.9045]

[88] Marcus R, Papaemmanouil O. Towards a hands-free query optimizer through deep learning. arXiv:1809.10212, 2018.

[89] Yang ZH, Chiang WL, Luan SF, Mittal G, Luo M, Stoica I. Balsa: Learning a query optimizer without expert demonstrations. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 931–944. [doi: 10.1145/3514221.3517885]

[90] Chen TY, Gao J, Chen HD, Tu YF. LOGER: A learned optimizer towards generating efficient and robust query execution plans. Proc. of the VLDB Endowment, 2023, 16(7): 1777–1789. [doi: 10.14778/3587136.3587150]

[91] Chen TY, Gao J, Tu YF, Xu M. GLO: Towards generalized learned query optimization. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 4843–4855. [doi: 10.1109/ICDE60146.2024.00368]

[92] Yu X, Chai CL, Li GL, Liu JB. Cost-based or learning-based? A hybrid query optimizer for query plan selection. Proc. of the VLDB Endowment, 2022, 15(13): 3924–3936. [doi: 10.14778/3565838.3565846]

[93] Woltmann L, Thiessat J, Hartmann C, Habich D, Lehner W. FASTgres: Making learned query optimizer hinting effective. Proc. of the VLDB Endowment, 2023, 16(11): 3310–3322. [doi: 10.14778/3611479.3611528]

[94] Anneser C, Tatbul N, Cohen D, Xu ZG, Pandian P, Laptev N, Marcus R. AutoSteer: Learned query optimization for any SQL database. Proc. of the VLDB Endowment, 2023, 16(12): 3515–3527. [doi: 10.14778/3611540.3611544]

[95] Zhong K, Sun LM, Ji T, Li CP, Chen H. FOSS: A self-learned doctor for query optimizer. In: Proc. of the 40th IEEE Int'l Conf. on Data Engineering. Utrecht: IEEE, 2024. 4329–4342. [doi: 10.1109/ICDE60146.2024.00330]

[96] Pirahesh H, Hellerstein JM, Hasan W. Extensible/rule based query rewrite optimization in starburst. ACM SIGMOD Record, 1992, 21(2): 39–48. [doi: 10.1145/141484.130294]

[97] Li ZDH, Yuan HT, Wang HM, Cong G, Bing LD. LLM-R^2: A large language model enhanced rule-based rewrite system for boosting query efficiency. Proc. of the VLDB Endowment, 2024, 18(1): 53–65. [doi: 10.14778/3696435.3696440]

[98] Zhou XH, Li GL, Chai CL, Feng JH. A learned query rewrite system using Monte Carlo tree search. Proc. of the VLDB Endowment, 2021, 15(1): 46–58. [doi: 10.14778/3485450.3485456]

[99] Sun ZY, Zhou XH, Li GL. R-Bot: An LLM-based query rewrite system. arXiv:2412.01661, 2024.

[100] Zhou Q, Arulraj J, Navathe S, Harris W, Wu JP. SIA: Optimizing queries using learned predicates. In: Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 2169–2181. [doi: 10.1145/3448016.3457262]

[101] Wang ZG, Zhou Z, Yang YC, Ding HR, Hu GS, Ding D, Tang CZ, Chen HB, Li JY. WeTune: Automatic discovery and verification of query rewrite rules. In: Proc. of the 2022 Int'l Conf. on Management of Data. Philadelphia: ACM, 2022. 94–107. [doi: 10.1145/3514221.3526125]

[102] Liu J, Mozafari B. Query rewriting via large language models. arXiv:2403.09060, 2024.

[103] Huang XM, Li HY, Zhang J, Zhao XX, Yao ZM, Li YY, Yu ZH, Zhang TY, Chen H, Li CP. LLMTune: Accelerate database knob tuning with large language models. arXiv:2404.11581v1, 2024.

[104] Lao JL, Wang YB, Li YF, Wang JP, Zhang YJ, Cheng ZY, Chen WH, Tang MJ, Wang JG. GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization. Proc. of the VLDB Endowment, 2024, 17(8): 1939–1952. [doi: 10.14778/3659437.3659449]

[105] Zhou XH, Li GL, Sun ZY, Liu ZY, Chen WZ, Wu JM, Liu JS, Feng RH, Zeng GY. D-bot: Database diagnosis system using large language models. Proc. of the VLDB Endowment, 2024, 17(10): 2514–2527. [doi: 10.14778/3675034.3675043]

[106] Xue SQ, Qi DR, Jiang CG, Cheng FY, Chen KT, Zhang ZP, Zhang HY, Wei GL, Zhao W, Zhou F, Yi H, Liu SD, Yang HJ, Chen FQ. Demonstration of DB-GPT: Next generation data interaction system empowered by large language models. Proc. of the VLDB Endowment, 2024, 17(12): 4365–4368. [doi: 10.14778/3685800.3685876]

[107] Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Proc. of the 22nd ACM SIGSAC Conf. on Computer and Communications Security. Denver: ACM, 2015. 1322–1333. [doi: 10.1145/2810103.2813677]

[108] Liu XM, Xie LH, Wang YP, Zou J, Xiong JB, Ying ZB, Vasilakos AV. Privacy and security issues in deep learning: A survey. IEEE Access, 2021, 9: 4566–4593. [doi: 10.1109/ACCESS.2020.3045078]

[109] Dwork C, Roth A. The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science, 2014, 9(3–4): 211–407. [doi: 10.1561/0400000042]

附中文参考文献

[7] 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. http://www.jos.org.cn/1000-9825/5909.htm [doi: 10.13328/j.cnki.jos.005909]

[9] 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. http://www.jos.org.cn/1000-9825/5908.htm [doi: 10.13328/j.cnki.jos.005908]

[10] 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. [doi: 10.7544/issn1000-1239.2019.20190446]

[11] 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. [doi: 10.11897/SP.J.1016.2020.02019]

[12] 宋雨萌, 谷峪, 李芳芳, 于戈. 人工智能赋能的查询处理与优化新技术研究综述. 计算机科学与探索, 2020, 14(7): 1081–1103. [doi: 10.3778/j.issn.1673-9418.1911063]

附录 A

论文中部分专业术语和缩略语如表 A1 所示.

表 A1 文中部分专业术语和缩略语

英文全称	英文缩写	中文名称	术语含义
Relational sum product network	RSPN	关系型求和乘积网络	一种用于捕获数据库中列之间联合概率分布的模型
Select project join	SPJ	选择投影连接	一种只包含选择投影连接的查询类型
Factorize sum split product network	FSPN	分解求和划分乘积网络	一种可以自适应分解联合概率分布的图模型
Conditional probability distribution	CPD	条件概率分布	条件概率分布
Multi set convolutional networks	MSCN	多集合卷积神经网络	一种以多集合为输入的卷积神经网络
Object-relational mapping	ORM	对象-关系映射	一种将编程语言中的对象与数据库中的记录对应起来的技术

作者简介

何家豪, 博士生, 主要研究领域为查询优化, 机器学习.

王嘉辰, 硕士生, 主要研究领域为代价估计, 机器学习.

王晓, 硕士生, 主要研究领域为连接优化, 强化学习.

张喜盈, 硕士生, 主要研究领域为基数估计, 机器学习.

李翠平, 博士, 教授, 博士生导师, CCF 杰出会员, 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.

陈红, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为数据库技术, 新硬件平台下的高性能计算.